

**ASSESSMENT OF OCR QUALITY AND FONT IDENTIFICATION IN  
HISTORICAL DOCUMENTS**

A Thesis

by

ANSHUL GUPTA

Submitted to the Office of Graduate and Professional Studies of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee, Ricardo Gutierrez-Osuna  
Committee Members, Richard Furuta  
Laura Mandell  
Head of Department, Dilma Da Silva

August 2015

Major Subject: Computer Engineering

Copyright 2015 Anshul Gupta

## ABSTRACT

Mass digitization of historical documents is a challenging problem for optical character recognition (OCR) tools. Issues include noisy backgrounds and faded text due to aging, border/marginal noise, bleed-through, skewing, warping, as well as irregular fonts and page layouts. As a result, OCR tools often produce a large number of spurious bounding boxes (BBs) in addition to those that correspond to words in the document. To improve the OCR output, in this thesis we develop machine-learning methods to assess the quality of historical documents and label/tag documents (with the page problems) in the EEBO/ECCO collections—45 million pages available through the Early Modern OCR Project at Texas A&M University.

We present an iterative classification algorithm to automatically label BBs (i.e., as text or noise) based on their spatial distribution and geometry. The approach uses a rule-based classifier to generate initial text/noise labels for each BB, followed by an iterative classifier that refines the initial labels by incorporating local information to each BB, its spatial location, shape and size. When evaluated on a dataset containing over 72,000 manually-labeled BBs from 159 historical documents, the algorithm can classify BBs with 0.95 precision and 0.96 recall. Further evaluation on a collection of 6,775 documents with ground-truth transcriptions shows that the algorithm can also be used to predict document quality (0.7 correlation) and improve OCR transcriptions in 85% of the cases.

This thesis also aims at generating font metadata for historical documents. Knowledge of the font can aid OCR system to produce very accurate text transcriptions, but getting font information for 45 million documents is a daunting task. We present an active learning based font identification system that can classify document images into fonts. In active learning, a learner queries the human for labels on examples it finds most informative. We capture the characteristics of the fonts using word image features related to character width, angled strokes, and Zernike moments. To extract page level features, we use bag-of-word feature (BoF) model. A font classification model trained using BoF and active learning requires only 443 labeled instances to achieve 89.3% test accuracy.

## **ACKNOWLEDGEMENTS**

I would like to thank my committee chair, Dr. Ricardo Gutierrez-Osuna, and my committee members, Dr. Richard Furuta, and Dr. Laura Mandell, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience. I also want to extend my gratitude to the eMOP team, which provided the historical document data, and the expertise needed to understand it.

## NOMENCLATURE

BB	Bounding box
ML	Machine learning
AL	Active learning
LP	Label propogation
hOCR	Microformat for OCR workflow and results
MLP	Multi-layer perceptron
ZM	Zernike moments
ZP	Zernike polynomials

## TABLE OF CONTENTS

	Page
ABSTRACT .....	ii
ACKNOWLEDGEMENTS .....	iv
NOMENCLATURE .....	v
TABLE OF CONTENTS .....	vi
LIST OF FIGURES .....	viii
LIST OF TABLES .....	xi
1. INTRODUCTION .....	1
2. BACKGROUND AND RELATED WORK .....	4
2.1 Background .....	4
2.2 Related work .....	5
3. AUTOMATIC ASSESSMENT OF OCR QUALITY .....	10
3.1 Methods .....	11
3.1.1 Pre-filtering .....	12
3.1.2 Column segmentation .....	13
3.1.3 Local iterative relabeling .....	14
3.2 Results .....	17
3.2.1 Datasets .....	17
3.2.2 Pre-filtering .....	18
3.2.3 Column extraction .....	20
3.2.4 Local iterative relabeling .....	20
3.2.5 Deriving a measure of document quality .....	22
3.2.6 Improving OCR transcriptions .....	24
3.3 Discussion .....	25
4. FONT IDENTIFICATION USING ACTIVE LEARNING .....	28
4.1 Feature extraction from document images .....	30
4.1.1 Font characteristics .....	30

4.1.2	Document image preprocessing.....	34
4.1.3	Mean and IQR stroke width for a word image .....	36
4.1.4	Slant line density .....	37
4.1.5	Zernike moments .....	40
4.1.6	Bag-of-words model .....	42
4.2	Active learning to build font identification model.....	43
4.2.1	Choosing base classifier .....	46
4.2.2	Active sampling .....	48
4.3	Results.....	51
4.3.1	Dataset .....	51
4.3.2	Feature extraction from document images .....	52
4.3.3	Bag-of-word features .....	55
4.3.4	Evaluation set-up for the active learning .....	56
4.3.5	Active sampling.....	57
4.4	Discussion .....	62
5.	CONCLUSIONS AND FUTURE WORK.....	66
	REFERENCES.....	70
	APPENDIX .....	76

## LIST OF FIGURES

	Page
Figure 1: OCR output for an eMOP document; BBs shown in green.....	11
Figure 2: Overview of the proposed BB classification method and features used for recognition at each stage.....	12
Figure 3: Segmenting columns by identifying troughs in the horizontal distribution of BBs .....	14
Figure 4: Finding nearest neighbors. Only those within $D_{max}$ from the corners of the target BB (outlined) are considered. Colors indicate the corner to which neighbors are assigned.....	16
Figure 5: Feature distributions for BBs in dataset 1.....	19
Figure 6: Column segmentation for two difficult test cases.....	20
Figure 7: (a) BB classification rate before and after local iterative relabeling; (b) Number of iterations required for convergence.....	21
Figure 8: Iterative relabeling results for the image in Figure 1. Color denotes MLP confidence: the more saturated, the higher the confidence. Red: noise; green: text .....	22
Figure 9: (a) BB-based quality measure ( $BB_{noise}$ ) vs. the Jaro-Winkler similarity ( $s_{JW}$ ) for 6,775 documents. (b) $s_{JW}$ before and after iterative relabeling; for most documents (those above the diagonal line) iterative relabeling improved $s_{JW}$ .....	23
Figure 10: Average change in Jaro-Winkler similarity as a function of document quality ( $BB_{noise}$ ).....	25
Figure 11: Example word images for Blackletter and Roman fonts. ....	29
Figure 12: Block diagram for active learning based font identification system. ....	29
Figure 13: Some anatomical characteristics of a font class [27].....	30
Figure 14: (a) Anatomical characteristics of Blackletter font; (b) Anatomical characteristics of Roman fonts.....	31



Figure 15: (a) Text snippet in Blackletter font; (b) Text snippet in Roman font. ....	31
Figure 16: (a) Output of the assessment algorithm, where red boxes denote predicted noise BBs and green boxes denote predicted text BBs; (b) Document image with predicted text BBs. ....	33
Figure 17: (a) Original word image with salt noise in the stems and pepper noise in the background; (b) Preprocessed word image after opening and closing operations.....	35
Figure 18: An example showing an intermediate step in the calculation of mean and IQR character width.....	37
Figure 19: An example of Hough transform applied to a word image. Here, the green line segments are the detected angled straight lines. ....	38
Figure 20: Pipeline for slant line density. ....	39
Figure 21: Visualization of the Hough transform output matrix, which represents count of the straight lines upon which the length of the perpendicular from the origin is $\rho$ and the perpendicular makes an angle $\theta$ with x-axis. In this figure color saturation represents the fraction of points (in the image) that lie on the line specified by certain $\rho$ and $\theta$ . ....	39
Figure 22: Zernike moment extraction process.....	41
Figure 23: Block diagram explaining bag-of-word features (BoF).....	43
Figure 24: Pool based active learning [37].....	45
Figure 25: Red and green points are labeled data; all other points are unlabeled data; dashed line is the decision boundary learned by a classifier. (a) shows the learned decision boundary for logistic regression; (b) shows the decision boundary for logistic labeled propagation [41]. ....	47
Figure 26: The red and green examples are the initial labeled examples; yellow examples are the most informative examples based on uncertainty; blue examples are the most diverse examples from the current labeled data; Navy blue examples are the most uncertain and diverse set of unlabeled examples. ....	49
Figure 27: Corss-validation F1-score for the classifiers trained using all 18 features, only Zernike moments (ZMs), and only character width (CW) mean & IQR and slant line density (SLD). ....	54
Figure 28: Coefficients of logistic regression (value averaged over 5 folds). ....	54

Figure 29: Scatter plot in PCA space where each dot is a document color coded by its class label.....	55
Figure 30: Learning curve (Red) for entropy ( <i>ScoreExploitation</i> ) based active sampling; Black curve represents the performance of random sampling.....	59
Figure 31: Learning curve (Red) for entropy ( <i>ScoreExploitation</i> ) based active sampling. Black curve represents the performance of random sampling. (a) Blue learning curve for <i>Scoresum, diversity</i> ; (b) Blue learning curve for <i>Scoresum, no diversity</i> ; (c) Blue learning curve for <i>ScoreSum – Diff, no diversity</i> ; (d) Blue learning curve for <i>ScoreSum – Diff, diversity</i> ; (e) Blue learning curve for <i>Scoreweighted sum, no diversity</i> ; (f) Blue learning curve for <i>Scoreweighted sum, diversity</i> ; (g) Blue learning curve for <i>Scorerandom exploration and exploitation</i> ; (h) Blue learning curve for <i>ScoreExploration</i> .....	60
Figure 32: Area under the learning curves for different sampling techniques (scores); {1,2,3,4,5,6,7,8,9} corresponds to the scoring functions. ....	62
Figure 33: (a) A Mixed class document misclassified as a Blackletter documents; (b) A Blackletter document misclassified into Mixed class; (c) A Roman document classified as a Mixed document. ....	64
Figure 34: eMOP post-processing pipeline. ....	66
Figure 35: Snapshot of Picasa used to annotate document images. ....	77
Figure 36: Sync service block diagram. ....	77

## LIST OF TABLES

	Page
Table 1: Features used during local iterative relabeling .....	17
Table 2: Datasets used for training and validation purposes.....	18
Table 3: Average change in Jaro-Winkler similarity ( $\Delta$ ) with application of the local iterative relabeling algorithm.....	24
Table 4: Features extracted from word images. ....	33
Table 5: List of labels used to annotate document images using Picasa. ....	52
Table 6: Number of labeled data for different classes.....	52

## 1. INTRODUCTION

Technology has transformed the way we access documents. Texts, images, sounds and videos are now easily accessible and searchable via internet services such as Google, Bing or YouTube. These services provide access to most of the documents published in the modern era: documents available in digitized forms such as pdf (texts), tiff (images) and mp4 (videos). With the outburst of these digital documents, historical texts—everything from pamphlets to ballads to multi-volume poetry collections in the hand-press period (roughly 1475-1800)—are becoming very difficult to locate by even the most devoted researchers.

This issue accelerated work to convert these historical texts to their digital counterparts. Through the use of Optical Character Recognition software (OCR), we can create machine readable versions of these texts. However, OCR of such documents is a challenging task due to the characteristics of the physical documents and the quality of their scanned images. Early printing processes (printing presses, mass paper production, handmade typefaces) produced texts with fluctuating baselines, mixed fonts, and varied concentrations of ink, among many other irregularities. To make matters worse, the existing digital collections for documents of that period largely consist of binary (i.e., as opposed to grayscale), low-quality and low-resolution images, the result of digitization from microfilm converted from photographs—four decades and three generations away from the originals [1].

To improve OCR quality, these documents generally require additional processing (e.g., image denoising, font identification) before the optical character recognition (OCR) transcriptions are of sufficient quality to undergo linguistic analysis (e.g., n-gram or dictionary lookup). Unfortunately, which document requires what type of processing is often unknown, and manually tagging/labeling each document in the collections is prohibitive. As a step towards improving OCR quality, this thesis will explore and develop machine-learning methods to assess the quality of historical documents and to label/tag documents (with the page problems) in the EEBO/ECCO collections—45 million pages available through the Early Modern OCR Project [2] at Texas A&M University. The two specific aims of this thesis are:

**Aim 1:** To develop a method to estimate the quality of OCR text output for which there is no ground truth, no typed text that it can be measured against. Obviously humans can look at the output and, comparing it to the page image, determine how well or poor an OCR engine has performed. But it becomes cost prohibitive to manually process massive amounts of textual data: OCR text for 45 million page images from the ECCO and EEBO databases. Hence, we must determine how well an OCR engine has performed on any given document automatically—that is, without human beings having to check each output.

**Aim 2:** To develop active-learning techniques to assist users in tagging the collections (EEBO and ECCO databases). In this thesis, we focus on font identification but the method can be extended to detect other page problems. In the digitization process, knowledge of which font is present in the document image can help improve the

performance of OCR [3-6]. Detection of font can be formulated as a supervised classification problem that requires labeled data for model building. However, getting labeled data from a corpus of 45 million page images, with varied font types, is a daunting task. To overcome this issue, we propose to use active learning to build the font classifier. Active learning is a learning paradigm where the Machine Learning (ML) algorithm guides the user (i.e., by suggesting high-value instances to tag) and the human guides the ML algorithm (i.e., by tagging those instances).

The rest of the document is organized as follows. Section 2 summarizes background of the eMOP project and related work for aim 1 and aim 2. Section 3 describes the proposed automatic quality assessment algorithm. Section 4 presents the active learning based system to identify font present on historical document images. Finally section 5 provides conclusion and directions for future work.

## **2. BACKGROUND AND RELATED WORK**

### **2.1 Background**

Motivated by issues in mass digitization of historical documents, researchers at Texas A&M University started in 2013 the Early Modern OCR Project (eMOP; <http://emop.tamu.edu>) with funding from the Andrew W. Mellon Foundation. eMOP is a two-year project that seeks to improve OCR for some 45 million pages from the Eighteenth Century Collections Online (ECCO) and Early English Books Online (EEBO) proprietary database products. The goal extends beyond producing accurate transcriptions for these collections, and also aims to create tools (dictionaries, workflows, and databases) to support scholarly research at libraries and museums. eMOP relies on the Tesseract open-source OCR engine available from Google [7]. For each document image, Tesseract produces a standard hOCR [8] data file containing the layout and logical structure of the document, including the coordinates of the bounding box (BB) of each recognized word along with its text transcription and recognition confidence.

In the second year of the eMOP project, the researchers working on improvement of OCR quality of historical documents realized that many document images are of such poor quality that no amount of training the recognition system would produce the desired accuracy/quality. They realized that the poor quality is mainly due to problems such as noisiness, bleed through, skewing and warping. EMOP's original triage process was

designed to examine OCR results and route documents to different tools such as automatic word correction, crowd-sourced line segmentation correction, by-hand font identification, or automated re-OCRing with different font training. However, due to the presence of many poor quality document images, eMOP required a different way for handling the hOCR output. Hence, eMOP started to focus on a triage process that would allow programmatic diagnosis of input documents based on the output of the OCR system. This diagnosis will output a page score, and documents with high enough page score will then be sent further for text analysis, including dictionary look-ups, to correct as much of the OCR output as possible [1].

## **2.2 Related work**

The ability to triage documents is critical in large-scale document digitization. Document triage prevents heavily degraded documents from entering the OCR pipeline, and instead directs them elsewhere for additional processing (e.g., rescanning, image denoising). In these cases, quality is generally defined as an objective property of the document image, such as OCR accuracy, though subjective measures (e.g., Mean Opinion Scores) have also been used. Image features that have been found to correlate with OCR performance include global properties, such as the amount of black background speckle, image sharpness and uniformity, as well as local properties of the text, such as stroke thickness and continuity, and character/word height-to-width ratio [9].



A few studies have focused on improving OCR performance by pre-applying image restoration techniques, such as deblurring, skew removal, and bleed-through removal, to mention a few. As pointed out by Lins et al. [10], however, these techniques should not be blindly applied but should be used selectively based on the type of noise or degradation present in the document. For this purpose, the authors developed a method to identify five types of noise (bleed through, skew, orientation, blur and framing) based on image features such as palette, gamut, or number of foreground pixels. The authors found that the overhead of this noise-classifier was far lower than running the image through unnecessary filters. In related work, Sandhya et al. [11] developed a taxonomy of image noises in historical documents that extends beyond the five categories of Lins, Banerjee and Thielo [10]. Their taxonomy considered four types of noise sources: aging, digitization and storage, physical factors (e.g., folding, burn, bleed-through) and document factors (e.g., varying fonts, mixed alphabets.). More recently, Farahmand et al. [12] reviewed image processing techniques to remove ruled-line noise, marginal noise, clutter noise, stroke-line pattern noise, background noise, and salt-pepper noise. More recently, Ben Salah et al. [13] proposed a method to detect missed text areas in historical documents. The method consists of building a classifier to discriminate between foreground pixels (those inside bounding boxes returned by the OCR engine) and background pixels. Once the classifier is built, it can be used to reclassify background pixels. The authors report an 84% recall rate for text components that were initially missed by the OCR engine.

