Towards Knowledge Discovery from the Vatican Secret Archives. *In Codice Ratio* – Episode 1: Machine Transcription of the Manuscripts.

Donatella Firmani
Roma Tre University
donatella.firmani@uniroma3.it

Marco Maiorino
Vatican Secret Archives
m.maiorino@asv.va

Paolo Merialdo
Roma Tre University
paolo.merialdo@uniroma3.it

Elena Nieddu
Roma Tre University
elena.nieddu@uniroma3.it

ABSTRACT

In Codice Ratio is a research project to study tools and techniques for analyzing the contents of historical documents conserved in the Vatican Secret Archives (VSA). In this paper, we present our efforts to develop a system to support the transcription of medieval manuscripts. The goal is to provide paleographers with a tool to reduce their efforts in transcribing large volumes, as those stored in the VSA, producing good transcriptions for significant portions of the manuscripts. We propose an original approach based on character segmentation. Our solution is able to deal with the dirty segmentation that inevitably occurs in handwritten documents. We use a convolutional neural network to recognize characters, and statistical language models to compose word transcriptions. Our approach requires minimal training effort, making the transcription process more scalable, as the production of training sets requires a few pages and can be easily crowdsourced. We have conducted experiments on manuscripts from the Vatican Registers, an unreleased corpus containing the correspondence of the popes. With training data produced by 120 high school students, our system has been able to produce good transcriptions that can be used by paleographers as a solid basis to speedup the transcription process at a large scale.

ACM Reference format:
Donatella Firmani, Marco Maiorino, Paolo Merialdo, and Elena Nieddu. 2018. Towards Knowledge Discovery from the Vatican Secret Archives. *In Codice Ratio* – Episode 1: Machine Transcription of the Manuscripts.. In *Proceedings of The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, United Kingdom, August 19–23, 2018 (KDD '18),* 10 pages.
https://doi.org/10.1145/3219819.3219879

1 INTRODUCTION

In Codice Ratio is a research project that aims at developing novel methods and tools to support content analysis and knowledge discovery from large collections of historical documents. The goal is to provide humanities scholars with novel tools to conduct data-driven studies over large historical sources. Paleographers and philologists could quantitatively analyze trends and evolution of writings and languages across time and countries; historians could examine and discover facts and correlations among information spread in vast corpora of documents.

The project concentrates on the collections of the Vatican Secret Archives (VSA)[1], one of the largest and most important historical archives in the world. In an extension of 85 kilometres of shelving, it maintains more than 600 archival collections of historical documents on the Vatican activities – such as, official correspondence of the Vatican, account books, correspondence of the popes – starting from the end of the eighth century. We are currently working on the collection of the Vatican Registers, which record the inbound and outbound correspondence of the popes: political letters that testify the broad activities of the popes in the ecclesiastical and temporal spheres; authoritative opinions on legal issues; documents addressed to sovereigns, religious and political institutions scattered throughout the globe; correspondence relating to the harvest of tithes and tributes due to the Church. A small illustration of a page from the Vatican Registers is shown in Figure 1.

The systematic and continuous preservation of these registers began in the middle age,[2] hence most of these documents are *manuscripts.* The VSA has begun to acquire digital images of these documents but, unfortunately, complete transcriptions for the earliest registers do not exist. Therefore, a first fundamental step to develop any form of data-driven content analysis is to perform a transcription of the manuscripts. The problem is challenging: on the one hand, a manual transcription is unfeasible (at least in a reasonable amount of time), due to the volume (hundreds of thousands of pages) of the collection. On the other hand, although these manuscripts are written with a uniform style (a derivation of the *Caroline* style), traditional OCR does not apply here, because of irregularities of writing, ligatures and abbreviations.

[1]The word secret does not denote confidentiality. "Secret" derives from the Latin word "secretum", which stands for "private", meaning that the archives are personal property of the Pope.
[2]Namely, under Pope Innocent III (1198-1216).

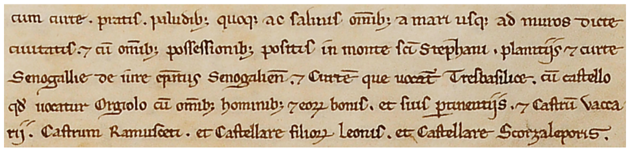
As many libraries and archives have begun digitizing their assets, several handwritten text recognition approaches have been recently developed to address text transcriptions also from medieval manuscripts. Since segmenting words in characters is tricky with hand written texts, these approaches typically aim at recognizing entire words. Because of the variability and the size of the lexicon, they need a huge amount of training data, i.e., hundreds of fully transcribed pages, and often end up focusing on specific tasks, such as *word spotting* [1, 11], which aim at recognizing words from a given small vocabulary (e.g. a list of names). To illustrate the problem of building a reliable training dataset, consider Figure 2: it reports the distribution of the occurrences of words in a corpus composed by a partial transcription of the Registers of Innocent III (in total, it is about 68,000 words). Observe that a few words (just 9) occur more than 100 times (the most occurring word is *"et"*, the Latin conjunction that corresponds to "and"), while the majority of words have less than 10 occurrences.

Our goal is to develop a full-fledged system that transcribes as much as possible from the manuscripts, letting the paleographer to complete the produced transcription with local corrections. To this end, we follow a different approach, based on character segmentation. Our idea is to govern imprecise character segmentation by considering that correct segments are those that give rise to a sequence of characters that more likely compose a Latin word. We have therefore designed a principled solution that relies on a convolutional neural network classifier and on statistical language models. For every word, we perform a segmentation that can produce more segments than those actually formed by the characters in the word. Every segment is labeled by a classifier, which essentially recognizes the most likely character. We then organize the sequence of segments in a directed acyclic graph: the paths of such a graph represent candidate transcriptions for the word, and the most likely solution is selected based on language statistics.

It is worth noting that compared to a segmentation-free approach, training the classifier requires labeled examples for the limited set of character symbols, with a twofold advantage. First, the size of the training set is several order of magnitude smaller, as we need to provide examples only for the limited set of character symbols, and not for a rich lexicon of words. Second, producing the examples is much easier, as it does not require to transcribe whole words, an activity that can be carried on by expert paleographers. In our system, the production of the training set is accomplished by a crowdsourcing solution that consists of simple visual pattern matching tasks, similar to captchas.

