Project Number: **770299**

NewsEye: A Digital Investigator for Historical Newspapers

Research and Innovation Action
Call H2020-SC-CULT-COOP-2016-2017

D2.7: Article separation (c) (final)

Due date of deliverable: M45 (31 January 2022)

Actual submission date: 31 January 2022

Start date of project: 1 May 2018 **Duration:** 45 months

Partner organization name in charge of deliverable: UROS

Project co-funded by the European Commission within Horizon 2020		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	-
RE	Restricted to a group specified by the Consortium (including the Commission Services)	-
CO	Confidential, only for members of the Consortium (including the Commission Services)	-

Revision History

Document administrative information		
Project acronym:	NewsEye	
Project number:	770299	
Deliverable number:	D2.7	
Deliverable full title:	Article separation (c) (final)	
Deliverable short title:	Article separation (final)	
Document identifier:	NewsEye-T23-D27-ArticleSeparation-c-final-Submitted-v6.0	
Lead partner short name:	UROS	
Report version:	V6.0	
Report preparation date:	31.01.2022	
Dissemination level:	PU	
Nature:	Report	
Lead author:	Johannes Michael (UROS)	
Co-authors:	Max Weidemann (UROS), Roger Labahn (UROS)	
Internal reviewers:	Günter Hackl (UIBK-DEA), Leo Leppänen (UH-CS)	
Status:		Draft
		Final
	x	Submitted

The NewsEye Consortium partner responsible for this deliverable has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Change Log

Date	Version	Editor	Summary of changes made
03/03/2021	1.0	Johannes Michael, Max Weidemann, Roger Labahn (UROS)	First full draft
03/03/2021	1.1	Johannes Michael, Max Weidemann (UROS)	Final changes before internal review
23/03/2021	2.0	Johannes Michael, Max Weidemann (UROS)	Taking reviewer comments into account
22/04/2021	3.0	Johannes Michael, Max Weidemann (UROS)	Final adjustments following quality management
03/01/2022	4.0	Johannes Michael, Max Weidemann, Roger Labahn (UROS)	First full draft of the updated deliverable
05/01/2022	4.1	Johannes Michael, Max Weidemann, Roger Labahn (UROS)	Final changes before internal review
21/01/2022	5.0	Johannes Michael, Max Weidemann, Roger Labahn (UROS)	Final update, taking reviewer comments into account
31/01/2022	6.0	Antoine Doucet (ULR)	Minor adjustments and submission

Executive summary

This report describes the third task of WP 2, namely the article separation task (Task T2.3). Our article separation approach is partially based on the layout analysis task (Task T2.1) giving us a set of baselines detected on a newspaper page. Therefore, this deliverable has to be seen in close connection to M24 (April 2020) public Deliverables D2.4 on layout analysis and D2.5 on automated text recognition, whose workflows provide us with helpful content information for further steps in the article separation process.

In year one we introduced a new error measure for article separation. Furthermore, we presented a first attempt of a simple clustering-based approach to extract articles. The underlying data structure for all our considerations is the well-established PAGE format[1], which is described in detail in Deliverable D1.9, Section 3.2.1.

In year two we introduced the concept of a 'news item', which is NewsEye's internal definition of an article. We use a two-stage strategy to extract them. The first stage of this methodology, namely text block segmentation, was the main goal of year two and the corresponding algorithms were tested and evaluated for a given set of newspaper pages.

In year three we focused on the second stage, namely forming news items from a combination of text blocks. To this end we introduce an updated article separation workflow. This includes a collection of new modules, which are described in detail in this report. Additionally, a competition regarding the text block segmentation was held during the last year and we briefly present its results. Besides, this public document also includes some crucial information from previous, non-public deliverables (D2.3, D2.6) about the article separation measure, the definition of a news item and the data formats.

In the prolongation phase of the project, new data sets were processed, a new user-feedback inspired measure was introduced and some additional experiments were done, including work in collaboration between UROS and ULR. Finally, the software modules from year three were migrated to the Transkribus platform.

The source code developed in this task is openly available on GitHub and is listed under the NewsEye repositories[2].

Overall, when comparing our final results to our original baseline method, we record relative error improvements of $73.7\,\%$, $49.6\,\%$ and $48.6\,\%$ (depending on the data set) regarding the article separation task. This is above the KPI goal which was set at a $20 - 40\,\%$ relative improvement over initial benchmarks.

[1]https://www.primaresearch.org/schema/PAGE/gts/pagecontent/2016-07-15/Simple%20PAGE%20XML%20Example.pdf
[2]https://github.com/NewsEye/Article-Separation

Contents

Executive Summary **4**

1 Introduction **7**

2 Proposed article separation measure **8**
2.1 Notation 9
2.2 R and P matrices 9
2.3 R-, P- and F-value for article separation 10

3 From articles to news items **11**

4 Data **13**
4.1 Ground truth format 13
4.2 NewsEye data sets 13

5 Workflow **15**

6 Modules **16**
6.1 Baseline detection 16
6.2 Automatic text recognition 16
6.3 Text block detection 16
6.3.1 Baseline clustering 19
6.3.2 Semantic segmentation using RU-Net 19
6.3.3 Instance segmentation using Mask R-CNN 20
6.4 Separator detection 21
6.4.1 RU-Net for separator detection 22
6.4.2 Post-Processing 23
6.4.3 Baseline correction 24
6.5 Heading detection 24
6.5.1 ARU-Net for heading detection 25
6.5.2 Stroke Width Transform for heading detection 25
6.5.3 Combination of both approaches 26
6.6 Content based features for text block relations 27
6.6.1 Word vector based text block similarities 27
6.6.2 BERT based text block similarities 28
6.7 Graph neural network 29
6.8 Text block clustering 31

7 Experiments and results **32**
7.1 Semantic context with BERT 33
7.2 GNN training with pseudo ground truth 34
7.3 GNN generalization capabilities 34
7.4 News item clustering 36
7.5 Internal user satisfaction 37

8 Competition on text block segmentation **38**

9 Prolongation Update **39**
9.1 New data sets 39

9.2 Comparative article separation measure 43
9.3 ULR text block similarity feature 43
9.4 Rule-based post-processing of confidence graph 44
9.5 Hierarchical text block clustering 46
9.5.1 Cluster distance metric 46
9.5.2 Cluster extraction 48
9.5.3 Evaluation 50
9.6 Internal user satisfaction survey 51
9.7 Future work 52

1 Introduction

The purpose of the NewsEye project is to enable historians and humanities scholars to investigate a great amount of newspaper collections. The newspaper pages are digitized and are available as scanned images. To ensure efficient work, the data processing steps should be as automatic as possible. Generally, newspapers are structured into large numbers of articles. These usually contain a distinct piece of content or describe a certain topic and can mostly be understood without any context. Newspaper articles are crucial entities for historians and humanities scholars who focus on a specific research area and are only interested in articles related to that topic. Additionally, some natural language processing (NLP) applications, like e.g. topic modeling (T4.1) or event detection (T3.3), rely on a logical structuring of the underlying text, to be able to extract meaningful information. For this reason it is important to tackle the article separation task, which tries to form coherent articles, based on the previously detected baselines (see public Deliverable D2.4) and their respective text (see public Deliverable D2.5).

In year three we introduce an updated workflow on article separation using a mixture of machine learning models and traditional layout analysis (LA) and clustering algorithms, incorporating results from tasks T2.1 and T2.2, respectively available in Deliverables D2.4 and D2.5. For the readers who are not familiar with the previous (non-public) Deliverables D2.3 and D2.6, this deliverable includes the descriptions of our article separation (AS) measure, news items and the used data formats in Sections 2, 3 and 4.1 respectively. A listing of the data sets used for training and evaluating the machine learning models as well as the three big use cases (one for each library partner) can be found in Section 4.2. A brief overview of the workflow is given in Section 5. We provide a more detailed description of each component in Section 6. In Section 7 different kinds of experiments are presented, e.g. on the central component, the graph neural network (GNN), and on the final articles/news items. Also, last year we presented the competition proposal for the ICPR2020 on text block segmentation. The competition was successful and presented in early 2021 at the postponed ICPR2020 conference, as described in Section 8.

In the prolongation phase of the project new data sets in Swedish and English/French were processed and some additional experiments were done in collaboration between UROS and ULR. Furthermore, we collected and evaluated AS feedback in an internal user satisfaction survey, which also inspired a new comparative AS measure. Finally, the software modules introduced in year three were migrated to the Transkribus platform. For the purpose of this deliverable, all updates regarding the prolongation have been collected in Section 9.

As a reminder, the description of the AS task in the NewsEye description of action is repeated below.

Task 2.3 – Article separation (AS)

This task aims to finally separate newspaper articles, give a coarse classification into basic types of text vs. non-text blocks, and detect basic named entities with a notable semantic meaning. We will follow two basic approaches: (a) Geometry and semantics based approach will employ geometric and textual output from preceding tasks, and combine them to cluster detected text lines to articles, unveil their types and distinguish entities like title, place or date of an article. That will be further improved by using available layout knowledge, by incorporating Language Models, and by exploiting its textual content found by preceding Text Mining/Topic Modeling techniques. Due to its novelty, the latter will mainly start as a research topic in synergy with WPs 3 and 4, and be joint work with UH-CS and ULR.

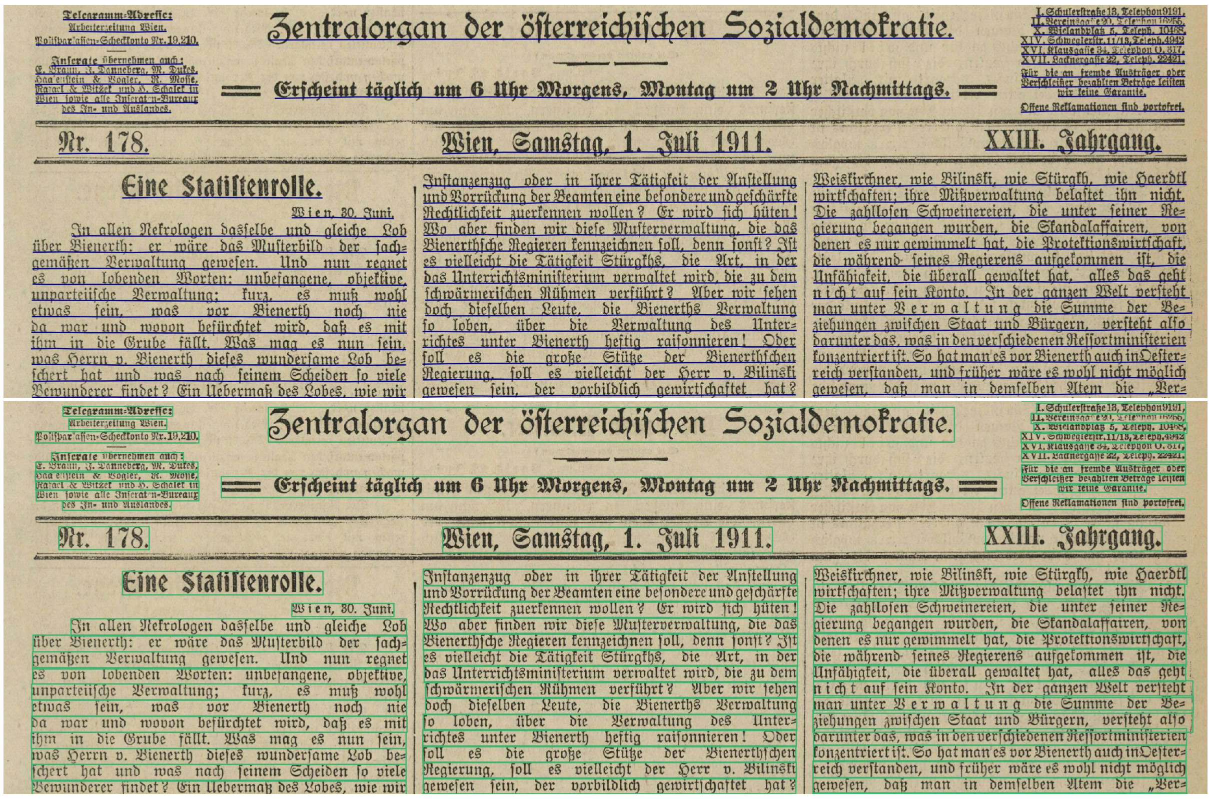
(b) Machine Learning based approach will investigate different concepts based on Machine Learning technology. For recovering text mining technologies' potential for separating newspaper articles, this approach again starts from a rather early research stage and is led in collaboration with ULR and UH-CS. The task comprises various neural methods for clustering text lines found in preceding Layout Analysis modules as well as an end-to-end approach using direct pixel labeling in the raw image.

2 Proposed article separation measure

Each article consists of a set of text blocks, which in turn consist of a set of baselines (or text lines). Thus, the AS task can be interpreted as a clustering problem over baselines and any clustering measure could be used to compute quantitative results. However, since the baselines are not given in real-world applications and must be determined first (see Task 2.1), it would be useful to be able to evaluate an end-to-end scenario, i.e. where the first step is to automatically detect baselines, which should then be clustered into articles. Therefore, we designed a new AS evaluation measure that can take the quality of the baselines into account. Since baselines are a central concept in this report, we want to remind the reader of their meaning.

Definition 1 (Baseline)**.** A *baseline* is defined in the typographical sense as the virtual line where most of the characters rest upon and descenders extend below.

In other words, a baseline can be described by a polygonal chain where the text rests upon, whereas the corresponding text line is a polygon that encloses the entire text. This is visualized in Figure 1.


Telegramm-Adreſſe:
Arbeiterzeitung Wien.
Poſtſparkaſſen-Scheckkonto Nr. 19.210.

Inſerate übernehmen auch:
E. Braun, J. Danneberg, M. Dukes,
Haaſenſtein & Vogler, R. Moſſe,
Rafael & Witzek und H. Schalek in
Wien ſowie alle Inſeraten-Bureaux
des In- und Auslandes.

Zentralorgan der öſterreichiſchen Sozialdemokratie.

Erſcheint täglich um 6 Uhr Morgens, Montag um 2 Uhr Nachmittags.

I. Schulerſtraße 13, Telephon 9191,
II. Vereinsgaſſe 20, Telephon 16255,
X. Wielandplatz 5, Teleph. 10468,
XIV. Schweglerſtr. 11/13, Teleph. 4942
XVI. Klausgaſſe 34, Telephon O. 317,
XVII. Lacknergaſſe 22, Teleph. 22421.
Für die an fremde Austräger oder Verſchleißer bezahlten Beträge leiſten wir keine Garantie.
Offene Reklamationen ſind portofrei.

Nr. 178. Wien, Samstag, 1. Juli 1911. XXIII. Jahrgang.

Eine Statiſtenrolle.

Wien, 30. Juni.

In allen Nekrologen dasſelbe und gleiche Lob über Bienerth: er wäre das Muſterbild der ſach-gemäßen Verwaltung geweſen. Und nun regnet es von lobenden Worten: unbefangene, objektive, unparteiiſche Verwaltung; kurz, es muß wohl etwas ſein, was vor Bienerth noch nie da war und wovon befürchtet wird, daß es mit ihm in die Grube fällt. Was mag es nun ſein, was Herrn v. Bienerth dieſes wunderſame Lob be-ſchert hat und was nach ſeinem Scheiden ſo viele Bewunderer findet? Ein Uebermaß des Lobes, wie wir Inſtanzenzug oder in ihrer Tätigkeit der Anſtellung und Vorrückung der Beamten eine beſondere und geſchärfte Rechtlichkeit zuerkennen wollen? Er wird ſich hüten! Wo aber finden wir dieſe Muſterverwaltung, die das Bienerthſche Regieren kennzeichnen ſoll, denn ſonſt? Iſt es vielleicht die Tätigkeit Stürgkhs, die Art, in der das Unterrichtsminiſterium verwaltet wird, die zu dem ſchwärmeriſchen Rühmen verführt? Aber wir ſehen doch dieſelben Leute, die Bienerths Verwaltung ſo loben, über die Verwaltung des Unter-richtes unter Bienerth heftig raiſonnieren! Oder ſoll es die große Stütze der Bienerthſchen Regierung, ſoll es vielleicht der Herr v. Bilinski geweſen ſein, der vorbildlich gewirtſchaftet hat? Weiskirchner, wie Bilinski, wie Stürgkh, wie Haerdtl wirtſchaften; ihre Mißverwaltung belaſtet ihn nicht. Die zahlloſen Schweinereien, die unter ſeiner Re-gierung begangen wurden, die Skandalaffairen, von denen es nur gewimmelt hat, die Protektionswirtſchaft, die während ſeines Regierens aufgekommen iſt, die Unfähigkeit, die überall gewaltet hat, alles das geht nicht auf ſein Konto. In der ganzen Welt verſteht man unter Verwaltung die Summe der Be-ziehungen zwiſchen Staat und Bürgern, verſteht alſo darunter das, was in den verſchiedenen Reſſortminiſterien konzentriert iſt. So hat man es vor Bienerth auch in Oeſter-reich verſtanden, und früher wäre es wohl nicht möglich geweſen, daß man in demſelben Atem die „Ver-

Telegramm-Adreſſe:
Arbeiterzeitung Wien.
Poſtſparkaſſen-Scheckkonto Nr. 19.210.

Inſerate übernehmen auch:
E. Braun, J. Danneberg, M. Dukes,
Haaſenſtein & Vogler, R. Moſſe,
Rafael & Witzek und H. Schalek in
Wien ſowie alle Inſeraten-Bureaux
des In- und Auslandes.

Zentralorgan der öſterreichiſchen Sozialdemokratie.

Erſcheint täglich um 6 Uhr Morgens, Montag um 2 Uhr Nachmittags.

I. Schulerſtraße 13, Telephon 9191,
II. Vereinsgaſſe 20, Telephon 16255,
X. Wielandplatz 5, Teleph. 10468,
XIV. Schweglerſtr. 11/13, Teleph. 4942
XVI. Klausgaſſe 34, Telephon O. 317,
XVII. Lacknergaſſe 22, Teleph. 22421.
Für die an fremde Austräger oder Verſchleißer bezahlten Beträge leiſten wir keine Garantie.
Offene Reklamationen ſind portofrei.

Nr. 178. Wien, Samstag, 1. Juli 1911. XXIII. Jahrgang.

Eine Statiſtenrolle.

Wien, 30. Juni.

In allen Nekrologen dasſelbe und gleiche Lob über Bienerth: er wäre das Muſterbild der ſach-gemäßen Verwaltung geweſen. Und nun regnet es von lobenden Worten: unbefangene, objektive, unparteiiſche Verwaltung; kurz, es muß wohl etwas ſein, was vor Bienerth noch nie da war und wovon befürchtet wird, daß es mit ihm in die Grube fällt. Was mag es nun ſein, was Herrn v. Bienerth dieſes wunderſame Lob be-ſchert hat und was nach ſeinem Scheiden ſo viele Bewunderer findet? Ein Uebermaß des Lobes, wie wir Inſtanzenzug oder in ihrer Tätigkeit der Anſtellung und Vorrückung der Beamten eine beſondere und geſchärfte Rechtlichkeit zuerkennen wollen? Er wird ſich hüten! Wo aber finden wir dieſe Muſterverwaltung, die das Bienerthſche Regieren kennzeichnen ſoll, denn ſonſt? Iſt es vielleicht die Tätigkeit Stürgkhs, die Art, in der das Unterrichtsminiſterium verwaltet wird, die zu dem ſchwärmeriſchen Rühmen verführt? Aber wir ſehen doch dieſelben Leute, die Bienerths Verwaltung ſo loben, über die Verwaltung des Unter-richtes unter Bienerth heftig raiſonnieren! Oder ſoll es die große Stütze der Bienerthſchen Regierung, ſoll es vielleicht der Herr v. Bilinski geweſen ſein, der vorbildlich gewirtſchaftet hat? Weiskirchner, wie Bilinski, wie Stürgkh, wie Haerdtl wirtſchaften; ihre Mißverwaltung belaſtet ihn nicht. Die zahlloſen Schweinereien, die unter ſeiner Re-gierung begangen wurden, die Skandalaffairen, von denen es nur gewimmelt hat, die Protektionswirtſchaft, die während ſeines Regierens aufgekommen iſt, die Unfähigkeit, die überall gewaltet hat, alles das geht nicht auf ſein Konto. In der ganzen Welt verſteht man unter Verwaltung die Summe der Be-ziehungen zwiſchen Staat und Bürgern, verſteht alſo darunter das, was in den verſchiedenen Reſſortminiſterien konzentriert iſt. So hat man es vor Bienerth auch in Oeſter-reich verſtanden, und früher wäre es wohl nicht möglich geweſen, daß man in demſelben Atem die „Ver-

Figure 1: An extract of a newspaper page showing the baselines (top) and respective text lines (bottom).

