Using deep learned vector representations for page stream segmentation by agglomerative clustering

submitted in partial fulfillment for the degree of master of science

Lukas Busch
12490334

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 31.10.2022

	UvA Supervisor
Title, Name	Maarten Marx
Affiliation	UvA
Email	M.J.Marx@uva.n

ABSTRACT

Page Stream Segmentation (PSS) is the task of segmenting streams of pages into their original source documents. The state of the art is neural start-of-document-page classification on representations of the text and image of pages. We wanted to improve performance by viewing PSS as a clustering problem instead, hypothesizing that pages from one document are similar to each other and different from pages in other documents. We compared the difference between the use of image and text pre-trained page vectors and finetuned vectors taken from a neural classification model on a new publicly available dataset. Clustering, neither with pre-trained, nor with finetuned neural page representations does unfortunately not perform better for PSS than start-of-document-page classification. Clustering with all representations is substantially more effective than the baseline, with finetuned outperforming pre-trained and early fusion of both image and text representations outperforming individual representations. Having the number of documents K as part of the input, in our use case a realistic assumption, has a surprisingly large positive effect. Additionally, we tried to predict K, but were not able to have accurate enough predictions to improve classification performance.

KEYWORDS

Page Stream Segmentation, Agglomerative Clustering, VGG16, Transfer learning

GITHUB REPOSITORY

https://github.com/luka5132/Agglomerative_Clustering_PSS

1 INTRODUCTION

The task of Page Stream Segmentation (PSS) is concerned with the segmentation of consecutive streams of pages into their original source documents. This task is of great importance to digitalization efforts in many different domains, including financial records [10], lawsuits [6] and historical documents [29].

In this research we perfrom PSS on a new open dataset containing streams released by the Dutch government under the Open Government Act, or *Wet Open Overheid* (WOO), formerly known as *Wet Openbaarheid Bestuur* (WOB). A previously released dataset, by van Heusden et al. [28], is similar to our dataset. The difference is that the dataset used in this study contains streams that are released by Dutch ministries and in the dataset by van Heusden et al. streams were released by Dutch provinces. In both cases these streams suffer from the same problems. One problem is that they are hard to read through as they can be long and unstructured. With many documents concatenated into one pdf file with no clear division between the documents. For the Open Government Act to function best, these streams have to be made easier to read and search through. For this, the first step approach is to separate the streams back into individual documents. Here PSS plays a big role.

The current state of the art approach to PSS is to see this problem of segmentation as a classification problem: binary classifiers are trained to detect whether or not a page is the start of a document [6, 10, 14, 29]. Instead we approach PSS as an agglomerative clustering problem in which each page is compared to its neighbouring pages, and pages are represented as either finetuned or pre-trained image/text vectors. This way we leverage the relationships between pages, instead of viewing them as stand alone objects. We do this under the hypothesis that pages in a document are more similar to each other than pages from other documents. This method has been used before, by Thompson and Nikolov in 2002 [9], but they used a combination of rule based representations and a classification model to determine similarity, instead of deep- and transfer-learned vector representations.

We build mostly upon the study of Wiedemann et al. [29] and use their Convolutional Neural Network (CNN) architecture for both image and text classification models. These models are used to obtain fintetuned representational vectors and to compare the performance of our clustering models against.

We also test performance using pre-trained representations. These are less costly as they would require less computational power and most importantly no annotations, which would be needed to train a new CNN model. To obtain visual representations we use a pre-trained version of the VGG16 model [25] and for textual representations we use BertSentenceTransformer [22].

Research Aim Our aim is to improve PSS performance by using restricted connectivity agglomerative clustering and vector representations of pages taken from finetuned and pre-trained deep learning models. In addition we ask the following questions:

- **RQ1** Does clustering performance improve when we use supervision? That is, with vector representations from a finetuned CNN model [29] instead of from a pre-trained classification model (VGG16) [25] or BertSentenceTransformer [22]?
- **RQ2** How can we adapt Agglomerative Clustering to PSS, and does that improve its performance?
- **RQ3** What is the effect of treating the number of documents K as given for classification?

Clustering, neither with pre-trained, nor with finetuned neural page representations does not perform better for PSS than start-of-document-page classification. Clustering with both representations is substantially more effective than the baseline, with finetuned outperforming pre-trained. A combination of visual and textual representations gives the best clustering performance. Lastly the number of documents K as part of the input, in our use case a realistic assumption, has a surprisingly large positive effect.

2 RELATED WORK

We discuss how agglomerative clustering has been used in NLP, and survey the most relevant and successful approaches to page stream segmentation.

2.1 Agglomerative Clustering

Agglomerative Clustering is developed by Joe Ward in 1963 [15]. In the field of text analysis, agglomerative clustering has mainly been used for two different purposes. The first is clustering sets of documents into groups. In 2013, Su et al. [26] did this using a simple keyword frequency comparison. In the same year, Alfred et al. [2] used Tf-Idf document vectors. In 2019, Franciscus et al. [12] used word embeddings instead of TfIDf to cluster tweets. The second main purpose is finding topics of interest in (large) texts. In 2011, Wu et al. [30] created a linear segmentation algorithm based on hierarchical clustering and in 2020 Bodrunova et al. [4] used agglomerative clustering based on sentence embeddings to find those topics.

2.2 Page Stream Segmentation

In 2002, Thompson and Nikolov [9] used a combination of hierarchical clustering and classification in their approach to Page Stream Segmentation. Using a rule based representation of papers they trained a classifier to determine the similarity of pages and then used bottom-up hierarchical clustering to separate documents. In 2009, Meilender and Belaïd [19] extracted specific fields from pages to use as features for a Bakis Model. The first study that viewed PSS as a solely classification problem was that of Agin et al. [1], who used a Bag of Visual Words (BoVW) representation and three different classifiers, Support Vector Machines (SVM), Random Forest and Multilayer Perceptron (MLP). In 2014, Rusinol et al. [23] used both image and text as input for their SVM and were the first to use multimodal machine learning for PSS. In 2021, with the use of transfer learning, Wiedemann et al. [29] obtained until now state of the art PSS performance by creating a multimodal classification model using the VGG16 architecture [25] for images and a pre-trained FastText [5] for word embeddings. Using domain specific Bert [11] models [14] only yielded a minor improvement. Lastly, in 2022, Demirtas et al. [10] included interpage context by adding page dependencies using 33 semantic classes, which increased performance substantially.