Methods to classify documents into types of noise, as presented by Lins, Banergee and Thielo [10], are fully supervised; as such, they incur a large cost for obtaining sufficient labelled data to train the classifier. In recent studies, active learning is being widely used to reduce the overhead of getting large labelled dataset for building good supervised ML models [14, 15]. In active learning, a (machine) learner queries the (human) teacher for labels on examples it finds worth labelling with respect to the problem at hand [16]. Bouguelia et al. [17] proposed a semi-supervised active learning approach for stream based document classification task. The developed method is used to classify documents into classes such as bank checks, medical receipts, invoices, prescriptions etc. The authors reported that the method gives 2-3% precision boost, as compared to model built with fully labelled training dataset, using on an average only 36% of the labelled data.

Most prominent page image problems in documents are due to the aging process. With time, the page quality degrades and the ink from the back side of the page starts showing up on the front side, especially in the background. This degradation gets enhanced when such pages are scanned and binarized. The end result is a reduction in the OCR recognition accuracy and generation of garbage text output. Hence, many image preprocessing techniques have been used to reduce such page image noise and enhance the character edges in order to boost OCR recognition accuracy. One such work is presented by Likforman-Sulem et al. [18]. The authors combined two widely used noise reduction methods: total variation regularization that reduces background noise, and non-local means filter that enhances character details. They compared OCR

recognition accuracy for multiple datasets of historical documents, with and without the enhancement process, and showed a 10% boost in OCR recognition accuracy for most of their datasets.

A number of studies have focused on post-correcting errors in OCR outputs by modeling typographical variations in historical documents; see Reffle and Ringlsetter [19], Reynaert [20] and references therein. As an example, Alex et al. [21] proposed two OCR post-correction methods for the problems of end-of-line hyphen removal and substitution of long-s (recognized as f) to letter s (e.g. “fenfible” to “sensible”). Using dictionary based methods, the authors reported a 13% reduction in word error rates. Furrer and Volk [22] used a combination resources to determine the correctness of every word given as an output from the OCR engine, including a large dictionary system that covers morphological variation and compounding, a list of local toponyms, and recognition confidence values from the OCR engine. For these techniques to be effective, however, noise BBs must be removed in advance.

Historical documents are generally printed in multiscript. OCR systems work best when they have to recognize just one kind of script/font. Therefore, recognition of text from documents with multiple fonts usually leads to OCR errors. The knowledge of the font can boost accuracy of the OCR system, but in mass digitization projects it is not feasible to manually tag each document image with the type of font. This triggered the need for automatic font recognition for the document images before they are processed by the OCR system. In a survey paper on script recognition, Ghosh et al. [23] describe different forms of writing systems and the scripts such as logographic system, syllabic

system and alphabetic system, then categorize script recognition methodologies based on the extracted features. The two categories are 1) structure based; and 2) visual appearance based. In structure based methods, the connected components of characters are extracted, and their shape and structure are analyzed to detect particular scripts. In contrast, appearance based methods try to make use of features that can capture the visual appearance of the individual characters and the way they are grouped into words, lines and paragraphs. Each of these categories are further divided based on the level of their application in the document image. This means that each of the above categories can be applied at page level, paragraph level, word level or at document level.

### 3. AUTOMATIC ASSESSMENT OF OCR QUALITY\*

As illustrated in Figure 1, when a document has poor quality, the OCR engine generally produces a large number of spurious BBs –in addition to those that correspond to words in the document. As part of Aim 1, in this section, we present a method to discriminate between noisy and text BBs by analyzing statistical differences in their geometrical properties and confidence score returned by the OCR engine. To extract BBs features, we use hOCR file not the underlying image. We use these features to develop an algorithm to classify BBs returned from OCR into text or noise. We then use the classification result to formulate a document quality measure such as fraction of predicted noise BBs. This approach is advantageous for two main reasons. First, the approach does not require dedicated image processing algorithms [12], which can become prohibitive for large document collections. Second, the approach becomes language-agnostic because it relies exclusively on geometrical properties of BBs rather than the text transcription associated with them.

In the sections that follow, we explain the algorithm to classify BBs into text or noise, and validate it on a dataset containing 159 mid-to-poor quality documents (over 72,000 manually-labeled BBs). Then, we illustrate how the method can be used to obtain an objective measure of document quality and improve OCR transcription performance by filtering out noise BBs before the document undergoes post-correction.

---

\*Reprinted with permission from “Automatic assessment of OCR quality in historical documents” by Anshul Gupta, AAAI 2015, Copyright 2015 by AAAI.

We presented this work at the 29th AAAI Conference on Artificial Intelligence (AAAI) in January 2015 [24].

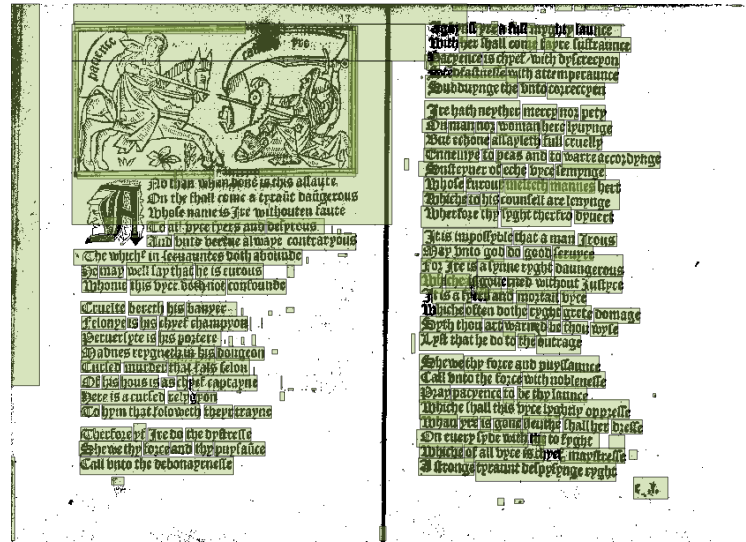
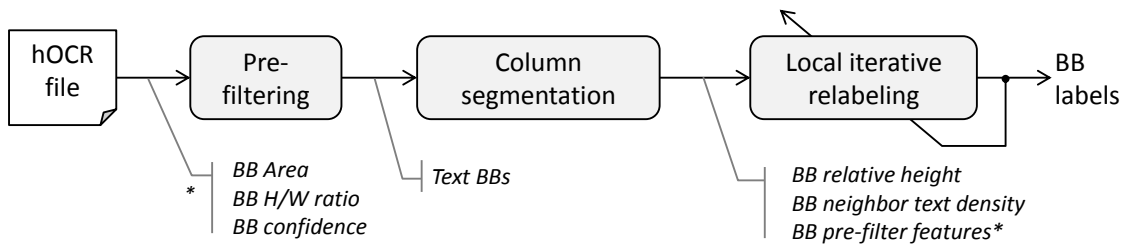


Figure 1: OCR output for an eMOP document; BBs shown in green

### 3.1 Methods

Our pipeline is based on the Tesseract open-source OCR engine available from Google [25]. For each document image, Tesseract produces an hOCR data file [8] (an open standard for formatted text from OCR) containing the layout and logical structure of the document, including the coordinates of the BB for each recognized word along with its text transcription and recognition confidence. It is the hOCR file, not the underlying image, that we use for analysis. Our overall approach for discriminating text and noise BBs is illustrated in Figure 2. The individual steps (pre-filtering, column segmentation, and local iterative relabeling) are described next.



**Figure 2: Overview of the proposed BB classification method and features used for recognition at each stage**

### 3.1.1 Pre-filtering

The first step in the process consists of generating initial labels for each of the BBs returned by Tesseract. For this purpose we use a rule-based classifier that considers three features for each BB: word confidence, height-to-width ratio and area. The rules are derived as follows:

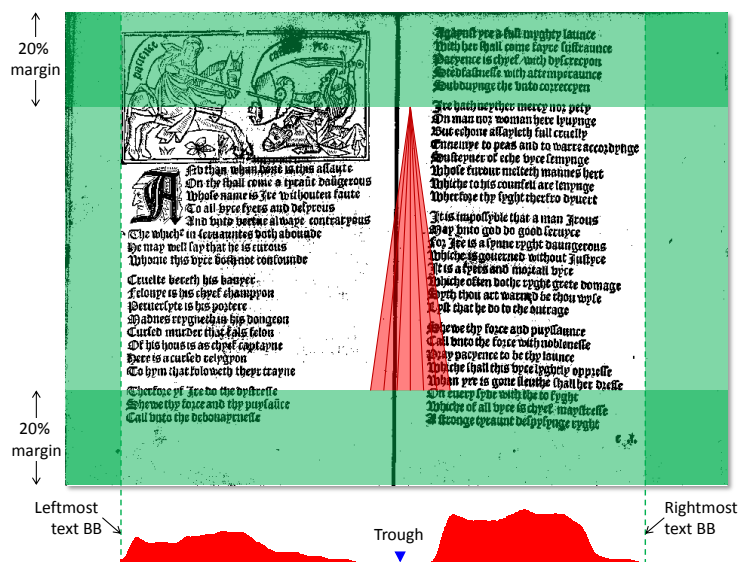
- **Rule 1: OCR word confidence.** BBs with very low or very high confidence predominantly consist of noise, and are flagged accordingly during pre-filtering.
- **Rule 2: Height-to-width ratio.** Most words are written horizontally, so the height-to-width ratio is generally lower for word BBs than for noise BBs. Consequently, if this ratio is less than a threshold we label the BB as text; otherwise, we label it as noise.
- **Rule 3: Area.** Tesseract has a tendency to misidentify speckles as legitimate text; fortunately, these areas are small as compared to normal text BBs. Accordingly, we label as noise all BBs in the lowest percentiles of the total area for the document.

Thresholds for the individual rules are optimized simultaneously with a manually-labeled subset of the corpus; see results section. The final filter is the conjunction of the three rules. BBs classified as text at this stage are used in the next stages to extract column layout and estimate the average font size of each document.

### *3.1.2 Column segmentation*

Documents in the eMOP collection generally have multiple pages and/or columns, each with its own set of problems (e.g., degree of skew or noise). For this reason, the second step in the process consists of dividing each image into its constituent pages and columns, so that each can be processed individually. First, we identify the leftmost and rightmost text BB from the pre-filtering stage; these coordinates define the text boundaries of the image. Then, we perform column segmentation by analyzing the distribution of BBs over the horizontal axis; the dominant troughs in this distribution define the column boundaries.





**Figure 3: Segmenting columns by identifying troughs in the horizontal distribution of BBs**

To compute this distribution of BBs, we divide the horizontal axis with 1,000 evenly-spaced points. At each point, we trace rays from the top margin to the bottom margin with slopes in the range  $90^\circ \pm 3^\circ$  in increments of  $0.2^\circ$ , then calculate the number of intersecting BBs for each ray. At each point, we then identify the ray with the fewest intersections, and that becomes the value of the distribution at that point. Since images tend to have a large number of spurious BBs at the margins, any BBs in the top and bottom 20% are discarded. The overall process is illustrated in Figure 3.

### 3.1.3 Local iterative relabeling

After each page has been split into columns, we apply an iterative relabeling algorithm to the BBs of each column. The rationale behind this final step is that BBs surrounded by text are more likely to contain text than those surrounded by noise.

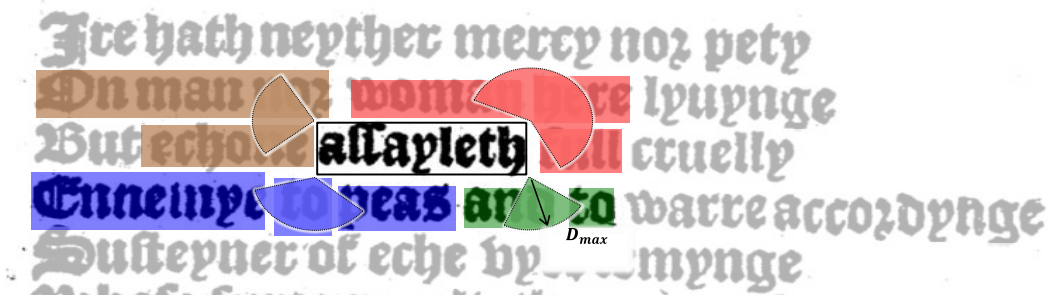
Accordingly, for each of the four vertices of each BB we find its nearest neighbors (see Figure 4). Then, we calculate a weighted score,  $S$ , based on the label of each neighbor penalized by its distance:

$$S(b) = \frac{\sum_{k=1}^P w_k L_k}{\sum_{k=1}^P w_k}, \quad \text{with } w_k = \frac{1}{\text{dist}(b,k)} \quad (1)$$

where  $b$  is the index of the BB,  $N$  is the number of BBs within distance  $D_{max}$  from the vertices of  $b$ , and  $P$  is the maximum number of nearest neighbors considered ( $P \leq N$ ).  $L_k$  is the predicted label (0: noise; 1: text) for the  $k$ -th nearest neighbor, initially taken from the pre-filtering step. As illustrated in Figure 4, the distance  $D_{max}$  limits the search area for nearest neighbors, preventing text BBs that are far from  $b$  to be considered in the computation. The distance  $D_{max}$  is computed relative to  $H_{med}$ , the median height of text BBs found in the pre-filtering stage, plus a tolerance defined by  $H_{IQR}$ , their interquartile range; both statistics are computed for each individual column in the image:

$$D_{max} = H_{med} + \alpha \times H_{IQR} \quad (2)$$

where  $\alpha$  defines the tolerance; the larger its value the more distant neighbors that are allowed in the computation of  $S$  of eq. (1). In our implementation, the value of  $\alpha$  is optimized to minimize the mean-square error between  $S$  and the ground-truth label for all BBs in a training set.



**Figure 4: Finding nearest neighbors. Only those within  $D_{max}$  from the corners of the target BB (outlined) are considered. Colors indicate the corner to which neighbors are assigned.**

The iterative process starts by initializing BB labels with those from the pre-filtering stage. From these labels, an initial score  $S$  can be computed for each BB. This score is then combined with six additional features (see Table 1), and passed as an input to a multilayer perceptron (MLP) previously trained to classify BBs as either text or noise. The additional features include those used in pre-filtering ( $C_{OCR}, H/W, A$ ) as well as the BB position relative to the document margins, and its height normalized to  $H_{med}$  and  $H_{IQR}$ . The resulting labels are used to re-compute  $S$  and the process is repeated until convergence, i.e., labels no longer change from one iteration to the next.

**Table 1: Features used during local iterative relabeling**

Features	Description
$S$	Score from nearest neighbors ; see eq. ((1))
$C_{OCR}$	OCR word confidence*
$H/W$	Height-to-width ratio of BB*
$A$	Area of BB*
$H_{norm}$	Normalized height: $H_{norm} = (H - H_{med})/H_{IQR}$
$H_{dist}$	Horizontal distance from the middle of the page
$V_{dist}$	Vertical distance from the top margin
<i>*available from the pre-filtering stage</i>	

## 3.2 Results

### 3.2.1 Datasets

To test the proposed algorithm we generated three separate datasets (see Table 2) consisting of binarized document images from the eMOP collection, carefully selected to represent the variety of documents in the corpora. This included single column, multi-page and multi-column document images, as well as images with artifacts due to ink bleed-through, multiple skew angles, warping, printed margins, printed column separators, and pictures. Each BB returned by Tesseract for each of the document images in all three datasets was then manually labelled (i.e., text/noise) to generate ground truth data, for a total of 72,366 BBs. As labeling criteria, we considered as noise any BB that spanned more than two lines of text, as well as BBs around pictures, small speckles, and printed margins. The remaining BBs were labelled as text. To guard against differences in image size, the coordinates of BBs for each document were [0,1] normalized. Dataset 1 was used to optimize thresholds in the pre-filtering stage whereas dataset 2

was used to optimize parameters  $\alpha$  and  $P$  in the local iterative relabeling stage. Dataset 3 was used to cross-validate the MLP and evaluate overall performance.