To summarize, our on-going experience for the transcription of the manuscripts of the VSA makes the following contributions:

- we have designed an end-to-end solution that builds on state-of-the-art technologies—such as, convolutional neural network, statistical language models, crowdsourcing—to support paleographers to transcribe large corpora of manuscripts;
- we have experimented our approach on the Vatican Registers: never having been transcribed in the past, these documents are of unprecedented historical relevance; we were able to generate the exact transcription for 65% of the word images of our dataset; for an additional 10% the correct transcription was at edit distance 2;



cum curte. pratiſ. paludib; quoq; ac ſaltuiſ omib; a mari uſq; ad muros dicte
ciuitatiſ. 7 cu omib; poſſeſſionib; poſitiſ in monte ſci Stephani. plantaijs 7 curte
Senogallie de iure eccl'ie Senogalien. 7 Curte que uocat Trebaſilice. cu caſtello
qd uocatur Orgiolo cu omib; hominib; 7 eo2 boniſ. et ſuiſ ptinentijs. 7 Caſtru Vacca
rij. Caſtrum Ramuſceri. et Caſtellare filio2 Leoniſ. et Caſtellare Scorzaleporiſ.

Figure 1: Sample text from the manuscript "Liber septimus regestorum domini Honorii Pope III" (Vatican Registers).

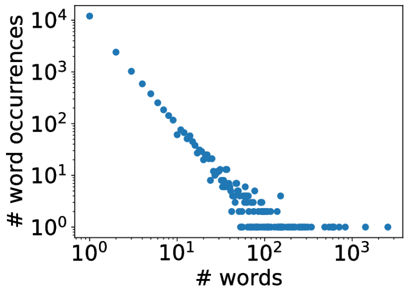


Figure 2: Word distribution in the Registers (log-log scale).

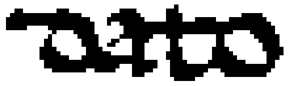


Figure 3: Sample input image of our work-flow. The correct transcription is the Latin word "dato" (meaning "given").

- our approach drastically simplifies the production of the training set; in our experiments the training set has been generated in a few hours involving 120 high school students and using just 2 folios of the Vatican Registers;
- all the material (code, ground-truth, training data) of the project is publicly online.[3]

2 OVERVIEW

Our raw input data consist of high-quality scanned images of whole manuscript *pages*. Each page undergoes a set of standard pre-processing steps, leading to a bag of word images.

(1) the color image is transformed into a bi-chromatic one;
(2) all the white margins are cropped;
(3) *skew* and *slant* are corrected, that are, page distortions due to acquisition process and letter slope due to calligraphy;
(4) lines are separated out, by looking at horizontal white spaces;
(5) words are extracted, by looking at vertical white spaces.

Each word image is finally submitted to our transcription system. Figure 3 shows a pre-processed word image, of size 66×18 pixels.

Main components. Our system consists of four main components.

- **Collection of training samples.** We implemented a custom crowd-sourcing platform, and employed 120 high-school students to label the dataset. To overcome the complexity of reading ancient fonts, we provided the students with positive and negative examples of each symbol. We trained a

[3]http://www.inf.uniroma3.it/db/icr/

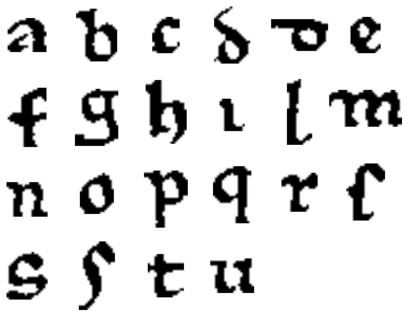


Figure 4: Our set of 22 symbols.

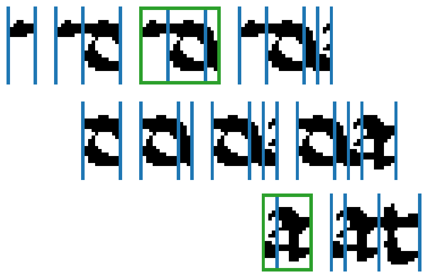


Figure 5: Breakdown of the word in Figure 3 into overlapping segments[4]. Local minima are shown in blue. Pieces fitting entirely one character are highlighted in green.

character classifier on this dataset, implemented with a deep convolutional neural network.

- **Character recognition.** Recognizing characters within a handwritten word is challenging, due to *ligatures*. To this end, we first partition the input word into elementary text segments. Most segments contain actual characters, but there are also segments with spurious ink strokes. Then, we submit all the segments to the trained classifier. Computed labels are very accurate when the input segment contains an actual character, but can be wrong otherwise.
- **Candidate transcriptions.** We reassemble noisy labels from the classifier into a set of candidate word transcriptions. Specifically, we select the best m candidate transcriptions for the input word image, using language models.
- **Word decoding.** We consider the m transcriptions from the previous step and revise character recognition decisions in a principled way, by solving a specific decoding problem on a high-order hidden Markov model. The most promising transcriptions are finally returned to the user.

Character symbols. We take into account minuscule characters of the Latin alphabet and their variations, shown in Figure 4.

3 SYSTEM WORK-FLOW

In this section, we describe in detail each component of our system. We use the word image in Figure 3 as running example.

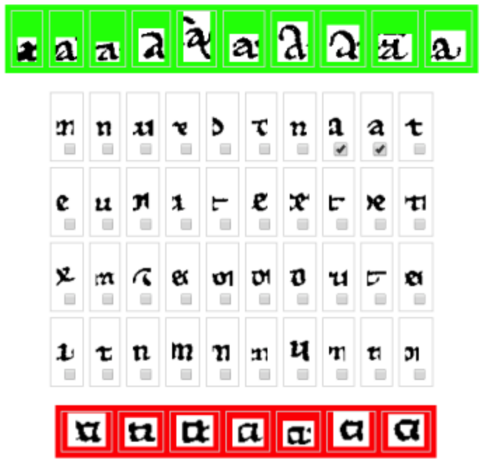


Figure 6: Sample screen of our platform. Positive and negative examples are highlighted in green and red respectively.

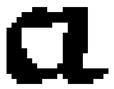


Figure 7: Sample pattern resembling modern hand-writing style of character "a", produced by wrong segmentation of the bi-gram "ci". Our target pattern of character "a" is given by the top-leftmost symbol in Figure 4.