3 METHODOLOGY

We divide the methodology in five subsections. First we introduce our dataset. Second we show which classification and clustering models were used. Third, we define and justify our metrics. Fourth, we create and explain a baseline PSS method. Fifth, we explore two improvement methods of clustering and one possible improvement for classification.

3.1 Data

We use a new open dataset — that has yet to be made public — which contains streams of governmental documents. These streams come from dutch ministries and have been annotated by members from Follow the Money [27] and students of the University of Amsterdam. There are a total of 110 streams of which we use 108, because we only consider streams that contain at least two documents, so that we do not cluster with only one group.

These 108 streams have an average stream (page)length of 827 and an average of 226 documents per stream, see fig 1. It is clear from the figure that there is a lot of variety in both document and page count per stream. Which, as we will see later on, makes it difficult to accurately predict the document count of a stream (K). We split the data in a train and test set with 70%, 30% of the total streams respectively. This results in a train set of 76 streams, with 63.815 pages and 17.834 documents and a test set of 32 streams with 25.124 pages and 6476 documents.

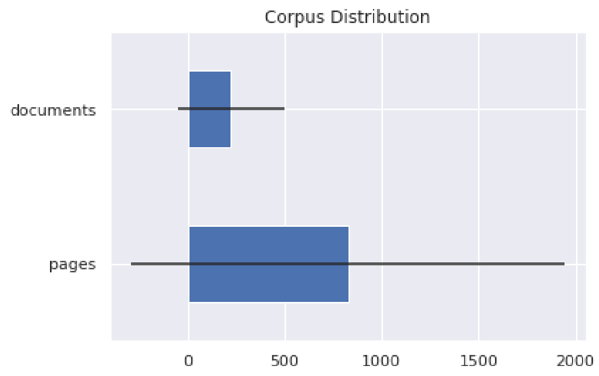


Figure 1: Distribution of documents and pages from dataset. The blue area is the mean, the black line the standard deviation

In the dataset pages from streams have already been extracted as images with 300 dpi. Similarly, text has been extracted using an Optical Character Recognition (OCR) model.

3.2 Models

In this subsection we first show which models were used for image and text classification as well as which hyperparameters were used. Then we go into detail on how we performed clustering.

3.2.1 Classification. We use the CNN classification architectures from the study of Wiedemann et al. [29], who obtained state of the art performance on the Tabacco800 [18] dataset and an in-house dataset with German documents.

Their image classifier is a deep CNN architecture that follows the works of Noce et al. [20] and Gallo et al. [13]. The basis of this architecture is a pre-trained VGG16 model [25] that takes 224x224 images as input. Wiedemann et al. remove the last dense layer of the VGG16 model and add two new layers. Finally they freeze the first four CNN blocks of the VGG16 model and keep the last block as well as the newly added layers trainable.

The text classifier is based on a study by Kim et al. [16] that had great performance on sentiment analysis tasks. Wiedemann et al. extend the model by Kim et al. by adding bidirectional Gated Recurrent Units (GRU) [8] layers before convolution. To embed word semantics they use a fastText [5] model that has been trained on Wikipedia.

We use the same architecture of Wiedemann et al. for image classification, except that we use 300x300 size images as input. We train this model for 20 epochs with batch size 256 and a learning rate of 1E-5.

For text classification we again use the architecture of Wiedemann et al. [29], but instead of using the English version of Fasttext [5] we use the Dutch version: *cc.nl.300.bin*. We then train this model for 20 epochs with a batch size of 256 and a learning rate of 1E-5. We perform no text cleaning since the page-texts contain many non-ascii characters that can be informative.

Because we perform clustering with the assumption that the number of documents (K) in a stream are known we test the effect of this on classification. We thus differentiate between *normal* classification predictions, in which we use a threshold of 0.5 and *K given* classification predictions, in which we set the K highest predicted scores to 1.

3.2.2 Clustering. For agglomerative clustering we use the sklearn *AgglomerativeClustering* model [21]. Finetuned vector representations are taken from the last dense layer of both the image and text CNN models. Pre-trained image vectors are taken from the last dense layer of the pre-trained VGG16 model [25] and pre-trained text vectors are created using BertSentenceTransformers [22] with the *bert-base-nli-mean-tokens* language model.

We use these representations to calculate the *cosine* distance for each page and its neighbouring pages. We then restrict connectivity to only neighbouring pages, normalize the distances and use the *average* linkage function to perform clustering. Both the use of cosine similarity and the average linkage function were found to yield the best results in a hyperparameter search.

3.3 Metrics

For both clustering and classification, we measure results using the macro F1 score for the "page starts a new document" class, which we call *Page F1*, and macro Panoptic Quality [17], which we call *Doc F1*.

The Doc F1 scores are precision, recall and F1 scores based on matching documents allowing some, but punishing for, mismatches. For each predicted document D_p, there is either no or exactly one true document D_t with more pages in the overlap of D_p and D_t than in their symmetric difference. The pairs (D_p, D_t) having enough overlap are the true positives. The only difference then with standard P, R and F1 calculations is that the numerator is not the number of TPs, but its *weighted count* in which we sum the Jaccard similarities of all TP pairs. We argue this metric is more important as PSS should focus on retrieving the documents as accurately as possible. We define the formulae for both metrics below.