**Table 2: Datasets used for training and validation purposes**

<b>Dataset</b>	<b># images</b>	<b>% Text/Non-Text</b>	<b># BBs</b>
1	39	69/31	14,705
2	34	71/29	15,896
3	86	66/34	41,765

### 3.2.2 *Pre-filtering*

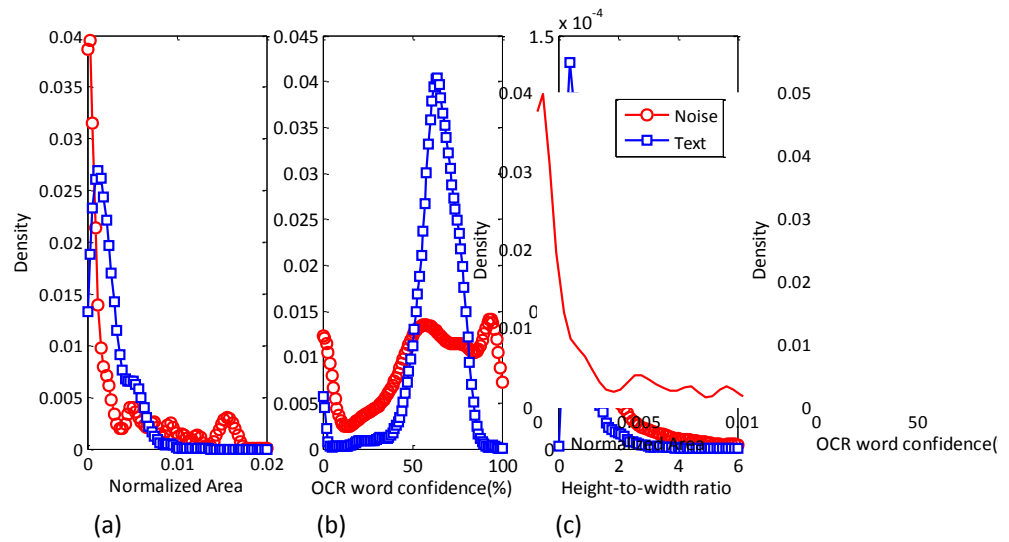
Figure 5 shows the distribution of features for noise and text BBs in the documents from dataset 1. The distribution of normalized areas in Figure 5a indicates that noise BBs tend to be smaller than text BBs, following our observations that Tesseract has a tendency to generate small spurious BBs whenever speckle noise is present in the image. Shown in Figure 5b, the distribution of OCR word confidence values for noise BBs is multimodal, with peaks near the extrema (0,1), whereas for text BBs it is normally distributed with a peak around 65% confidence. Finally, the distribution of H/W ratios in Figure 5c shows clear differences between the two types of BBs, with text generally having a much lower H/W ratio, as could be anticipated.

To optimize the threshold values for the three rules in the pre-filtering stage, we performed a receiver-operating-characteristic (ROC) analysis of the binary classification problem on dataset 1. Namely, we performed exhaustive search for the word confidence (two thresholds), height-to-width ratio and area thresholds (a 4-dimensional search space) to find the operating point with maximum F1-score on the precision-recall curve.

The derived rules were:

- **Rule 1:** If  $0 < C_{OCR} < 0.95$ , then **TEXT**
- **Rule 2:** If  $H/W < 2$ , then **TEXT**
- **Rule 3:** If  $A > 1^{\text{st}}$  percentile, then **TEXT**

which, when used as a conjunction, yield a F1-score of 0.93 (0.94 precision; 0.91 recall). Thus, pre-filtering can identify a significant number of noisy BBs, but it also mislabels a large proportion (9%) of text BBs in the documents. This is largely due to the fact that it does not consider information local to each BB, a problem that is handled by the last step in the process: local iterative relabeling.



**Figure 5: Feature distributions for BBs in dataset 1.**

### 3.2.3 Column extraction

The bottom panel in Figure 3 illustrates the horizontal distribution of BBs for one of the images in the collection. The limits for the two columns in the document are clearly indicated by troughs in the distribution. Figure 6 shows segmentation results for two additional and more challenging documents due to noise and skew.

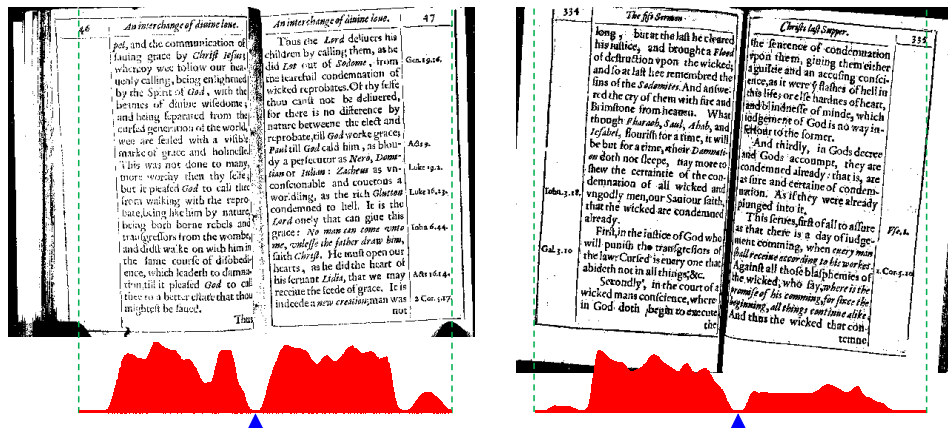


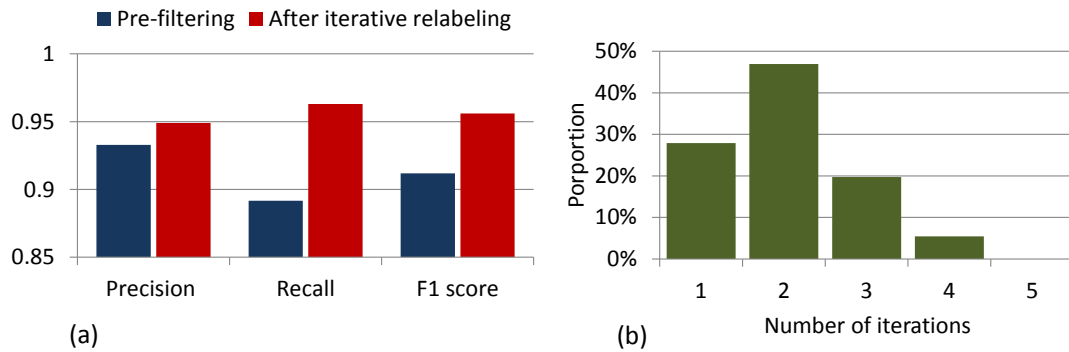
Figure 6: Column segmentation for two difficult test cases.

### 3.2.4 Local iterative relabeling

The MLP for the iterative process consisted of a hidden layer with 8 tangent-sigmoidal neurons, and 2 output neurons (i.e., one per class) with soft-max activation function to ensure MLP outputs could be interpreted as probabilities. The number of hidden units ( $N_H = 8$ ) was optimized through three-fold cross-validation over dataset3 with the F1-score as the objective function. Parameter  $P$  in eq. (1), the maximum number of neighbors, was set to 84 (21 per vertex), and parameter  $\alpha$  in eq. (2), was set to

10. These optimal values were extracted by minimizing the mean square error between  $S$  and the ground-truth label for all BBs in dataset 2.

We also performed three-fold cross-validation over dataset 3 to compare model performance before and after iterative relabeling. Results are summarized in Figure 7a; precision, recall and the F1 score improve when compared to pre-filtering results on dataset 3, with the largest gains obtained for recall (from 0.89 to 0.96). Figure 7b summarizes the convergence rate; in 95% of the cases the algorithm converges within three iterations.

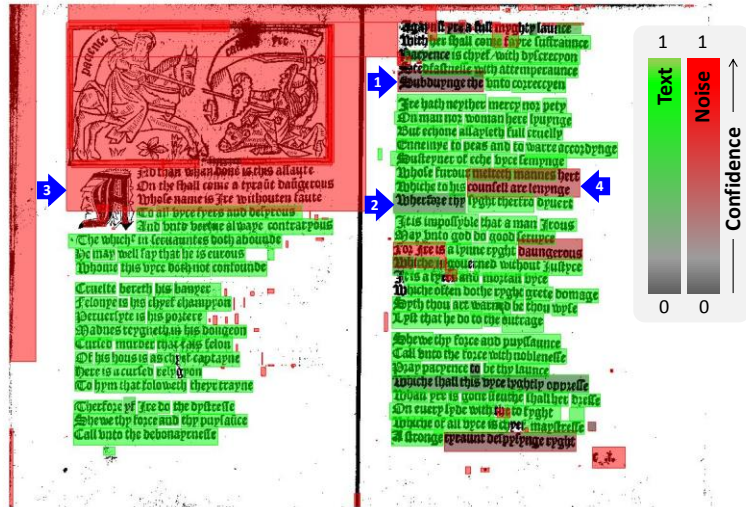


**Figure 7: (a) BB classification rate before and after local iterative relabeling; (b) Number of iterations required for convergence**

Figure 8 shows a document overlaid with the BBs returned by Tesseract. The fill color (green vs. red) represents the MLP prediction (text vs. noise, respectively), with higher color saturation denoting higher confidence; see color-bar insert. Arrows 1 and 2 illustrate two cases for which prediction was correct but the MLP had low confidence, hence the gray tone. Arrow 3 points to a BB that covers graphics and a decorative drop



cap, neither of which is likely to lead to a good OCR transcription. Finally, arrow 4 points to a BB that contains two lines of text; as such, the OCR transcriptions are likely to contain garbage.



**Figure 8: Iterative relabeling results for the image in Figure 1. Color denotes MLP confidence: the more saturated, the higher the confidence. Red: noise; green: text**

### 3.2.5 Deriving a measure of document quality

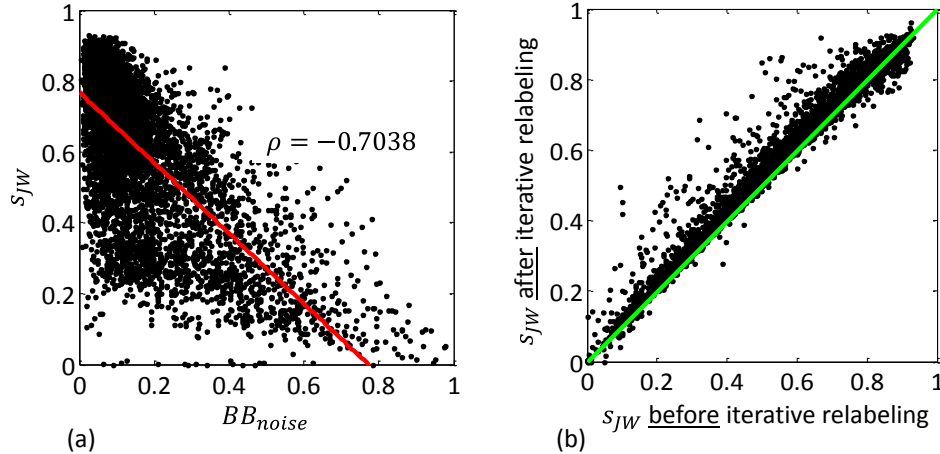
As shown in the previous subsection, the classifier can label BBs as text or noise with remarkable accuracy, which suggests that it may be used to estimate the overall quality of each document. Low-quality documents tend to produce a large number of spurious BBs, whereas high-quality documents produce mostly text BBs. Thus, the proportion of noise BBs returned by the OCR engine tends to be representative of the document's quality:

$$BB_{noise} = \frac{\# \text{ noise } BBs}{\# BBs} \quad (3)$$

We evaluated this quality measure on a large dataset of 6,775 document images from the EEBO collection that had manually-annotated transcriptions. For each document, we computed the similarity  $s_{JW}$  between the OCR output and the manual transcription:

$$s_{JW} = 1 - d_{JW} \quad (4)$$

where  $d_{JW}$  is the Jaro-Winkler distance [26], a measure of dissimilarity between the two text strings. For the purpose of this work, we used the ‘juxta’ command-line implementation of the Jaro-Winkler distance available in [juxtacommons.org](http://juxtacommons.org).



**Figure 9: (a) BB-based quality measure (BB\_noise) vs. the Jaro-Winkler similarity (s\_JW) for 6,775 documents. (b) s\_JW before and after iterative relabeling; for most documents (those above the diagonal line) iterative relabeling improved s\_JW**

Results in Figure 9(a) show a strong negative correlation ( $-0.704; p \ll 0.001$ ) between the proposed noise measure ( $BB_{noise}$ ) and the Jaro-Winkler similarity ( $s_{JW}$ ). Thus, as the proportion of noise BBs in a document increases, so do differences between OCR and manual transcriptions also increase. The significance of this result is that  $s_{JW}$  cannot be computed in practice since it requires the manual transcription, whereas  $BB_{noise}$  can be computed directly from the output of the OCR engine. As such, it may be used to automatically triage documents of poor quality and focus computational resources on those whose quality is more likely to generate good OCR transcriptions.

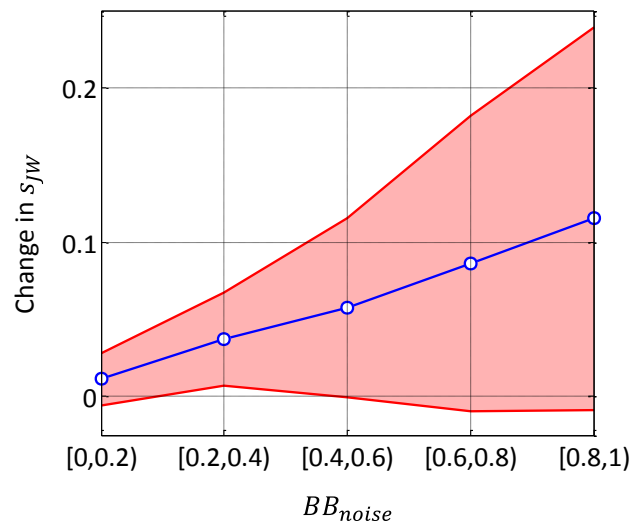
### 3.2.6 Improving OCR transcriptions

In a final step, we tested whether our algorithm could be used to improve the overall OCR performance. For this purpose, we ran the algorithm on the previous dataset (6,775 documents), removed any BBs labeled as noise, and computed  $s_{JW}$  between the resulting transcription and the manual transcription. Results are summarized in Figure 9b and Table 3. On 85.4% of the documents the algorithm improved  $s_{JW}$  (avg: +6.3%), whereas on 10.6% of the documents it lead to a decrease (avg: -3.0%).

**Table 3: Average change in Jaro-Winkler similarity ( $\Delta$ ) with application of the local iterative relabeling algorithm.**

	$\Delta > 0$	$\Delta < 0$	$\Delta = 0$
% documents	85.4	10.6	4.0
Avg. change	6.3	3.0	0.0

Lastly, we analyzed the impact of local iterative relabeling as a function of document quality; results are shown in Figure 10. Regardless of document quality ( $BB_{noise}$ ), local iterative relabeling increases the Jaro-Winkler similarity. These improvements are modest for high-quality documents (i.e., low  $BB_{noise}$ ), but become quite significant (up to 0.25) for documents of poor quality, where they are most needed



**Figure 10: Average change in Jaro-Winkler similarity as a function of document quality ( $BB_{noise}$ ).**

### 3.3 Discussion

In this section we have presented an approach to assess the quality of OCR using information about the spatial distribution and geometry of word BBs. The approach uses a pre-filtering step to initialize BB labels. From these, the document is segmented into columns by finding troughs in the horizontal distribution of BB coordinates. In a final

step, an iterative filtering algorithm is used to incorporate local information from neighboring BBs. When cross-validated on a dataset of 159 historical document images, the algorithm achieves 0.95 precision and 0.96 recall.

The pre-filtering step is designed to minimize false-positive rates since noisy BBs can compromise the subsequent column-segmentation step. As such, the pre-filter tends to miss short text BBs (e.g., short words such as ‘a’, ‘I’, ‘An’) since they violate rule 2. These initial labeling errors are corrected by the iterative relabeling algorithm, which also considers neighborhood information, the relative height of BBs relative to other BBs in the document, and their spatial location in the document. Relabeling generally converges within three iterations, a cost-effective investment considering the improvements in classification performance that it provides.

Tesseract assumes that the document image to be OCRed consists of only one column. When an image is passed through Tesseract, it first segments it into paragraphs, lines, and words. It then recognizes text for identified words on a page image. Since the EMOP collections have document images with multiple columns, the column segmentation step becomes essential for the analysis of the document images.