3.1 Training samples

The fundamental step of our transcription system consists in collecting labelled samples of character symbols in Figure 4. In the HTR domain, there is a common understanding that there is no advanced strategy for partitioning a word image into its character constituents, without being aware of its transcription somehow. Therefore, we do something basic at this point, which turns out to be very effective for our purposes. As shown in Figure 5, we break down a word into overlapping segments of different sizes, bounded by *local minima* of the black pixel count function over columns of the image matrix. Then, we ask the crowd to select only segments in which a given visual pattern (among those in Figure 4) fits *entirely*. We observed that almost all the "non-characters" (including small, large, and misplaced segments) are filtered out, leaving nicely cropped segments such as those highlighted in green in Figure 5.

We first describe our crowd-sourcing application and then detail our basic segmentation technique, which we call *slice-segmentation*.

Crowd-sourcing. We developed a custom crowd-sourcing platform and enrolled 120 high-school students in the city of Rome, that did the labeling as a part of a work-related learning program. Each student was required to perform a sequence of tasks: each task consists of selecting any image on the platform that visually matched (fitting entirely with least possible extra-strokes) a given character symbol. In Figure 6, we show a screen-shot of a task. Each task relates to a symbol, and consists specifically of:

[4]We show only 10 out of the 30 segments from slice-segmentation, for lack of space.

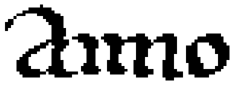


Figure 8: A sample occurrence of the Latin word "anno" (meaning "year") yielding many OCR interpretations. Interpretations include "aiiiio", "aimo", "amio", "aniio', "aiino", and "ainio", but none of these is a Latin word.

- sample occurrences of the symbol (in green), best capturing hand-writing variations identified by paleographers;
- 40 images randomly drawn among a pool of unlabelled segments, arranged in a grid, each with its own check-box.

Every time the check-box is marked, the corresponding image receives a vote. Finally, the image is labelled with the most voted character symbol. If no clear majority emerges, then the image is labelled with a special *non-character* class ⊗. Students were told to match visual patterns, rather than trying to *read*. However, enrolled students do use modern variants of certain symbols in their daily lives, and is reasonable to assume that some tasks have been solved by using reading skills. In order to prevent mislabeling of patterns that casually resemble modern writing styles of symbols, such as Figure 7, we also provided examples of *negative* cases (in red).

Slice-segmentation. Given the input image word, we count the ink pixels for each column of the matrix encoding the image. Local minima in the resulting function are considered as possible character boundaries. Our intuition is that, typically, ligatures correspond to vertical "bands" with less ink. In the case of adjacent local minima – which can occur if the black pixel count function has plateaus – only the leftmost candidate point is chosen.

3.2 Character recognition

The core of our transcription system is a character classifier that takes as input a segment of a word image. The classifier estimates the probability that the image belongs to each of the character classes in Figure 4 and to the special non-character class ⊗. The choice to introduce a non-character class, rather than discarding results with classification score below a certain threshold, is motivated by the lack of consistent scoring of correct segmentations versus the incorrect ones. The classifier is implemented with a *deep convolutional neural network*, and trained with an augmented set of the crowd-sourced image labels. The resulting model, which has been described in [7], achieves 96% accuracy, which is one of the highest results reported in the literature so far. One may think that such an accurate classifier allows the user to just feed it with images from slice-segmentation of the input word w and let it filter out almost everything but the nicely cropped segments, which together constitute w. However, there are two problems that a transcription system needs to solve.

- While humans can easily distinguish character symbols from spurious strokes, this turns out to be a hard task for an automatic classifier. Respective to the class ⊗ (non-character), indeed, our classifier achieves 95% precision but only 74% recall. This means that our transcription system needs to deal with a certain amount of "false characters".

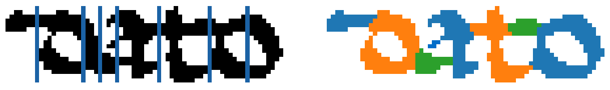

(a) slice-segmentation (b) jigsaw-segmentation

Figure 9: Slice-segmentation vs. jigsaw-segmentation.

- Depending on the writing-style, the visual features of certain handwritten characters components, such as the individual sticks of a "m", are indistinguishable from different whole characters, such as the "i" . In these cases, the "i"-shaped elements and the whole "m" ought to be considered equally valid OCR instances. Only when we consider the word holistically, indeed, we can rule out infrequent cases such as "iii". Figure 8 provides an extreme example of this phenomenon.

Our solution is two-fold.

- We introduce a more sophisticated segmentation technique, which we refer to as *jigsaw-segmentation*, with several benefits, including reducing "false characters" in the recognition phase. The difference between the slice-segmentation and the jigsaw-segmentation is schematically shown in Figure 9.
- We handle ambiguity intrinsic in the character classification process by using a custom version of the so-called *transcription lattice* data-structure, that has been introduced in [14] for *on-line* HTR systems. In the on-line setting the handwriting is given as a temporal sequence of coordinates that represents the pen tip trajectory, whereas in our case – which is typically referred to as *off-line* HTR – the handwriting is given as an image of the text, without any time sequence information.

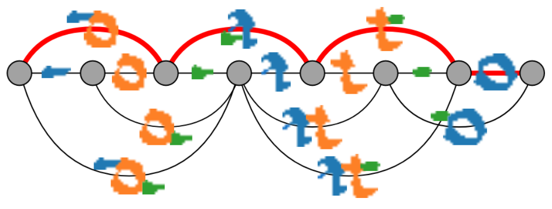
The jigsaw-segmentation is discussed first, then we give details of character classifier. Finally, we describe our lattice data-structure.

Jigsaw-segmentation. For each connected component of the input word image, we compute its upper and lower contour[5]. We then compute local minima for the upper contour, and local maxima for the lower contour. Having both functions aligned to the column index of the word image, we connect each local minimum from the upper contour to the closest local maximum from the lower contour. Resulting "regions", or segments, can be combined together into larger ones by using the lattice. Figure 9b shows regions extracted by the jigsaw-segmentation on a sample word, with different colors.

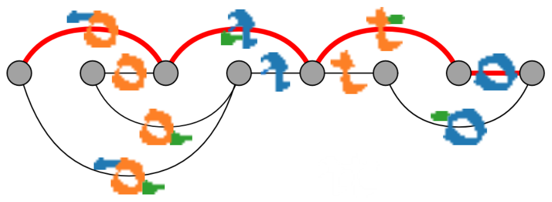
Character classifier. Our classifier is implemented with the deep convolutional neural network described in [7]. For sake of completeness, we mention that it takes as input 56 × 56 single-channel images. The input is then propagated through 8 adaptable layers: 3 convolutional layers, each applying 2 × 2 stride 2 max-pooling, and 2 feed-forward layers. All layers use ReLU activation function, except for the output, where softmax activation is applied. This design is inspired to recent neural networks models [12].