Page F1: In the formula below, TP and FP stand for True- and False-positives and TN and FN stand for True- and False-negatives. A true positive is when a page is predicted to be a start page and this is true. If the page is not a start page we speak of a False-positive. This works similarly for TN and FN.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Page\ F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

$$Page\ F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

Doc F1: Inspired by the formula notation of van Heusden et al. [28] we define D_t to be a ground truth block and D_p is a predicted block. A block is a set of (connected) pages. Intersection of both of these blocks is called *Intersection over Union* or $IoU(D_t, D_p)$, this is calculated using Jaccard similarity. We consider a pair of blocks (D_t, D_p) True Positive if $IoU(D_t, D_p) > 0.5$. This means that more than half of the pages of D_t and D_p need to overlap. Then we define TP to be the set of all True Positives and FP the set that contains all predicted blocks (D_p) that are not in a TP pair. Similarly FN represents the set of all ground truth blocks (D_t) that are not in a TP pair. Because we do not require a complete overlap we essentially create *Weighted True Positives*, let this be WTP. The formula is then defined as follows

$$WTP = \sum\{IoU(D_t, D_p) \mid (D_t, D_p) \in TP\} \tag{5}$$

$$Doc\ Precision = \frac{WTP}{|TP| + |FP|)} \tag{6}$$

$$Doc\ Recall == \frac{WTP}{|TP| + |FN|)} \tag{7}$$

$$Doc\ F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{8}$$

$$Doc\ F1 = \frac{WTP}{|TP| + 0.5(|FP| + |FN|)} \tag{9}$$

We report the average scores of these metrics over all streams — macro scores — in the test set.

3.4 Baseline

We do not only compare our clustering results to the classification models, but use a baseline method as well, as reference. We call this baseline method the *Naive Mean*, because we simply assume every document in the test set to have the micro average document length of documents in the train set. The average (rounded) document length of streams in the train set is 4. With this mean value we obtain the following results:

Table 1: Performance of *Naive Mean* baseline method on test set of 32 streams.

Metric	Precision	Recall	F1
Page	0.38	0.27	0.29
Doc	0.18	0.14	0.15

As one would expect this method does not perform well. We see that the scores are higher for the Page metric, implying that it is easier to make a 'lucky' prediction on pages than on documents. Finally we note that precision is higher than recall, especially for the Page metric. This is because for every four pages only one page is predicted to be a positive, whereas three pages are predicted to be negative. With somewhat random predictions — because of the simplicity of the method — recall will naturally be lower than precision.

3.5 Improvement Methods

With respect to RQ2 we try several methods to best adapt clustering to PSS. First we explain how to best combine textual and visual representations in an attempt to increase performance. Then we go into a clustering method we developed that we call the *switch method* which tackles the *similarity problem* that occurs when performing agglomerative clustering with finetuned representation vectors, we will dive deeper into the *similarity probem* in section 5.2.1. Finally, we further inspect the effect of K (RQ3) and propose an approach to predict K. This did not directly lead to an improvement, but results show that it is valuable to explore this area further.

3.5.1 *Multi modality.*

We are primarily interested in the effect of multi-modal representations on clustering. Since we already know from the works of Wiedemann et al. [29] and Guha et al. [14] that this approach has a positive impact on classification. We distinguish between two methods of combining textual and visual features: *early* and *late fusion.*

In *early fusion* we first combine the image and text representations and then calculate the distance between them. In essence this can be done by simply concatenating the two representational vectors. However, one has to be careful of the scale of both vectors. If one vector has values that are considerably larger than the other, this vector will be dominant when calculating the distance between the resulting combined vectors. We see that this is not so much a problem for the finetuned vectors, as the average absolute value in the finetuned image vectors is 1.05, compared to 1.35 for text vectors. For pre-trained vectors the difference is much larger, with an average value of 0.401 for image and only 0.029 for text. If one were to simply concatenate the pre-trained image and text vectors the results obtained with these combined representations would be nearly identical to those of using only image vectors.

A simple approach to solving this would be to normalize both vectors and then combine them. However, as Attig and Perner showed in 2011 [3], normalizing vectors has a negative effect on obtaining an accurate similarity measure. Therefore we equalize the scale using a different method. Let the image vector be V^i with size (N^i, M^i) and the text vector be V^t with size (N^t, M^t). We first calculate the average absolute value for both vectors, let these be a_i and a_t. Depending on which value is larger we calculate the quotient (q) with the larger a value as the denominator and the lowe ar value as the numerator. We then divide the vector with the largest a value by this quotient. For clarification see the formulas below:

$$a_i = \frac{\sum_i^{N^i} \sum_j^{M^i} |V_{ij}^i|}{N^i * M^i} \tag{10}$$

$$a_t = \frac{\sum_i^{N^t} \sum_j^{M^t} |V_{ij}^t|}{N^t * M^t} \tag{11}$$

$$q = \begin{cases} a_i/a_t, & \text{if } a_i > a_t \\ a_t/a_i, & \text{if } a_t > a_i \end{cases} \tag{12}$$

For example, if $a_i > a_t$ we would then divide the image vector by the quotient: $V^i = V^i/q$. This way both vectors will have the same scale without it interfering with the similarity measures.