When evaluated on a collection of documents with manual transcriptions, the proportion of BBs labeled as noise ( $BB_{noise}$ ) shows a strong correlation with OCR performance, measured as the Jaro-Winkler similarity between OCR and manual transcriptions. As such,  $BB_{noise}$  may be used to triage heavily-degraded documents, allowing the OCR engine to focus on documents that have the highest chance of producing accurate transcriptions. Beyond triage, the spatial distribution of noise BBs

may be used to provide additional diagnostics for poor-quality documents and direct them to the appropriate process (e.g., rescanning, image denoising). As an example, salt-and-pepper noise tends to generate a large proportion of small BBs, graphics generally result in large and overlapping BBs (see Figure 8), and marginalia text (see Figure 6) can be detected by the presence of high-confidence BBs outside the text boundaries. This is particularly important in mass digitization efforts, such as early modern OCR project (eMOP) that motivates this work [1], where indiscriminate application of image restoration algorithms is prohibitive.

#### 4. FONT IDENTIFICATION USING ACTIVE LEARNING

Most documents in the eMOP collections are printed using variants of two font types: Roman or Blackletter; see Figure 11. In this section, we describe a method to classify documents into either class (Roman, Blackletter) or Mixed—documents that contain both fonts. Figure 12 illustrates the sequence of steps for the font identification method. We start the training process by selecting a set of seed training images from the ECCO/EEBO collections. We then feed these document images to Tesseract to generate the corresponding hOCR files. Next, we denoise the hOCR output by passing it through the OCR quality assessment module (section 3) to produce denoised hOCR. Denoising the hOCR output before the font identification is important since any kind of page noise may give false cues regarding the font present on a document image. We use the filtered hOCR and the document image to extract image features to be used by the font classifier. We train the classifier iteratively: after each round of training the output of the font classifier is used to perform active sampling on the document collection to select a few informative documents. We then present the selected documents to the user for tagging and include them in the labeled dataset for the next round of training. This training-tagging process is repeated until a performance criterion is met, e.g., a target precision/recall rate is reached. We evaluate the font identification system on a dataset containing a mix of documents, printed in Roman and Blackletter fonts, selected from the eMOP collections.

In the following subsections we discuss the proposed method in detail. First, we describe the characteristics of historical fonts and the features that we use for font recognition. We then discuss the learning algorithm we use for building the font classifier. Next, we present active sampling strategies, which we use to select the most informative unlabeled data instance for the user to tag. We conclude by presenting and discussing the results for the developed method.

Blackletter	Roman
contrary	Wherein
Good Sir	declaration
assigned	Creature
forgive	Sentiments

Figure 11: Example word images for Blackletter and Roman fonts.

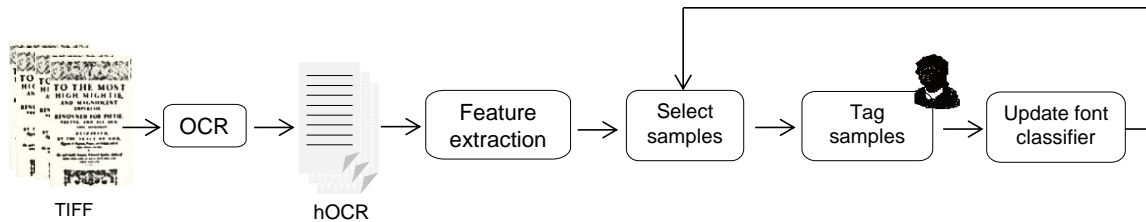


Figure 12: Block diagram for active learning based font identification system.



## 4.1 Feature extraction from document images

### 4.1.1 Font characteristics

A font (typeface) is a design for a set of characters. Fonts are classified based on differences in their anatomy such as strokes (curved or straight lines), stem width, type of ascenders and descenders, type of serifs and spacing between consecutive characters; see Figure 13. For more details on the anatomy of fonts please refer to [27]. Blackletter (old English or Gothic) and Roman font classes are the two main types used in early modern printing. Since the first printed book these font classes have been evolved into multiple subclasses, all of which are present in the EEBO and ECCO collections. In this thesis, we aim to recognize the font class for a document; hence, we need to first understand the key differences between Blackletter and Roman font classes.







 <b>Ascender: upward vertical stroke.</b>	 <b>Horizontal stroke.</b>	 <b>Descender: downward angled stroke.</b>
 <b>Angled stroke.</b>	 <b>Serif: Non-structural details at the end of some strokes.</b>	 <b>Stem: Primary vertical strokes.</b>

Figure 13: Some anatomical characteristics of a font class [27].

Blackletter typeface has tall, narrow letters compared with other fonts. Blackletters are usually formed by sharp, straight, angular lines and they do not connect with each other especially in round letters (see Figure 14a) [28]. They have drastic differences between thick and thin strokes, and most of the letters possess diagonal, thin serifs. Within each word the between-letter spacing is very small that makes text written/printed with this font difficult to read; see Figure 15a and 5b.

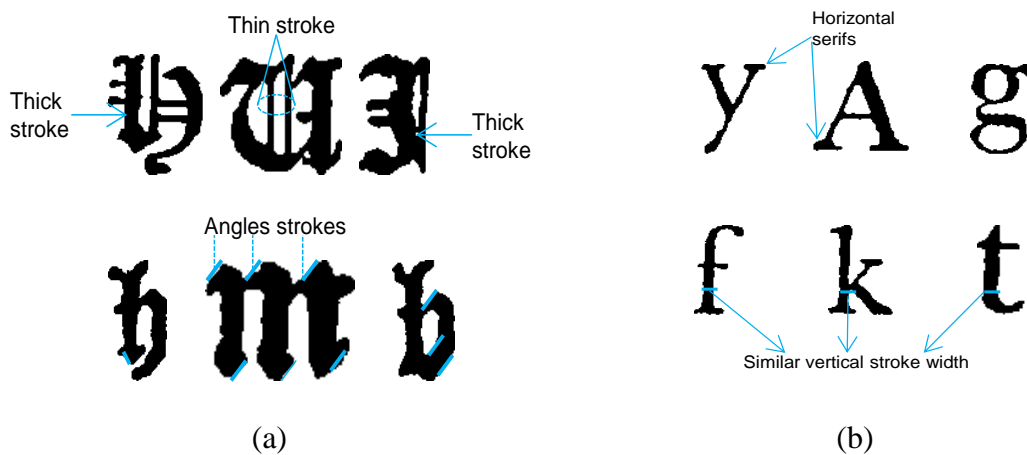


Figure 14: (a) Anatomical characteristics of Blackletter font; (b) Anatomical characteristics of Roman fonts.

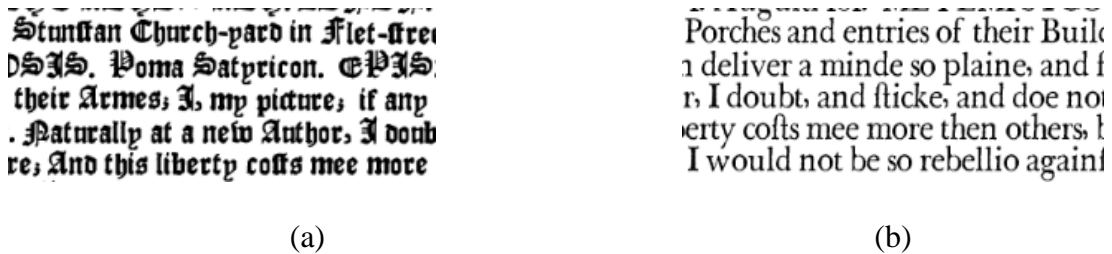


Figure 15: (a) Text snippet in Blackletter font; (b) Text snippet in Roman font.

Roman typeface is also popularly recognized as a serif typeface. It is called serif because of the small lines attached to the main strokes of characters; see Figure 14b. The historical styles of this font have curved angled strokes, and these angled strokes are also the thinnest parts of a character. The major differences with respect to Blackletter fonts are:

1. The Roman typeface gives less emphasis to the angled strokes than the Blackletter typeface.
2. Absence of the diagonal serifs in Roman typeface.
3. Within each word, Roman typeface exhibits uniformity in the vertical stroke widths of individual characters.
4. Roman typeface usually has thinner strokes.
5. Roman characters take space proportional to their shape. This is why they are very easy to read.

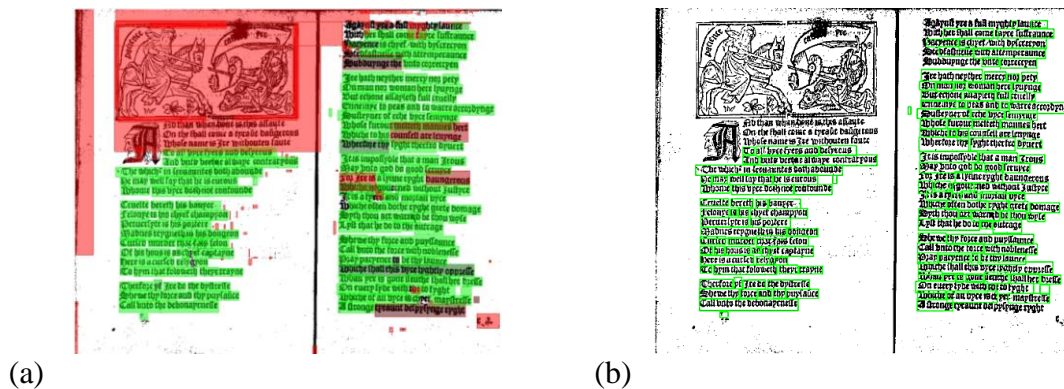
With the above differences between the Blackletter and Roman font classes, we can now easily differentiate between texts written in these fonts. However, the challenge is to represent these differences using features extracted from the denoised hOCR and the corresponding document image.

Figure 16 shows a document image overlaid with the denoised hOCR output. This shows that the bounding boxes (BBs) from the denoised hOCR can be used as a mask to extract word images from its document image. Since the developed system will be used to build a font database for 45 million pages, we make the choice of working at the word level instead of at the character level to extract features. Accordingly, we

process each word image from a document to extract the image features shown in Table 4. Since our goal is to identify font on a page image, we convert features extracted from the word images to page image features. For this, we use vector quantization to generate a bag-of-visual words feature representation of a page image [29]. The feature extraction process is explained in the following subsections.

**Table 4: Features extracted from word images.**

Word Features	Font characteristic
Mean stroke width	Roman fonts have smaller vertical stroke width than the blackletter.
IQR stroke width	To capture drastic differences in the stroke widths, which are the characteristics of black letter fonts.
Number of angled lines per character (slant line density)	To capture the amount of angled straight lines in a word image. Blackletters fonts are characterized by angled lines and serifs.
Zernike moments	To capture the overall shape (visual appearance) of the font present on a word image.



**Figure 16: (a) Output of the assessment algorithm, where red boxes denote predicted noise BBs and green boxes denote predicted text BBs; (b) Document image with predicted text BBs.**

#### 4.1.2 Document image preprocessing

Images of historical documents, especially the EEBO and ECCO digital collections, are binarized (i.e., as opposed to grayscale), are low-quality and have low-resolution, the result of digitization from microfilm converted from photographs –four decades and three generations away from the originals. The conversion of these documents to images has resulted into image problems such as skew and warping. In addition, aging effects can lead to various types of noise such as salt-and-pepper noise, bleed-through and black blotches due to torn pages. To avoid the effect of noise in the extracted features, we perform the following preprocessing on each word image of a document:

- a) Denoise hOCR: Using the denoised hOCR avoids noisy BBs around black blotches, table boundaries, musical scripts, pictures, decorative page elements etc; see Figure 16.
- b) Normalize the height of word images: We use the ‘imresize’ function from Matlab’s image processing toolbox to resize each word image to have the same height of 400 pixels [30]. In order to maintain the aspect ratio, the image width is chosen automatically by the resizing algorithm.
- c) Remove salt and pepper noise: Word images have pepper noise in the background and salt noise on the character strokes (Figure 17a) that can lead to incorrect estimation of character width. Hence, we perform a series of image morphological operations [31]. First, we perform erosion to remove background salt noise. Then we dilate the eroded image to reduce the pepper noise present in

the character strokes. This set of operations is called opening of an image by a structuring element (in our case it is a 3x3, all '1' element). Opening removes the noise but also introduces some unwanted artifacts, hence we also perform closing operation; See Figure 17.



(a)



(b)

**Figure 17: (a) Original word image with salt noise in the stems and pepper noise in the background; (b) Preprocessed word image after opening and closing operations.**

- d) Correct skew: We estimate the stroke width (details presented in the following subsection) by moving in a row of the word image and counting the number of continuous black pixels. However, if the word image is skewed, the method can over-estimate the stroke width. This may result into false cues about the type of font in which the word is written/printed. We correct the skew by calculating a

time-frequency distribution (Wigner-ville distribution) for different skew angles; we use the Matlab's 'wvd' function to calculate the distribution. The angle at which the distribution shows a peak is the estimate of the skew in the word image. Details of this method are included in [32].

#### 4.1.3 Mean and IQR stroke width for a word image

To estimate the mean stroke width and IQR (interquartile range of the stroke widths), we scan 10% rows (41 rows; middle row  $\pm$  20 rows) of the preprocessed word image to store two type of locations: 1) background to foreground transition points that mark the left boundary of the character stroke (point A, shown as green points in Figure 18); and 2) foreground to background transition points that mark the right boundary (point B; shown as red points in Figure 18). Difference in the x-coordinates of point A ( $x_A$ ) and point B ( $x_B$ ) serves as an estimate of the stroke width. In a single row multiple such stroke widths are found; we store count ( $C_i$  is the count for  $i^{\text{th}}$  row) of the stroke widths for each of the 41 rows. We then find the maximum of these  $C_i$ 's as shown in eq(5).

$$C_{max} = \max_{i=1 \text{ to } 41} C_i \quad (5)$$

where  $C_i$  is the count for  $i^{\text{th}}$  row;  $C_{max}$  is the max of  $C_i$  taken over 41 rows (middle row  $\pm$  20 rows). Next, we select rows that have  $C_i$  equal to  $C_{max}$  and calculate trimmed mean and IQR over the stroke widths found along the selected rows.



**Figure 18: An example showing an intermediate step in the calculation of mean and IQR character width.**

The advantage of using trimmed mean and IQR is that these statistics are robust to outliers, which can occur due to noise left after preprocessing the word image. Calculating statistics over the max-evidence rows is advantageous because it provides significantly large number of stroke widths over which the calculated trimmed mean and IQR are more accurate. Since we work on word images extracted using the BB as a mask, and the BB is the smallest rectangle that encloses a word, we can safely assume that most of the stroke widths can be found along the rows near middle of the word image. Hence, we scan rows whose y-coordinates lie between (middle-20) to (middle +20) – only 10% rows are processed. Restricting the y-coordinates helps in reducing the overhead of scanning the entire word image, and since we are dealing with huge amount of data (45 million page images with approximately  $45 \times 200$  million words), extracting features quickly is also one of the priorities of this thesis.

#### 4.1.4 *Slant line density*

Since Blackletter font shows heavy use of angled strokes, we try to capture that by performing a Hough transform [33] on the word image. The Hough transform is a powerful method which automatically analyzes a digital image to detect simple shapes



such as lines, circles or ellipses. The transform is used in many applications such as image segmentation, skew detection, and feature extraction in more complex computer vision problems. The Hough transform works best on edge images, usually calculated by applying edge operator such as Canny [31]. The edge image gives us points which lie on a desired curve in the image. The Hough transform groups these edge points into objects (e.g. straight line) by performing an explicit voting procedure over a set of parameterized image objects. An example of Hough transform applied to a word image is shown in Figure 19.



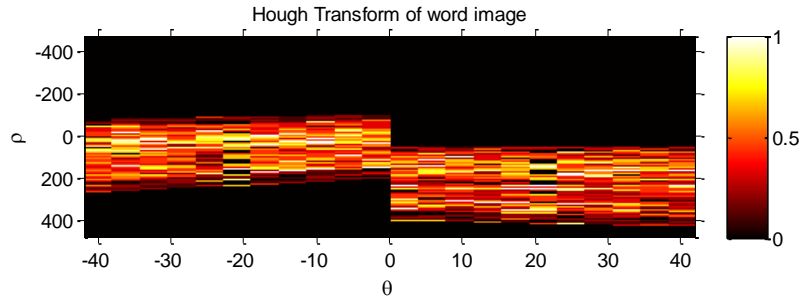
**Figure 19: An example of Hough transform applied to a word image. Here, the green line segments are the detected angled straight lines.**

For the font recognition problem, we focus on estimating number of straight lines in a word image with slope between  $45^\circ \pm 5^\circ$  and  $-45^\circ \pm 5^\circ$ . The block diagram for the feature extraction process is shown in Figure 20. We begin by extracting the edge image from the preprocessed word image. Once the edge image is extracted, we apply the Hough transform to detect straight lines. The Hough transform outputs a matrix where value at a cell gives the count of the number of lines with a particular slope and

intercept; rows of the output matrix correspond to the intercept of straight lines and columns represent slopes; see Figure 21. Next, we perform a filtering operation that selects significant lines (whose counts are greater than a threshold) of length greater than 7 pixels. Finally, we calculate the number of remaining lines and divide it by number of recognized characters in the word image; we get the number of recognized characters from the hOCR output. We estimate the slant line density as the number of filtered straight lines per character in a word image.