Transcription lattice. We consider the jigsaw-segmentation of the input word, the x coordinates of segment centroids, and a "start"

[5]We apply a smoothing of 3 pixels to both the resulting functions



(a) $\eta = 1.0$



(b) $\eta < 1.0$

Figure 10: Sample lattices with different thresholds.

centroid corresponding to the $x = 0$ coordinate. Using these data, we build a directed acyclic graph where:

- each vertex is representative of a centroid c;
- each edge (c_1, c_2) represents the *group* of adjacent segments whose centroids c are in the interval $(c_1, c_2]$, and possibly form together a character shape.
- each path yields a *partition* of the word into groups of adjacent segments (i.e., candidate characters).

We use the term "edge" for referring both to the mathematical object and to the group of adjacent segments. Edges that span a character-sized length – that is, that have endpoint centroids not further[6] than a threshold σ – are submitted to our classifier. Then, only those that get a probability score for the class $\otimes$ lower than a threshold η are effectively added to the lattice. We set the default values for σ and η to 25 pixels (we observed experimentally that edges larger than 25 pixels can be safely dropped) and 0.1, respectively.

Example 3.1. Figure 10a shows the complete transcription lattice for our word in Figure 3, with the path yielding the correct partition in red. Seven distinct segments produce eight vertices, one for each segment plus the "start" vertex. Edges between vertices represent alternative groupings for the distinct segments. Figure 10b shows the same graph, without the edges which would likely be discarded by the classifier as $\otimes$. Discarding edges reduces the number of paths, and therefore the number of unlikely partitions for a word.

Each edge in the lattice gets an array of *labels*, corresponding to the top-ranked classes from the classifier. Even though the top class is correct in the 96% of the cases, considering a few alternatives from the probability distribution returned by the classifier can be useful in the most difficult instances. Therefore, we allow each edge in the lattice to have multiple labels, using the following heuristic.

(1) We sort classes according to probability.
(2) We pick the top classes that account together for at least θ_1 probability (that is, the sum of their scores is $\geq \theta_1$), and that account singularly for at least θ_2 (that is, their score is $\geq \theta_2$).
(3) If we select the class $\otimes$ in this process, we drop the edge.
(4) Otherwise, we set selected classes as its labels.

Default values for θ_1 and θ_2 are 0.8 and 0.1, respectively [7].

Example 3.2. Let $\{l_1 : p_1, l_2 : p_2, l_3 : p_3\}$ be the top-3 classes from the classifier. We show some computations of our heuristic in the default setting for deciding labels in a few sample cases.

- {"a" : 0.8, "o" : 0.1, "d" : 0.05} yields a single label "a".
- {"a" : 0.75, "o" : 0.05, "d" : 0.05} also yields a single label "a", because for scoring total 0.8 we would need the low-probability label "o" (which violates θ_2).
- {"a" : 0.5, "o" : 0.4, "d" : 0.05} yields a double label "a/o".

3.3 Candidate transcriptions

Let the *origin* vertex be the start centroid, and the *destination* vertex be any of the sink vertices (i.e., those with no outgoing edges). Note that one sink vertex corresponds to the rightmost centroid, while possible other sinks are produced by the edge selection process. We generate all the edge-disjoint paths from the origin to all possible destinations: each corresponds to a candidate transcription. An edge with multiple labels can be thought of as a set of *parallel* edges (i.e., a set of edges that have the same endpoints), each with its own label. We generate transcriptions in a depth-search-first fashion, and grow each transcription incrementally from left to right.

Example 3.3. Consider the lattice in Figure 10b. The candidate transcriptions are "dato", "daid", "diid", "dito".

Generated transcriptions are ranked in non-increasing order of *word probability*, based on a *model* of the Latin language. We first describe this step, and then make time-efficiency considerations.

Ranking transcriptions. We use a Latin corpus for estimating q-gram frequencies in the Latin language. Probability of each transcription is estimated under the assumption that the frequency of each character only depends from the preceding $q-1$ characters. Consider the transcription "dato" in Example 3.3, and let $q = 3$. There are two classical methods for evaluating its probability.

- We can think of the transcription as a *sub-string* of a possibly larger word, and compute its *sub-string probability* as $p(dato) = p(d)p(a|d)p(t|da)p(o|at)$.
- We can think of the transcription as a *word* of the language, and compute its *word probability* as $p(\$dato\wedge) = p(d|\$)p(a|\$d)p(t|da)p(o|at)p(\wedge|to)$, where \$ and $\wedge$ are special symbols denoting the beginning and the end of a word.

Terms in the above formulae are part of the language model. Word probability better represents whether a given transcription corresponds to a Latin word, whereas sub-string probability is more robust to word image extraction failures, where the word image lacks the first (or the last) few symbols. We rank transcriptions according to their word probability, with $q = 6$ by default, and use sub-string probability for optimization (as discussed in the following). Some sinks can be close to the start vertex due to noise in the

[6]Centroid distance is just $c_2 - c_1$.

[7]In practice, we never select more than three labels.

jigsaw-segmentation, yielding short transcriptions even for long word images. Such shorter paths can have higher word probability than longer ones, therefore, we introduce a simple yet robust method for getting rid of them.

(1) We compute word image length.
(2) We estimate the size that each transcription would have on paper, by multiplying the average character length observed in our experiments (that is, 19 pixels) by the number of characters in the transcription.
(3) We drop all the generated transcriptions with length smaller than 90% of the word image length.

Speed-up. Generating all the paths can be time-consuming, and avoiding uninteresting transcriptions can save time without affecting result quality. For this purpose, we use sub-string probability. Since sub-string probability decreases monotonically as the transcription grows longer, as soon as the sub-string probability of a path from the origin vertex drops under a threshold β, we "prune" all the paths with that prefix from our computation. Even though word probability may have different value than sub-string probability, it is reasonable to assume that low sub-string probability transcriptions correspond to low word probability transcriptions. We set the default value for β to 10^{-16} (note that path probability is the product of possibly many terms due to chain rule).