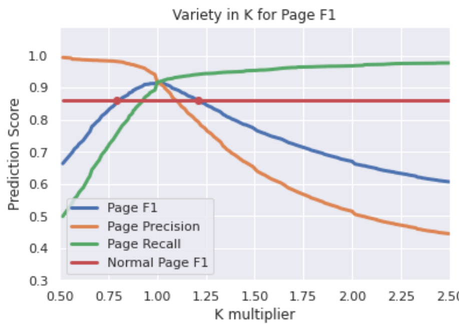
Late fusion is easier to perform. First we calculate the cosine distances between all pages of a stream for both modalities and store these in a list: $D = [d(p_1, p_2), d(p_2, p_3), ..., d(p_n, p_{n+1})]$. Let D_{min} be the minimum distance and D_{max} be the maximum distance in D. We then normalize the distances in the list:
$D = [(d_i - D_{min})/(D_{max} - D_{min})], for\ d_i \in D$.
Let the normalized list using images be (D^i) and text be (D^t), to obtain a combined list of distances (D^c) we then simply add the distances from both lists together:
$d^c(p_i, p_{i+1}) = d^i(p_i, p_{i+1}) + d^t(p_i, p_{i+1}), for\ d^i \in D^i, d^j \in D^j$

3.5.2 *Switch method.*

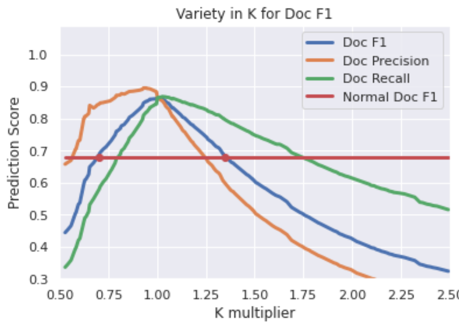
As we will see later in the study, clustering causes some problems. To address one of these we created a so-called 'switch' cluster method. Here, if a distance between two pages $d(p_{i-1}, p_i)$ is higher than a threshold T — which is the K_{th} highest distance from that stream — we set ('switch') the distance of the next pair of pages:
For distance $d(p_i, p_{i+1})$

$$f(d) = \begin{cases} d(p_i, p_{i+1}) = 2 \cdot \mu_d - d(p_i, p_{i+1}), & \text{if} d(p_{i-1}, p_i) > T \\ d(p_i, p_{i+1}) = d(p_i, p_{i+1}), & \text{otherwise} \end{cases}$$

With μ_d being the average distance for that stream. The rationale behind this becomes clear in our error analysis in section 5.2.1.



(a) Effect on Page performance



(b) Effect on Doc performance

Figure 2: Effect of variety in K values between 0.5 and 2.5 times the actual value. Measured in Page and Doc performance on 32 streams of the test set

3.5.3 *K prediction.*

With our dataset it is not unrealistic to assume K as given. However, in other scenarios K might be harder to obtain and thus would have to be predicted. Therefore we investigated the value of predicting K. To know how accurately K should be predicted we ran an experiment on our test set where we multiplied the true value of K with a values between 0.5 and 2.5 and measured the impact that the different K values have on our best performing model: the image classifier.

Results of this are shown in Figure 2. In the figure we can see that if we manage to predict the value of K within 80% and 120% of the true value we can improve page F1 performance, see 2a. There is more leniency — and more to gain — for doc F1, with a range between 70% and 135%, see 2b. If our prediction of K is outside these ranges, we see that performance quickly gets worse.

Another observation we make from Figure 2 that substantiates our choice to include Doc F1 as a metric comes from looking at the precision and recall scores. As one would expect, a lower K value increases Page precision, as the model takes less risks and thus lowers the amount of false positives, similarly this increases the amount of false negatives, hence the decrease in recall performance. For the Doc scores we see that precision actually goes down as well, this happens because the loss of weighted true positive documents (WTP) is greater than the reduction of false positive documents (FP). In other words, these extra predicted start page have a big influence on properly identifying documents. We will expand on this in section 5.1

To predict the value of K we use the cosine distances between pre-trained representations of pages. Based on our hypothesis we expect high distances between pages to indicate document boundaries and low distances to indicate that pages belong to the same document. Therefore we create histograms of the distances for both visual and textual pre-trained representations of pages in a stream. So that these can be used as training data to predict the total number of documents in a stream. We created this method ourselves in an attempt to generate more predictive features that can be used by a regression model.

With these histograms we have two 10-dimensional vectors — one for image and one for text — of which the first value represents how many distances are between 0-0.1, the second value is the amount of distances between 0.1-0.2 and so forth. We then use these individually, or we concatenate both vectors, and add the page count to create 11- or 21-dimensional training vectors.

The model we use to predict K is the Extreme Gradient Boosting (XGBoost) regression model, which is based on decision trees and was developed by Chen and Guestrin in 2016 [7]. We measure performance using the mean squared error (MSE) metric, which is a common metric to use for regression. We do not do any hyperparameter tuning.

4 RESULTS

Table 2 contains the results for all methods using only one modality. All methods outperform the baseline, and most of them substantially. In Table 3 we see the effect of combining modalities for clustering, using both early and late fusion. In Table 4 the MSE scores of our K-precition models are shown.

We now briefly answer our research questions. Our main conclusion is that boundary page classification is far superior to clustering for PSS, even if the number of documents is not known. Thus we did not reach our aim of increasing performance with clustering compared to classification.

Regarding RQ1 we find that supervision does help substantially with clustering. The best performing finetuned clustering method — multi-modal with early fusion — outperforms its pre-trained counterpart with 37% page F1 score (0.49 to 0.67) and 142% doc F1 score (0.24 to 0.58). Secondly, with respect to RQ2, we observe that the best adaption to PSS for clustering is to use both improvement methods discussed in section 3.5. We see that the use of the switch method (3.5.2) increases performance across all different modalities

Table 2: Macro F1 scores over 32 streams (6.476 docs with 25.124 pages) for the classification and clustering methods using only one modality as well as the mean document length baseline.

Method	modality	Page F1	Doc F1
Mean doc. length baseline	-	0.29	0.15
Classification	Image	0.86	0.68
	Text	0.83	0.70
Classification (K given)	Image	**0.92**	**0.87**
	Text	0.86	0.80
Cluster finetuned	Image	0.54	0.25
	Text	0.54	0.28
Cluster finetuned (switch)	Image	**0.61**	**0.52**
	Text	0.59	0.49
Cluster pre trained	Image	0.49	0.24
	Text	0.42	0.18
Cluster pre trained (switch)	Image	0.43	0.29
	Text	0.38	0.23

and that both fusion methods of image and text representations (3.5.1) outperform the use of only one modality. The best performing method using finetuned representations without these improvements uses image representations and gives a page F1 score of 0.54 and a doc F1 of 0.24. With improvements the best performing model — multi-modal with early fusion and using the switch method — improves these results by 24% page F1 score (0.54 to 0.67) and an impressive 132% doc F1 score (0.25 to 0.58).