**Figure 20: Pipeline for slant line density.**



**Figure 21: Visualization of the Hough transform output matrix, which represents count of the straight lines upon which the length of the perpendicular from the origin is  $\rho$  and the perpendicular makes an angle  $\theta$  with x-axis. In this figure color saturation represents the fraction of points (in the image) that lie on the line specified by certain  $\rho$  and  $\theta$ .**

#### 4.1.5 Zernike moments

The features discussed till this point capture structural differences, such as stroke structure and connection style between the Roman and Blackletter fonts. In this subsection, we explain the shape descriptors (Zernike Moments) that we use to capture the visual appearance of the text (words). Figure 15 shows a sample set of words printed in Roman and Blackletter fonts. We can see that both types differ in the shape of the individual characters and the way these characters are grouped into words.

The Zernike moment is a special class of geometric moments. Geometric moments became very popular in computer vision following a seminal contribution by Hu' [34], which showed that for an image  $I(x, y)$  one can set one-to-one correspondence to a unique set of moments (commonly referred to as variance and mean). A geometric moment is defined as:

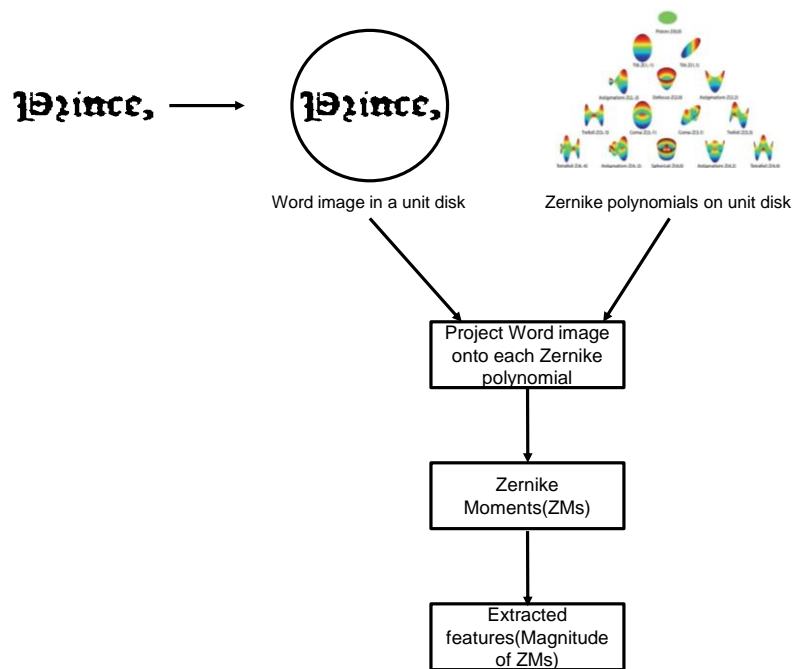
$$m_{p,q}(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q I(x, y) dx dy \quad (6)$$

where  $x$  and  $y$  are the coordinates in the image space, and  $p$  and  $q$  are the order of the moment. Usually, these geometric moments are calculated w.r.t the centroid of the image so that they become translational invariant; see eq (7).

$$\mu_{p,q}(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x - \bar{x})^p (y - \bar{y})^q I(x, y) dx dy \quad (7)$$

where  $\bar{x}$  and  $\bar{y}$  are the coordinates of the image centroid, and  $p$  and  $q$  are the moment order. Centering facilitates interpretation because moments are represented in terms of deviation from the centroid. For instance, higher order central moments capture information that is farther away from the centroid. For recognition purposes, Hu [34] derived seven moment invariants that are invariant under translation, similitude, and

orthogonal transformation that involves rotation and reflection. We can derive higher order moments, but they may contain redundant information and also capture large amount of noise. In addition, their extraction poses high computational overhead because all lower order moments need to be extracted first. These issues with higher order geometric moments make them infeasible for use in problems with large datasets.



**Figure 22: Zernike moment extraction process.**

The Zernike moments (ZMs) are the projections of the image on some orthogonal basis functions, known as Zernike polynomials (ZPs), which are defined on a unit disk and are orthogonal to each other; see Figure 22. Using these ZPs, ZMs are defined as:

$$Z_{m,n} = \frac{n+1}{\pi} \sum_x \sum_y I(x,y) V_{nm}^*(p, \theta) \quad (8)$$

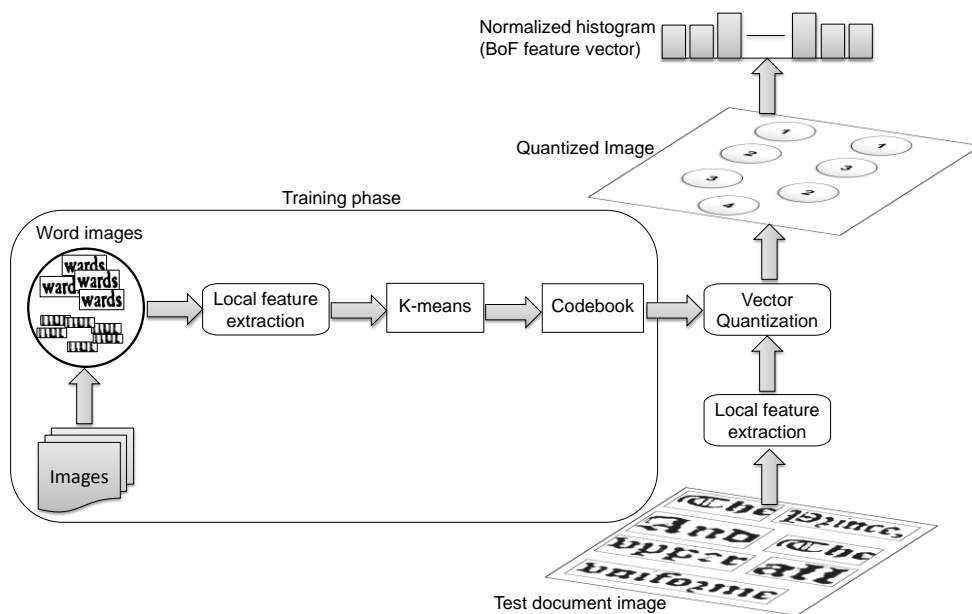
where  $x$  and  $y$  are the coordinates in image space,  $V_{nm}^*$  are the ZPs;  $m$  and  $n$  are the order of the ZM. The orthogonal property of ZMs allows calculation of higher order ZMs without having to calculate low order ZMs. Also, each moment captures unique information about the shape present on an image, which is crucial for good shape description. In our work, we calculate magnitude of first 6 ZMs along with their transformations (total of 15 ZM features) similar to the ones used in tumor classification problem by Tahmasbi et al. [35].

#### 4.1.6 Bag-of-words model

State-of-the-art image classification systems use bag-of-word features (BoF) [29] to represent images as a histogram of their local features, usually extracted from small square patches in the image. BoFs are used for capturing semantic information in the image, and are being widely used in problems such as texture classification and object classification. In our work, we consider word images as the image patches, and the local features are the 18 features (mean and IQR stroke width, slant line density and 15 ZMs) extracted from these patches. In font identification, BoFs do not capture any semantic information but rather the characteristics of font subclasses - variants of blackletter and roman fonts.

Figure 23 shows the steps involved in extracting BoFs for a page image. We first extract local features for all the word images on a page. We then cluster these local features using k-means [36] and the set of cluster centers forms a codebook. We use the

codebook to vector quantize each word of a page image, which assigns each word image to a cluster center. Finally, we generate the BoF by counting the number of quantized words assigned to each cluster center of the codebook. The page images from eMOP databases have different word count. To achieve word count invariance, we divide the BoF for a page image with the total number of words in the page.



**Figure 23: Block diagram explaining bag-of-words features (BoF).**

## 4.2 Active learning to build font identification model

The fundamental way to classify pattern vectors is to learn a classifier such as SVM, random forest and neural networks. All of these classifiers are learnt using training data; for example, in our case we need to build a training dataset with documents labelled as Blackletter, Roman or Mixed. A better classifier can be learnt by

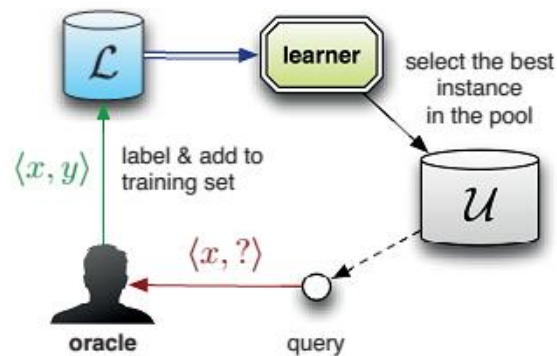
using a larger training dataset, but the process of labelling by hand requires extensive human labor. Hence, in this thesis we use active learning [37] methods to train a font classifier while minimizing the number of training examples that need to be manually labeled.

Active learning is a learning paradigm where an active learner (e.g. a classifier) acquires its own training data by querying the labels for few selected examples from an Oracle (e.g. human annotator). To select the examples for querying, the most informative instances are sampled from the unlabeled data. Active learning is most appropriate in scenarios when the unlabeled data are in abundance, and there is a need for a large labeled dataset to training a classifier accurately. Active learning helps to achieve high classification accuracy with few labelled examples [38]. For font identification, we aim to build a classifier that can work for 45 million document images from the eMOP databases; hence, we use active learning to build a robust classifier using as few labeled document images as possible.

According to the source of unlabeled data, active learning methods are generally divided into two categories: 1) pool-based methods, and 2) stream-based methods. In the pool based methods, the learner has access to all the unlabeled data at any time, whereas in stream based methods the unlabeled data arrive sequentially. For our font identification problem, we use a pool-based active learning method; see Figure 24.

Formally, in pool-based active learning there is a pool of unlabeled data (i.e.  $n$  document images)  $U = \{x_1, \dots, x_n\}$ , where  $x_i$  is a BoF feature vector for the  $i$ -th document. Each instance  $x_i$  can have a label  $y_i \in \{Blackletter, Roman, Mixed\}$ .

Another set of documents images ( $L$ ) is present that consists of  $\{x_i, y_i\}$  pairs (i.e., labeled examples).



**Figure 24: Pool based active learning [37].**

Typically, an active learning method comprises of two parts: a learning engine and a sampling engine. To begin with, a base classifier is trained on the small amount of labeled data ( $L$ ) available. Next, the sampling engine uses the trained classifier to select informative instances ( $X$ ) from  $U$ . Selection is done using a selective sampling algorithm [37] that uses certain utility measure to evaluate all instances in  $U$ . Then, the human annotator labels all instances in  $X$  and these labeled instances are added to  $L$ . The learning engine retrains the classifier using the updated labeled dataset. The whole process of training and sampling is repeated until a performance criterion is met, e.g., a target precision/recall rate is reached.

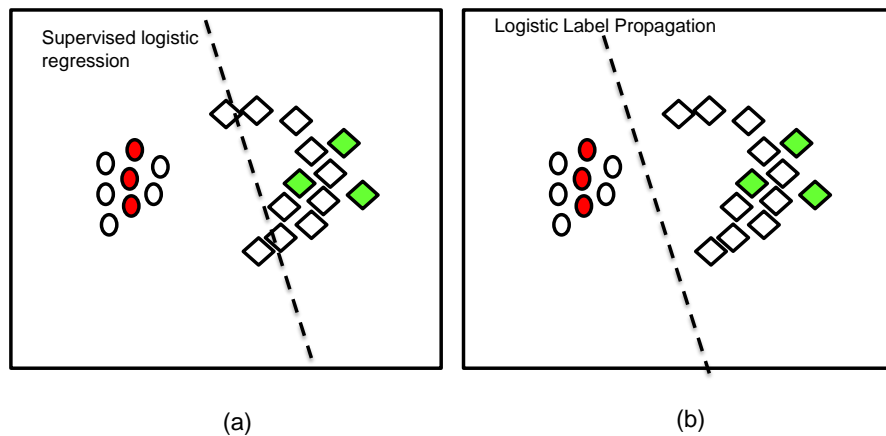


#### 4.2.1 *Choosing base classifier*

In an active learning (AL) setting, any of the strong supervised classifiers such as SVM, random forests, and neural networks can be used. Since these are supervised learning algorithms, they just use whatever limited labelled data is available, and ignore the vast amount of unlabeled instances. Instead, using a semi-supervised classifier as the base classifier can help in building better classification models when labeled data are limited [39]. Among these, label propagation (LP) is one of the most frequently used graph based semi-supervised algorithm [40]. In LP, all (labeled+unlabeled) data points (nodes) form a fully connected graph, and on this graph the labels are propagated to unlabeled data points according to their proximity (similarity) to the labeled data. In a nutshell, labeled data act as a source that transmits labels to the unlabeled instances. The propagation is done iteratively until the class-posterior probability for each unlabeled data point stops changing. Predictions for unlabeled data points are then done by choosing the class with maximum class-posterior probability.

The major drawback of LP models is that they are computationally expensive in estimating label for a new data point. Since new data point are not part of the similarity matrix, LP reconstructs the whole similarity matrix after adding the newly arrived test point. It then re-estimates the class-posterior probabilities for all the unlabeled data to predict a label for the test data point. Since the aim of this thesis is to build a model that can be applied to predict font for a large dataset (testing phase), LP models cannot be used in this case because of the computational overhead. Hence, we use a modified LP model proposed by Kobayashi et al. [41] known as Logistic label propagation (LLP).

The LLP model employs logistic classifiers to estimate labels for a new test point. The cost function for LLP model has two parts: 1) the cost derived from label propagation to cope with the unlabeled samples in a semi-supervised manner, and 2) the negative log-likelihood as in logistic regression to measure classification errors across the labeled samples. As shown in Figure 25, when cost due to the unlabeled data is incorporated, the decision boundary for LLP shifts in order to properly classify unlabeled diamonds (diamonds misclassified by logistic regression). Since the LLP model estimates label for a new input based on logistic classifier, it has a very low computational cost in testing phase.



**Figure 25: Red and green points are labeled data; all other points are unlabeled data; dashed line is the decision boundary learned by a classifier. (a) shows the learned decision boundary for logistic regression; (b) shows the decision boundary for logistic labeled propagation [41]<sup>2</sup>.**

---

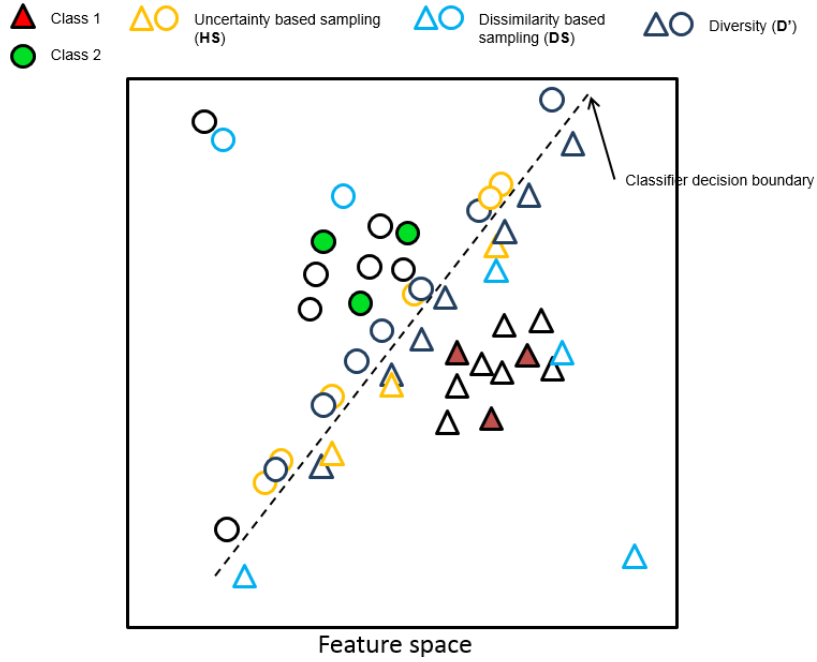
<sup>2</sup> Reprinted from Takumi Kobayashi, Kenji Watanabe, Nobuyuki Otsu, “Logistic label propagation”, Pattern Recognition Letters, Copyright (2012) with permission from Elsevier.