2.1 Notation

For the rest of this report we define some notations. Since a baseline is described by a polygonal chain, i.e. a list of a finite number of ordered 2-dimensional points ($=$ vertices of the chain), it is to be understood as a vector. Furthermore, to evaluate the quality of our AS system, we have to compare the results of the algorithm ($=$ hypotheses (HY)) with the ground truth (GT) data provided by our project partners. GT means, in our context, the ideal system output generated by humans. Per page we define:

- $\mathbf{g}_k$ is the GT baseline with index k given by human annotators, $k \in \{1, \dots, K\}$, where K is the number of all GT baselines of the page.
- $\mathbf{h}_l$ is the HY baseline with index l detected by the LA system, $l \in \{1, \dots, L\}$, where L is the number of all HY baselines of the page.
- $A_{g,i} = \left\{\mathbf{g}_{i_1}, \dots, \mathbf{g}_{i_{m_i}}\right\}$ is the GT article with index i as a set of GT baselines, $i \in \{1, \dots, M\}$, where M is the number of all GT articles of the page.
- $A_{h,j} = \left\{\mathbf{h}_{j_1}, \dots, \mathbf{h}_{j_{n_j}}\right\}$ is the HY article with index j as a set of HY baselines, $j \in \{1, \dots, N\}$, where N is the number of all HY articles of the page.

Remark Since we ignore the reading order of the baselines, we only consider **sets and not lists** of baselines defining articles.

2.2 R and P matrices

In the following, we want to compare the given M GT articles with the generated N HY articles. To this end we compute two different types of evaluation scores between every GT article and every HY article. So, we get two matrices of dimension $M \times N$.

At this point the baseline detection (BD) measure presented in [1] is used, which is composed of the so called R- and P-value helping us to study the similarity between two sets of baselines.

- The **R-value** $\in [0, 1]$ (see [1], Section 3) indicates, loosely spoken, how well a set of GT baselines is covered by a set of HY baselines. Hence, this score has similar properties like the well-known recall value.
  $\Rightarrow$ Segmentation errors, e.g. a baseline is split into two lines or two lines are merged to one, are not penalized, because we measure how reliable the text is detected (ignoring layout issues).
- The **P-value** $\in [0, 1]$ (see [1], Section 3) indicates, loosely spoken, how well a set of HY baselines is covered by a set of GT baselines. Hence, this score has similar properties like the well-known precision value.
  $\Rightarrow$ Segmentation errors are penalized, because we measure how reliable the structure of the text lines (layout) of the page is detected. So, this score gives us information about the over- and under-segmentation of the lines.

Definition 2 (R-matrix)**.** The R-matrix between the GT articles $A_{g,i}, i \in \{1, \dots, M\}$, and the HY articles

$A_{h,j}, j \in \{1, \dots, N\}$, is defined as

$$\boldsymbol{R}(\{A_{g,1}, \dots, A_{g,M}\}, \{A_{h,1}, \dots, A_{h,N}\}) := \begin{array}{c} \\ A_{g,1} \\ \vdots \\ A_{g,i} \\ \vdots \\ A_{g,M} \end{array} \begin{array}{c} \begin{array}{ccccc} A_{h,1} & \cdots & A_{h,j} & \cdots & A_{h,N} \end{array} \\ \left[\begin{array}{ccc} & & \\ & & \\ & R_{\text{BD}}(A_{g,i}, A_{h,j}, \mathcal{T}_i) & \\ & & \\ & & \end{array} \right] \end{array}$$

in which $R_{\text{BD}}(A_{g,i}, A_{h,j}, \mathcal{T}_i)$ is the R-value defined in [1]. The set $\mathcal{T}_i$ contains tolerance values for each GT baseline included by $A_{g,i}$ affecting that minor deviations in the BD are not penalized.

Definition 3 (P-matrix)**.** The P-matrix between the GT articles $A_{g,i}, i \in \{1, \dots, M\}$, and the HY articles $A_{h,j}, j \in \{1, \dots, N\}$, is defined as

$$\boldsymbol{P}(\{A_{g,1}, \dots, A_{g,M}\}, \{A_{h,1}, \dots, A_{h,N}\}) := \begin{array}{c} \\ A_{g,1} \\ \vdots \\ A_{g,i} \\ \vdots \\ A_{g,M} \end{array} \begin{array}{c} \begin{array}{ccccc} A_{h,1} & \cdots & A_{h,j} & \cdots & A_{h,N} \end{array} \\ \left[\begin{array}{ccc} & & \\ & & \\ & P_{\text{BD}}(A_{g,i}, A_{h,j}, \mathcal{T}_i) & \\ & & \\ & & \end{array} \right] \end{array}$$

in which $P_{\text{BD}}(A_{g,i}, A_{h,j}, \mathcal{T}_i)$ is the P-value defined in [1]. The set $\mathcal{T}_i$ contains tolerance values for each GT baseline included by $A_{g,i}$ affecting that minor deviations in the BD are not penalized.

2.3 R-, P- and F-value for article separation

After the calculation of the R- and P-matrices based on the evaluation scheme for BD, we determine the maximum entries in these matrices in a greedy manner.

Remark 'Greedy manner' means, in this context, that one by one the maximal values of a given matrix are chosen with following deletion of the corresponding rows and columns. Afterwards, the resulting values are summed up (see Algorithm 1).

Algorithm 1 Greedy Function

```
1: procedure GREEDY($\boldsymbol{B}$) with $\boldsymbol{B} \in \mathbb{R}^{M \times N}$
2:     Sum $\leftarrow 0$
3:     $\boldsymbol{B}' \leftarrow \boldsymbol{B}$
4:     while $\boldsymbol{B}'$ is not empty do
5:         $b \leftarrow$ one of the maximal elements of $\boldsymbol{B}'$
6:         Sum $\leftarrow$ Sum $+ b$
7:         $\boldsymbol{B}' \leftarrow$ take $\boldsymbol{B}'$ and delete corresponding row and column of $b$
8:     end while
9:     return Sum
10: end procedure
```

Furthermore, it seems to make sense to create a monotony between the BD and the proposed AS measure that holds

$$\text{proposed AS measure} \leq \text{BD measure} . \tag{1}$$

The equality is taken if and only if the articles have been found perfectly. In an end-to-end scenario, this relation is useful to realize in which step the errors happened (in the BD or in the AS step). One can compute both measures and compare them. In case of equality, it is obvious that the errors only occurred in the BD. In case of inequality, the discrepancy between both measures gives an indicator of how big of an impact the AS has, since its measure is bounded above by the BD measure.

To ensure (1), each row of the R-matrix is weighted by the percentage of the GT article (compared to all GT articles) corresponding to this row (analogous, each column of the P-matrix is weighted by the percentage of the HY article (compared to all HY articles) corresponding to this column).

For example, we consider a page with three given GT articles $A_{g,1}, A_{g,2}, A_{g,3}$ and two detected HY articles $A_{h,1}, A_{h,2}$. Hence, the R-matrix has the dimension 3×2. The set $A_{g,1}$ has 10 baselines and the sets $A_{g,2}$ and $A_{g,3}$ contain together 30 baselines. Therefore, the first row in the R-matrix (this row corresponds to the GT article $A_{g,1}$) is multiplied by $1/4$, since the article $A_{g,1}$ includes $25\,\%$ of all GT baselines assigned to articles.

After the explained multiplication/weighting step, the above described Greedy Function is applied on the resulting matrices. We want to express this process with

$$\text{GREEDY}_{\text{weighted}}(\boldsymbol{\mathcal{R}}) \quad \text{or} \quad \text{GREEDY}_{\text{weighted}}(\boldsymbol{\mathcal{P}}) \ .$$

Definition 4 (R-, P- and F-value for Article Separation)**.** The R- and P-value for the generated HY articles are defined as

$$R_{\text{AS}}\left(\{A_{g,1},\dots,A_{g,M}\},\{A_{h,1},\dots,A_{h,N}\}\right) := \text{GREEDY}_{\text{weighted}}(\boldsymbol{\mathcal{R}}) \ \in [0,1] \ ,$$

$$P_{\text{AS}}\left(\{A_{g,1},\dots,A_{g,M}\},\{A_{h,1},\dots,A_{h,N}\}\right) := \text{GREEDY}_{\text{weighted}}(\boldsymbol{\mathcal{P}}) \ \in [0,1] \ .$$

Thus, we obtain the F-value for AS, i.e. the harmonic mean of the R- and P-value,

$$F_{\text{AS}}\left(\{A_{g,1},\dots,A_{g,M}\},\{A_{h,1},\dots,A_{h,N}\}\right) := \frac{2 \cdot R_{\text{AS}} \cdot P_{\text{AS}}}{R_{\text{AS}} + P_{\text{AS}}} \ \in [0,1] \ .$$

The target value is 1 in all three cases.

We think that these three values give us an appropriate tool to evaluate the result of an algorithm merging a given set of detected baselines to a number of articles. The R_{BD} and P_{BD} values ensure, that HY articles with too many or too few baselines in comparison with the corresponding GT articles are penalized with a lower evaluation score.

3 From articles to news items

During the first year of the NewsEye project there was a serious problem with the AS task, namely how to exactly define an article. This is important for the GT data generation, because everyone has a different understanding of an article. But consistent GT data is required for machine learning approaches, as detailed already within Task T1.3 on 'Data Generation', which we hereby quote from Deliverable D1.5, Section 3.1.1: "One of the advantages of machine learning approaches is that the user - and not the machine - controls the output of the computing process: the results reflect the GT data set and its rules. In other words, although we will go for a comprehensive definition of 'articles' in the NewsEye project it is of course possible that libraries or DH groups use their own definitions. As long as the GT data is consistent it is to be expected that the neural networks will produce models which will achieve good results nonetheless."

Therefore, an alternative concept to define newspaper articles in a consistent way was introduced, namely the 'news item'. This is the result of a working group within the NewsEye project including libraries, computer science and digital humanities researchers. From now on, the concept of a news item will be our internal definition of a newspaper article. **Hence, in the following, 'articles' and 'news items' should be understood interchangeably.** The definition follows the International Press Telecommunications Council (IPTC), which also names its main concept 'news item'[3].

Definition 5 (News item)**.** Based on the considerations of the working group, we want to define the following set of rules.

- News items are 'distinct pieces of content' which can be understood without any context.
  - E.g., a news item is a report about the progress of political negotiations, or about a car accident, or about a story about a crime case at court, but also a job announcement, or a letter to the editor.
  - A paragraph in an article, or a row in the stock exchange table are pieces of the news item, but are not understandable on their own – they need some context which is provided by the 'rest' of the news item to be fully understood.
- A newspaper issue can be seen as a collection of a large number of news items.
- News items are usually separated from each other with some layout or markup, such as titles, highlighted words, or graphical separators.
- Though newspapers are usually arranged in sections, we do currently not take into account any hierarchical order of news items.
- News items are most often written by one author or come from one source.

However, we also want to remark that as already mentioned in the above quotation this rule set is not always unambiguous. This problem occurred particularly during a research stay in August 2019 of a NewsEye project partner of the University of Innsbruck (UIBK-ICH). Digital humanities often work for their research questions on suitable sub-corpora, i.e. they only need some parts of a newspaper for their investigations. A 'distinct piece of content which can be understood without any context' now depends on the specific research question (see also [2], Section 4.2), e.g. if you look for information about a special person you will probably only need one paragraph of a report in which the person is mentioned and not the whole report itself with a lot of other unimportant facts. So, it seems to be very hard to have a gold standard definition of a news item. But Definition 5 gives us much more consistency for GT generation than the broad understanding of an article.

[3]https://www.iptc.org/std/NewsML-G2/latest/QuickStart-NewsML-G2-ItemBasics

4 Data

4.1 Ground truth format

We use the PAGE format in our workflow which is also used within the Transkribus platform[4]. All objects (regions, groups etc.) on, e.g. a newspaper page, are identified with a unique ID within the whole PAGE file. Regions are defined by their type, outline (polygon), and attributes. Especially the baselines are of interest for the AS task. For that, a GT file contains among other things:

- the coordinates of the baselines (= polygonal chain) with the corresponding text
- the reading order of the baselines
- the IDs of the articles the baselines belong to

The article IDs are set with the help of a 'custom tag'. An example for this tag of the PAGE entry for a single text line/baseline is given by

```
<TextLine id="tl_1" custom="readingOrder {index:0;} structure {id:a10; type:article;}">.
```

Of course, there can be baselines having no article ID in the PAGE, i.e. these lines do not belong to an article but to a 'None' class. This special class is not represented with corresponding rows and columns in the R- and P-matrices.

Remark Under the consideration of the 'None' class, the relation (1) should only hold for the BD measure applied on the 'not None' baselines. That makes sense, because inaccurately detected lines with a 'None' tag pull down the BD score but do not affect the AS value. Thus, the AS measure can exceed the BD measure in such a case.

All in all, the PAGE format gives us a valid and very useful tool to store and evaluate the results of an LA, automated text recognition (ATR) and/or AS system.

4.2 NewsEye data sets

Data is needed to train and evaluate the various machine learning modules that will be presented in Section 5. It should contain GT for all three tasks of WP 2, namely for LA, for ATR, and for AS. This includes baselines, the textual content of their respective text lines, text blocks and article tags, but also separators and headings.

In year two we used the so called `ONBv2` data set. This data set, provided by the Austrian National Library (ONB), consists of 232 historical German newspaper pages (partially binarized) ranging from the 19th to 20th century. The newspapers comprise the titles 'Arbeiter Zeitung', 'Illustrierte Kronen Zeitung', 'Innsbrucker Nachrichten' and 'Neue Freie Presse' (for more details see Deliverables D2.5, Section 4.1 and D2.4, Section 2.4). Originally, we were not satisfied with the quality of the text block and separator information. Therefore we also manually created an `ONB_100` data set for text block detection, which was also used for the competition in Section 8, and an `ONB_120` data set for separator detection. In year three we removed 2 pages from the `ONBv2` data set, because their top half is rotated 180 degrees, which should normally not be the case. The resulting data set is called `ONB_230`.

[4] https://transkribus.eu/Transkribus/

Further GT data was provided by the National Library of Finland (NLF) and the National Library of France (BnF). They include 200 pages and 184 pages of newspapers, respectively. The first data set is called `NLF_200` and consists of historical Finnish newspapers from the 19th century with the title 'Uusi Suometar'. From the second data set one page had to be removed since it did not contain any article tags. The set is called `BnF_183` and consists of historical French newspapers from the 19th to 20th century, comprising the titles 'Le Gaulois' and 'Le Matin'. Some example pages can be seen in Figure 2.

The GT generation process for these data sets was not standardized and we want to briefly present the details, which were provided by the partners UIBK-DEA.

- `ONB_230`: The ONB provided ALTO[5] files for this data set with most of the necessary information. These were then manually corrected regarding baseline and text information.
- `NLF_200`: The files provided by the NLF contained most of the necessary information, but it is not clear how these were obtained and they contained various layout errors. The baselines and their text were then redone automatically by our LA and ATR engines to improve their quality.
- `BnF_183`: The files provided by the BnF contained almost none of the necessary information. First of all, text block and separator information was added, which was automatically produced by ABBYY FineReader[6], a popular optical character recognition (OCR) engine. Baseline and text information was then produced by our LA and ATR engines. Finally, text blocks were marked as headings manually.
- Article tags were done manually on all three data sets by native speakers of the respective language.

As outlined above, the quality of the GT data differs between the three data sets. Both `NLF_200` and `BnF_183` contain semi-automatically generated data. But whereas in `NLF_200` only the baselines and their text was improved, `BnF_183` contains mostly automatically generated information. Also, we did not have the capacities to manually correct a specific subset, like in the case for `ONB_230`. Furthermore, the layouts in `NLF_200` and `BnF_183` seem to be strictly more complex than in `ONB_230`. This will show in the final results.

Before working with the data we pre-processed it by removing every text block that did not contain any baselines or article tags. `ONB_230` contained 16498 text blocks, from which 45 had to be removed (0 without baselines, 45 without article tags). `NLF_200` contained 28447 text blocks, from which 764 had to be removed (739 without baselines, 25 without article tags). `BnF_183` contained 73376 text blocks, from which 10814 had to be removed (10153 without baselines, 661 without article tags). The high amount of removals is an indicator about the overall quality of the provided GT text blocks.

During year two, three focused project-wide work groups were set up in order to process some key document collections. For each involved library one data set was chosen to be taken through the whole NewsEye pipeline. They are called *'DE-ArbeiterZeitung-migration'*, *'FI-Uusi Suometar-isms'* and *'FR-L'Œuvre-gender'* for the ONB, NLF and BnF respectively. These are the first bigger data sets that were entirely processed by our new updated AS workflow. For simplicity, we will refer to these data sets as use cases. Table 1 summarizes the data sets used in this deliverable.

[5]ALTO (Analyzed Layout and Text Object) is an open XML scheme developed by the EU-funded project 'METAe' for the description of text and layout information of digitized pages.
[6]https://www.abbyy.com/de/ocr-sdk/

(a)

(b)

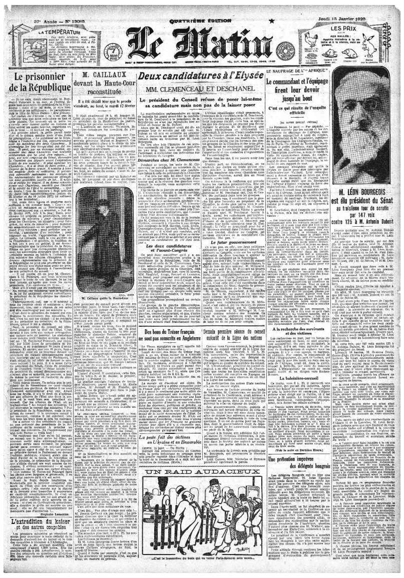

(c)

Figure 2: Example pages for the three GT data sets `ONB_230` (a), `NLF_200` (b) and `BnF_183` (c).

Table 1: NewsEye data sets used in this deliverable.

Data set	Language	# Pages	Type
`ONB_230`	German	230	GT
`NLF_200`	Finnish	200	GT
`BnF_183`	French	183	GT
`use_case_DE`	German	70.418	use case
`use_case_FI`	Finnish	86.308	use case
`use_case_FR`	French	64.724	use case

5 Workflow

In year two we started developing a bottom-up AS workflow that clustered baselines into text blocks, which were in turn merged into news items. We still follow this two-stage strategy, but have now a more refined workflow, using a GNN as our central component. We now want to give a brief overview of the updated workflow and will describe all components in the next section in more detail. How the individual modules interact with each other and for which modules GT data is required can best be seen from the two overviews in Figures 3 and 4.