3.4 Word decoding

The final step of our work-flow consists in revising character recognition decisions, and possibly making punctual changes to candidate transcriptions. In the case of wrong classifier decisions, we may not have generated the correct transcription of a word image. So-called *decoding* algorithms, such as the Viterbi algorithm [29], are one classical approach for solving those cases in HTR and speech-to-text applications. The decoding problem consists indeed in finding the most likely sequence of hidden states of a Hidden Markov Model (HMM), that resulted in a given sequence of observed events.

In the HTR domain:

- the *hidden sequence* is the underlying word in the image;
- the *observed sequence* is the the generated transcription from the lattice, with labels computed by the classifier;
- the *transition probabilities* are the language model: for instance, frequency of bi-gram "ci" corresponds to transition probability from hidden state "c" to hidden state "i" (that is, a q-gram model yield a $q-1$-th order HMM);
- the *emission probabilities* depend on the precision and recall of the classifier on each class: for instance, if a certain symbol s_1 is frequently mislabelled with s_2, then the probability for the hidden state s_1 of emitting s_2 has to be set accordingly.

We also solve a decoding problem, although we observe two specific features of our setting.

- We use up to 8-grams for computing word probability, which translate into 7-th order HMM. Unfortunately, we observed that even a 2-nd order version of the Viterbi algorithm [29] can take up to 3 seconds for each generated transcription.
- Only few components of our confusion matrix have value significantly above 0. Such components are in correspondence of character pairs with similar shapes, which we call *counterparts*. We show in Figure 11 sample occurrences of counterparts, that can be challenging for humans too.

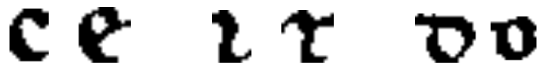


Figure 11: Sample character pairs that can be confused due to their similar shape, or to noise in the jigsaw-segmentation or in the pre-processing phases, leading to punctual errors (akin to misspelling) in the generated transcriptions.

For the above reasons, we trade off model accuracy for efficiency. We implement a fast procedure which optimizes the same objective function of the Viterbi algorithm, under a simplifying assumption on emission probabilities which makes sense in our setting. Specifically, we assume that the probability that the hidden symbol h emits the observed symbol o, i.e., $p(o|h)$, is either 0 or $\frac{1}{k}$, where k is the number of counterparts of h (included h itself). Sets of counterparts are decided both by selecting the higher components of the confusion matrix, and by empirical experience. Our procedure takes as input a generated transcription and returns all the alternative transcriptions obtained by counterparts permutation, in non-increasing order of word probability. Complete list of counterpart classes in our experiments are: "i" and "r", "o" and "d", "n" and "m", "l" and "f", "c" and "e".

Example 3.4. Consider the wrong transcription "dito" in Example 3.3, and let (for sake of this example) "a" be counterpart of "i", and "c" be counterpart of "o". Alternative transcriptions are "dito" (itself), "dato", "ditc", and "datc". In case "dito" were the only generated transcription from previous phases, the decoding step would have considered revising it into "dato", which is correct.

Final transcription set. We solve our specific decoding problem on the top m transcriptions generated in the lattice phase, and produce further M transcriptions. Among those $m + M$ transcriptions, we select m new transcriptions in non-increasing order of word probability. We observed experimentally that transcriptions from the lattice are neither over- or under-represented in the final set.

4 EXPERIMENTS

Our experiments were performed on a machine with an Intel Core i7-4980HQ CPU with 8 cores running at 2.80GHz, and 16GB RAM. Neural network training and prediction were performed on an Nvidia GeForce GTX 980M GPU, with CUDA Toolkit v9.0 and cuDNN v7.0. The operating system was Ubuntu 16.04 LTS, with kernel version 4.4.0-112-generic, and Python version 3.6.3. We implemented the neural network using TensorFlow 1.3, and the character-level language model using KenLM.

4.1 Dataset

Training data. We use annotations from 2 pages of Vatican Register 12. This results in approximately 15K characters. Characters with less than 1K examples were augmented to match the required quantity and balance the training set. The augmentation process involves slight random rotation, zooming, shearing and shifting, both vertical and horizontal. The final dataset comprises 23K examples evenly split between 23 classes.

Test data. We test our system on four pages belonging to the same Vatican Register, but spanning different ages and writers, transcribed entirely by volunteer paleographers. After undergoing the pre-processing and jigsaw-segmentation, each word is transcribed independently by the system. Our system currently considers only the word images without *abbreviated forms*. This is further discussed in Section 6.

Corpus for language model. Our corpus is composed of 716 ancient Roman Latin texts, spanning different ages and subjects, for a total of over 14M words. It is worth observing that the Latin language used in the Vatican Registers exhibits some differences with the ancient Roman Latin, which is typically used in publicly available corpora. These differences introduce some drawbacks, that we are currently overcoming, as we discuss later in Section 6.

4.2 Results

We now compare different configurations of our transcription system. Our experiments are summarized below.

- We show the benefit of the jigsaw-segmentation over the basic slice-segmentation, in the **character recognition** step.
- We compare different settings of the parameters used for the lattice creation and the **candidate transcriptions** steps.
- We show results for different q-gram sizes in the language model and measure how close are the top 3 ranked transcriptions to the correct transcription. We also demonstrate the impact of the sub-string probability on the result accuracy.
- Finally, we quantify the benefit of the **word decoding** phase.

For the considered configurations, performance of our system are measured by the metrics below.

- We define the *reciprocal rank* of a word image as the multiplicative inverse of the rank of the correct transcription: 1 for first place, $\frac{1}{2}$ for second place, and so on. The *Mean Reciprocal Rank* (**MRR**) is the average of the reciprocal ranks of word images in our test set.
- We define the *Mean Word Processing Time* (**MWPT**) as the average time taken by our system for returning top m transcriptions of word images in our test set.
- We define the m-**precision** as the fraction of word images in our test set, for which the correct transcription is i) generated by our system, and ii) ranked in the top m positions. Classical definition of precision is captured by 1-precision.
- For the top few transcriptions, we also provide edit distance statistics (**ED**) with respect to the correct transcription. Specifically, we use the *Levenshtein distance* metric [6].

Results of our experiments are computed in the default parameter setting and without word decoding, unless specified otherwise.

Character recognition. Detailed results on character recognition are described in [7]. For the purpose of this paper, we only report that average precision and recall among all classes are both 96%. Precision ranges from 86% to 99%, whereas recall ranges from 74% to 99%. As frame of comparison, we trained a logistic regression model on the same dataset. Such baseline scored 80% and 79% average precision and recall.