### 4.2.2 Active sampling

The performance of active learning (e.g. % labeled data needed to achieve desired accuracy) depends on the choice of the sampling strategy (query function) that selects the most informative unlabeled instances for manual labelling. For sampling documents in our work we use a query function that evaluates the importance (w.r.t. trained classifier) of an unlabeled sample according to three measures:

- a) **Uncertainty.** In uncertainty active sampling, unlabeled instances, for which the classifier is confused, are selected for querying.[37]; see Figure 26. The uncertainty measure promotes exploitation near decision boundary. In multi-class settings, entropy (eq. (9)) is one of the most popular measures of uncertainty. Entropy is defined on the class- posterior distribution  $P(y|L)$  where  $L$  is the labelled data and  $y$  is the label variable that ranges over all possible labeling of an unlabeled sample  $x$ .  $H(y|x)$  is high when the classifier outputs similar probabilities for all the classes, and is low when the classifier is highly confident for one of the classes. Since we use the LLP model, which outputs  $p(y|x)$  for each unlabeled instances, it is convenient to use entropy as the uncertainty measure.

$$H(y|U_k, L) = - \sum_y p(y|U_k, L) \log p(y|U_k, L) \quad (9)$$



**Figure 26:** The red and green examples are the initial labeled examples; yellow examples are the most informative examples based on uncertainty; blue examples are the most diverse examples from the current labeled data; Navy blue examples are the most uncertain and diverse set of unlabeled examples.

b) **Dissimilarity to the nearest labeled data.** Entropy based sampling just samples instances near the decision boundary. In order to promote exploration, it is useful to select instances that lie in unexplored regions of feature space; see Figure 26. For this, we use LLP model's similarity matrix, which calculates similarity between two documents  $i$  and  $j$  as follows :

$$S_{ij} = \exp\left(-\frac{\|X_i - X_j\|^2}{\sigma^2}\right) \quad (10)$$

where  $X_i$  and  $X_j$  are the features vectors for document  $i$  and  $j$ , and  $\sigma$  is the bandwidth hyper-parameter. For an unlabeled instance ( $U_k$ ), we first find the

5 nearest (most similar) labeled instances ( $L_n$ ) and store the similarity ( $S_{nk}$ ) value between  $L_n$  and  $U_k$ . The dissimilarity is then calculated as:

$$D_k = \sum_{n=1}^5 1 - S_{nk} \quad (11)$$

The samples with high dissimilarity values are selected for querying.

**c) Dissimilarity between unlabeled data.** Many active sampling strategies select just the most informative sample at each iteration. This is uneconomical because retraining is required in each iteration of the active learning algorithm. For our problem retraining after adding just one example is computationally very prohibitive because we are dealing with millions of document images. Hence, we use batch-mode active sampling, which selects  $h$  ( $h > 1$ ) most informative unlabeled samples at each iteration. Selecting the  $h$  most informative samples, based on uncertainty or dissimilarity, may result into selection of instances that are similar to each other. Hence, to incorporate diversity, we select  $h$  unlabeled samples that have high dissimilarity from the remaining unlabeled data; see Figure 26. For an unlabeled data point ( $U_{k'}$ ), we calculate dissimilarity using equation Eq. (11). We then select the 5 most dissimilar unlabeled instances to  $U_{k'}$ , and calculate the dissimilarity to the remaining unlabeled data as :

$$D'_{k'} = \frac{1}{n} \sum_{i=1}^5 D_{k'i} \quad (12)$$

where  $D_{k'i}$  is calculated using equation Eq. (11);  $n = 5$  which is the top  $n$  dissimilar unlabeled data points.

We use combination of the evaluation metrics ( $H(y|U_k)$ ,  $D_{nk}$ , and  $D'_{k'}$ ) to define a scoring function  $Score_{comb} = f(H(y|U_k), D_{nk}, D'_{k'})$  in each iteration of active learning, and use this function to sample  $h$  unlabeled instances with highest  $Score_{comb}$ . In the results section, we evaluate different  $f(H(y|U_k), D_{nk}, D'_{k'})$ .

## 4.3 Results

### 4.3.1 Dataset

To test the active learning based font identification system, we create a dataset consisting of binarized document images from the eMOP collections (ECCO and EEBO databases). The dataset includes documents printed in Blackletter font, Roman font and Mixed- document with text in both fonts. In total, the dataset contains 3,272 documents labeled by experts from eMOP team using a labelling tool based on Picasa [42] that was also developed as a part of this thesis.

Since most of the documents in the eMOP corpora consist of two pages (left and right), experts from eMOP provided the labels shown in Table 5. In order to get labeled dataset with labels belonging to the three categories- Blackletter, Roman and Mixed, we label a document image as being a Mixed document if its right and left page are labelled with different fonts. Table 6 shows the summary of the dataset generated.

**Table 5: List of labels used to annotate document images using Picasa.**

<b>Label</b>	<b>Meaning</b>
Left-black	Left page printed in blackletter font
Right-black	Right page printed in blackletter font
Left-roman	Left page in roman font.
Right-roman	Right page in roman font
Black	Both pages are printed in blackletter font
Roman	Both pages are printed in roman font
Mixed	Both pages consists of test printed in blackletter and roman fonts

**Table 6: Number of labeled data for different classes.**

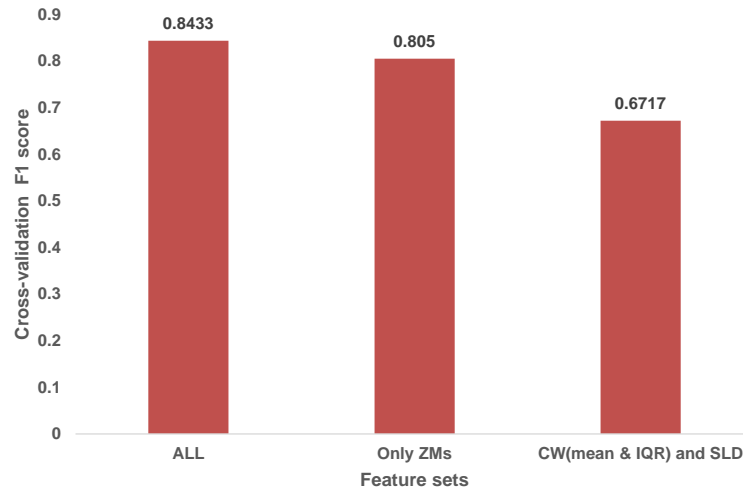
<b>Class</b>	<b># of document images</b>
Blackletter	1005
Roman	1768
Mixed	498

#### 4.3.2 *Feature extraction from document images*

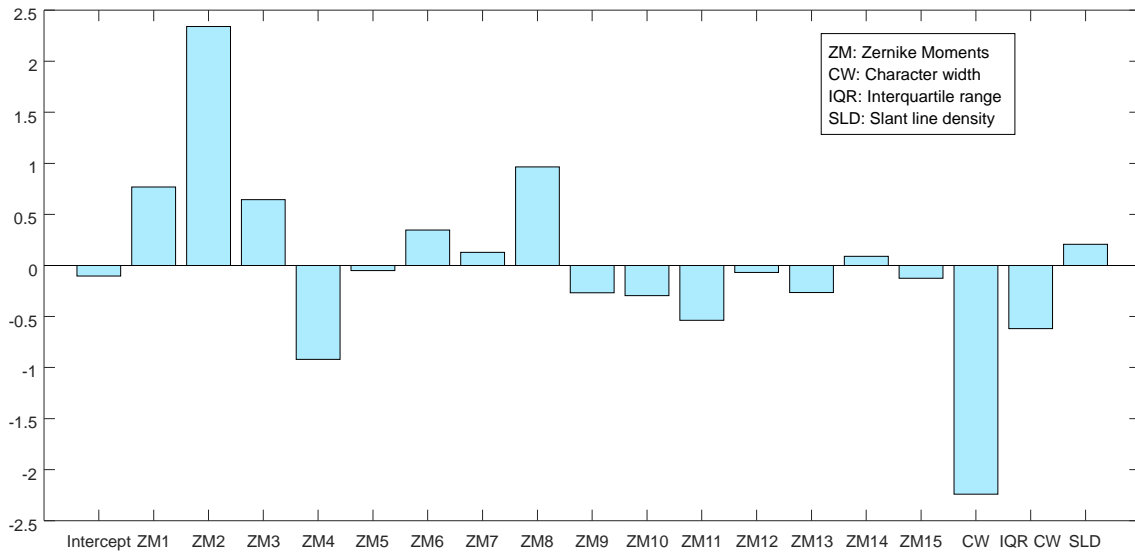
To evaluate the effectiveness of the extracted features, we conduct an experiment to evaluate their class separation capability. In this experiment, we train a classifier using 18 features (see Table 4) to predict font class (Blackletter or Roman) for a word image. We first create a dataset by randomly selecting 500 Blackletter documents and 500 Roman documents, and extract the 18 features (Table 4) for all the words present in the documents. To generate ground truth for each word image, we label all the word images from Blackletter documents as class-1 (Blackletter word) and all the word images from Roman documents as class-2 (Roman word). We then train a classifier using Logistic regression. A logistic regression model outputs, for each input word image, a score between [0 1] that can be compared against a threshold to obtain a class label. For selecting the best threshold, we perform 5 fold cross validation and select the threshold

for which we get maximum cross-validation score F1-score (84.18%; threshold=0.5). Each logistic regression coefficient represents a change in log-odds when the corresponding feature value is changed by 1 unit. Hence, we use logistic regression coefficients to assess quality of features; larger the coefficient, more important a feature is for font identification. The mean logistic regression coefficients over the 5-folds are plotted in Figure 28. We see that all orders of Zernike moments have significant effect on the log-odds. Among the mean character widths, IQR character width and slant line density, the mean character width has the maximum effect on the log-odds. Apart from mean logistic regression coefficients, we also store the cross-validation F1-scores for the classifiers that we train using all (18 features), only Zernike moments, and only slant line density and mean & IQR character widths features. The cross-validation F1-scores for all the three classifiers are shown in Figure 27. The score for the classifier trained using all 18 features is highest. Hence, we conclude from the experiment that the extracted features (Table 4) are good representative of the font classes and also shows good class separability.





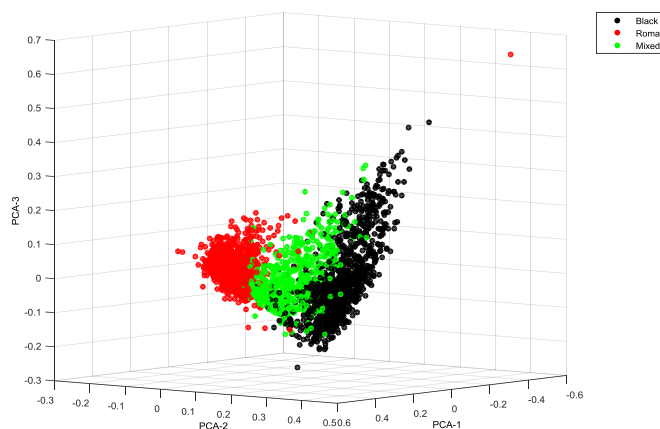
**Figure 27: Corss-validation F1-score for the classifiers trained using all 18 features, only Zernike moments (ZMs), and only character width (CW) mean & IQR and slant line density (SLD).**



**Figure 28: Coefficients of logistic regression (value averaged over 5 folds).**

### 4.3.3 Bag-of-word features

In order to evaluate the class separability of the Bag-of-word features (BoF), we visualize all the 3272 document images color coded by their class labels. For this, we first reduce the dimensionality of BoF from 20 to 3 by performing principal component analysis (PCA). We then plot all the document images in the 3-dimensional PCA space. Figure 29 shows the scatter plot where red circles represent Roman documents, black circles represent Blackletter documents, and green circles represent Mixed-font documents. From this figure, we see that the Roman class forms a compact cluster whereas the Blackletter class shows large intra-class variability. This could be because the Blackletter fonts were used to imitate handwritings hence these fonts vary from one book to other and show larger variability compared with the Roman fonts. However, we see that the three classes have good separability, hence we use BoF for building model to classify document images into fonts present on them.



**Figure 29: Scatter plot in PCA space where each dot is a document color coded by its class label.**

#### 4.3.4 Evaluation set-up for the active learning

To evaluate the font identification system, we randomly select a set of 600 documents (200 from each class) as the test set, and use the remaining 2672 documents as the training set to train the font classifier using active learning. To start the experiment, we randomly select a set of 3 labeled documents (from the training set and one from each class) as a seed-set to train the font classifier (LLP). The remaining 2669 documents constitute the unlabeled dataset. After training the font classifier, the active learning algorithm picks the 20 most informative documents from the unlabeled dataset using  $Score_{comb}$ , and queries their labels from an oracle. In our evaluation, instead of asking human to label, the active learning algorithm automatically picks the labels from the training dataset. We then re-train the font classifier after adding these new labeled documents to the previously chosen seed-set, and measure the performance of the re-trained classifier on the test set. We repeat this process of selecting informative instances, querying labels and retraining until all unlabeled data get consumed. At each iteration, we record the size of the labeled dataset and the test accuracy.

As a baseline, we perform a second experiment where, instead of selecting the most informative documents, we randomly select a set of 20 documents at each iteration. We again measure the performance of the retrained classifier on the test set after every iteration, and repeat the process of sampling, querying labels and retraining until unlabeled data get fully used. We repeat the two experiments 20 times and calculate

average test accuracy at each iteration. In next subsection, we present the results of the experiments for different  $Score_{comb}$ .

#### 4.3.5 Active sampling

In section 4.2.2, we discussed three ways of ranking unlabeled data for active sampling: uncertainty measure ( $H(y|U_k, L)$ ), dissimilarity to the nearest labeled data ( $D'_{k'}$ ), and dissimilarity between unlabeled data ( $D_{nk}$ ). When we evaluate the active learning using the set-up defined in section 4.3.4, in each iteration we perform active sampling by first ranking the unlabeled data, according to some combination of the above mentioned three measures, and then select the top ranked 20 unlabeled examples for labeling. In this subsection, we test the performance of the combinations (of the three measures) that we use for active sampling. We run the evaluation set-up to test the following combinations:

- 1)  $Score_{Exploitation} = H(y|U_k, L)$ . Here, we just use class entropy (uncertainty measure) to rank the unlabeled data.
- 2)  $Score_{Exploration} = D_{nk}$ . Here, we use dissimilarity from the labeled data to rank the unlabeled data.
- 3)  $Score_{sum,diversity} = H(y|U_k, L) * D'_{k'} + D_{nk}$ : We sample a batch of most informative instances that are as diverse as possible from remaining unlabeled data. Hence, we take product of  $H(y|U_k)$  and  $D'_{k'}$  (diversity factor). We also add an exploration factor ( $D_{nk}$ ).
- 4)  $Score_{sum,no\ diversity} = H(y|U_k, L) + D_{nk}$  with no diversity factor.

5)  $Score_{Sum-Diff, no\ diversity} = \frac{H(y|U_k, L) - D_{nk}}{H(y|U_k, L) + D_{nk}}$  with no diversity factor. Here, we

give more weightage to the unlabeled instances which are close to the decision boundary and lie in the dense part of the labeled data (low  $D_{nk}$ ). We achieve this by taking difference of the two measures ( $H(y|U_k, L), D_{nk}$ ).

6)  $Score_{Sum-Diff, diversity} = \frac{H(y|U_k) * D'_{k'} - D_{nk}}{H(y|U_k) * D'_{k'} + D_{nk}}$ . It is similar to score in 3, but with

diversity included.

7)  $Score_{weighted\ sum, no\ diversity} = 2 * H(y|U_k) + D_{nk}$  with no diversity factor.

Here, we give more weightage to samples with high entropy ( $H(y|U_k)$ ).