As a first step we extract the text block structure of a given newspaper page based on geometric information. For this, we apply a baseline detection in conjunction with an additional baseline clustering to form text blocks. The current parametrization of this clustering leads to an over-segmentation regarding the actual news items. We believe this approach is easier to handle than an under-segmentation, since we can keep merging text blocks in the second step, and not worry as much about finding specific splits. Therefore, this clustering approach tends to be seen more as a pre-processing step which is fast and robust, at least as long as we have baselines that are not ranging over multiple columns. Though, this problem can later be addressed by including separator information.

A motivation for this two-stage strategy is that the computation of the reading order of the single lines within the blocks is trivial. Hence, after the first step we are already able to deliver text blocks and the corresponding text in the right order for further NLP tasks in the NewsEye workflow pipeline like named entity recognition (NER) or topic modeling, which are covered in work packages WP 3 to WP 5. However, the reading order between blocks is much more complicated.

To merge the over-segmented text blocks into larger news items, we use a GNN, followed by a final text block clustering. The GNN is the most crucial junction in our workflow, since many other modules feed into it, and because it predicts the actual text block relations regarding the news items. To this end, also semantic content of the extracted text blocks will be used, mainly by means of bidirectional encoder representations from transformers (BERT) (for more details on BERT see [3] and [4]).

One drawback of this AS workflow are certain dependencies between some modules. This means that we have to deal with error-propagation in the system, which will affect the quality of the final AS. For example, the BERT is pre-trained on the text coming from the ATR module. Or, falsely merged baselines to text blocks can not be corrected in subsequent modules.

6 Modules

6.1 Baseline detection

The BD module is already described in Deliverable D2.4. It takes as input an image of the newspaper page and outputs a set of baselines and their corresponding text lines, represented by polygonal chains and polygons respectively. Additionally, if there are baselines that are ranging over multiple columns which in turn are separated by visible dividers (i.e. separators), the baselines get split. This is described in Section 6.4.

6.2 Automatic text recognition

The ATR module is already described in Deliverable D2.5. It takes as input a set of text line images of the newspaper page and outputs their corresponding textual content.

6.3 Text block detection

For text block detection we consider a rule-based approach as well as machine learning approaches. The former was our starting point in year one, which uses a clustering algorithm to form text blocks. In year two we did some first experiments on text block detection with machine learning approaches. In particular, we performed semantic segmentation, i.e. pixel labeling, using an RU-Net, to directly extract text blocks from newspaper images. In year three, we also applied instance segmentation for the text block detection task using Mask R-CNNs. However, the results were not satisfactory enough. As a consequence, we still mainly stick with the clustering approach. But, we will see in the competition results in Section 8 that the right choice and configuration of an instance segmentation model should yield better results. Nevertheless, we use the RU-Net as additional input for the later GNN in the form of abstract visual features.

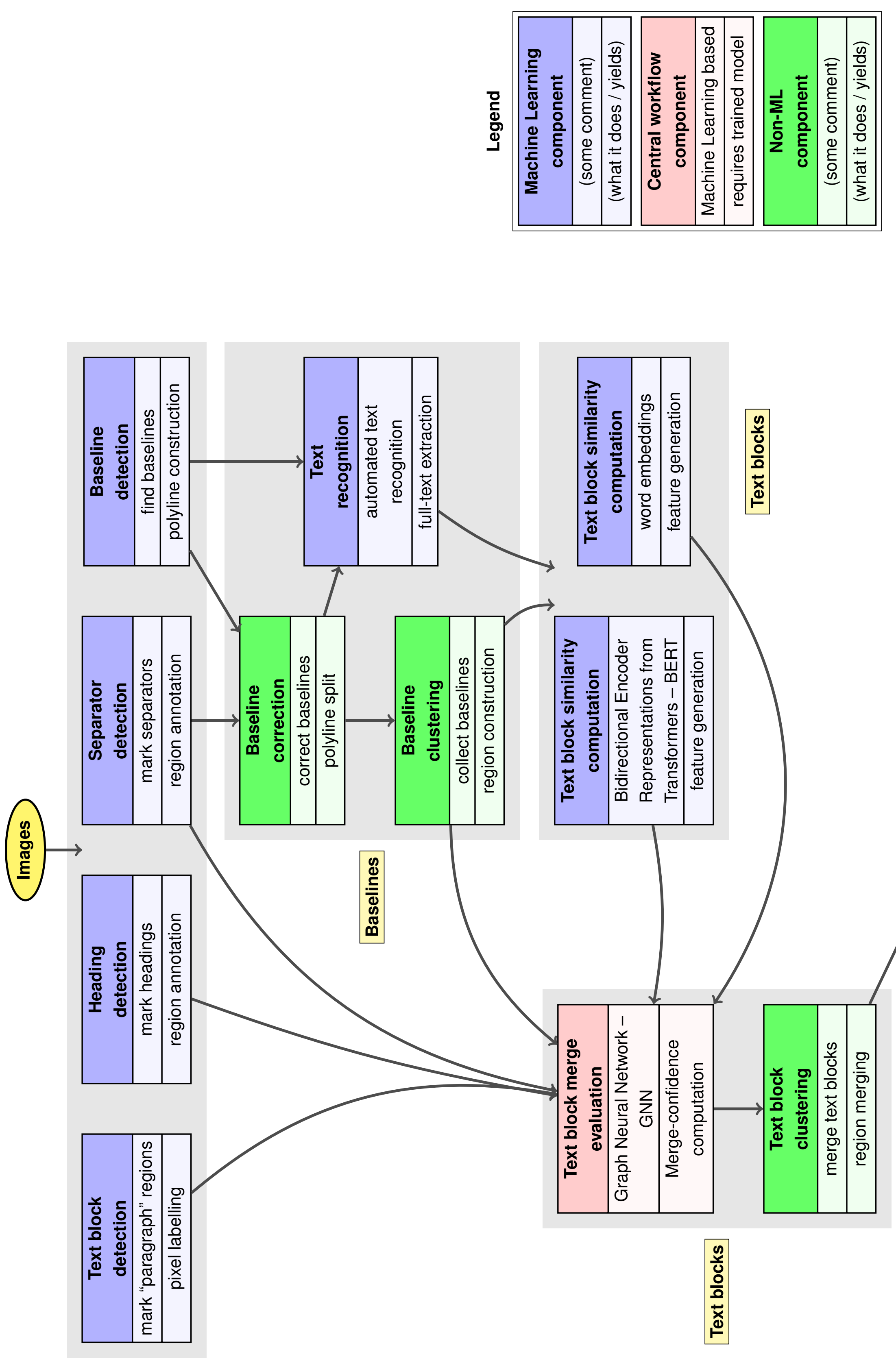


Figure 3: NewsEye's article separation workflow

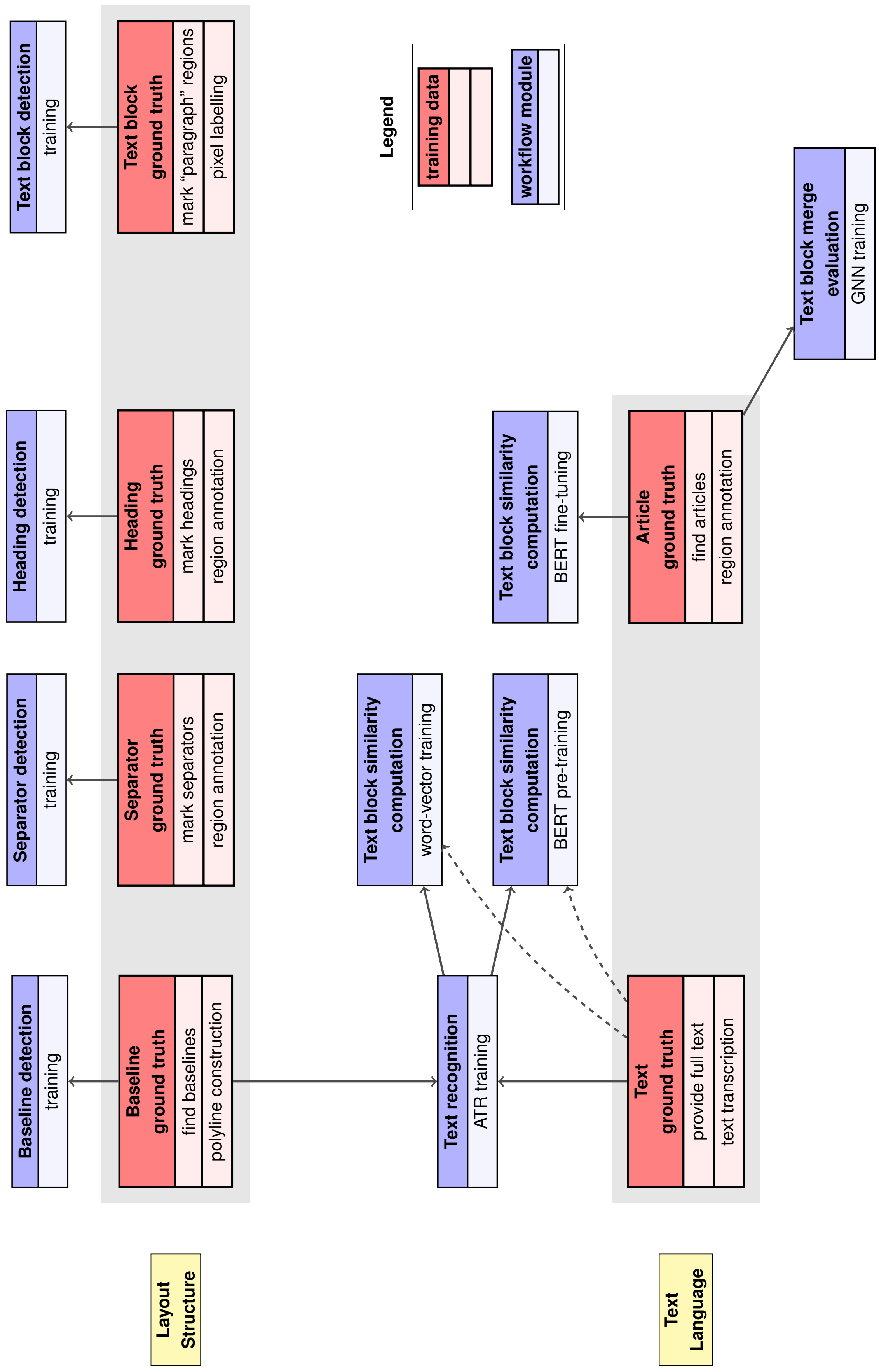


Figure 4: NewsEye's ground truth / training data for the article separation workflow

6.3.1 Baseline clustering

We developed a simple but robust clustering approach based on the DBSCAN algorithm (density-based spatial clustering of applications with noise, see [5]), to extract the text block structure of a given newspaper page based on the geometric information of the previously detected baselines. The main idea of the algorithm is that, given a set of points in some space, it groups together points that are closely packed together, marking outliers (noise) that lie alone in low-density regions. This density-based procedure does not need a hyper parameter determining the number of clusters to be detected. This is very advantageous, because we do not know a priori how many text blocks are present in the given page. Note that instead of working with points like in the traditional setting, we work with the bounding boxes of the respective text lines.

The current parametrization of this clustering leads to an over-segmentation regarding the actual news items, since it is designed to form text blocks. Additionally, we include any found outliers in the clustering process, such that every text line is assigned to a text block. This means that even a single line can also form a text block. We believe this approach is easier to handle than an under-segmentation, since we can merge text blocks later in the workflow, and not worry as much about finding specific splits. Therefore, this clustering approach tends to be seen more as a pre-processing step. This approach can be error-prone to baselines that range over multiple columns, because these will form text blocks over different columns. Though this problem can later be addressed by including separator information.

After the modified DBSCAN clustering is done, we need to outline the baseline clusters to generate the final text blocks. To this end, we apply the alpha-shape algorithm (see [6]) on every baseline cluster. The advantage of alpha-shapes, in contrast to e.g. the convex hull, is that we can enclose the underlying text more closely, since these shapes are usually not convex. Exemplary results are shown in Figure 6 (a). For a comparison to the text block detection output of the OCR engine ABBYY see Deliverable D2.4, Section 6.

6.3.2 Semantic segmentation using RU-Net

The second approach that was introduced in Deliverable D2.4 is based on the ARU-Net (see also [7]), the same network that we also use for other tasks such as baseline, separator and heading detection. In the following, we will summarize the key elements from Deliverable D2.4 on the ARU-Net.

ARU-Net At its core, the ARU-Net is an extension of the U-Net (see [8]) which is achieved by adding two key concepts, spatial attention (A) and depth (residual structures (R)). Both modules are optional, so that you can add both (ARU), only one (AU and RU) or neither of them (U). A more detailed description is given in the following.

The U-Net is based on fully convolutional networks (FCNs) that use convolutional neural networks (CNNs) to incorporate the spatial relationships of an image. The main purpose of FCNs is to perform semantic segmentation (pixel labelling) on an image (see [9]), i.e. predicting a class for each image pixel. For the text block detection task we have the classes *text_block* and *other*. The FCN combines local features to produce more meaningful high level features using pooling layers. These pooling layers reduce the spatial dimension of the input. Thus, the result suffers from a coarse resolution. The U-Net[7]

[7]The ‘U’ in the name reflects the U-shaped form of the architecture, see Figure 5 (a).

is derived by applying a deconvolutional network on the subsampled output of the FCN, such that the output has the same spatial dimension as the input. Additionally, shortcuts between layers of the same spatial dimension are introduced, which allows for an easier combination of local low-level features and global higher-level features (see [8]). Finally, as stated above, the ARU-Net extends the U-Net attention (A) and depth (R). The attention mechanism makes it possible to handle various font sizes, especially mixed font sizes on a single page. The residual structure together with the skip connections helps very deep neural networks with error propagation, i.e. makes them still trainable and yield state-of-the-art results. In Figure 5 the basic architecture of the U-Net and the combination with the attention network is depicted. The residual connections are defined for each depicted CNN block whereas the skip connections are depicted as the 'Idendity' arrows. For more details see Deliverable D2.4.

However, instead of extracting text blocks from this neural network we use its different feature maps in the network architecture directly as an input to the GNN to provide abstract visual features tailored to the text block detection task. The feature maps are depicted as rectangles in Figure 5 (a), where we only use the output of each two-layer CNN block as input to the GNN. We did experiments on the `ONB_100` data set comparing the RU-Net with the ARU-Net where the training images were randomly resized to an image height of 500 to 1500 pixels while keeping the original aspect ratio. The best performing model was an RU-Net with a pixel accuracy of $97.03\,\%$, a precision of $98.06\,\%$ and a recall of $96.80\,\%$. Example outputs for the text block detection task can be seen in Figure 6 (b).

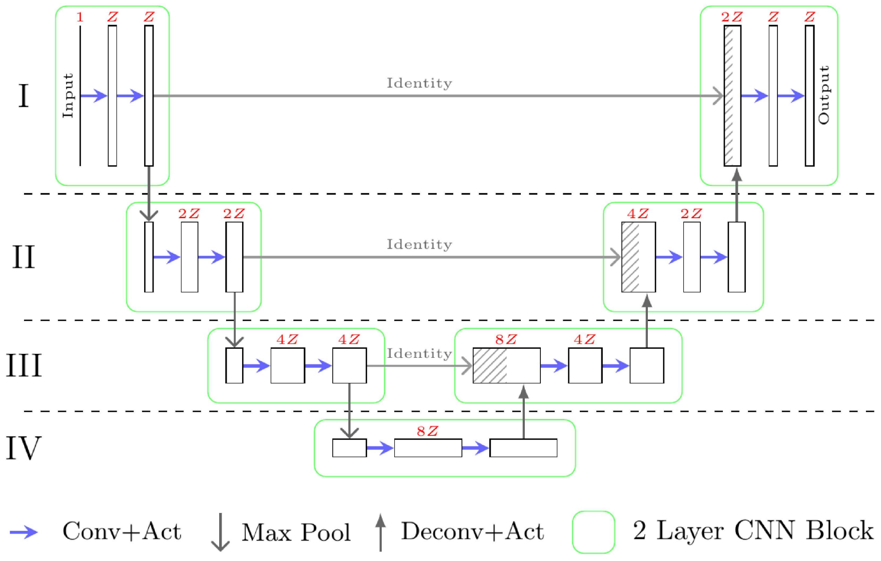
6.3.3 Instance segmentation using Mask R-CNN

Instance segmentation is a combination of semantic segmentation and object detection. In semantic segmentation we give each pixel a label, as is done with the ARU-Nets. Object detection, on the other hand, is used for detecting specific objects in an image by localizing and classifying them. A combination of both approaches provides a class label for each pixel in the image with the addition of treating multiple objects of the same class as individual instances. Thus, in the case of text block detection, we are not only interested in identifying where text blocks are, but we also want to distinguish between them. This would overcome the problem of applying a subsequent post-processing to remove uncertainties on the borders of neighboring text blocks as it is the case for semantic segmentation approaches, see Deliverable 2.4, Section 3.1.

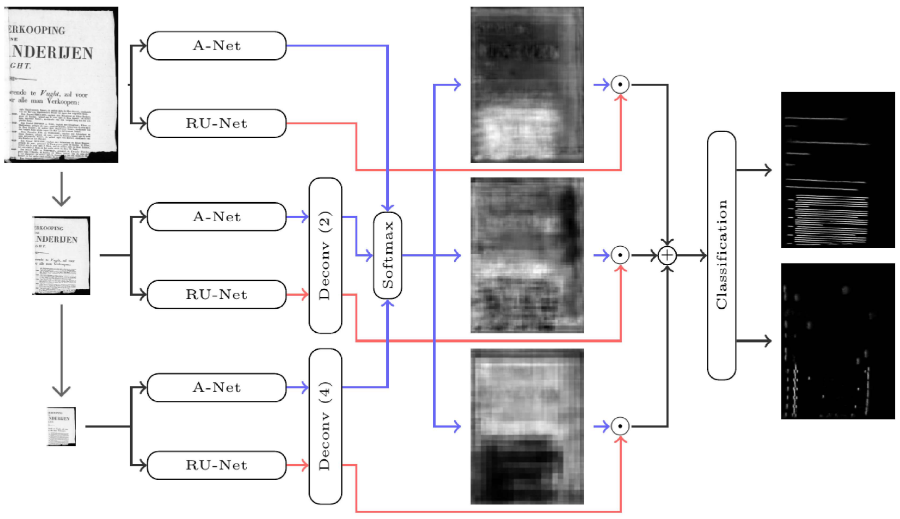
Mask R-CNN (see [10]) is a specific model performing instance segmentation and is based on the R-CNN model and its extensions (see [11, 12, 13]), which are solely used for object detection. In addition to detecting the objects in an image with bounding boxes, Mask R-CNNs perform pixel labeling inside these bounding boxes.

For the experiments we used a TensorFlow implementation[8] using the default configurations. First results show that for simple pages the approach works fine, sometimes missing a few lines as a result of too small bounding boxes. For more complex pages, however, there are cases where no text blocks are found at all. Examples are given in Figure 6 (c).

[8] https://github.com/matterport/Mask_RCNN



(a) U-Net architecture.



(b) RU-Net and A-Net applied to different scales of the image.

Figure 5: The U-Net architecture with residual connections in the CNN blocks and the combination with an attention network to perform the pixel labeling task as done in [7].

6.4 Separator detection

In newspapers, one can often observe that adjacent text blocks are visually split by separators, horizontal and vertical ones. Most often, vertical separators are used for dividing the newspaper page into columns, whereas horizontal separators are used to start a new topic and thus possibly a new article. Hence, in order to help the process of identifying text blocks belonging to the same article this is a valuable feature to use.