Table 3: Macro F1 scores over 32 streams (6.476 docs with 25.124 pages) for multi-modal (Image + Text) clustering methods.

Method	fusion method	Page F1	Doc F1
Cluster finetuned	early	0.56	0.28
	late	0.57	0.30
Cluster finetuned (switch)	early	**0.67**	**0.58**
	late	0.65	0.54
Cluster pre trained	early	0.49	0.24
	late	0.48	0. 25
Cluster pre trained (switch)	early	0.44	0.30
	late	0.42	0.27

Concerning classification, the best performing modality is to use images. As explained in the methodology, we did not look into the use of multimodalies here, as it is already shown to give small improvements [14, 29]. The page F1 score of .86 obtained using images on our corpus is in line with that reported in [29] on their corpus. Our main positive result — related to RQ3 — is that when the number of documents K is known, we obtain 28% performance gain in doc F1 (0.68 to 0.87). In our use case, this K is easily obtained through knowledge extraction. If not, this result shows that it pays off to invest in estimating the number of documents in a stream.

However, as one can see in Table 4 this is non trivial, the lowest MSE score we obtained was 8519, this was with the use of only textual representations. This means that on average the predicted value of K was 93 ($\sqrt{8519}$) documents lower or higher than the actual value. From section 3.1 we know that the average K value for streams in the test set is 202. We know from Figure 2 that predictions need to be within 80% and 120% of the actual value of K to improve Page F1 and between 70% and 135% to increase Doc F1. Therefore, in Table 4, we show how far off the predictions are in percentage score. We differentiate here between predictions that were lower than the actual value — *avg mult lower* — and predictions that were above the actual value, *avg mult higher*. We make this distinction because we see in Figure 2 that performance gets worse much quicker when the predicted K value is lower than the actual value, compared to when the predicted value is higher.

Table 4: Macro MSE scores and average percentage scores of XGBoost K-prediction model for different pre-trainend representations on 32 streams in test set

Modality	MSE	avg mult lower	avg mult higher
Image	15641	**0.79**	1.96
Text	**8519**	0.73	2.14
Image+Text	11460	0.76	**1.84**

5 DISCUSSION

5.1 RQ3: How much does knowing K help?

In this section we first go into detail on the effect of K on classification by looking in depth at the prediction performance. Then we discuss the results of our XGBoost K-prediction model and the implications of this.

5.1.1 Effect of K. We focus only on the performance of the image model here, as this model gave the best results. If we zoom in on the effect of K on page level classification results we see in Table 5 that top-K and the normal method make almost the same total number of mistakes but with the top-K method false positives and false negatives are completely balanced while they are not without K given. The equal precision and recall caused the increase in their harmonic mean. With K given, the classifier is forced to assign more pages to the True class, increasing recall. This naturally comes with more false positives, but the increase in correctly identified start pages is so large that precision even goes up as well.

Table 5: FP's and FN's for start page classification using images

	FP	FN	Total
Normal	239	707	946
Top-K	477	477	954

The effect on doc F1 is even larger, a 28% increase to .87. Indeed, having K known causes 230 extra correctly identified startpages, but almost an increase in weighted true positives WTP of nearly 1300. Of these 1300, 830 blocks are 1 or 2 page documents, for which Panoptic Quality requires a complete match between the predicted and true pages and thus rewards with an IoU score of 1. Our task, PSS, is after all about segmenting a stream into documents, so a high score on the document metric is the ultimate goal.

5.1.2 Predicting K. Our K-prediction model is not accurate enough to give predictions that can improve classification performance. We directly see this by looking at how far off the predictions are relative to the actual value of K. In Table 4 we see that none of the three different models can predict K accurately enough to be within the necessary range to improve Page F1 score (0.8-1.2). We do see that when predictions are too low — *avg mult lower* — they fall just in range of the doc F1 score (0.7-1.35), but when the prediction of K is too high it is far above the range.

If we look at the direct effect of the predicted K values on the performance of the classification model — viz Fig 6 — we find that using the predicted K values of the model with the lowest MSE score actually leads to the worst classification performance. Predictions for K of both other models score slightly higher. If we compare the results of either using K values predicted with image representations or image and text representations to the results of using a *normal prediction* we find that page F1 score drops by 8% (from 0.86 to 0.79) and 3% for doc F1 (from 0.68 to 0.64). We include classification with K given — i.e. using the true K value — as well, to show the upper bound of what can be achieved with effective K prediction.

Interestingly we find that MSE does not necessarily accurately reflect how effective a K prediction model is in increasing classification performance. Although prediction using text representations gave a MSE score nearly half that of prediction using images, its impact on classification is worse. We do see however that when MSE is significantly lower — as is the case for the normal classification method — it reflects performance better.

With only 76 streams as training data and a high variety in both page and doc length of these streams (see fig 1) the poor results of our prediction models can somewhat be explained. We would expect that more training data would lead to an increase in K prediction scores. As we can see by the relatively low loss in doc F1 performance, the prediction model does not need much to already improve classification results. More research in this area could therefore be valuable.

Table 6: Effect of using K values on classification scores, either by giving true value of K (K given) or predicting K using image and/or text representations

	MSE	Page F1	Doc F1
classification	786	0.86	0.68
classification (K given)	**0**	**0.92**	**0.87**
classification (K predicted)			
Using Image	15641	**0.79**	**0.66**
Using Text	**8519**	0.78	0.64
Using Image+Text	11460	**0.79**	**0.66**

5.2 RQ2:How to adapt clustering to PSS?

In this section we go into how to use agglomerative clustering for PSS. We focus only on finetuned vectors, as we will discuss pretrained vectors in section 5.3. We have already explained how we perform basic clustering in section 3.2.2, but as we can see in the results in Table 2, base performance can be improved greatly by adding our improvement methods.