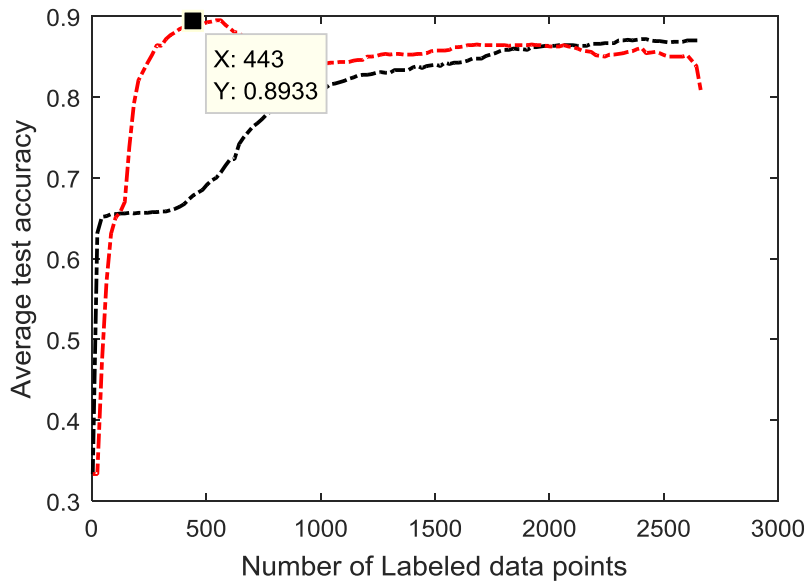
8)  $Score_{weighted\ sum, diversity} = 2 * H(y|U_k) * D'_{k'} + D_{nk}$  weighted score with diversity.

9)  $Score_{random\ exploration\ and\ exploitation} = \begin{cases} H(y|U_k) * D'_{k'} , & \text{when } p > 0.5 \\ D_{nk} , & \text{when } p \leq 0.5 \end{cases}$

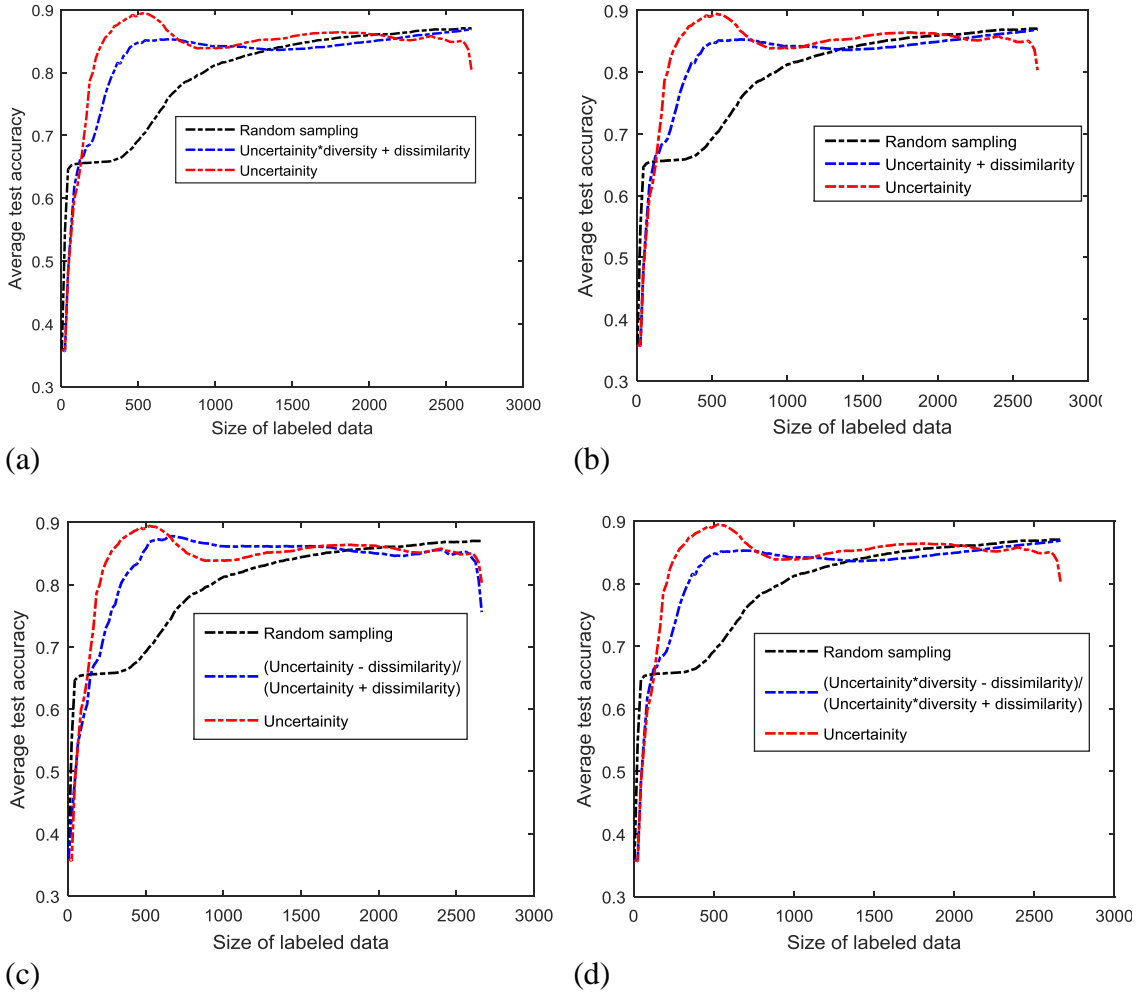
where  $p$  is random number drawn between  $[0,1]$ .

For the evaluation, bandwidth parameter for LLP model is set to 300. As explained in section 4.3.4, after the evaluation is over we calculate mean (over the 20 repetitions of the experiment) of test accuracy for each iteration and plot it to get learning curve as shown in Figure 30. Similarly, we calculate learning curve (Figure 31) for each combination (active sampling strategy) and then find area under the curve (AUC) for the learning curves corresponding to different sampling techniques; the higher the AUC the better is the active sampling. Figure 32 shows the bar plot, where each bar represent the AUC for each sampling technique and the red line is the AUC for

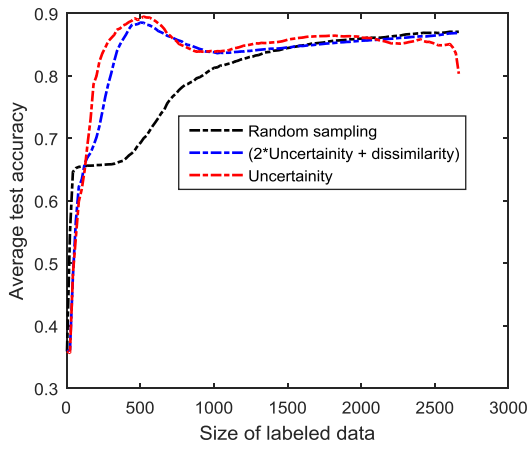
random sampling. As shown, active sampling based on  $Score_{Exploration}$  performs worse than random sampling. It has been shown that dissimilarity based active learning works well in complex classification problem such as XOR, but gives poor performance in problems with simple structure [43]. In our case, Figure 29 shows that the data for our problem have a simple structure, which can be a possible reason for the poor performance of the dissimilarity-based active sampling. All other sampling techniques, which combine both dissimilarity and uncertainty, have similar performances. The best performer is  $Score_{Exploration}$  based active sampling that uses 443 labeled data to achieve a test accuracy of 89%.



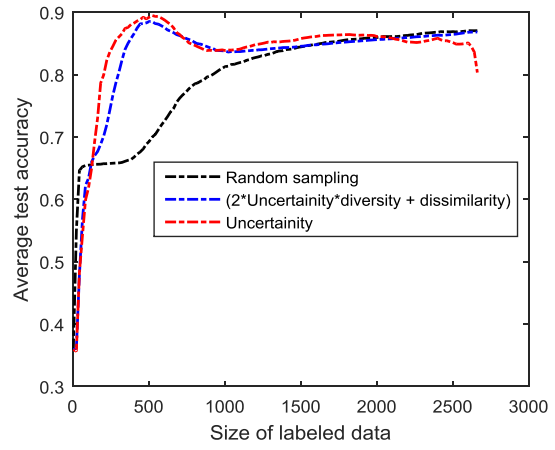
**Figure 30: Learning curve (Red) for entropy ( $Score_{Exploitation}$ ) based active sampling; Black curve represents the performance of random sampling.**



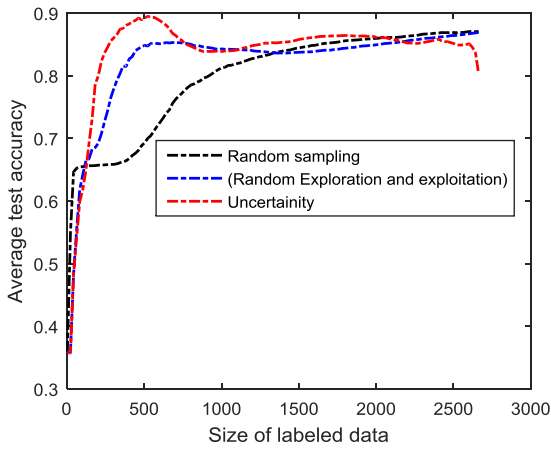
**Figure 31: Learning curve (Red) for entropy ( $Score_{Exploitation}$ ) based active sampling. Black curve represents the performance of random sampling. (a) Blue learning curve for  $Score_{sum,diversity}$ ; (b) Blue learning curve for  $Score_{sum,no\ diversity}$ ; (c) Blue learning curve for  $Score_{Sum-Diff,no\ divesity}$ ; (d) Blue learning curve for  $Score_{Sum-Diff,diversity}$ ; (e) Blue learning curve for  $Score_{weighted\ sum,\ no\ diversity}$ ; (f) Blue learning curve for  $Score_{weighted\ sum,diversity}$ ; (g) Blue learning curve for  $Score_{random\ exploration\ and\ exploitation}$ ; (h) Blue learning curve for  $Score_{Exploration}$ ;**



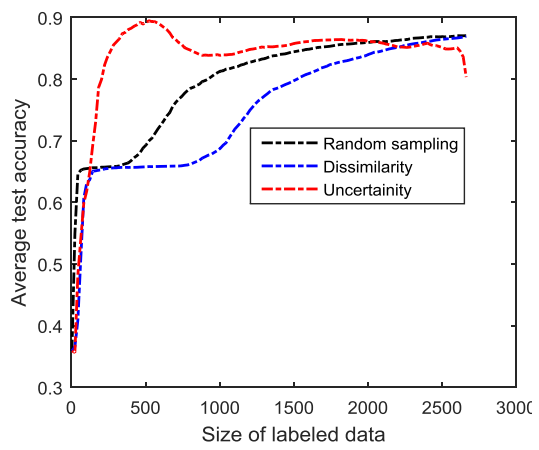
(e)



(f)



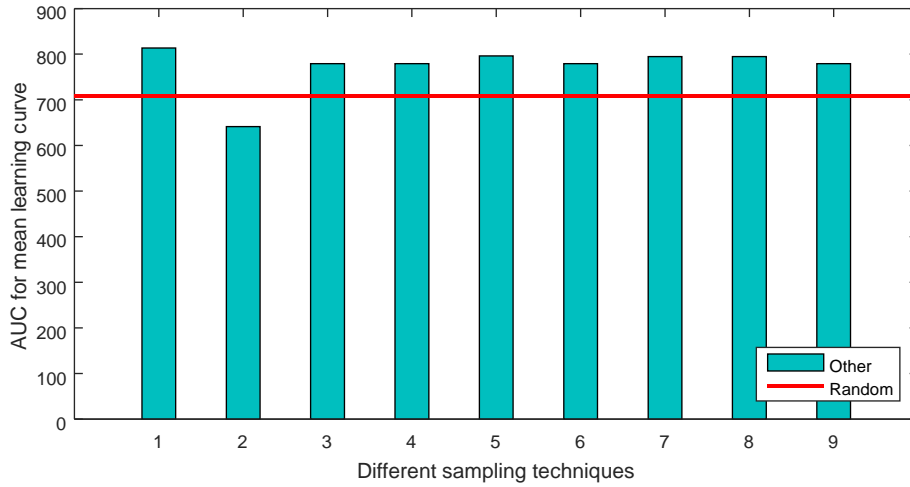
(g)



(h)

**Figure 33 continued.**





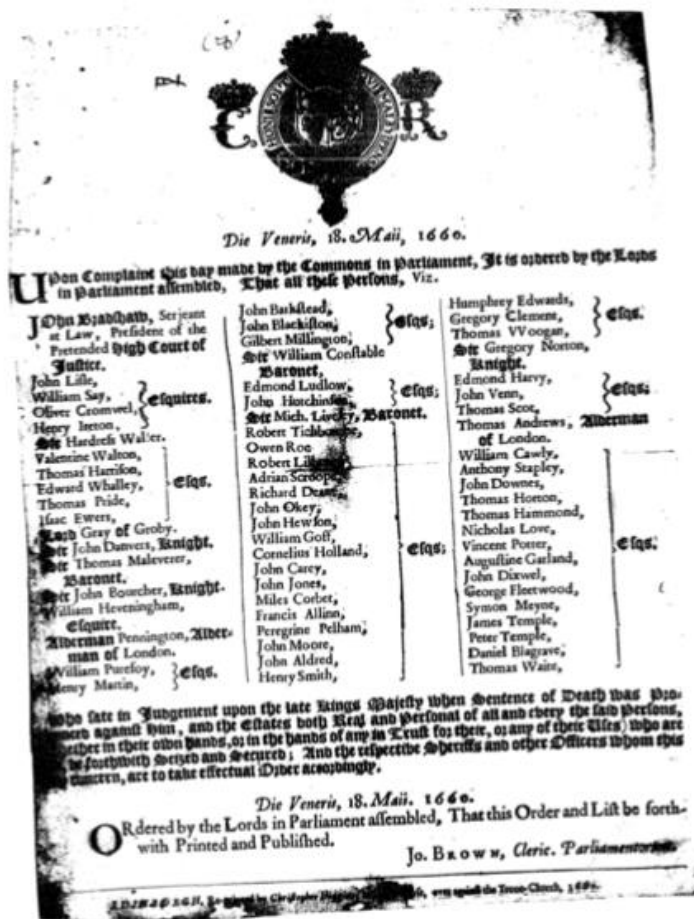
**Figure 32: Area under the learning curves for different sampling techniques (scores); {1,2,3,4,5,6,7,8,9} corresponds to the scoring functions.**

#### 4.4 Discussion

We have developed an active learning based font identification system for historical documents. We first critically analyzed the characteristics of different fonts (Blackletter and Roman), and designed features to capture them. The 18 features used to represent font characteristics have significant information for building a good font classifier. According to the results shown in Figure 28, the most significant feature is the mean character width followed by Zernike moments. Zernike moments prove to be important since they capture the visual appearance of the font classes. The bag-of-word features (BoF) provide good representation of different types of fonts present in the eMOP databases. The analysis of the BoF confirms its high class separability as the three classes (Blackletter, Roman, and Mixed) form compact clusters in the PCA space (Figure 29).

We used both the labeled and unlabeled data to train a logistic label propagation (LLP) model. We tested several active sampling techniques to select the most informative unlabeled instances for querying. For the problem of font identification, we showed that active sampling works better than random sampling. Furthermore we showed that uncertainty based active sampling technique performs best for our problem. Using uncertainty based active learning we achieved 89.33% test accuracy with only 443 labeled examples; see Figure 30.

When we analyzed the set of documents that are consistently misclassified during the active learning, we found that most of them belong to Mixed category. This could be because of the labelling errors in the dataset. In the dataset a document is labeled as a Mixed document when the proportion of Blackletter text and Roman text appeared to be equal visually. This could have resulted into false labels for certain documents that are then misclassified; see Figure 33a and Figure 33b. Another interesting misclassification result is shown in Figure 33c where a Roman document is misclassified into Mixed category document. The document contains italicized Roman words, which may have resulted into high slant line density that in turn can become the reason for misclassification.



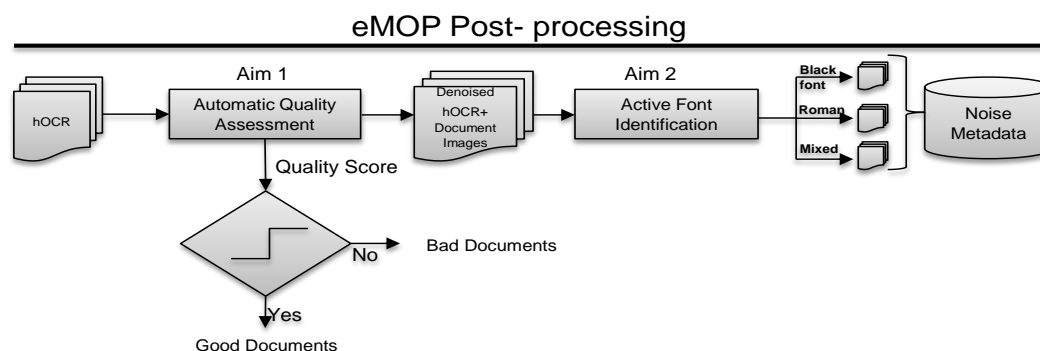
(a)

Figure 33: (a) A Mixed class document misclassified as a Blackletter documents; (b) A Blackletter document misclassified into Mixed class; (c) A Roman document classified as a Mixed document.