Different approaches exist for detecting separators on a document page, using machine learning (see e.g. [14]) or traditional LA methods (see e.g. [15]). How well these approaches work depends heavily on the quality and complexity of the newspaper pages. On good quality pages that have a relatively simple layout, the latter usually work better, while for more general cases machine learning approaches

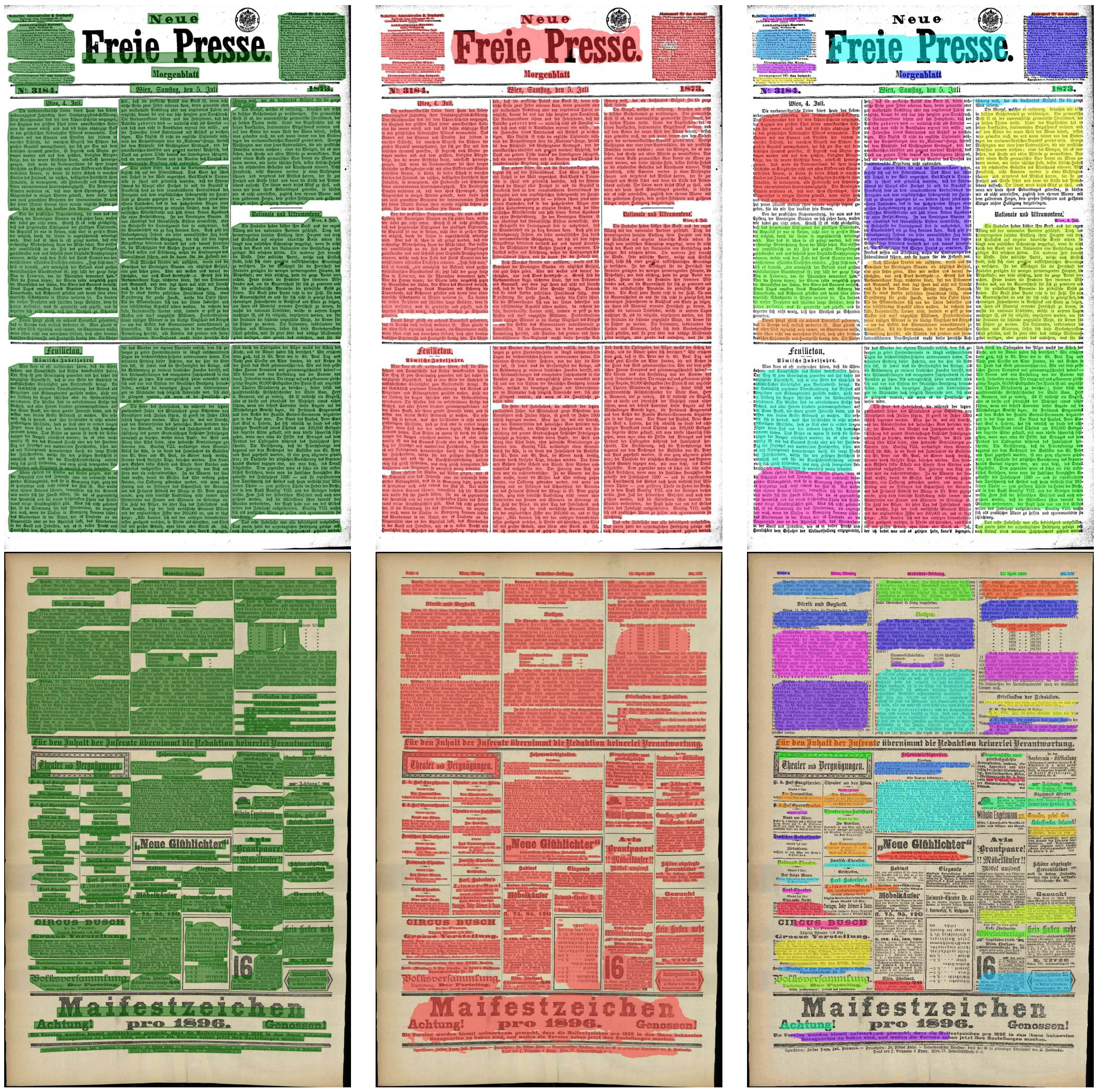


Figure 6: Text block detection results on a simple (top) and a complex page (bottom) with baseline clustering (a), semantic segmentation using an RU-Net (b) and instance segmentation using a Mask R-CNN (c).

are the better choice. Our method is using machine learning and an additional post-processing applying common algorithms like connected component (CC) analysis and morphological operations. The network and the post-processing steps are described in the following.

6.4.1 RU-Net for separator detection

For separator detection we use the same neural network as for baseline detection (ARU-Net), i.e. we perform a pixel labeling task on the input image with the classes *separator* and *other*. A more detailed description of the network architecture was given in Deliverable D2.4 and can also be found in [7]. For training the network we used 100 of the pages from the `ONB_120` GT data set and the remaining 20

for testing. Note that no distinction was made between horizontal and vertical separators in the GT. The input images were randomly resized to an image height of 880 to 2200 pixels while keeping the original aspect ratio. We also compared the versions of the network with (ARU-Net) and without (RU-Net) attention and finally used the best model in terms of pixel accuracy, precision and recall. Overall, the RU-Net performed slightly better with an accuracy of $94.05\,\%$, a precision of $90.20\,\%$ and a recall of $88.11\,\%$.

6.4.2 Post-Processing

Additionally, post-processing is applied to the net output to remove noise, distinguish between horizontal and vertical separators and to convert the binary pixel map to polygons that can be stored in the PAGE format. In a first step, a threshold of 0.05 is applied to the net output to convert the confidences to binary values. In other words, confidences that are greater than 0.05 are mapped to 1 and confidences that are lower or equal to 0.05 are mapped to 0. We initially set a low threshold to avoid discarding too much information. Later, in a post-processing step, the noise is removed (see below).

Afterwards, we apply CC analysis to extract a first possible set of separators. The CC algorithm is used to segment and identify parts in binary images, where two pixels are connected if they have the same value and are neighbors. To define the neighborhood of a pixel we differentiate between the four-connectivity and the eight-connectivity. The four-connectivity defines the top, right, bottom and left pixel as neighbors whereas the eight-connectivity also adds the four diagonal pixels. Brief experiments yielded the eight-connectivity as a better fitting approach, since too much information was discarded with the four-connectivity. Connected components that are smaller than 100 pixels are interpreted as noise and are removed. Note, that for inference we input images that have a fixed height of 1500 pixels. Experiments on the `ONB_120` data set showed, that on average 11 CCs are found per image before the noise removal. After the post-processing 3 to 4 CCs (34.55% of all CCs) are removed.

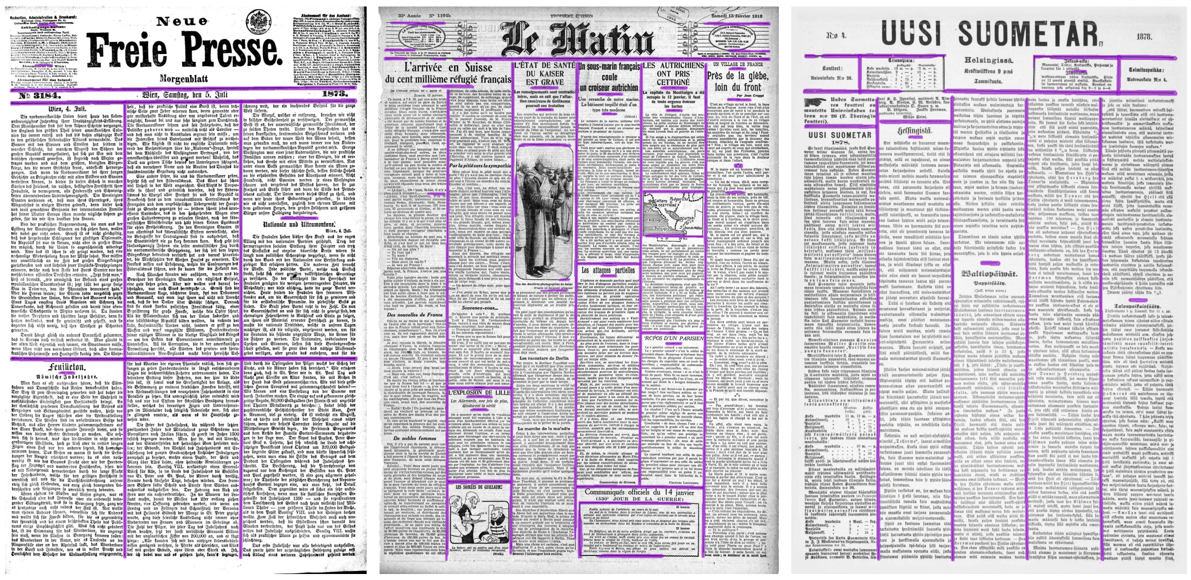
In a first version we made no distinction between vertical and horizontal separators. This led to complicated polygons combining multiple vertical and horizontal separators in one and seemed to confuse the GNN later on, also see Sections 7.5. To fix this, the morphological *opening* operation is applied to the CCs with different structuring elements for vertical and horizontal separators. Originally, the opening is an operation to remove noise from an image and is a successive execution of the erosion and dilation operations (possibly multiple times). The structuring element for detecting vertical separators has a shape of $H_S \times 1$ where $H_S = 0.02H$ and H is the image height, only extracting vertical lines that have a minimum height of H_S. Analogously, the structuring element for detecting horizontal separators has a shape of $1 \times W_S$ where $W_S = 0.015W$ and W is the image width only extracting horizontal lines that have a minimum width of W_S. Again, the parameters H_S and W_S were determined by brief experiments on a small data set. Note that also experiments on deskewed images had been done, but did not bring major improvements.

Finally, with Python modules like Shapely[9] and Rasterio[10] the CCs are converted to polygons and can be saved in the PAGE format. In general, there is no option to distinguish between vertical and horizontal separators in the PAGE format, there is just one `SeparatorRegion` type. Therefore, we added a custom tag to the region like in the following example where `horizontal` and `vertical` describe the orientation.

[9] https://github.com/Toblerity/Shapely
[10] https://github.com/mapbox/rasterio

An example output of the algorithm can be seen in Figure 7.

```
<SeparatorRegion id="SeparatorRegion_1" custom="structure {orientation:horizontal;}">
```



(a) (b) (c)

Figure 7: Separator detection results on a German (a), French (b) and Finnish newspaper page (c).

6.4.3 Baseline correction

As mentioned in Section 6.1, we also use separators to split baselines/text lines that range across multiple columns. Since columns are delimited from each other by vertical separators, only these are used for splitting. If the text line already has text assigned to it, one should also make sure that the text is split as well. We apply the following two approaches.

- If the words have coordinates, assign every word to the left/right of the separator to the left/right split.
- If there are no word coordinates, split the baseline and copy the text to both splits.

6.5 Heading detection

Another important logical structure type in newspapers are headings. They give us information about where an article may start. With headings we also include the big title that can be found on the first page of most newspapers. For heading detection there are also machine learning based approaches for labeling pixels in an image (see e.g. [14]), rule-based approaches, e.g. taking into account the height of a text line (see e.g. [16]) and approaches combining both (see e.g. [17]). Our method combines the initial BD and text block segmentation with a neural network and a modified Stroke Width Transform (SWT) to classify the headings in an image.

The input of the algorithm are the text lines we get from the baseline detection module (see Section 6.1). For each text line we want to determine if it is a heading or not. To do so, we combine an ARU-Net with a distance transformation of the image. The former performs pixel labeling on the input image. The latter produces features that can be used to get information about the stroke width and height of the text in a text line. We describe both sub-modules in the following.

6.5.1 ARU-Net for heading detection

As with separator detection in Section 6.4, we use an ARU-Net with the classes *headings* and *other*. We used the GT from the three data sets introduced in Section 4.2 (`ONB_230`, `NLF_200` and `BnF_183`). For testing we used 60 pages including issues from each data set, while using the remaining pages for training the network. The input images were randomly scaled to an image height ranging from 1200 to 2400 pixels, where we use a fixed image height of 900 pixels in the inference phase. Again, we compared the network with and without attention and stick with the ARU-Net in the end.

Remark Note, that 900 is not lying in the range the network is trained on, but nevertheless yields good results compared to other bigger image heights, also achieving a good trade-off between performance and run-time of the algorithm. One reason why the smaller image height also works well could be due to the attention that is used in the model, where the input images are further down-scaled a few times.

Then, for each text line we take its bounding box and crop the corresponding sub-image from the network output. Finally, we calculate the average value over all pixel confidences of that sub-image which tells us how likely it is that the given text line is a heading.

6.5.2 Stroke Width Transform for heading detection

The SWT (see [18]) is a local image operator that determines for each pixel in an image the width of the most likely stroke containing the pixel. Thus, the output is an image of the same size as the original input. In our implementation we adapt the algorithm such that we rather apply a distance transform on a binary image which calculates the distance to the closest zero pixel for each pixel of the source image. Here, zero pixel refers to the background (white), since we first invert the image. By considering the maximum values in this distance transform, we get similar results to the much more compute intensive SWT. An example output can be seen in Figure 8. Given a newspaper page, for each text line we then do the following steps.

- Get its bounding box and crop the corresponding excerpt from the distance transformation image.
- Create CCs based on the distance transformation excerpt via connected components analysis.
- Remove all CCs that are too small (minimum width and height of 3 pixel), too big (maximum width and height of 500 pixel) or with too extreme aspect ratios (up to $8 : 1$ or $1 : 8$). The respective parameters were determined by some brief experiments on a smaller data set.
- Based on the remaining CCs compute the text line height by taking the maximum value of all CC heights and compute the stroke width by taking the median of the maximum values of each CC's stroke width.

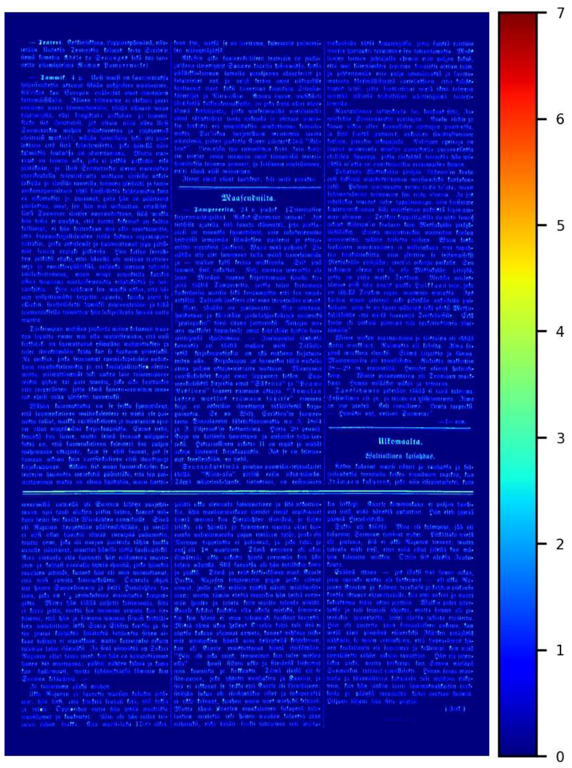


Figure 8: Example output of the distance transformation where the pixel values refer to the distance to the closest zero pixel in the source image.

6.5.3 Combination of both approaches

The final decision whether a text line is a heading or not is determined by a linear combination of the approaches from the last two subsections. This is

- the network output probability for each text line,
- the deviation to the most common stroke width across a page (mapped to the interval $[0,1]$) and
- the deviation to the most common text height across a page (mapped to the interval $[0,1]$).

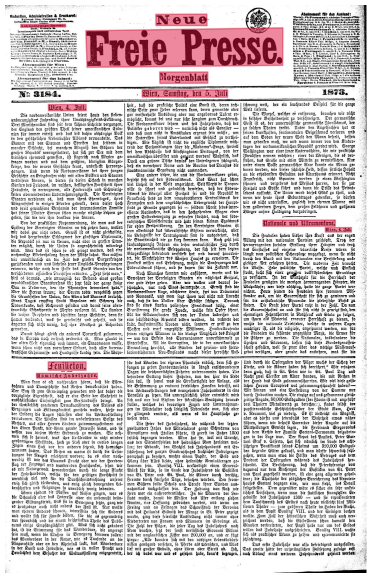
Experiments on the GT data, calculating the P-, R- and F_1 scores, yielded a linear combination with a weight of 0.8 for the network output probability and 0.1 for the stroke width weight as well as the text height weight. This shows us that the machine learning approach gives us much more valuable information than the SWT features. However, if the deviation to the most common stroke width or text height across a page is very large (which is reflected by values close to 1 after scaling to the interval $[0,1]$), we then classify the text line as a heading regardless of the value of the network output. Conversely, if the network probability is very high, we rely only on this information. In general, a text line is considered a heading if its corresponding final heading confidence value succeeds a threshold of 0.4. Overall we reach an F_1 score of $66.53\,\%$ with a precision of $82.14\,\%$ and a recall of $55.91\,\%$ where we are more interested in a high precision value since false positives could result in wrong merges of news items later on. An example output of the algorithm can be seen in Figure 9.

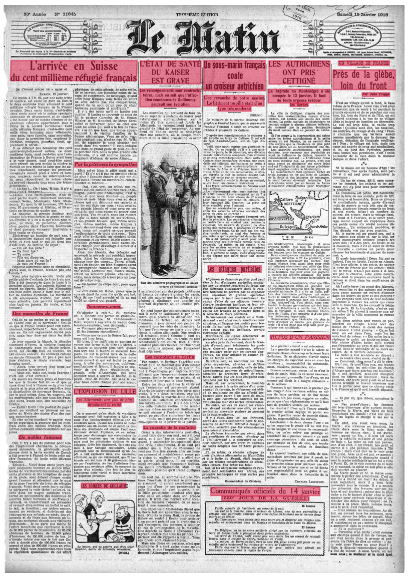
Remark In the PAGE format headings are defined at text region level. In cases where a heading is part of a bigger text region mostly containing non-heading text this would lead to problems. If a heading is part of such a paragraph text region we either can only say, that the whole paragraph is a heading or we would loose the heading information when declaring the region as non-heading. Either case would lead to problems in the GNN later on.

To overcome this issue we also store the information on text line level using the custom tag, ending up with a more fine-grained option. If at least $80\,\%$ of the text lines in a text region are headings we define the whole text region as a heading. Two examples for defining the heading on text region level and on text line level in PAGE are given in the following.

```
<TextRegion id="tr_1" custom="readingOrder {index:0;}" type="heading">
```

```
<TextLine id="tl_1" custom="readingOrder {index:0;} structure={semantic_type:heading;}">
```


(a)


(b)

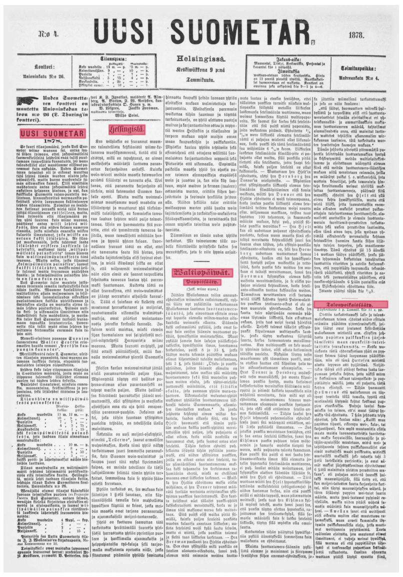

(c)

Figure 9: Heading detection results on a German (a), French (b) and Finnish newspaper page (c). One can see that some headings, e.g. the big titles, are not found, which is due to the low recall value. However, in these examples we have no false positive examples, which reflects the higher precision value.