First, we see that using both modalities — image and text — increases overall performance and second we find that using the switch cluster method gives the biggest improvement, especially for the doc F1 metric. We explain why the switch method is so effective by discussing a problem we discovered with clustering that we named the *similarity problem* in subsection 5.2.1. Lastly we highlight another problem with clustering — that we were not able to solve — that in part explains the low performance, we call this problem the *entry problem* and discuss this is subsection 5.2.2

5.2.1 Similarity Problem. Since the finetuned representations are taken from a classifier these representations reflect its prediction; that is, whether or not a page starts a new document (class 1 or 0). The cosine distance between two fine tuned representations of the same type — (1,1) and (0,0) – is low, and is high between a start-page and a non start page, viz. Table 7. Two of these 4 scenarios cause a problem: we want (1,1) to be clustered into different groups which they will not, given a low distance. Conversely, (1,0) should be clustered together as the 0 here represents the second page of a new document. We call this the *similarity problem* and solve it by using the 'switch' cluster method described in section 5.2.1. Here the distance after a potential start page is switched which causes the distance between two start-pages (1,1) to be higher, so that they are clustered apart, and the distance between a start page and non-start page (1,0) to be lower, so that they are clustered together.

Table 7: Mean normalized cosine distances of finetuned image representations, grouped by classifier predictions.

Prediction	1	0
1	0.25	0.76
0	0.73	0.25

Table 8: Highlighting the 'switch' cluster method

(a) Without Switch

Clus pred	distance	Class pred	Gold
...	...	...	...
1	0.357	1	1
0	0.330	1	1
0	0.655	1	1
1	0.554	0	0
1	0.270	1	1
...	...	...	...

(b) With Switch

Clus pred	distance	Class pred	Gold
...	...	...	...
1	0.762	1	1
1	0.789	1	1
1	0.465	1	1
0	0.566	0	0
1	0.542	1	1
...	...	...	...

We highlight this method with an example from one of our test streams in Table 8. We can see in Table 8a that while the classification model is correct the cluster predictions are wrong. Because the first three pages of our example are all start-pages (1,1,1) they have a low distance and are clustered together. Although the third and fourth page belong to the same document (1,0), the distance is high. Thus they are predicted to be different documents. The predictions with the switch method are shown in Table 8b. Because of the switch, the distance between the first and second, and second and third page are reversed and are correctly predicted to be start pages. Likewise the distance between the third and fourth page, which was high before, is now low enough to be clustered together. We can see the effectiveness of this method back in the results: the highest score using clustering was obtained with the switch.

5.2.2 Entry Problem. Clustering still suffers from another problem, in which one mistake is multiplied into many. We call this the *entry problem*. Figure 3 contains the binary predictions of the switch clustering method using finetuned image representations versus the gold standard of an example from our test set. We only consider the part of the stream indicated by the 0s and 1s. For the first five pages of the highlighted area we show the clustering prections as well as the switched distances and gold standard in Table 9. We see that the first two pages are in reality both start pages $(1, 1)$, but the switched distance (0.503) is not enough to classify them apart, and thus they are grouped together in the prediction $(1, 0)$. What follows after this first mistake is that the cluster model is able to properly identify document boundaries, but because the *entry value* of this chain was wrong all predictions are exactly the opposite of what they should be (the 40 page long red part in the figure).

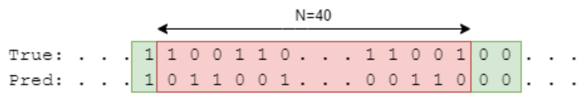
Table 9: Predictions and distances of example stream highlighting the entry problem

distance (with switch)	Clus pred	Gold
...	...	...
0.503	1	1
0.722	0	1
0.869	1	0
0.151	1	0
0.270	0	1
...	...	...

We consider the *entry problem* to be the case when at least 10 pages with a minimum of 4 different document boundaries are predicted to be the opposite of the ground truth. With this definition we can calculate the impact of the *entry problem*, results are shown in Table 10. For all modalities only the best performing model is showcased. This means that only finetuned representations are considered and that the Image+Text representations were combined using early fusion.

One thing that stands out is that the entry problem almost exclusively occurs in clustering with the switch method. Rather than see this as a shortcoming of the switch method we attribute this effect to the success of the method. This is because the entry problem can only occur when document boundaries are properly identified,

Figure 3: The entry problem on a stream from the test set.


N=40
True: . . . 1 1 0 0 1 1 0 . . . 1 1 0 0 1 0 0 . . .
Pred: . . . 1 0 1 1 0 0 1 . . . 0 0 1 1 0 0 0 . . .

which is not the case for clustering without the switch method. If we consider the results from Table 2 and Table 3 we can make an even stronger observation: that the better the clustering model, the more it suffers from the entry problem. More research is certainly needed to solidify this hypothesis, but our results indicate that the entry problem might be a byproduct of a strong agglomerative clustering model in PSS.

Unfortunately we are not able so solve both the similarity problem and the entry problem together, as we can not predict where the cluster models will go wrong. The impact of the entry problem is large, as it is the cause of misidentification of nearly 5% of all 25124 pages. If we could repair the entry problem and only count the initial mistake, page F1 would increase by 18% and doc F1 by 19% for image clustering; 12% and 16% for text and 15% and 17% for image+text respectively.