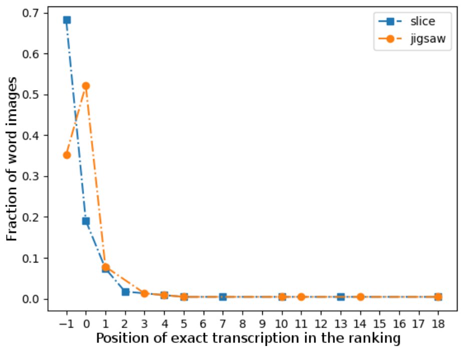


Figure 12: Comparison of different segmentation techniques. The value -1 represents words images for which the correct transcription has not been generated.

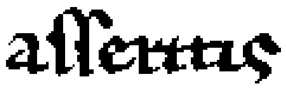
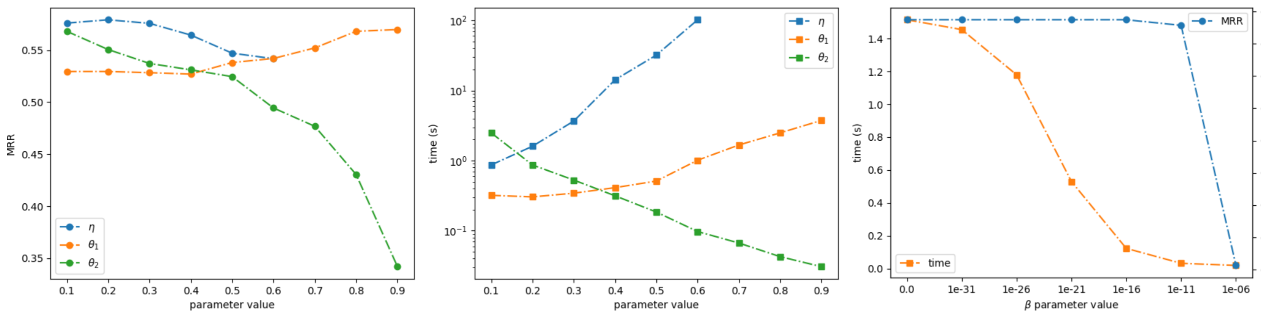


Figure 13: The correct transcription of this word image is "asseritis", while the first-ranked transcription is "afferitis".

Segmentation. Figure 12 shows the performance of our system with different segmentation techniques. The plot considers the position in the ranking of the correct transcription of word images in our test set. The value -1 is a special index for when the correct transcription has not been generated. We observe that the fraction of words for which jigsaw-segmentation yields the correct transcription is approximately 65% (decoding can recover the correct transcription of approximately 9% of the remaining 35%), compared to much lower 20% achieved by slice-segmentation. Also, transcriptions generated by jigsaw-segmentation let language model rank the correct transcription of almost all the word images[8] in the top 5. For remaining 35%, 16% of first-ranked transcriptions is at edit distance 1 from the correct transcription, 15% at distance 2 and 28% at distance 3. Figure 13 shows a sample word image of the 15% group, for which the first-ranked transcription is at distance 2 from the correct transcription.

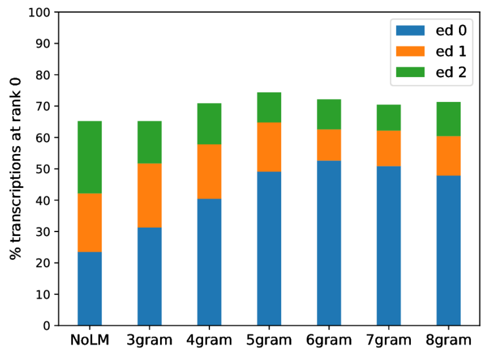
Parameters tuning. Figures 14a and 14b show the performance of our system for different settings of the parameter for lattice creation, that are, η, θ_1 and θ_2 (defined in Section 3.2). Changing such parameters affects indeed the *size* – that is, the number of edges and labels – of the transcription lattice, trading off quality of the transcription for computing time. Dropping edges with non-character probability $> \eta$ has great impact on performances, and the smaller the threshold the better. We recall indeed that $\otimes$ labels (i.e., non-character) returned by our classifier are correct in the 95% of the tested cases[9]. Dropping edges with character probability $< \theta_2$ can decrease running time akin to η. However, faster computation

[8]For which correct transcription is available.
[9]That is, false positive are very infrequent.

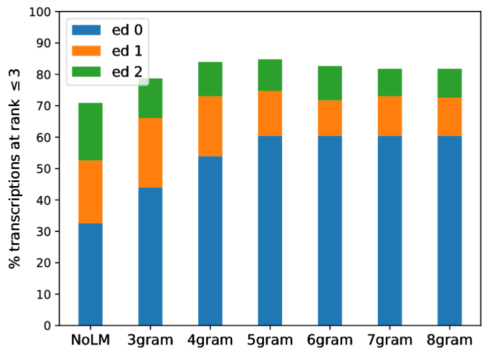


(a) MRR for η, θ_1 and θ_2 (b) MWPT for η, θ_1 and θ_2 (c) MRR and MWPT for β

Figure 14: Different parameter settings for the lattice creation and the transcriptions generation. Computation for $\eta > 0.6$ exceeded 2 min timeout. In Figure 14c, we show MRR and MWPT with shared x-axis (in log-scale). The value 0 is not on scale.



(a) 1-precision



(b) 3-precision

Figure 15: Different values of q in the language model. Bar labeled "NoLM" represents ranking without language model, relying on character classification score only.

has a quality cost here, and we identify the best MRR-MWPT trade-off to $\theta_2 = 0.1$. θ_1 has little impact, due to skew in the probability distribution returned by our character classifier. After building the lattice, the parameter β limits the number of paths that are visited during candidate transcriptions generation, as detailed in Section 3.3. Figure 14c shows that MRR stays high up to $\beta = 10^{-11}$ and drops afterwards. $\beta = 10^{-16}$ has best MRR-MWPT trade-off.

Ranking transcriptions. The blue bars in Figure 15 show the 1-precision and 3-precision of our system for different q-gram sizes in the language model. The bar labeled as "NoLM" shows ranking results obtained without taking language models into account. The NoLM ranking was produced by multiplying network classification scores for each character. Figure 15b considers top 3 transcriptions of all the word images in the test set. We observe that, by using 6-grams, almost 80% of our results are away from correct transcriptions by no more than 2 characters, and approximately 60% corresponds exactly to the underlying manuscript word. Figure 15a reports the corresponding results when considering only top 1 transcriptions. Another way for reading results in Figure 15a is the following. As shown in previous Figure 12, we generate the correct transcription of more than 65% of the word images in our dataset. This means that approximately 77% of such word images have the correct transcription ranked at position 1 when using 6-grams (which is our default setting), but approximately 23% does not get the optimal ranking[10]. Improving on the ranking produced requires a better model of the language used in the Vatican Register, included models of sentences, and is discussed in Section 6.