## 5. CONCLUSIONS AND FUTURE WORK

The challenges in mass digitization of historical documents need to be addressed in order to make these documents easily accessible. EMOP project has taken this initiative - to improve digitization quality of historical documents (printed between 1400-1800), by performing document triaging and then selectively processing on the historical documents.



**Figure 34: eMOP post-processing pipeline.**

The work presented in this thesis is being used in the eMOP post-processing pipeline [1]; see Figure 34. The quality assessment algorithm (Aim 1) is used to segregate documents into good and bad categories based on a certain threshold. In mass digitization projects, this reduces the overhead of re-OCR'ing the good quality documents and sends the bad quality documents for further processing. The active font identification algorithm (Aim 2) uses the denoised hOCR and document image to label/tag all the documents in order to build font metadata for the EEBO/ECCO

collections. This meta-data will be used to apply appropriate pre-processing steps needed for better recognition of a particular font.

Our results from the automatic quality assessment algorithm [24] indicate that the standard output from an OCR engine (spatial distribution, geometry and confidence of bounding boxes) provides sufficient information to (1) accurately identify text and noise in a document image, (2) estimate the document's overall quality, and (3) improve OCR transcription performance. Whenever additional pre-processing (e.g., image restoration) is not viable, our algorithm may still be used to boost OCR accuracy by filtering out noise BBs before the document is submitted for linguistic analysis to correct character recognition errors against historical dictionaries and n-gram models. As illustrated in Table 3 and Figure 10, this simple filtering step can lead to significant gains in OCR performance: an average of 6.3% improvement for 85.4% of the documents analyzed. Additional improvements in BB labeling may be obtained by using information from linguistic processing as additional features for the MLP. Denoising then would become an iterative process throughout the post-processing pipeline of improving OCR transcriptions for degraded page images.

Our results for the font identification problem show that the characteristics of Blackletter and Roman fonts can be successfully captured by statistics related to character widths, angled strokes and Zernike moments. A logistic regression model built using the extracted features can classify word into type of font with a F1-score of 84%. The bag-of-word feature provides a good feature representation for the problem in hand. When a font classification model for classifying a document image is trained using

active learning, on an average 14% (out of 3272 documents) of labeled data are required to achieve 89% test accuracy; see Figure 30. In terms of eMOP project, the work presented in this thesis has laid down strong foundation to achieve the goal of good quality digitization of historical documents. The work presented on font identification can be easily adapted to build active learning based identification systems for other page problems such as bleedthrough, identifying pictures, musical scripts and decorative page elements. These systems can be wrapped around a user interface to allow linguists (subject experts) to guide machine learning models by tagging documents. The developed algorithms can utilize these newly tagged data to fine tune their machine learning (ML) models. Another extension to the active learning based system can be done by making use of interactive machine learning strategies. Such systems (e.g. Cueflik [44]) are useful when the set of tags(or labels) may evolve as the user's understanding of the collection (or the ML results) improves. This suggests that the user should not be viewed as a mere "data labeler" but as an explorer and a designer in the overall process. This can be achieved by asking the user to select what kind of features he would like to use for a specific problem.

We can also make use of the denoised output for analyzing and exploring unknown noise types in the eMOP databases (ECCO and EEBO). Based on experts' advice, we considered that these databases just have problems like bleedthrough, skew, decorative page elements, non-unified font classes etc. The quality assessment algorithm outputs a label for each BB, using which we can extract the noisy BB images, and then

select a set of diverse noisy BB images [45]. We can analyze these diverse noisy BB images to find sources of other noises apart from ones explained by the eMOP experts.

The work presented in this thesis will aid the mass-digitization process at eMOP. The open source versions of this work will be made available to researchers working on mass digitization projects. We expect that the research presented in this thesis, along with the tools developed, will mobilize the scholarly research at libraries and museums by improving the digitization quality of historical documents available from the EEBO/ECCO collections (45 million documents).



## REFERENCES

- [1] M. J. Christy, L. Auvil, R. Gutierrez-Osuna, B. Capitanu, A. Gupta, and E. Grumbach, "Diagnosing Page Image Problems with Post-OCR Triage for eMOP," presented at the *Proceedings of Digital Humanities Conference*, 2014.
- [2] eMOP. (2012). *OCR'ing Early Modern Texts*. Available: <http://emop.tamu.edu/>
- [3] M. B. Imani, M. R. Keyvanpour, and R. Azmi, "Semi-supervised Persian font recognition," *Procedia Computer Science*, vol. 3, pp. 336-342, 2011.
- [4] S. La Manna, A. Colia, and A. Sperduti, "Optical font recognition for multi-font OCR and document processing," in *10th International Workshop on Database and Expert Systems Applications*, 1999, pp. 549-553.
- [5] R. Rani, R. Dhir, and G. S. Lehal, "Script identification of pre-segmented multi-font characters and digits," in *12th International Conference on Document Analysis and Recognition (ICDAR-2013)*, 2013, pp. 1150-1154.
- [6] M. Zahedi and S. Eslami, "Farsi/Arabic optical font recognition using SIFT features," *Procedia Computer Science*, vol. 3, pp. 1055-1059, 2011.
- [7] R. Smith, "An Overview of the Tesseract OCR Engine," in *9th International Conference on Document Analysis and Recognition (ICDAR-2007)*, 2007, pp. 629-633.
- [8] T. M. Breuel, "The hOCR microformat for OCR workflow and results," in *9th International Conference on Document Analysis and Recognition (ICDAR-2007)*, 2007, pp. 1063-1067.

- [9] P. Ye and D. Doermann, "Document Image Quality Assessment: A Brief Survey," in *12th International Conference on Document Analysis and Recognition (ICDAR-2013)*, 2013, pp. 723-727.
- [10] R. D. Lins, S. Banerjee, and M. Thielo, "Automatically detecting and classifying noises in document images," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 33-39.
- [11] N. Sandhya, R. Krishnan, and D. Babu, "A language independent Characterization of Document Image Noise in Historical Scripts," *International Journal of Computer Applications*, vol. 50, 2012.
- [12] A. Farahmand, A. Sarrafzadeh, and J. Shanbehzadeh, "Document Image Noises and Removal Methods," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2013.
- [13] A. Ben Salah, N. Ragot, and T. Paquet, "Adaptive detection of missed text areas in OCR outputs: application to the automatic assessment of OCR quality in mass digitization projects," presented at the *Proceedings SPIE Document Recognition and Retrieval*, 2013.
- [14] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and information systems*, vol. 35, pp. 249-283, 2013.
- [15] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *ICML*, 2000, pp. 839-846.
- [16] B. Settles, "Active learning literature survey," *University of Wisconsin Madison*, vol. 52, p. 11, 2010.

- [17] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd, "A stream-based semi-supervised active learning approach for document classification," in *12th International Conference on Document Analysis and Recognition (ICDAR-2013)*, 2013, pp. 611-615.
- [18] L. Likforman-Sulem, J. Darbon, and E. H. B. Smith, "Enhancement of historical printed document images by combining total variation regularization and non-local means filtering," *Image and vision computing*, vol. 29, pp. 351-363, 2011.
- [19] U. Reffle and C. Ringlstetter, "Unsupervised profiling of OCRed historical documents," *Pattern Recognition*, vol. 46, pp. 1346-1357, 2013.
- [20] M. Reynaert, "Non-interactive OCR post-correction for giga-scale digitization projects," in *Computational Linguistics and Intelligent Text Processing*, ed: Springer, 2008, pp. 617-630.
- [21] B. Alex, C. Grover, E. Klein, and R. Tobin, "Digitised historical text: Does it have to be mediOCRe," in *Proceedings of the First International Workshop on Language Technology for Historical Text(s) (LThist) at KONVENS 2012*, pp. 401-409.
- [22] L. Furrer and M. Volk, "Reducing OCR errors in Gothic-script documents," in *Proc. RANLP 2011 workshop on Language Technologies for Digital Humanities and Cultural Heritage*, 2011, pp. 97-103.
- [23] D. Ghosh, T. Dube, and A. P. Shivaprasad, "Script recognition—A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2142-2161, 2010.

- [24] A. Gupta, R. Gutierrez-Osuna, M. Christy, C. Boris, A. Loretta, L. Grumbach, R. Furuta, and L. Mandell, "Automatic assessment of OCR quality in historical documents," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-2015)*, 2015.
- [25] R. Smith, "An Overview of the Tesseract OCR Engine," presented at the *Proc. 9th Int. Conf. Document Analysis and Recognition (ICDAR)*, 2007.
- [26] W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage," presented at the *Proc. Section on Survey Research Methods (American Statistical Association)*, 1990.
- [27] Typedia.com. (2006–2015, 05/15). *Learn: Anatomy of a Typeface*.
- [28] W. contributors. (2006–2015, 05/15). *Blackletter*. Available: <https://en.wikipedia.org/wiki/Blackletter>
- [29] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *9th IEEE International Conference on Computer Vision, 2003* 2003, pp. 1470-1477.
- [30] I. The MathWorks. (1994-2015, 05/15). *imresize*. Available: <http://www.mathworks.com/help/images/ref/imresize.html>
- [31] R. C. Gonzalez and R. E. Woods, "Digital image processing," ed: Prentice hall Upper Saddle River, NJ, 2002.
- [32] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis, "Skew angle estimation for printed and handwritten documents using the Wigner–Ville distribution," *Image and Vision Computing*, vol. 20, pp. 813-824, 2002.

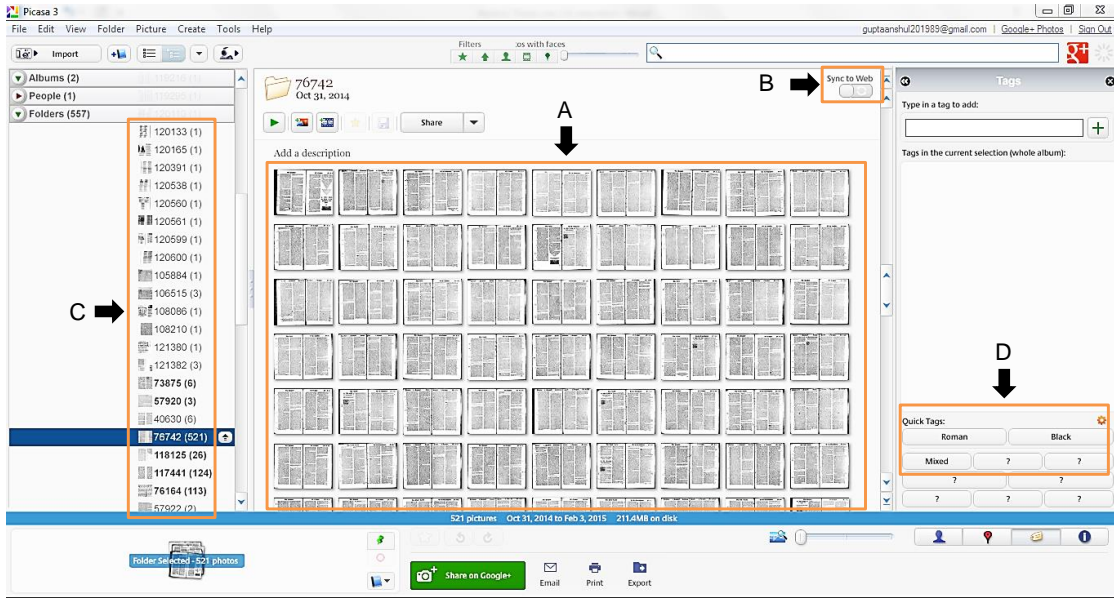
- [33] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, pp. 111-122, 1981.
- [34] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179-187, 1962.
- [35] A. Tahmasbi, F. Saki, and S. B. Shokouhi, "Classification of benign and malignant masses based on Zernike moments," *Computers in Biology and Medicine*, vol. 41, pp. 726-735, 2011.
- [36] C. Elkan, "Using the triangle inequality to accelerate k-means," in *ICML*, 2003, pp. 147-153.
- [37] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, pp. 1-114, 2012.
- [38] B. Settles, "From theories to queries: Active learning in practice," *Active Learning and Experimental Design*, pp. 1-18, 2011.
- [39] J. Long, J. Yin, W. Zhao, and E. Zhu, "Graph-based active learning based on label propagation," in *Modeling Decisions for Artificial Intelligence*, ed: Springer, 2008, pp. 179-190.
- [40] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, pp. 321-328, 2004.
- [41] T. Kobayashi, K. Watanabe, and N. Otsu, "Logistic label propagation," *Pattern Recognition Letters*, vol. 33, pp. 580-588, 2012.
- [42] G. Inc. (2015, 05/15). *Picasa*. Available: <https://picasa.google.com/>

- [43] Y. Baram, R. El-Yaniv, and K. Luz, "Online choice of active learning algorithms," *The Journal of Machine Learning Research*, vol. 5, pp. 255-291, 2004.
- [44] J. Fogarty, D. Tan, A. Kapoor, and S. Winder, "CueFlik: interactive concept learning in image search," presented at the *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008.
- [45] C. Dima, M. Hebert, and A. Stentz, "Enabling learning from large datasets: Applying active learning to mobile robotics," in *Robotics and Automation, 2004. Proceedings. ICRA'04*, 2004, pp. 108-114.

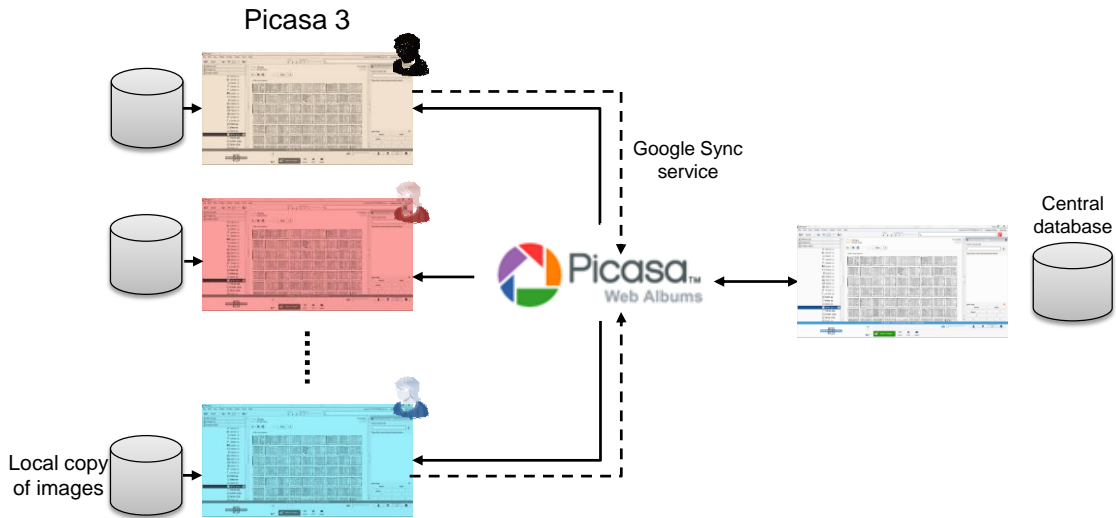
## APPENDIX

### Picasa tool

Picasa [42] is a photo management tool developed by Google Inc. It provides an easy-to-use interface to quickly visualize and manage thousands of photos/images. In the eMOP project, we use Picasa to quickly annotate document images according to the type of font present. The only drawback of Picasa is that it is a standalone PC application and does not allow collaboration. Since eMOP project has its experts residing in different parts of the country, we want to share the labels provided by one expert on his/her PC with others. Hence, we make use of Google sync - a label pushing service by Google, which automatically updates the labels of the document images on Picasa web albums; block diagram of the label pushing service is shown in Figure 36. At any time, when the other experts open their Picasa tool, the labels get pushed from Picasa web albums to their PC. Figure 36 shows a snapshot of the Picasa tool. Here, C represents the directory tree for easy navigation through the image folders; A is the region where thumbnails of the images are displayed; B is the sync button that uploads labels to Picasa web albums; and D is the tagging facility where the user can define tags and annotate images.



**Figure 35: Snapshot of Picasa used to annotate document images.**



**Figure 36: Sync service block diagram.**