Table 10: Number of misclassified pages because of the entry problem for different methods of clustering using finetuned representations, image+text was combined with early fusion

	Image	Text	Image+Text
Switch	1268	1138	1341
Normal	0	0	16

5.3 RQ1: How much does supervision help clustering?

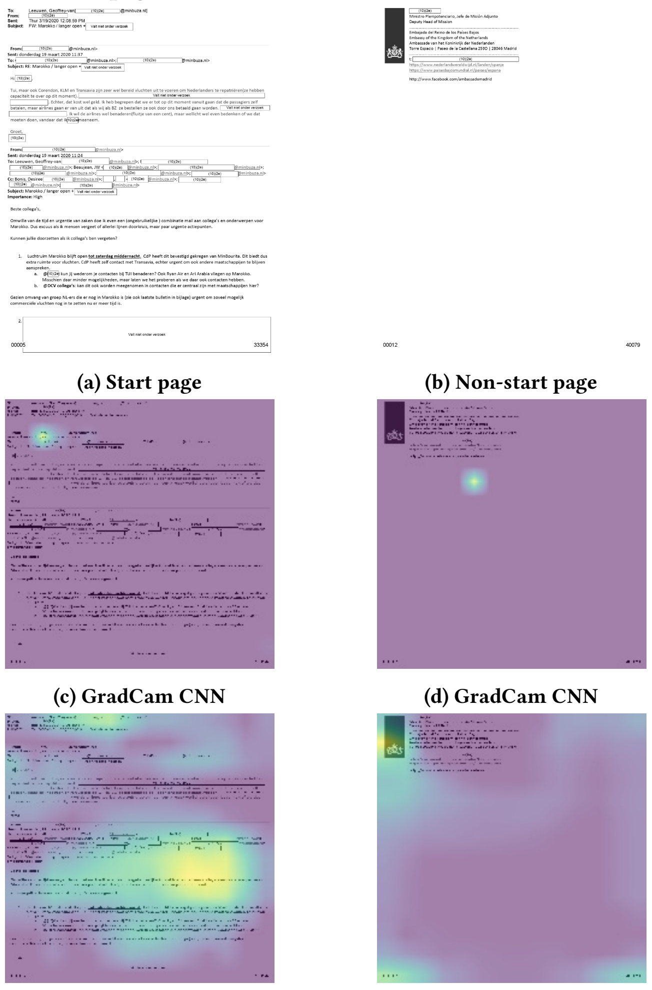
The performance of clustering using finetuned vectors is much higher than that with the use of pretraind vectors. In this subsection we will go into the reason behind this. We explained in subsection 5.2.1 that finetuned representations reflect the binary output of the classification model. We now further analyse this by focusing on the image model. With the use of GradCam [24] we found that the image classification model focuses primarily on email headers or contact information on letters. For the pre-trained vectors this is not the case. Instead of being guided by the classification model, the VGG representations are based on more general features such as the page structure, or whether or not a page contains an image or a table.

This is highlighted in Figure 4, we see that the finetuned CNN model either focuses on the header when the image is a start page (see 4c) or 'looks away' when the image is not a start page (see 4d). The VGG16 model on the other hand does not know what to focus on and explores large areas of the page, regardless on whether or not the page is a start page, as can be seen 4e and 4f.

We believe this is a more intuitive separation of pages. For example, with VGG representations, the distance is high between a page with only an image and a page with only text. The classification model on the other hand, sees both these pages as 'uninteresting', classified as 0, which causes the distance to be low.

Unfortunately, as we can see in the results (Table 2 and Table 3) this intuitive way of separating pages is too simplistic for a complex task as PSS. We calculated the average distance between pages from the same documents and with pages from the other documents using pre-trained visual representations and found that there is almost no difference, with an average cosine distance of 0.232 and 0.236 respectively. Therefore we have to conclude that our hypothesis was wrong and that pages from the same document are not more similar to each other than pages from different documents. In fact, documents in our dataset do not have a big variety and start pages can be identified by considering only a few features. Because the hypothesis does not hold there is no possible way for the cluster algorithm to differentiate the documents using pre-trained representations. We suspect that with more variety in documents clustering might perform better. Simultaneously, if there are more types of start pages, classification would likely perform worse.

Figure 4: Original images and GradCam visualization of CNN and VGG16 of pages from example stream, one start and one non start page



(a) Start page **(b) Non-start page**

(c) GradCam CNN **(d) GradCam CNN**

(e) GradCam VGG16 **(f) GradCam VGG16**

6 CONCLUSION

We tried to increase PSS performance by viewing PSS as a clustering problem instead of a classification problem, which is the current standard approach. We compare our clustering performance to classification scores based on the models developped by Wiedeman et al. [29] .

We found that classification substantially outperforms clustering for PSS, and even more when the number of documents in a stream is given. Although it is not unrealistic for out data to consider K as given we created a XGBoost model [7] to predict K. The best performing version of this model uses either pre-trained image representations or a combination of pre-trained image and pre-trained text combinations and is not accurate enough to increase performance. We attribute this mainly to the limited training set of only 76 streams.

In addition, we found that clustering with finetuned representations of the classification model gives better results than with pre-trained representations. We created two improvement methods: multi modal clustering with early fusion and the switch method. Both these methods have a positive effect on clustering performance. However, clustering with finetuned representations still comes with two problems: the *similarity problem*, which we solved using 'switch clustering' and the *entry problem* which we were not able to solve.

Finally, we conclude that pre-trained representations are too simplistic for a complicated task as PSS. We suggest not to rule out clustering entirely, as more variety in documents could both favour clustering and hinder classification.

Therefore we suggest future research in this area to apply clustering using pre-trained representations to a dataset with bigger variety in document types. A big improvement for clustering with finetuned representations can be made by adressing the *entry problem*. If somehow one is able to detect when this occurs and make an intervention, performance could increase substantially. Classification can be improved by researching alternative ways of predicting K, possibly by including more features or combining different datasets to obtain more training data.

Although the results obtained in this study were below expectations, there are many areas to explore for further research and make possible improvements.