Fast decoding. We observe that the fraction of word images for which our system does not generate the correct transcription is approximately 35%. (It is worth noticing that for such word images, most of first-ranked transcriptions have no more than 3 spelling errors.) Decoding can recover the correct transcription of approximately 9% of such word images. Other effects of the decoding phase is that top-ranked transcriptions become closer to correct transcriptions. For instance, the amount of word images having correct transcription ranked as second increases by 30%. We note that the amount of word images having correct transcription ranked as first does not change significantly. Overall, only top-ranked positions benefit from decoding. Indeed, decoding has limited effect on the overall MRR, which changes from 0.57 to 0.58 by using decoding.

5 RELATED WORKS

Transcription goal. Handwritten Text Recognition (or HTR) is a branch of research concerning the automatic transcription of

[10]Correct transcription, when generated, is in the top 5 for almost all the word images.

handwritten text. Even though it extends to live-captured handwriting (online recognition), that is clearly not the case for historical documents. The offline recognition task is generally regarded as harder than the online, due to the lack of temporal information: online handwriting recognition can leverage the order and timing of character strokes, while offline recognition cannot. Due to the many challenges involved in a fully automatic transcription system of historical handwritten documents, many researchers in the last years have focused on solving sub-problems, including word spotting [20, 21], handwriting classification [4], and text line segmentation [18]. Our goal is rather the creation of a full-fledged off-line HTR system: an effort shared by several ongoing projects, as more and more libraries and archives worldwide digitize their collections [10, 23, 25].

Crowdsourcing. These transcription systems generally work by a segmentation-free approach, where it is not necessary to individually segment each character. While this removes one of the hardest steps in the process, it is necessary to have full-text transcriptions for the training corpus, in turn requiring expensive labelling procedures undertaken by paleographers with expertise on the period under consideration. [8] details all the steps for the creation of the IAM-HistDB, one of the first publicly available historical handwritten text databases containing medieval script. Our character-level classification has instead much smaller training cost, and allows the collection of a large corpus of annotated data using a cheap crowdsourcing procedure. Crowdsourcing solution for cultural heritage has been experienced in many projects. One of the pioneering initiative to crowdsource the transcription of manuscripts is the *Transcribe Bentham* project, a collaborative platform for crowdsourcing the transcription of the philosopher Jeremy Bentham's unpublished manuscripts [5]. Also the Transcriptorium project [19, 24] exposes HTR tools through specialized crowd-sourcing web portals, supporting collaborative work. Our solution is more focused than the above ones: since it aims at producing training data, it relies on a much simpler solution based on visual pattern matching task that can be performed by unskilled workers.

Pre-processing. In order to perform page, line and word segmentation, we apply image preprocessing on our high-resolution page images. A variety of preprocessing steps as those applied in our system are discussed in [13, 15]. Common steps include normalization of size, density, rotation (slope), and slant.

Neural networks. Our approach employs a convolutional neural network for character image classification. There has been an interest in applying recent results in recurrent and convolutional neural networks to achieve improved classification accuracy: [27] performs word spotting through a deep convolutional neural network, outperforming various word spotting benchmarks; while [9] adopts a bidirectional Long Short-Term Memory neural network to transcribe at word level, with high accuracy. In order to achieve complete transcriptions, these approaches would need thousands of word-level annotations, which is not a scalable task due to the expertise required. We will come back on this point when discussing future research directions for our project.

Segmentation lattice. To deal effectively with ambiguity in segmentation and transcription, we map each word image to a lattice,

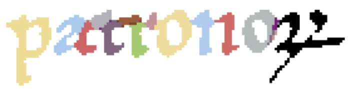


Figure 16: A word (*patrono*rum) from the Vatican Registers containing an abbreviation (the symbol in black).

whose source-to-sink paths represent alternative segmentations and their associated transcriptions. Our approach is similar to the technique proposed in [14], as we discussed in Section 3.

Word decoding. Language modeling for on-line handwriting recognition bears many similarities with the use of language information in OCR and speech recognition, which often employ statistical *n*-gram models on the character or word level [2, 26]. HMM technology is widely used in segmentation-free recognizers [3, 17, 22, 28]. We employ HMM-like decoding to generate likely transcriptions that might not have been generated by our pipeline due to classification errors, simplifying some assumptions on emission probabilities.

6 CONCLUSIONS AND FUTURE WORK

Data science can deeply contribute to analyze and understand our historical and cultural heritage. Data acquisition and preparation from manuscript historical documents is done by means of a transcription process, whose scalability is limited, as it must be performed by expert paleographers. In this paper, we have presented a system, developed in the context of the *In Codice Ratio* project, to support the transcription of medieval manuscripts in order to improve the scalability of the process. We have followed an original approach that requires minimal training effort, and that is able to produce correct transcriptions for large portions of the manuscripts. Our approach, which relies on word segmentation, neural convolutional network, and language models, has been successfully experimented on the Vatican Registers.

We are currently working on the system in order to extend the set of symbols, and hence to improve the overall effectiveness of the process. In particular, we are adding the most frequent abbreviations, i.e., short-hands used by the scribes to save room or to speed up writing. In our process, the main issue with abbreviations is the lack of statistics on their occurrences, which prevents us to effectively apply the language models. Gathering statistics for the abbreviation is not trivial: the usage of these symbols depends both on the age and on the domain of the manuscripts. For instance, in the Vatican Registers, which have diplomatic and legal contents, some abbreviations are more frequent than in manuscripts with of literary works, even from the same age. Indeed, we have already collected training samples for the classifier also for many abbreviations: our crowdsourcing approach to collect labeled examples worked well also for these symbols, as it is based on simple visual pattern matching tasks. Figure 16 shows an example of a one of the most frequent abbreviations. The last symbol, in black, is a shorthand for the Latin desinence *"rum"*: notice that it is simple, given some sample images, to recognize it also without any paleography knowldge. Also the neural network performs well with the extended set of symbols, as abbreviations are typically well distinguishable from other symbols.