6.6 Content based features for text block relations

During the NewsEye project work it became more and more clear that in order to properly get adequate, consistent news items from merging text blocks, one has to comprise also the similarity of their content. This amounts to calculating NLP based features, which mathematically are weighted relations between text blocks. We will think of the weights as estimates for a certain *text block similarity*.

6.6.1 Word vector based text block similarities

The first approach is to employ usual word embeddings. This has been investigated and implemented in a cooperation between the partners UROS, UIBK-ICH and ULR. We used the rather straightforward standard idea: to every text block, assign an embedding as the (arithmetic) mean of all (non stop-word)

word embeddings, and calculate the cosine of two block vectors for assigning a text block similarity to every pair of text blocks. For the underlying word embeddings, freely available sources (see e.g. NLTK Project[11]) were used. Note that these embeddings are language specific.

These features turned out to be moderately helpful. In practice, they seem not to provide better results, when the structure-based features are combined with the BERT features (Section 6.6.2). On the other hand, such features can be calculated easy and fast. Word vector based text block similarity features shall remain in the toolbox for first and fast starting results on new and less extensively prepared data sets. As soon as time and resources allow for preparing and using the much more advanced BERT technologies, those should be preferred.

6.6.2 BERT based text block similarities

The second approach is to use a BERT model to generate such features. These are context-based language models that are first pre-trained on unlabeled data and then fine-tuned to a specific task on labeled data.

As shown in Figure 10, the basic BERT model consists of N similar encoder layers connected in series, each consisting of a multi-head attention layer, followed by a feed-forward layer. Each time a feed-forward layer or multi-head attention layer is applied, the residual input is added back to the actual output and a layer normalization is applied to the result. In order for the BERT model to be able to process text, the text must first be converted into a sequence of tokens by means of a tokenizer, so that this can then be converted into a sequence of vectors by means of a trainable embedding. Since the encoder layers consist only of attention layers and feed-forward layers, the BERT must be given additional information about the position of each element in the input sequence. This is done via a so-called positional encoding, by adding a fixed position vector to each embedding vector of the input sequence. For a more detailed description of the BERT architecture see [3] and [4].

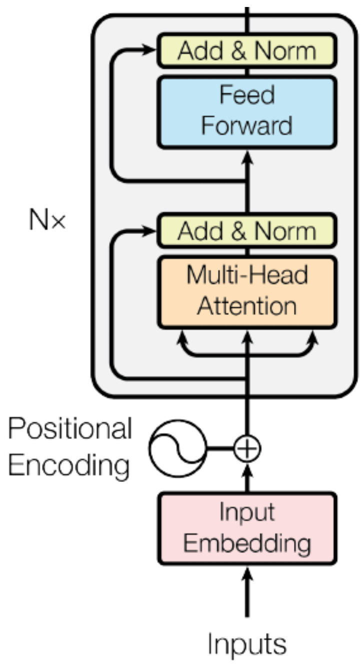


Figure 10: General architecture of a BERT model.

For the feature generation we first did a pre-training of the BERT analogously to [3] on the Mask-Language-Model (MLM) task and the Next-Sentence-Prediction (NSP) task. The MLM task first tokenizes the text input with a subword tokenizer, then $15\,\%$ of the tokens are masked randomly. Of these masked tokens, $80\,\%$ are replaced by a special masked token, $10\,\%$ are replaced by a randomly chosen other token, and the remaining $10\,\%$ keep the original correct token. The goal of the MLM task is to correct the masked tokens to their original state, which is only possible by ‘understanding’ the language and in turn learning of a robust language model. Claiming that language models understand language is somewhat controversial (see e.g. [19]), but that is out of the scope of this report. The goal of NSP is to learn if two parts of text belong to the same document or came from different documents. For each of the two pre-training tasks an additional classification layer is added on

[11] https://www.nltk.org/index.html

top of the architecture. In this way both tasks can be trained with a combined loss.

For fine-tuning, the goal is to determine whether two text blocks belong to the same news item. Normally, only the pre-trained layers as shown in Figure 10 are reused for fine-tuning. But because the NSP task from pre-training is already pretty similar to the question of whether two text blocks belong to the same news item, we reuse the classification layer of the NSP task here. For training, GT is needed, where we have example pages that contain information about which text blocks belong to the same news item. A particular input sample consists of 2 text blocks that are separated by a separator token and entered into the BERT. If the two text blocks are too long to process for the BERT, i.e. the maximum permissible sequence length is exceeded, a corresponding number of tokens are cut out of the text blocks in the middle so that the maximum possible sequence length can be maintained again. After the fine-tuning is done, the BERT model can be used to predict text block similarities on unseen data.

6.7 Graph neural network

In the last year we introduced the general concept of GNNs. These are artificial neural networks that can be applied to graph data, which is able to represent a set of objects with complex relationships and interdependency between them (for a survey on GNNs see [20]). Here we want to focus on how we integrate them into our AS workflow. The goal is to solve a relation prediction task. Given a set of text blocks on a newspaper page, the GNN should predict which of them belong to the same news item. This is realized as an edge classification scenario. The GNN computes confidence scores for every possible pair of text blocks, which are interpreted as the probability for two particular text blocks to belong to the same news item. The text blocks are provided by the text block detection module (Section 6.3). This is a highly unbalanced classification scenario, because a text block belonging to a particular news item has negative relations to every text block in different news items.

Since GNNs work on graph data, we need to create a graph representation of a newspaper page, where relevant objects are represented by nodes and their relations by edges between them. Nodes and edges can then be enriched with various features to increase the representational strength of the data. For the graph construction on a particular newspaper page, we first match the text blocks to nodes. Secondly, a Delaunay triangulation (see [21]) between the nodes is used to build the edge set of the graph. In particular, the center points of each text block are matched to the nearest vertex of an underlying regular 50×50 pixel grid, to create a more homogeneous layout for the triangulation. For n nodes, this (Delaunay) graph contains $O(n)$ edges, which makes it computationally more feasible, in comparison to e.g. a fully-connected graph which contains $\binom{n}{2}$ edges. Any missing long-range dependencies should not be a problem, since feature information flows over multiple time steps in a GNN.

The next step is to assign features to the nodes and edges. We include both handcrafted features as well as machine-learning based features. Many of the previously introduced modules are used for this.

Nodes get assigned geometric features that represent the position and size of their respective text block. Additionally, we use a distance transformation over the newspaper image to approximate the stroke width and the height of the text for each text block and assign those features to each node. Furthermore, the output of the heading detection module (Section 6.5) is used to assign a feature, indicating whether a node contains a heading or not. Finally, an RU-Net trained on text block detection (Section 6.3.2) is used to extract abstract visual features over the bounding box area of each text block.

Edges get assigned separator information and features containing semantic context. The output of the separator detection module (Section 6.4) is used to assign two features, indicating whether a pair of connected nodes is separated by a horizontal and/or vertical separator. This distinction is useful, since a horizontal separator is a way stronger indicator for two text blocks to not belong to the same news item. This can often times be seen, when a single news item stretches over multiple newspaper columns, which are still split by vertical separators. This scenario is illustrated in Figure 11. Additionally we use a BERT (Section 6.6.2) to integrate semantic context between two text blocks. In particular, it predicts a probability that those text blocks belong to the same news item, solely based on their textual embeddings. This confidence score is then assigned as the final edge feature. This is even done for the training process of the GNN. There, we use the trained BERT to generate its predictions on the GT data, since the actual targets used for the BERT fine-tuning are the article relations between the text blocks. These are the same targets the GNN is trying to predict. If we would use those as edge features while training, it would be possible for the GNN to just focus on this one particular edge feature and disregard everything else. By using the BERT predictions as training features, we introduce enough uncertainty into the feature, so that the GNN can not solely rely on it.

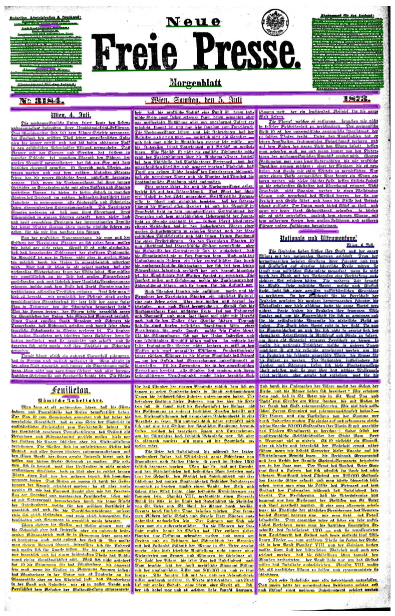
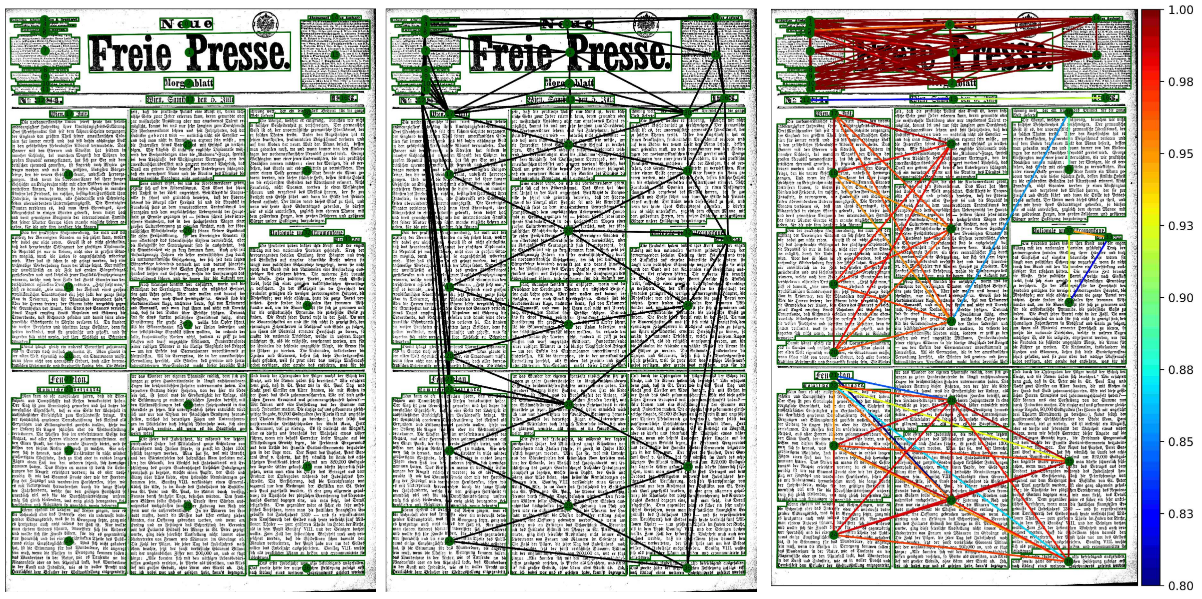


Figure 11: Impact of vertical and horizontal separators (marked as purple regions). A distinction is useful, since text blocks belonging to the same news item (uniformly colored) can still be split by vertical separators.

At this point a GNN is applied for the feature extraction. First, nodes aggregate features from their neighboring nodes and edges, employing an attention mechanism. Second, they update their own state with a Long Short-Term Memory. These steps are repeated for multiple time steps to allow information flow over longer paths in the graph. Afterwards, a binary edge classifier uses the network features to output a confidence that two specific nodes belong together. When this classifier is applied on every possible node pair, the result is a fully-connected confidence graph in which nodes represent text blocks and the edge weights represent the probability that two nodes belong to the same news item. This confidence graph is the input to the final module in our AS workflow. The process of the GNN module is depicted in Figure 12.

As mentioned in Section 4.2, the low amount of proper GT data is a problem. To artificially increase the number of training samples for the GNN, we use data augmentation on the generated features. Though, this is limited to geometric feature augmentation of the text blocks, since generating new sophisticated newspaper pages in the style, language and layout of the respective data sets would be an extremely challenging task in its own. The geometric augmentation modules include random scaling (horizontally and vertically independent), random rotation (coherent across all nodes) and random translation (combination of coherent part across all nodes and incoherent part for each individual node). For each training sample, these modules are applied independently with a probability of $50\,\%$. This does not generate new proper newspaper pages, since the textual content, the general layout as well as the information about headings and separators stay the same, but it at least introduces some more variations in the training process.

As mentioned in Section 5, the GNN depends entirely on the outputs of the previous modules in the AS workflow, and in turn is prone to an error propagation. Especially the text block clustering is a crucial step, since once a text block is formed, it will not be split further down the workflow. These text blocks represent the basic objects that the GNN is working on and that get clustered to news items afterwards (Section 6.8). Originally, there was an idea to build more of an end-to-end network, which operates directly on baselines. This would prevent an error propagation from the text block detection module. Though, some preliminary experiments showed that this approach is currently computationally not feasible, since many of the relevant newspaper pages include thousands of baselines. Therefore, this approach was not further pursued.



(a) (b) (c)

Figure 12: GNN module process. For the graph construction phase, nodes are matched to text blocks (a) and edges are built as a Delaunay triangulation over the nodes (b). The GNN output is a fully-connected confidence graph (c). Note that only high confidences are visualized.

6.8 Text block clustering

Since the GNN only outputs a confidence graph, some sort of post-processing is needed as a final step in the AS workflow to form distinct news items. The idea is to cluster the text blocks, based on the information contained in the confidence graph. We use three different approaches to tackle this clustering problem.

Our first approach is a **modified DBSCAN algorithm**. Another version of the DBSCAN algorithm was already used in Section 6.3.1 to cluster baselines to text blocks. Here, instead of working with a distance metric, we define the neighborhood of a node as the set of nodes whose edges satisfy a certain confidence threshold in the graph. During the algorithm, we only allow new nodes to get picked up by an existing cluster if their average confidence over all other nodes in that cluster meets a second threshold. This change makes the algorithm more robust against false positives in the confidence graph, which would otherwise result in multiple clusters getting merged by a single edge with high confidence.

Our second approach follows a **hierarchical clustering** analysis. In particular, an agglomerative clustering method is chosen, i.e. a bottom-up approach, where each text block starts in its own cluster and the closest pairs of clusters are merged continuously, until one big cluster is created. The end result is a strict hierarchy, since the structure is only coarsened during the algorithm. To decide which clusters should be combined, a cluster distance needs to be defined. We chose the distance of centroids as the criterion to specify the similarity of clusters. Once the hierarchy is formed, a distance threshold needs to be chosen, which determines at which point in the hierarchy the clusters are extracted. Brief experiments on a smaller data set suggested that a threshold T_{HC} based on the mean (d_{mean}) and median (d_{median}) distances over the merged clusters works best, specifically $T_{\text{HC}} = (d_{\text{mean}} + d_{\text{median}})/2$.

Our third approach is a straightforward **greedy clustering** algorithm. This is best justified by interpreting the GNN edge (confidence) weights as the probability that the two incident (text block) vertices belong to the same (news item) cluster. Then, assuming their independence implies, that given the GNN outputs, the probability of a certain clustering amounts to the product of all edge weights over intra-cluster edges times the product of all (1-edge weight)s over inter-cluster edges. This respectively its logarithm is a natural objective function to be maximized over all possible clusterings.

This is a typical problem of discrete optimization, where the number of all possibilities is far too large for searching them all. Hence, we might follow a traditional greedy strategy. For this, we start with the finest partition, i.e. every single vertex forms a single cluster. Then we join clusters step-by-step, such that in every step one merge is chosen which gives the maximal increase of the above objective function. The procedure stops as soon as no merge leads to an increase anymore.

Compared with the exponential overall number of clusterings, this strategy is of polynomial complexity in the number of (text block) vertices. Thus, it is realistic for practical computations, although, as it is known, taking the local (step-wise) optimum does not need to lead to a global optimum.

The text block clusters formed by any of the three mentioned algorithms represent the final news items (articles). Figure 13 shows the resulting clustering on the previous example, using the greedy clustering approach. The results for the other two approaches are not shown, since they are almost identical on this particular example. In fact, the clustering algorithms are mostly interchangeable, which is supported by the results shown later on in Table 6.

7 Experiments and results

In this section we want to present the major experiments and results obtained during year three. This mainly includes quantitative results from the BERT, the GNN and the final AS output. Minor experiments for hyperparameter tuning on the various modules are excluded.

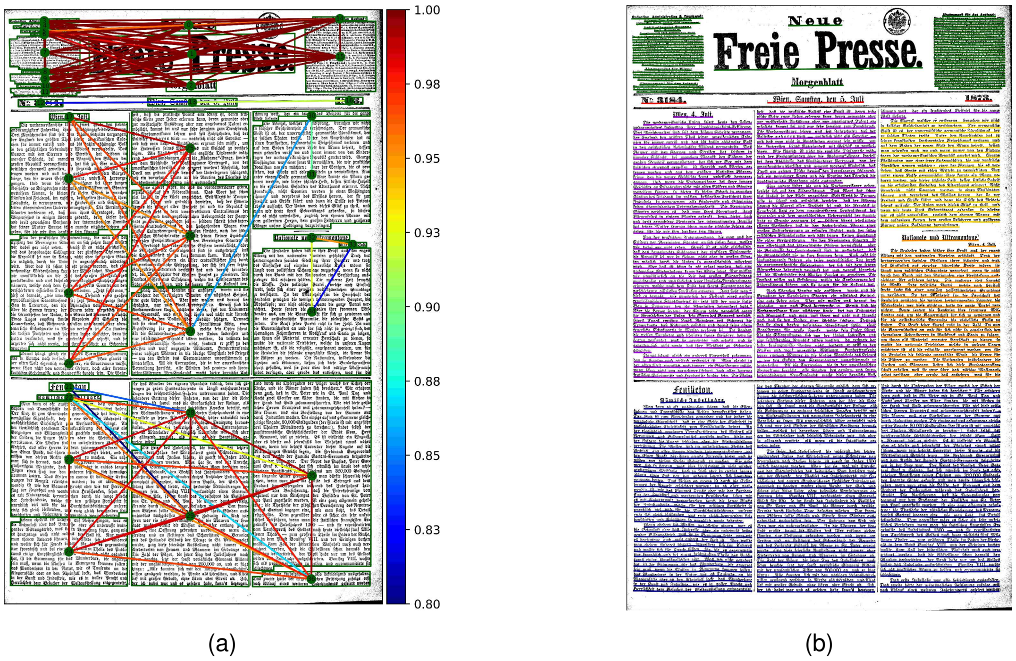


Figure 13: Text block clustering to news items. The greedy clustering algorithm uses the graph confidences (a) to form text block clusters (b). Note that only high confidences are shown in (a). Text block clusters are uniformly colored.

7.1 Semantic context with BERT

As mentioned in Section 6.6.2, the final goal for the BERT (fine-tuning) is to determine whether two text blocks belong to the same news item. For pre-training we used the raw text predictions from the ATR module (Section 6.2) on the three use-case data sets described in Section 4.2. For fine-tuning additional AS information is necessary. Here we use the three GT data sets described in Section 4.2. From each data set $10\,\%$ of the pages were randomly chosen for testing.

Experiments showed that BERT models trained on a single language performed the best. The underlying architecture for all of these models is the same, but we used a separately trained instance for each of the three data sets. With these mono-lingual BERTs we got the results depicted in Table 2 for the fine-tuning. It can be seen that the quality for the `ONB_230` and `BnF_183` data sets is relatively similar. It is not entirely clear why the BERT performs worse on the `NLF_200` data set. This could be due to the quality of the ATR, which impacts the pre-training performance, and in turn the fine-tuning results as well by means of the embeddings. In the Finnish language the words tend to be longer, i.e. consist of more characters on average. Maybe the BERT has harder time with longer words, because they decompose into more tokens. More extensive experiments are necessary to explain this discrepancy.