REFERENCES

[1] Onur Agin, Cagdas Ulas, Mehmet Ahat, and Can Bekar. 2015. An approach to the segmentation of multi-page document flow using binary classification. In *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, Yulin Wang, Xudong Jiang, and David Zhang (Eds.), Vol. 9443. International Society for Optics and Photonics, SPIE, 216 – 222. https://doi.org/10.1117/12.2178778

[2] Rayner Alfred, Tan Soo Fun, Asni Tahir, Chin Kim On, and Patricia Anthony. 2013. Concepts Labeling of Document Clusters Using a Hierarchical Agglomerative Clustering (HAC) Technique. In *The 8th International Conference on Knowledge Management in Organizations*, Lorna Uden, Leon S.L. Wang, Juan Manuel Corchado Rodríguez, Hsin-Chang Yang, and I-Hsien Ting (Eds.). Springer Netherlands, Dordrecht, 263–272.

[3] Anja Attig and Petra Perner. 2011. The Problem of Normalization and a Normalized Similarity Measure by Online Data. *Ibai=publishing* 4, 1 (2011), 3–17.

[4] Svetlana S. Bodrunova, Andrey V. Orekhov, Ivan S. Blekanov, Nikolay S. Lyudkevich, and Nikita A. Tarasov. 2020. Topic Detection Based on Sentence Embeddings and Agglomerative Clustering with Markov Moment. *Future Internet* 12, 9 (2020). https://www.mdpi.com/1999-5903/12/9/144

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[6] Fabricio Ataides Braz, Nilton Correia da Silva, and Jonathan Alis Salgado Lima. 2021. Leveraging effectiveness and efficiency in Page Stream Deep Segmentation. *Engineering Applications of Artificial Intelligence* 105 (2021), 104394.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *CoRR* abs/1603.02754 (2016). arXiv:1603.02754 http://arxiv.org/abs/1603.02754

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Kevyn Collins-Thompson and Radoslav Nickolov. 2002. A clustering-based algorithm for automatic document separation. In *SIGIR 2002 Workshop on Information Retrieval and OCR: From Converting Content to Grasping, Meaning*. Tampere, Finland.

[10] Mehmet Arif Demirtaş, Berke Oral, Mehmet Yasin Akpınar, and Onur Deniz. 2022. Semantic Parsing of Interpage Relations. *arXiv preprint arXiv:2205.13530* (2022).

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[12] Nigel Franciscus, Xuguang Ren, Junhu Wang, and Bela Stantic. 2019. Word Mover's Distance for Agglomerative Short Text Clustering. In *Intelligent Information and Database Systems*, Ngoc Thanh Nguyen, Ford Lumban Gaol, Tzung-Pei Hong, and Bogdan Trawiński (Eds.). Springer International Publishing, Cham, 128–139.

[13] Ignazio Gallo, Lucia Noce, Alessandro Zamberletti, and Alessandro Calefati. 2016. Deep neural networks for page stream segmentation and classification. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 1–7.

[14] Abhijit Guha, Abdulrahman Alahmadi, Debabrata Samanta, Mohammad Zubair Khan, and Ahmed H Alahmadi. 2022. A Multi-Modal Approach to Digital Document Stream Segmentation for Title Insurance Domain. *IEEE Access* 10 (2022), 11341–11353.

[15] Joe H. Ward Jr. 1963. Hierarchical Grouping to Optimize an Objective Function. *J. Amer. Statist. Assoc.* 58, 301 (1963), 236–244. https://doi.org/10.1080/01621459.1963.10500845 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1963.10500845

[16] Y Kim. 2014. Convolutional neural networks for sentence classification., arXiv. *preprint* (2014).

[17] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. 2019. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9404–9413.

[18] D. Lewis, G. Agam, S. Argamon, O. Frieder, D. Grossman, and J. Heard. 2006. Building a Test Collection for Complex Document Information Processing. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. Association for Computing Machinery, New York, NY, USA, 665–666. https://doi.org/10.1145/1148170.1148307

[19] Th Meilender and Abdel Belaïd. 2009. Segmentation of continuous document flow by a modified backward-forward algorithm. In *Document Recognition and Retrieval XVI*, Vol. 7247. International Society for Optics and Photonics, 724705.

[20] Lucia Noce, Ignazio Gallo, Alessandro Zamberletti, and Alessandro Calefati. 2016. Embedded textual content for document image classification with convolutional neural networks. In *Proceedings of the 2016 ACM Symposium on Document Engineering*. 165–173.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[22] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[23] Marçal Rusinol, Volkmar Frinken, Dimosthenis Karatzas, Andrew D Bagdanov, and Josep Lladós. 2014. Multimodal page classification in administrative document image streams. *International Journal on Document Analysis and Recognition (IJDAR)* 17, 4 (2014), 331–341.

[24] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR* abs/1610.02391 (2016). arXiv:1610.02391 http://arxiv.org/abs/1610.02391

[25] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

[26] Chunhua Su, Jianying Zhou, Feng Bao, Tsuyoshi Takagi, and Kouichi Sakurai. 2013. Collaborative agglomerative document clustering with limited information disclosure. *Security and Communication Networks* 7, 6 (2013), 964–978. https://doi.org/10.1002/sec.811

[27] Follow the Money. [n. d.]. Follow the money - platform voor onderzoeksjournalistiek. https://www.ftm.nl/

[28] Ruben van Heusden, Jaap Kamps, and Maarten Marx. 2022. WooIR: A New Open Page Stream Segmentation Dataset. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval* (Madrid, Spain) *(ICTIR '22)*. Association for Computing Machinery, New York, NY, USA, 24–33. https://doi.org/10.1145/3539813.3545150

[29] Gregor Wiedemann and Gerhard Heyer. 2021. Multi-modal page stream segmentation with convolutional neural networks. *Language Resources and Evaluation* 55, 1 (2021), 127–150.

[30] Ji-Wei Wu, Judy C.R. Tseng, and Wen-Nung Tsai. 2011. An Efficient Linear Text Segmentation Algorithm Using Hierarchical Agglomerative Clustering. In *2011 Seventh International Conference on Computational Intelligence and Security*. 1081–1085. https://doi.org/10.1109/CIS.2011.240