Our plan to collect statistics for the abbreviations is to use our current system to produce partial transcriptions for a number of pages, a few dozens, highlighting the words where the character classifier recognizes an abbreviation. Then, we will ask to the paleographers to transcribe these words. Based on these semi-automatic transcriptions, we will progressively update the language models.

So far, we took a probabilistic approach on language modeling: we plan, however, to investigate character-level neural language modeling, similarly to [16].

Another direction that we are currently exploring is to change the human interaction interface of the crowdsourcing platform. The idea is to show a complete word, segmented by the jigsaw-segmentation method, and to ask to the worker to highlight the segments corresponding to a target symbol. We expect to obtain more accurate examples, which could improve the accuracy of the neural network, especially for the class of non-character $\otimes$. Also, these annotations could allow us to build synthetic manuscripts, which could be used to train a recursive neural network.

ACKNOWLEDGMENTS

We thank Andrea Rossi for his fundamental contribution on a earlier version of the system and Simone Scardapane for his work on the deep convolutional neural network. We thank prof. Serena Ammirati and the paleoghapers Marco Miglionico and Michela Ventriglia, who volunteered for transcribing our test set. We are indebted to prof. Marica Ascione and the students of Liceo Keplero and Liceo Montale of Rome who did the labelling effort. We thank Sébastien Bratières and Lukasz Kaiser for helpful discussion, and the staff of PiCampus School for supporting the author Elena Nieddu. We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Quadro M5000 GPU for future work.

REFERENCES

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE transactions on pattern analysis and machine intelligence*, 36(12):2552–2566, 2014.

[2] I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open-vocabulary ocr system for english and arabic. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(6):495–504, 1999.

[3] R. Bertolami and H. Bunke. Hidden markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognition*, 41(11):3452 – 3460, 2008.

[4] M. Bulacu and L. Schomaker. Automatic handwriting identification on medieval documents. In *14th International Conference on Image Analysis and Processing, 2007. ICIAP 2007.*, pages 279–284. IEEE, 2007.

[5] T. Causer and M. Terras. âĂŸmany hands make light work. many hands together make merry workâĂŹ: Transcribe bentham and crowdsourcing manuscript collections. *Crowdsourcing Our Cultural Heritage*, pages 57–88, 2014.

[6] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB'03, pages 73–78, 2003.

[7] D. Firmani, P. Merialdo, E. Nieddu, and S. Scardapane. In codice ratio: Ocr of handwritten latin documents using deep convolutional networks. In *Proceedings of the 11th Int. Workshop on Artificial Intelligence for Cultural Heritage*, 2017.

[8] A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, and M. Stolz. Ground truth creation for handwriting recognition in historical documents. In *Proceedings of the 9th IAPR Int. Workshop on Document Analysis Systems.* ACM, 2010.

[9] A. Fischer, M. Wuthrich, M. Liwicki, V. Frinken, H. Bunke, G. Viehhauser, and M. Stolz. Automatic transcription of handwritten medieval documents. In *15th Int. Conf. on Virtual Systems and Multimedia, 2009.* IEEE, 2009.

[10] I. Flaounas, O. Ali, T. Lansdall-Welfare, T. De Bie, N. Mosdell, J. Lewis, and N. Cristianini. Research methods in the age of digital journalism: Massive-scale automated analysis of news-content-topics, style and gender. *Digital Journalism*, 1(1):102–116, 2013.

[11] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. A novel word spotting method based on recurrent neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 34(2):211–224, 2012.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[13] S. Jaeger, C. L. Liu, and M. Nakagawa. The state of the art in japanese online handwriting recognition compared to techniques in western handwriting recognition. *Int. J. Doc. Anal. Recognit.*, 6(2):75–88, Oct. 2003.

[14] D. Keysers, T. Deselaers, H. A. Rowley, L.-L. Wang, and V. Carbune. Multi-language online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[15] J. Kim and B.-K. Sin. Online handwriting recognition. In *Handbook of Document Image Processing and Recognition*, pages 887–915. Springer, 2014.

[16] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2741–2749. AAAI Press, 2016.

[17] A. L. Koerich, R. Sabourin, and C. Y. Suen. Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis & Applications*, 6(2):97–121, 2003.

[18] L. Likforman-Sulem, A. Zahour, and B. Taconet. Text line segmentation of historical documents: a survey. *International Journal on Document Analysis and Recognition*, 9(2):123–138, 2007.

[19] A. Marcus, A. Parameswaran, et al. Crowdsourced data management: Industry and academic perspectives. *Foundations and Trends® in Databases*, 6(1-2), 2015.

[20] J. Puigcerver, A. H. Toselli, and E. Vidal. Icdar2015 competition on keyword spotting for handwritten documents. In *13th International Conference on Document Analysis and Recognition (ICDAR), 2015*, pages 1176–1180. IEEE, 2015.

[21] T. M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9(2):139–152, 2007.

[22] V. Romero, V. Alabau, and J. M. Benedí. Combination of n-grams and stochastic context-free grammars in an offline handwritten recognition system. In *Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Part I*, IbPRIA '07, pages 467–474, Berlin, Heidelberg, 2007. Springer-Verlag.

[23] J. A. Sánchez, V. Bosch, V. Romero, K. Depuydt, and J. de Does. Handwritten text recognition for historical documents in the transcriptorium project. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pages 111–117. ACM, 2014.

[24] J. A. Sánchez, G. Mühlberger, B. Gatos, P. Schofield, K. Depuydt, R. M. Davis, E. Vidal, and J. De Does. transcriptorium: a european project on handwritten text recognition. In *Proceedings of the 2013 ACM symposium on Document engineering*, pages 227–228. ACM, 2013.

[25] J. A. Sánchez, V. Romero, A. H. Toselli, and E. Vidal. Icfhr2014 competition on handwritten text recognition on transcriptorium datasets (htrts). In *Conference on Frontiers in Handwriting Recognition (ICFHR), 2014 14th International*, pages 785–790. IEEE, 2014.

[26] A. Stolcke et al. Srilm-an extensible language modeling toolkit. In *Interspeech*, volume 2002, page 2002, 2002.

[27] S. Sudholt and G. A. Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *15th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2016*, pages 277–282. IEEE, 2016.

[28] A. H. Toselli, A. Juan, and E. Vidal. Spontaneous handwriting recognition and classification. In *ICPR (1)*, pages 433–436, 2004.

[29] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13(2):260–269, 1967.