Table 2: Fine-tuning results of the BERT for the AS task.

Data set	Accuracy [%]	F_1 [%]
ONB_230	94.16	76.06
NLF_200	95.81	54.65
BnF_183	93.03	77.83

7.2 GNN training with pseudo ground truth

Originally GT data from the ONB_230 data set contains information about the basic layout (baselines and text blocks), the contained text, separators and distinctive headings. These (and other) features will be used by the GNN for the training process. Later on, when working on actual use cases, these features need to be generated by other modules in the AS workflow, in order to be available for the GNN prediction. This means that there is a certain discrepancy between the data the GNN is learning from and the data on which the GNN makes predictions. It is easy to imagine, that especially the text blocks, the basic building blocks for the graph construction, will differ between the GT data and the results from the text block detection module. Partly missing separators or headings are also likely, since no module works perfectly accurate. This creates an additional difficulty for the GNN, where it not only has to try to solve the relation prediction task on data similar to the GT data, but also generalize and adapt to (erroneous) data coming from previous modules.

One idea is to create a kind of pseudo GT data set which incorporates results from these modules, so the GNN can learn and adapt to the kind of features it will later have to rely on. Adding pixel labeling based separators (Section 6.4) to the GT text lines can generate new baselines via splits. In this case the previous text is split onto the two new baselines and both get assigned the original article tag. Text blocks are formed by using the baseline clustering algorithm (Section 6.3.1) and their article tag is assigned by a majority vote of their respective baselines in the GT data, since it is possible for a text block to contain text lines of different news items after the clustering process. Headings are assigned as described in Section 6.5. Both data sets are divided into the same (randomly chosen) training, validation and test sets with 190, 20 and 20 pages respectively. The two data sets can be summarized by Table 3.

Table 3: Two training sets for the GNN. ONB_230_GT is the classical handmade GT. ONB_230_PS is pseudo GT which uses the GT text lines, but then incorporates features about separators, text blocks and headings from other modules (PL = Pixel labeling, BC = Baseline clustering).

Data set	Baselines & Text	Separators	Text blocks	Headings	Articles
ONB_230_GT	GT	GT	GT	GT	GT
ONB_230_PS	GT	PL	BC	PL	Vote

Models were trained and evaluated on both data sets. Since we are dealing with a classification task, a threshold T needs to be set, that determines at what confidence level a prediction is considered positive (two text blocks belong to the same news item) or negative. A standard threshold would be 0.5, i.e. we consider the most probable class (in a conventional sense). The higher the threshold, the less positive relations are predicted. Note that this is only necessary to compute the various measures for the GNN output, in order to compare the different results. In the context of our AS workflow, we do not make a binary decision, but instead work with the confidence graph itself in the text block clustering module. In the upcoming results, we chose the threshold T which maximizes the corresponding F_1 score. Looking at Table 4, the experiments showed the superiority of the GNN model trained on the ONB_230_GT data set for the validation and test sets. From this point on we only conducted experiments using the GT data, instead of incorporating more pseudo GT.

7.3 GNN generalization capabilities

In year two we worked internally solely with the ONBv2 data set. Now we have additional data sets available for training, namely NLF_200 and BnF_183. These data sets contain newspapers from different

Table 4: Evaluation of GNNs trained on different training sets. The model name corresponds to the data set it was trained on. Results are chosen based on best F_1 score (at threshold T, in %).

Model	Data set	ONB_230_GT				ONB_230_PS			
		T	P	R	F_1	T	P	R	F_1
GNN_onb_gt	train	0.64	92.8	95.4	94.1	0.30	75.3	79.5	77.3
	val	0.91	88.0	83.2	85.5	0.95	83.3	75.9	79.4
	test	0.78	89.0	84.2	86.5	0.64	84.3	84.1	84.2
GNN_onb_ps	train	0.70	83.8	83.0	83.4	0.66	89.8	90.3	90.1
	val	0.88	77.8	82.2	79.9	0.94	79.2	78.6	78.9
	test	0.86	81.6	79.6	80.6	0.80	75.1	83.0	78.9

time periods and with different languages and layouts. Since scarcity of AS GT data is one of our major problems, it makes sense to try to train joint GNN models on multiple of those data sets. Ideally the GNNs will be able to generalize across the different data sets. We are interested in seeing whether these combination of models can perform on par with models trained for specific newspapers or even benefit from the additional data variations.

To this end, we created a cross-evaluation setup, where we trained GNNs on every possible combination of the three GT data sets and evaluated them on the relevant training, validation and test sets. These subsets were randomly chosen. For ONB_230 the split is identical to the previous section (190, 20, 20 pages). The NLF_200 data set was split into 160, 20, 20 pages and the BnF_183 data set was split into 149, 17, 17 pages for training, validation and testing respectively. Note, that the GNNs that are trained on multiple data sets still obtain their semantic edge features from multiple mono-lingual BERTs. This is due to the fact that experiments showed worse overall results on multi-lingual BERTs compared to mono-lingual ones. Furthermore, a single heading module was used for the experiments, which was jointly trained on all GT data sets. Of the remaining modules in the workflow, baseline detection, text recognition, separator detection and text block detection (for visual features) were only trained on the ONB_230 data set, since it contained the GT of best quality.

The results are depicted in Table 5. Models specialized on a particular training set generalize decently well to other data sets, but do not outperform the other respective specialized models. E.g., GNN_o performs well, but not better than GNN_n, on NLF_200. The overall performance on BnF_183 is worse compared to ONB_230 and NLF_200. This can mostly be explained with the quality of the underlying GT data. First of all, the BnF_183 GT was partly semi-automatically generated and secondly, it contains way more text blocks than the other two data sets. Furthermore, it can be seen that the models perform considerably worse on the BnF_183 validation set compared to its test set. This is mainly due to the fact that the validation set contains more complicated newspaper pages. Since the subsets were randomly generated, the average total number of text blocks in the validation set was found to be higher (407 vs 309). In particular, it contains two pages about the stock market with a lot of tables and over 1000 text blocks each. These have a big impact on the overall results, since the subsets only contain 17 pages each. Generally, all specialized models slightly overfit on the training data, which indicates that additional GT data (of sufficient quality) would be useful.

Regarding multi-models (i.e. models trained on more than one data set), the results show that the GNNs can profit from the additional training data. The best test results (marked in bold in Table 5) for both NLF_200 and BnF_183 were obtained with multi-models. Though, this is a very minor benefit compared to the specialized models and it is still in the range of possible statistical errors, since these are the results from a single training run (due to time constraints). It is also noticeable, that the multi-models trained on two data sets usually perform better on the third excluded data set than their specialized counter parts. E.g., GNN_ob performs better on `NLF_200` than GNN_o and GNN_b. This is another indication, that additional training data helps generalization purposes. This can also be seen in the final model (trained on all three training sets), since it yields good results across the board and would probably be the best candidate to handle new unseen newspapers.

Note that the high variance in optimal thresholds T should not be of concern, since looking at more internal data shows, that for more average thresholds, i.e. near 0.5, the F_1 scores only drop by a small amount.

Table 5: Cross-evaluation of GNNs trained on different training sets. The model name corresponds to the data set(s) it was trained on (o = `ONB_230`, n = `NLF_200`, b = `BnF_183`). Specialized models are highlighted in grey. Results are chosen based on best F_1 score (at threshold T, in $\%$). Best performance for each test set is marked bold.

Model	Set	`ONB_230`				`NLF_200`				`BnF_183`			
		T	P	R	F_1	T	P	R	F_1	T	P	R	F_1
GNN_o	train	0.62	94.2	96.2	95.2	0.87	74.7	70.6	72.6	0.66	60.6	72.9	66.2
	val	0.89	87.7	84.5	86.1	0.91	77.7	70.1	73.7	0.78	42.3	59.0	49.3
	test	0.76	86.3	86.4	**86.4**	0.88	68.0	71.6	69.7	0.74	61.1	58.4	59.7
GNN_n	train	0.05	47.0	47.4	47.2	0.94	89.7	90.2	90.0	0.22	60.9	59.9	60.4
	val	0.91	70.3	69.2	69.8	0.91	82.8	90.8	86.6	0.30	51.2	40.9	45.5
	test	0.87	74.5	74.6	74.6	0.91	89.3	83.6	86.4	0.46	70.5	71.7	71.1
GNN_b	train	0.05	47.1	63.8	54.2	0.94	67.1	68.3	67.7	0.82	90.6	91.3	90.9
	val	0.97	70.1	69.7	69.9	0.91	58.5	62.6	60.5	0.82	51.5	84.5	64.0
	test	0.90	59.6	64.1	61.8	0.91	63.4	67.8	65.6	0.85	74.9	71.8	73.3
GNN_on	train	0.53	88.9	90.9	89.9	0.91	87.9	86.4	87.1	0.50	63.6	64.3	64.0
	val	0.92	86.6	84.0	85.3	0.88	84.1	89.0	86.5	0.50	45.2	51.5	48.1
	test	0.86	85.7	86.1	85.9	0.87	87.7	85.4	86.5	0.62	75.5	69.2	72.2
GNN_ob	train	0.14	70.7	83.2	76.5	0.93	76.7	73.0	74.8	0.71	83.7	86.6	85.1
	val	0.91	85.9	79.8	82.7	0.93	79.6	79.4	79.5	0.83	56.0	69.7	62.1
	test	0.71	85.6	84.9	85.3	0.92	74.8	71.4	73.1	0.75	71.6	70.4	71.0
GNN_nb	train	0.05	44.9	65.8	53.4	0.93	87.6	87.5	87.6	0.68	82.6	87.0	84.7
	val	0.96	76.0	69.1	72.4	0.90	81.8	86.1	83.9	0.86	54.8	80.7	65.2
	test	0.95	77.7	71.2	74.4	0.91	88.3	85.1	**86.6**	0.75	74.1	71.3	72.7
GNN_obn	train	0.22	67.3	73.9	70.4	0.92	85.1	84.3	84.7	0.63	78.5	83.5	80.9
	val	0.90	83.9	81.9	82.9	0.90	81.4	87.0	84.1	0.75	54.8	78.9	64.7
	test	0.84	85.3	82.9	84.1	0.88	81.3	83.1	82.1	0.75	76.4	70.9	**73.6**

7.4 News item clustering

Finally, the output of the GNNs needs to be clustered in order to form the actual news items. We compare all three clustering algorithms (see Section 6.8) on each GT data set, by evaluating our introduced AS measure, using the specialized GNN models from the previous section. We chose the specialized models to have more comparable results and because the differences to the best performing models was only marginal. For the hierarchical clustering, the threshold described in Section 6.8 was used.

For the modified DBSCAN clustering, we chose high confidence and clustering thresholds (see Section 6.8) of 0.8 each, which are based on the findings in Table 5. The results in Table 6 suggest that there is no clear favorite amongst the different clustering algorithms. Though, more internal data shows that the choice of the hyperparameter thresholds can have a significant impact on the resulting clusters. In contrast, the greedy approach is simple and does not rely on any hyperparameters. It also has the best test results on ONB_230 and NLF_200. All clusterings perform better on the training sets, which is explained by the slight overfitting of the GNNs. The overall performance on the different data sets fits our assessment of the quality of the underlying GT data, i.e. we see a drop in performance from ONB_230 over NLF_200 to BnF_183. One can conclude that the remaining errors are due to inaccurate confidence predictions from the GNNs. Nevertheless, compared to our original rule-based baseline clustering method, we can record relative error improvements (i.e. comparing $1 - F_1$ scores) of $73.7\,\%$, $49.6\,\%$ and $48.6\,\%$ for the test sets of ONB_230, NLF_200 and BnF_183 respectively.

Table 6: News item clustering performance (F_1 score from AS-measure in $\%$) on all three GT data sets for different clustering algorithms. The relative error improvements refer to $(1-F_1)$ scores. Best performance for each overall data set and respective test set is marked bold.

Algorithm	ONB_230				NLF_200				BnF_183			
	train	val	test	all	train	val	test	all	train	val	test	all
Greedy	85.2	85.5	**85.3**	85.2	80.0	77.7	**75.7**	79.3	69.9	60.0	65.2	68.5
DBSCAN	88.2	80.0	83.0	**87.1**	83.6	78.6	75.4	82.3	75.5	66.5	**69.7**	**74.1**
Hierarchical	85.5	84.5	84.4	85.4	86.7	77.9	**75.7**	**84.8**	73.1	65.3	69.0	72.0
Baseline clustering [old]	54.8	49.9	44.2	53.4	51.7	50.0	51.8	51.6	38.4	36.7	41.1	38.5
Relative error improvement (in $\%$)	73.9	71.1	73.7	72.3	72.5	57.2	49.6	68.6	60.2	41.1	48.6	57.9

7.5 Internal user satisfaction

After having computed the first 1000 pages from the use_case_DE data set entirely in UROS' AS workflow, we started a short round of collecting qualified user feedback. For this, we arbitrarily picked 10 pages which have never been used for any training and/or GT production. For those pages, all 3 different clustering methods were applied.

These results were presented to DH specialists from NewsEye partners of the University of Innsbruck in DH (UIBK-ICH) and the data specialists of UIBK-DEA. We received valuable feedback on the overall quality and specific errors along with a ranking between the 3 clustering methods.

The main results are:

- There is a wide quality range from fairly well separated pages to pages with far too many failures, which obviously should not appear.
- Essentially, errors are already included in the GNN confidence output. Mainly, they are clearly visible and due to not considering horizontal separators well enough.
- The 3 clustering methods do not lead to essentially different results. If any at all, the greedy method seems to have slight advantages.

From this, we draw the following conclusions:

- Much more weight should be given to the horizontal separators. Since this requires to re-train the GNN models and re-compute all subsequent workflow steps, we plan a second round of processing the use-case data sets at the very end of the project.
- This might even lead as far as forbidding to join text blocks across such horizontal separators. This idea remains to be tested for real application scenarios.
- As for the 3 different clustering methods, we restrict all further processing within this project work to use the greedy method only.

For further user feedback loops on the AS quality, we apply the per-page-evaluation scheme proposed by partner UIBK-DEA [22]:

1. true number of articles
2. number of correctly separated articles
3. number of correct article beginnings
4. number of wrong splits
5. number of wrong merges

From this, various global measures (relative to 1.) could be derived according to specific needs. Here, it seems to be important to include also correct article start lines as (e.g.) ‘half’ correct, and to distinguish between wrong splits (less serious) against (really serious) wrong merges.

8 Competition on text block segmentation

We organized a competition at the ICPR2020[12] named ‘ICPR2020 Competition on Text Block Segmentation on a NewsEye Dataset’. The conference was held online from January 10th to 15th, with the competition and its results presented on January 14th in a one hour slot on the underline platform[13]. To recap, the participants had the task to cluster baselines into text blocks in two different tracks, one simple and one complex track each having 40 pages training data and 10 pages test data. Overall, of a total of five registrations, three participants took part with two machine learning based approaches applying instance segmentation and one rule-based approach. The participants were:

- Cinnamon AI and Ho Chi Minh City University of Technology (HCMUT),
- École des Hautes Études commerciales Montreal (HEC) and
- Lenovo Research and South China University of Technology (SCUT),

with HCMUT using the rule-based method and the other two teams using the machine learning algorithms.

For the simple track, everyone submitted their results, while for the complex track, only the machine learning approaches participated. The results in Table 7 are copied from the original competition paper where we provided a baseline method performing a simple clustering on the baselines. The F-values are averaged over the 10 pages of the corresponding test sets. A more detailed description of the competition tasks and the methods of the teams can be found on the competition website[14] and in the competition paper [23].

[12] https://www.micc.unifi.it/icpr2020/
[13] https://underline.io/
[14] https://www.mathematik.uni-rostock.de/forschung/projekte/citlab/projects/text-block-segmentation-competition-icpr2020/

Table 7: Results of the participants (including our baseline method) in terms of the in Section 2 presented F-value on the test sets. The winning teams are shown in bold face.

	F-value simple track	F-value complex track
UROS baseline	0.934	0.768
Cinnamon AI & HCMUT	$\mathbf{0.999}$	-
HEC	0.995	0.887
Lenovo Research & SCUT	0.997	$\mathbf{0.954}$

9 Prolongation Update

We formulated two goals for the prolongation phase of the NewsEye project at the end of M36 for further sustaining and disseminating the results presented to this point. This includes the end-to-end processing of data sets suitable for testing and the consolidation of the implementation source code for further and public software integration.

Regarding the first point, we processed four big data sets covering two new languages: English and Swedish. More details can be found in Section 9.1. Addressing the software integration in the scope of the collaboration of NewsEye with READ-COOP[15], we have started incorporating the software modules into the Transkribus platform. The integration in the backend is complete so that the AS workflow can be executed in Transkribus. However, an application in the frontend does not yet make sense for various reasons. To name an example, an even more intensive exchange is needed between UROS and READ-COOP's Transkribus team together with partner UIBK-DEA to clarify questions about e.g. which hyperparameters can be set by user in the end. In particular, this implies that further steps are beyond the scope of this project. New experiments were also made in collaboration between project partners. The details can be found in the following subsections.

9.1 New data sets

In addition to the three big key use cases from Section 4.2 covering newspapers in German, Finnish and French, we processed three Swedish document collections including the titles '*Åbo Underrättelser*', '*Hufvudstadsbladet*' and '*Västra Finland*' covering $252,732$ pages overall and a bilingual (English and French) newspaper called the 'New York Herald' covering $84,765$ pages. An overview is given in Table 8 and example pages can be seen in Figure 15.

Table 8: Data sets processed in the NewsEye prolongation covering four newspaper issues.

Data set	Newspaper	Language	# Pages	Type
`use_case_EN`	New York Herald	English & French	$84,765$	use case
`use_case_SV`			$252,732$	
$\hookrightarrow$	Åbo Underrättelser	Swedish	$103,216$	use case
$\hookrightarrow$	Hufvudstadsbladet	Swedish	$137,280$	use case
$\hookrightarrow$	Västra Finland	Swedish	$12,236$	use case

Since no GT was available for these data sets, we applied the various models trained on the other three languages, omitting the BERT model from Section 6.6.2. This also means that we can only measure

[15] https://readcoop.eu

the quality of the results heuristically based on the visual outputs. Overall, most of the existing models seem to generalize well to the new languages and layouts from the new data sets. However, the AS model was not always able to cluster all text blocks to the corresponding article. Some example results can be seen in Figure 16.

We also observed in some cases that prior binarization of the input images gave better results of the visual models than applying them directly to the corresponding grayscale images (cf. Figure 14). We assume that this may be due to some binarized images occurring in the GT data the visual models were trained on.

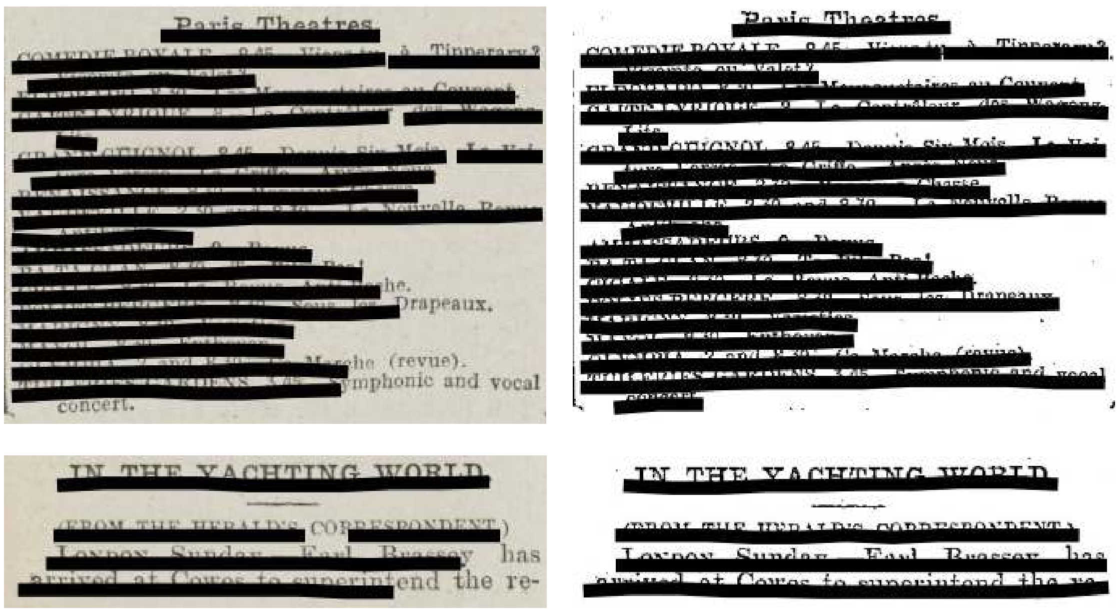


Figure 14: Comparison of the baseline detection model applied to the original image (left column) and to a binarized version of it (right column).

(a) New York Herald

(b) Åbo Underrättelser

(c) Hufvudstadsbladet

(d) Västra Finland

Figure 15: Example pages for the four newspapers covered in the NewsEye prolongation.

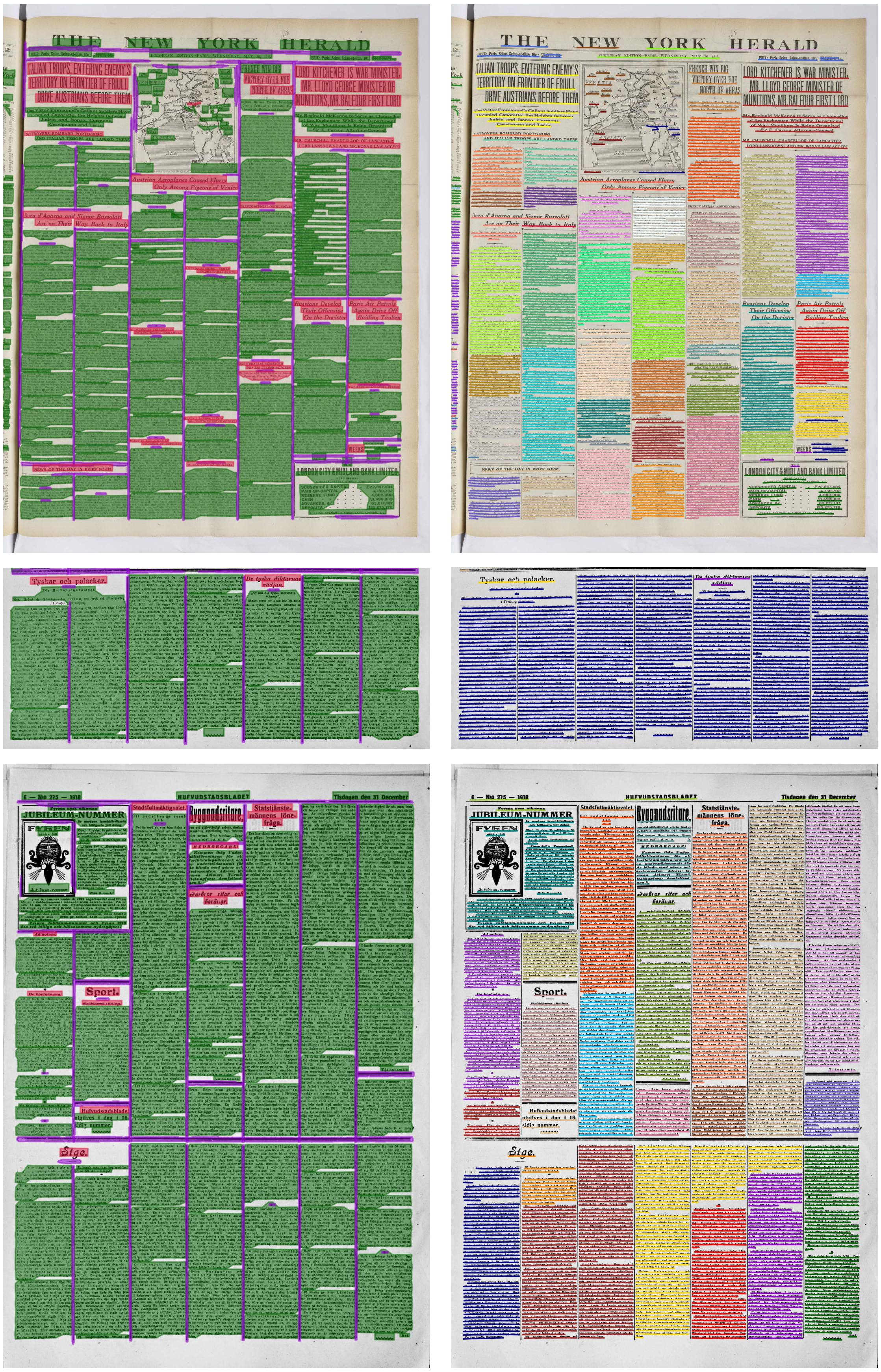


Figure 16: Example outputs for the separator detection, heading detection and text block detection models (left column) and the subsequent AS output of the GNN (right column).

9.2 Comparative article separation measure

In further developing the quality assessment approach described in Section 7.5, we implemented a measure, which compares an AS to its GT by counting correctly marked articles along with positions where an article was erroneously split or where two articles where erroneously merged.

Mathematically, the collection of text blocks into articles can be seen as a (combinatorial) *partition* over the set of all text blocks of a page, where every article then consists of the text blocks contained in one subset, called *blocks* of the partition. Both the GT and the hypothesis correspond to one partition each, and on this background, the AS quality can be measured by the distance between those two partitions in the so-called *partition lattice*, i.e. the ordered set of all partitions.

Altogether, we describe the quality of an AS on a per-page level by the following four figures compared to the GT separation: numbers of corrects, wrong splits, wrong merges, and distance (in partition lattice). In order to further compare such quadruples, it appeared to be reasonable to order them by distance first and then by the number of corrects second. This ordering can finally be used to compare the quality of different ASs against the given GT for a page. For two ASs, the AS with the higher ranked quadruple (according to the aforementioned ordering) receives one point, where the points are added up over all pages in the data set. In this case, the highest possible score is determined by the number of pages $|D|$ in the data set D (one AS scored higher on every single page than the other one). For more than two ASs (e.g. k), the comparisons are made for each possible pairing and all respective scores for each AS are added together. In this case, the highest possible score would be $(k-1)|D|$ (one AS scored higher on every single page than all the $k-1$ remaining ones). Note that because we count draws, i.e. same distances and same number of corrects, as scores for both contenders, that the sum of the scores can surpass the number of pages in a particular data set.

Given this new user-inspired measure, it was natural to repeat the evaluation of the clustering algorithms of year three. Their original results with respect to the initial AS measure can be found in Table 6 page 37. Any chosen hyperparameters were kept the same. Since the new measure is based on a per-page comparison of the news items for all involved algorithms, we conducted a series of comparisons, where in each step the algorithm with the worst score is removed. In this way, you get a more direct comparison of the best performing algorithms, whose scores are not influenced by worse results. In this case, only two steps are needed for each data set, since we only compare three different clustering algorithms (so only one needs to be removed until the top two get compared). Note that the scores are computed for the entire data set and not only the test set. The results are depicted in Table 9.

Comparing the results to Table 6 (columns 'all'), we observe a similar behavior for `ONB_230` and `NLF_200` in both measures. In fact, the order of the algorithms is identical, with the best being DBSCAN for `ONB_230` and Hierarchical for `NLF_200`. To get a more general comparison, we can sum up the scores of the algorithms across all three data sets: Greedy (1244) and DBSCAN (1228) are relatively equal, whereas the hierarchical clustering (1439) performs best. This suggests that the latter approach seems to be the most reliable choice for generally unknown use cases, that do not really fit any of the three GT data sets.

9.3 ULR text block similarity feature

In collaboration with ULR, UROS incorporated more semantic context in the GNNs, besides the BERT features. Similar to our approach in Section 6.6.1, the idea was to generate text block similarity scores

Table 9: News item clustering performance (comparative measure) on all three GT data sets for the original three clustering algorithms. The columns in each data set show a progression (left to right), where the worst algorithm is removed from the comparison. Draws count as a score for all involved algorithms.

Algorithm	ONB_230		NLF_200		BnF_183	
	All	Top-2	All	Top-2	All	Top-2
Greedy	250	-	159	-	**222**	**96**
DBSCAN	**291**	**143**	192	71	132	-
Hierarchical	**291**	136	**330**	**159**	205	89

based on a combination of word embeddings of the underlying text blocks. ULR provided us such similarity scores for each pair of text blocks in each page of the three NewsEye GT data sets (ONB_230, NLF_200, BNF_183).

The given texts were initially stripped of special characters and punctuation symbols. Additionally, after tokenization (using the freely available NLTK tokenizer[16]), any (language-specific) stop-words were removed. Afterwards, for the three languages (German, Finnish, French), pre-trained FastText[17] models were used to convert word tokens into word vectors. Now, text blocks could be assigned the average of their word vectors and similarity scores could be computed using the cosine distance between two text block vectors. Finally, these scores were normalized by a simple linear mapping $x \mapsto (x+1)/2$ from $[-1, 1]$ to $[0, 1]$, so that they could be interpreted as probability scores representing the likelihood of two text blocks belonging to the same article.

The ULR similarity scores can be incorporated as additional edge features, in order to train new GNN models for each data set. These can then be compared to matching models that do not use the new features. The general model setup corresponds to the models of Section 7.3. For each data set we trained two new models, one with and one without the ULR scores. Since the models without the new features are identical to the specialized models of Table 5, we chose the best performing one for the comparative results depicted in Table 10.

For ONB_230 and NLF_200 we did not observe any significant changes in the final F_1 scores, when using the ULR similarity scores. This result is in line with the behavior of models using our own word-vector based similarity scores, as already described in Section 6.6.1. For BNF_183 a slight improvement on the test set should be noted, although this is offset by a more significant performance loss on the validation set. Overall, simple word-vector based similarity features seem to not consistently benefit the GNNs.

9.4 Rule-based post-processing of confidence graph

In an attempt to manually improve the results of the GNN prior to the final text block clustering, we experimented with some geometrically rule-based post-processing of the resulting confidence graph. In particular, we wanted to correct false positives, i.e. positively predicted text block relations from the GNN, that should not belong together in practice. This in turn should alleviate some of the wrongful merges that could be observed in the subsequent text block clustering.

[16] https://www.nltk.org/api/nltk.tokenize.html?highlight=word_tokenize#nltk.tokenize.word_tokenize
[17] https://fasttext.cc/docs/en/crawl-vectors.html

Table 10: Evaluation of GNNs trained with and without the ULR similarity scores. Results are chosen based on best F_1 score (at threshold T, in %).

Data set	Setup	Train				Val				Test			
		T	P	R	F_1	T	P	R	F_1	T	P	R	F_1
ONB_230	-	0.79	97.5	97.9	97.7	0.88	84.3	88.1	86.1	0.79	87.7	86.2	**87.0**
	+ULR	0.74	97.5	97.9	97.7	0.90	84.9	87.7	**86.2**	0.76	87.5	86.5	**87.0**
NLF_200	-	0.94	88.7	89.6	89.1	0.88	82.0	89.3	**85.5**	0.90	89.6	84.2	**86.8**
	+ULR	0.92	89.1	88.6	88.9	0.83	81.5	89.6	85.4	0.87	88.5	84.9	86.7
BNF_183	-	0.68	77.5	82.5	79.9	0.91	59.9	73.6	**66.0**	0.84	66.1	65.5	65.8
	+ULR	0.76	80.9	82.4	81.7	0.92	59.3	66.8	62.8	0.87	68.4	65.0	**66.6**

After discussions with project partners at UIBK-DEA, we made two basic assumptions regarding the formation of articles:

(1) Structurally, an article should usually start with a heading, followed by a text body. The next heading should then signal the end of the current article as well as the beginning of the next one.
(2) Visible horizontal separators should be an indicator for article borders.

Both, headings and horizontal separators are extracted by previous modules in the AS workflow and can therefore be used to implement rules that try to correct text block relations which do not fulfill the aforementioned assumptions. In this case, the corresponding edge in the graph would be masked, i.e. its confidence set to zero. Especially assumption (2) was of interest, since one of the conclusions of the previous internal user satisfaction survey (see Section 7.5) was that much more weight should be given to horizontal separators.

The rule set consists of the following rules, but only covers some of the possible cases in (1) and (2), since the proper reading order over the text blocks is not available (this would be necessary to make decisions for text block pairs in different columns on the newspaper page):

- We only consider relations with a confidence above a certain threshold (e.g. $p > 0.3$).
  The reason is that, because we want to correct false positives, we are not interested in relations that were assigned a low confidence by the GNN, since those are not likely to be clustered together later on.
- Given a pair of text blocks, for which exactly one of them is marked as a heading, we mask their edge in the confidence graph, if they are both horizontally aligned on the page (i.e. located in the same column, based on their bounding boxes) and the heading is located vertically below the other text block.
  This solves some of the cases in (1): It is not possible to make a reasonable decision for a pair of two headings. Furthermore, if the text blocks are not horizontally aligned, it is still possible that they are part of the same article which spans multiple columns.
- Given any pair of text blocks (A, B) and a set of horizontal separators $\mathcal{S}$, we first look for separators $S \in \mathcal{S}$ that vertically separate A and B, i.e. we either have an alignment of the form 'A over S over B' or 'B over S over A'. We then mask the corresponding edge, if the triple (A, B, S) is also horizontally aligned, in the sense that there needs to be at least a minimal horizontal overlap of both A and B with S respectively.
  This solves some of the cases in (2): Technically two text blocks can be separated by a horizontal

separator even if they are neither vertically nor horizontally aligned (e.g. when they are located in different columns), but this is impossible to check geometrically without the proper reading order.

These rules are pretty restrictive, since they can only ever mask pairs of text blocks which are horizontally aligned. But even then, experiments showed, that when incorporating this kind of post-processing to the confidence graph, the final text block clustering results actually decrease in quality. This behavior was consistent across all three NewsEye GT data sets and is due to the fact that the rules also mask true positive text block pairings in practice. The errors are mainly due to many different structural inconsistencies in the GT and underlying data, regarding the two assumptions mentioned above. These include, but are not limited to:

- visible lines beneath headings or underlining of text, that are interpreted as horizontal separators
- multiple sub-headings within a single article
- smaller horizontal lines or ornaments within articles which are interpreted as horizontal separators
- big headers on front newspaper pages, that consist of multiple headings and lines
- tables which consist of many horizontal separators and text blocks
- interweaved advertisements in articles, that function as separators

Some examples of the masking process (with selective errors) are shown in Figure 17. Although these errors occur more rarely than correctly masked edges, they have a big negative impact on the final clustering, which outweighs any performance gain from the post-processing.

9.5 Hierarchical text block clustering

Given the overall good results of the hierarchical text block clustering across different data sets, as supported by the results of Table 6 and Table 9, we wanted to expand on this approach.

As described in Section 6.8, the algorithm is an agglomerative clustering method, i.e. a bottom-up approach. It begins with a forest of clusters that have yet to be used in the hierarchy being formed. It then iteratively combines the two *closest* clusters, until one big cluster remains in the forest. To this end, a cluster distance metric needs to be defined. Afterwards, the hierarchy needs to be processed in some way, to extract the actual clusters. In our original approach, we chose the distance of centroids as the cluster distance metric and applied a distance threshold on the formed hierarchy to extract the final clusters.

9.5.1 Cluster distance metric

The algorithm maintains a distance matrix for each cluster in the forest, which gets updated at each step. Let $d(\cdot,\cdot)$ be the cluster distance metric and s and t the two closest clusters that form a new cluster u. In the algorithm, s and t will therefore be removed from the forest, u will get added and the distance matrix will get updated. In the following, suppose there are $|u|$ original nodes $u_1, \ldots, u_{|u|}$ in cluster u and $|v|$ original nodes $v_1, \ldots, v_{|v|}$ in cluster v, where $v \neq u$ is any remaining cluster in the forest. Also, let D be the original pairwise distance matrix.

Besides the *distance of centroids* (UPGMC algorithm - *unweighted pair group method centroid*)

$$d(u, v) = ||c_u - c_v||_2,$$

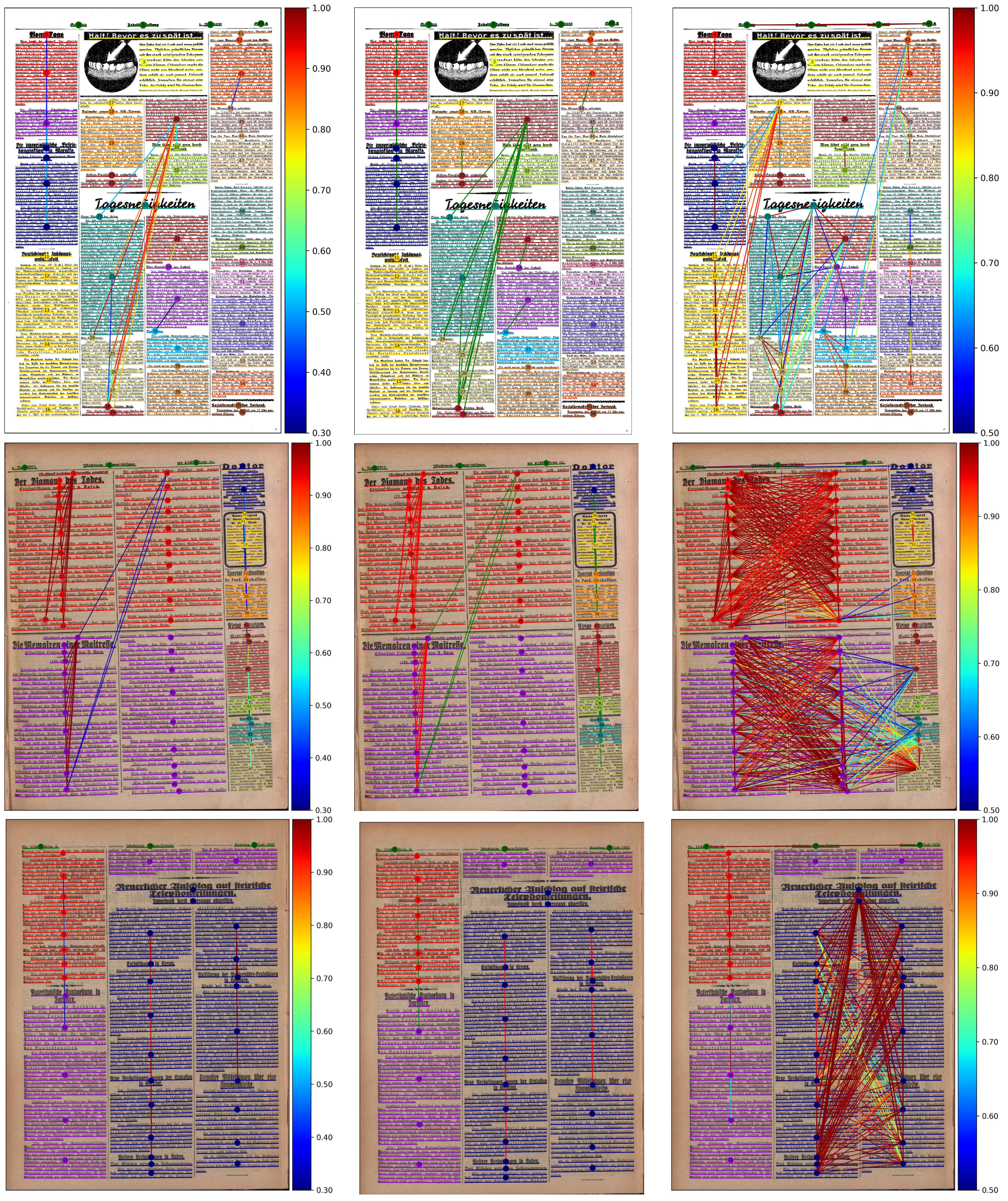

(a) (b) (c)

Figure 17: Post-processing of confidence graph for three different example pages. Masked edges and their confidence are shown in (a), whereas (b) categorizes them into corrects (green) and errors (red) regarding the GT articles. The resulting confidence graph is depicted in (c). The errors here are due to horizontal lines under headings (middle) and multiple sub-headings in a single article (bottom).

where c_u and c_v are the centroids of clusters u and v, and c_u is computed as the Euclidean centroid over all original nodes in clusters s and t, there are many other options for the choice of $d(\cdot, \cdot)$. We list some of them in the following:

- Method *Single* (Nearest Point Algorithm)

$$d(u,v) = \min_{i \in [|u|], j \in [|v|]} D_{u_i,v_j}.$$

- Method *Complete* (Farthest Point Algorithm)

$$d(u,v) = \max_{i \in [|u|], j \in [|v|]} D_{u_i,v_j}.$$

- Method *Average* (UPGMA Algorithm - *unweighted pair group method average*)

$$d(u,v) = d(s \cup t, v) = \frac{|s| \cdot d(s,v) + |t| \cdot d(t,v)}{|s| + |t|}.$$

- Method *Weighted* (WPGMA Algorithm - *weighted pair group method average*)

$$d(u,v) = d(s \cup t, v) = \frac{d(s,v) + d(t,v)}{2}.$$

The choice of the method will influence the shape of the final hierarchy and its individual cluster distances. One can also observe some inherent properties of the resulting hierarchy, given the cluster distance metric.

Using the method *Single*, clusters may be forced together due to single elements being close to each other, even though many of the elements in each cluster may be very distant to each other. In the context of the AS task, this would be a problem, since we have to assume some false positive in the confidence graph. The method *Complete* tends to find compact clusters of approximately equal diameters, but splits large clusters. Method *Centroid* has the possibility of inversions, i.e. two clusters that are merged may be more similar than the pair of clusters that were merged in a previous step. For other methods, the distance between merged clusters monotonically increases. Also, technically, it requires Euclidean distances between the original objects to be correctly defined. This is not guaranteed for the GNN confidences. Methods *Average* and *Weighted* both do simple averaging with different weightings. In theory *Average* should be preferred, because all distances contribute equally to each average and there is no practical reason to believe that individual nodes should have different weights.

Another idea to evaluate the different cluster distance metrics, is to compute the **Cophenetic Correlation Coefficient**. It measures how faithfully a hierarchy preserves the pairwise distances between the original objects and it can be argued that a hierarchy is an appropriate summary of the data if the correlation between the original distances and the cophenetic distances is high. We let $D \in (0,\infty)^{n \times n}$ be the original pairwise distance matrix, $\overline{d}$ its mean, $T \in (0,\infty)^{n \times n}$ the pairwise cophenetic distance matrix (distance between two objects in the hierarchy) and $\overline{t}$ its mean. The Cophenetic correlation coefficient $c \in [-1,1]$ is defined as

$$c = \frac{\sum_{i<j}(D_{i,j} - \overline{d})(T_{i,j} - \overline{t})}{\sqrt{\sum_{i<j}(D_{i,j} - \overline{d})^2 \sum_{i<j}(T_{i,j} - \overline{t})^2}}.$$

In practice we found, that the method *Average* produced the highest cophenetic correlation coefficients, if only slightly ahead of *Centroid*. The other methods performed worse overall. Therefore, we will be using the *Average* cluster distance metric going forward.

9.5.2 Cluster extraction

Once the hierarchy is formed, the actual clusters need to be extracted. As already mentioned, one way is to compute a distance threshold at which the hierarchy is cut. Any clusters that were merged with a distance metric below the threshold are kept and clusters formed above the threshold are split. Another option is to evaluate multiple possible clusterings using different cluster validity indices or methods with the goal to determine the optimal number of clusters. Given a hierarchy over n objects, there are n possible clusterings, from a single cluster containing all objects to n clusters with a single object each. For each of these clusterings the chosen cluster validity index is computed, in order to choose the optimal number of clusters (according to that index). There is wide range of such cluster validity indices available. Here, we want to focus on two methods.

The first method uses the **Silhouette Coefficient** as a measure of cluster validity. Let n objects be clustered into k clusters $C_1, \dots, C_k$. For a single object $u \in C_i$ we define

$$a(u) = \frac{1}{|C_i| - 1} \sum_{v \in C_i, v \neq u} D_{u,v} \qquad \text{as the distance of } u \text{ to its own cluster and}$$

$$b(u) = \min_{j \neq i} \frac{1}{|C_j|} \sum_{v \in C_j} D_{u,v} \qquad \text{as the distance of } u \text{ to its nearest cluster.}$$

The *Silhouette Score* $s(u) \in [-1, 1]$ of u is then computed as

$$s(u) = \begin{cases} 0, & \text{if } |C_i| = 1 \\ \frac{b(u)-a(u)}{\max\{a(u),b(u)\}}, & \text{if } |C_i| > 1 \end{cases} .$$

Note that $a(u)$ is not clearly defined for clusters with size 1, in which case we set $s(u) = 0$. For a high Silhouette Score we require $a(u) \ll b(u)$. Since $a(u)$ is a measure of how dissimilar u is to its own cluster, a small value represents a good cluster match. Furthermore, a large value of $b(u)$ implies that u is badly matched to its neighboring cluster. Thus, a Silhouette Score close to 1 means that the data is appropriately clustered and a value near 0 means that u is on the border of two clusters. A value close to -1 indicates that the clustering can be improved if u would be matched to its neighboring cluster. Finally, the *Silhouette Coefficient* $s \in [-1, 1]$ is the mean Silhouette Score over all objects

$$s = \frac{1}{n} \sum_{i=1}^{k} \sum_{u \in C_i} s(u).$$

For this method, the optimal number of clusters is the one that maximizes s.

The second method extracts the cluster distances that were used to merge clusters during the hierarchical clustering algorithm and associates them with the corresponding number of clusters. This data is then plotted and the method looks for an ‘elbow’ in the graph, i.e. for a change of slope from steep to shallow, to determine the optimal number of clusters. The merge distances at the start of the clustering algorithm, when each object still forms its own cluster, will be fairly small. In comparison, merge distances at the end of the clustering algorithm, when only a couple of clusters are left, will be fairly large. The idea of this **Elbow method** is, that the bend of the graph indicates, that additional clusters beyond that point have little value for a more meaningful cluster separation, since all objects are already relatively close to their clusters.

Figure 18 shows a newspaper example, its corresponding hierarchy and visualization of how both methods determine the optimal number of clusters.

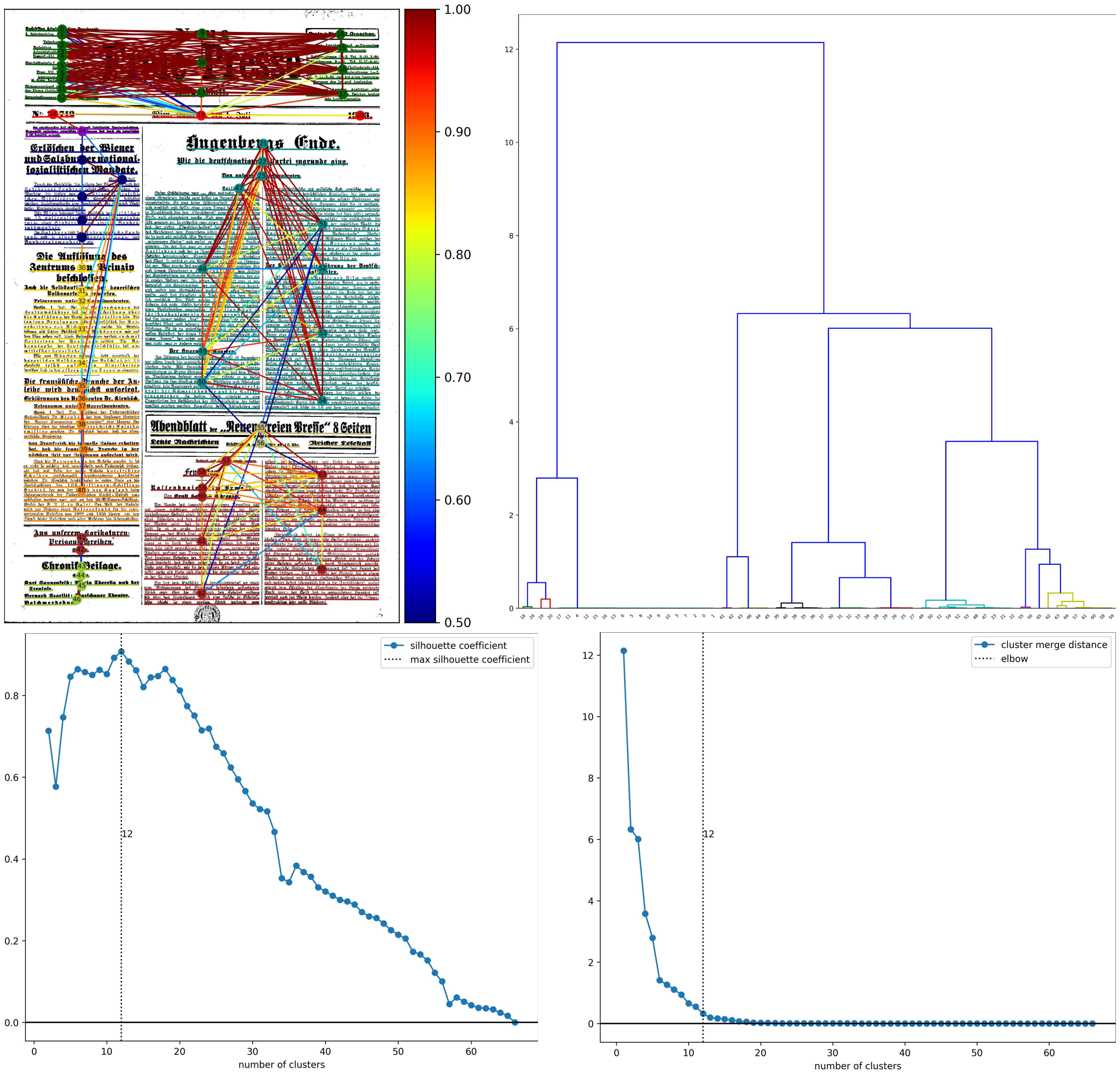


Figure 18: Visualization of the new hierarchical clustering methods. Top left: newspaper sample with overlayed confidence graph. Top right: Hierarchy formed by the clustering algorithm. Bottom: Graphs for cluster validity methods (left: Silhouette, right: Elbow), including the resulting optimal number of clusters (dotted lines).

9.5.3 Evaluation

We setup two alternative hierarchical clustering algorithms using the *Average* cluster distance metric and the two aforementioned cluster extraction methods. We compare them against the three candidates of Section 6.8 using the new comparative measure. The results for all GT data sets are depicted in Table 11.

The Elbow method is on par with the previous approach on the `ONB_230` data set. For `NLF_200` we can see that the threshold method still outperforms the new methods. On `BnF_183`, where we had the worst overall AS and GNN results (compared to the other two data sets), we see a clear improvement over any of the previous algorithms. This indicates that these cluster validity measures are more robust to noise in the confidence graph and could be good candidates for future unseen data sets, where no additional GT is available.

Table 11: News item clustering performance (comparative measure) on all three GT data sets for different clustering algorithms. The columns under Evaluation show a progression (left to right), where the worst algorithm is continuously removed from the comparisons. Draws count as a score for all involved algorithms.

Data set	Algorithm	Evaluation			
		All	Top-4	Top-3	Top-2
ONB_230	Greedy	544	406	-	-
	DBSCAN	562	423	275	-
	Hierarchical (threshold)	**621**	**461**	**306**	**170**
	Hierarchical (silhouette)	530	-	-	-
	Hierarchical (elbow)	611	458	301	158
NLF_200	Greedy	307	-	-	-
	DBSCAN	354	233	-	-
	Hierarchical (threshold)	**626**	**455**	**296**	**135**
	Hierarchical (silhouette)	505	358	218	88
	Hierarchical (elbow)	512	343	194	-
BnF_183	Greedy	260	134	-	-
	DBSCAN	155	-	-	-
	Hierarchical (threshold)	263	147	58	-
	Hierarchical (silhouette)	**617**	**441**	**273**	**114**
	Hierarchical (elbow)	605	430	262	108

9.6 Internal user satisfaction survey

We continued collecting user feedback for the AS results on the three NewsEye use-case data sets (cf. Section 4.2) for the project languages German, French and Finnish. Moreover, the designated subset of Swedish language newspapers was added for processing during NewsEye's prolongation phase. Hence, altogether four data sets have been entirely processed by UROS' AS workflow (cf. Section 5). For every language, 100 pages were chosen randomly, all of which were pure test data and have never been used for training of any of the machine learning components.

We conducted the survey by asking the project partners at the National Libraries in Vienna, Paris and Helsinki, respectively

(a) to provide a star-like ranking (0-5 stars) for the overall impression of the AS quality per page
(b) to count, per page, correctly separated articles and faults in incorrectly split or merged articles or wrong article beginnings (cf. Section 7.5)

We are very grateful to have received almost complete feedback for (a) and partial feedback for (b), where this approach apparently generated a too extensive workload for figuring out all details. An overview of user feedback (a) is given in Table 12. We only received feedback for 41 pages on the French data set and the 3 missing pages on the German data set are due to users classifying them as irrelevant to the task at hand. These answers allow for a first qualitative estimate, and show that the final AS results are considered to be of rather weak to medium quality.

Table 12: User Satisfaction Survey: Feedback on ranking with 0 – 5 stars.

	DE	FR	FI	SV
0	3	1	10	32
1	8	6	45	39
2	31	25	33	15
3	31	9	10	6
4	21	-	2	7
5	3	-	-	1
total	97	41	100	100
average	2.70	2.02	1.49	1.20

However, from our overall point of view, this overview also stands as a proof of the concept: It shows that NewsEye's AS workflow works in principle – but also reveals that very different efforts have been implemented for the four data sets. While for the German texts, we were able to tune very specific training data, the respective quality was essentially different for French and Finnish data sets and has not been tuned in any manner. Finally, for the recently added Swedish data set, we only exploited structural features and did not even apply a BERT-type language model based semantic feature.

Therefore, we may directly conclude that much more preparatory work and training effort for machine learning based algorithms has to be done for adjusting the engines and components to the specific data and the particular requirements of specific applications.

However, in spite of earlier hopes, the overall feedback did not allow to draw resilient quantitative conclusions on what amount and/or type of separation failures may affect the users' impression on the separation quality or the usability of the separated page. Our conjecture is that this is mostly due to the very different expectations and view points that different users may have. Consequently, for processing data sets and consistently estimating the resulting AS quality, one should agree upon and provide evaluation guidelines prior to the survey.

As an immediate consequence, the project partner UIBK-DEA (WP1 leader) proposed to run a very specific follow-up project on very homogeneous data and extremely clean GT. This will allow to precisely measure the resulting quality and to better estimate the true capabilities of the established workflow.

9.7 Future work

At the time of writing this report, project data sets with AS results are used and the AS workflow is investigated for usage by three NewsEye partners: UIBK-DEA (READcoop / Transkribus, Austria), ONB (Austrian National Library), and UH-DH (Helsinki Computational History Group, University of Helsinki, Finland). All teams will further be supported by UROS' software developers.

UROS, as the leading technology partner for AS, continues to process the quality assessment data set (see Section 9.6). In the context of exploitation and sustainability, it will further evaluate those results together with READcoop's Transkribus team and partner UIBK-DEA, and propagate the findings to interested NewsEye partners.

For this, following advice from UIBK-DEA, the evaluation measure for comparing automatically-generated article separations against a ground truth (see Section 9.2) is modified: It now also counts based on baselines directly (rather than on the intermediate text blocks, as done earlier), which allows to take all possible failures into account.

Finally, as part of the implementation of the sustainability plan and as already mentioned at the beginning of Section 9, we plan a more intensive exchange with the Transkribus team to further advance the integration of the AS workflow into Transkribus.

References

[1] Tobias Grüning, Roger Labahn, Markus Diem, Florian Kleber, and Stefan Fiel. “READ-BAD: A New Dataset and Evaluation Scheme for Baseline Detection in Archival Documents”. In: *CoRR* abs/1705.03311 (2017). arXiv: 1705.03311. URL: http://arxiv.org/abs/1705.03311.

[2] Andrew Naoum. “Article Segmentation in Digitised Newspapers”. PhD thesis. University of Sydney, 2020. URL: https://hdl.handle.net/2123/21725.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Technologies* 1 (2019), pp. 4171–4186.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[6] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. “On the shape of a set of points in the plane”. In: *IEEE Trans. Information Theory*. 1981.

[7] Tobias Grüning, Gundram Leifert, Tobias Strauß, and Roger Labahn. “A Two-Stage Method for Text Line Detection in Historical Documents”. In: *CoRR* abs/1802.03345 (2018). arXiv: 1802.03345. URL: http://arxiv.org/abs/1802.03345.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Proc. of the Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. “Mask R-CNN”. In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[11] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[12] Ross B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[13] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[14] David Hébert, Thomas Palfray, Stéphane Nicolas, Pierrick Tranouez, and Thierry Paquet. “Automatic article extraction in old newspapers digitized collections”. In: *ACM International Conference Proceeding Series* (May 2014). DOI: 10.1145/2595188.2595195.

[15] Andrea Britto Mattos, Igor Dos, Santos Montagner, Alexandre Crivellaro, Bruno Klava, and Marcel Brun. “Computer vision research at IBOPE Media: automation tools to reduce human intervention”. In: (Jan. 2011).

[16] B. Gatos, S. L. Mantzaris, K. V. Chandrinos, A. Tsigris, and S. J. Perantonis. “Integrated algorithms for newspaper page decomposition and article tracking”. In: *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR ’99 (Cat. No.PR00318)*. 1999, pp. 559–562. DOI: 10.1109/ICDAR.1999.791849.

[17] Anukriti Bansal, Santanu Chaudhury, Sumantra Dutta Roy, and J.B. Srivastava. “Newspaper Article Extraction Using Hierarchical Fixed Point Model”. In: *2014 11th IAPR International Workshop on Document Analysis Systems*. Apr. 2014, pp. 257–261. DOI: 10.1109/DAS.2014.42.

[18] B. Epshtein, E. Ofek, and Y. Wexler. “Detecting text in natural scenes with stroke width transform”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2963–2970. DOI: 10.1109/CVPR.2010.5540041.

[19] Emily M. Bender and Alexander Koller. “Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5185–5198. DOI: 10.18653/v1/2020.acl-main.463. URL: https://www.aclweb.org/anthology/2020.acl-main.463.

[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[21] B Delaunay. “Sur la sphere vide”. In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2.

[22] Günter Mühlberger. *Private communication*.

[23] Johannes Michael, Max Weidemann, Bastian Laasch, and Roger Labahn. “ICPR 2020 Competition on Text Block Segmentation on a NewsEye Dataset”. In: Feb. 2021, pp. 405–418. ISBN: 978-3-030-68792-2. DOI: 10.1007/978-3-030-68793-9_30.