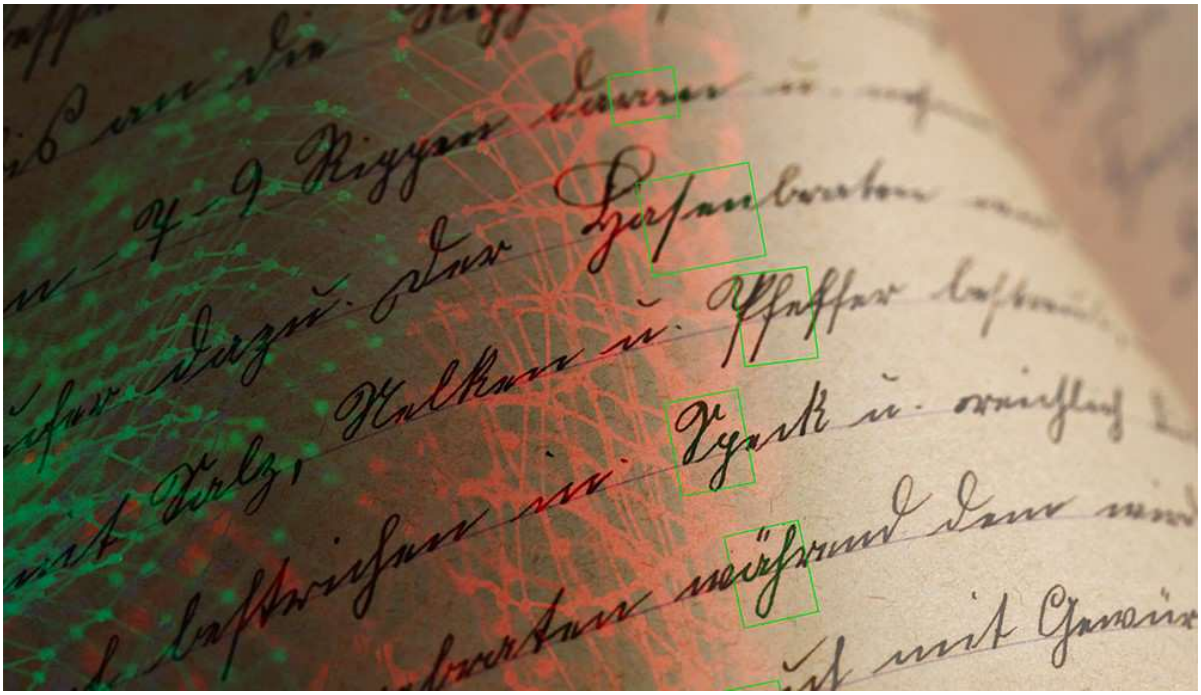
Università degli Studi di Salerno

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Tesi di Dottorato/Ph.D. Thesis

N-gram Retrieval for Word Spotting in Historical Handwritten Collections

Giuseppe De Gregorio



Supervisor: **Prof. Angelo Marcelli**

Ph.D. Program Director: **Prof. Pasquale Chiacchio**

Dottorato di Ricerca in Ingegneria dell'Informazione
Ciclo 35 – anno 2023

Università degli Studi di Salerno

.DIEM

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Dottorato di Ricerca in Ingegneria dell'Informazione
Ciclo 35

TESI DI DOTTORATO / PH.D. THESIS

N-gram Retrieval for Word Spotting in Historical Handwritten Collections

GIUSEPPE DE GREGORIO

SUPERVISOR: **PROF. ANGELO MARCELLI**

PHD PROGRAM DIRECTOR: **PROF. PASQUALE CHIACCHIO**

Anno 2023

Contents

Chapter 1

Introduction

The use of Artificial Intelligence for document management is one of the most interesting topics of digital transformation, which has attracted interest in a wide variety of fields in recent years. The availability and increasing diffusion of IT platforms capable of processing large amounts of data allow the complete or partial automation of the flow of information, ranging from the acquisition, classification, archiving and analysis of documents, to opening up solutions that were impossible until a few years ago. In the context of archives of historical material, this digitisation process represents a radical revolution and offers solutions that are fundamental to the future of humanistic studies. This new vision makes it possible to access historical material of interest quickly and easily through the online publication of the contents of the collections held in the institutions. Moreover, when the digitisation process is structured and technologies are well integrated, libraries can offer innovative forms of analysis and interaction with manuscripts that were difficult or impossible before the advent of these new technologies. On the other hand, digitisation processes can also be helpful in the internal management of library documents by simplifying research, indexing and collection handling processes, leading to a significant reduction in time.

In this context, the concept of the "*digital library*" is now a solid and es-

tablished one. According to the Digital Libraries Federation (DLF) [140]:

> «*digital libraries are organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use by a defined community or set of communities.*»

It is interesting to note that the digital library concept thus encompasses not only the technologies that enable the shift to the digital library but also *the institution's hardware resources, personnel and proprietary knowledge.* It is therefore important that not only paper documents are digitised, but that the entire structure is prepared to deal with the new data format and that the staff is trained to handle the data in order to fully exploit the potential of this new architectural and organisational form.

This generational shift certainly affects large institutions and organisations, but small businesses attracted by the promise of making the work product more efficient and powerful are also rapidly gaining interest. While for large organisations hardware and human resources are not a problem for the digital crossroads, for small organisations and archives this could be an insurmountable problem. Hardware resources of small organisations are often limited and processing large amounts of data can be difficult. Furthermore, modern image processing techniques based on artificial intelligence or deep learning technologies not only sometimes require special and specific hardware, but also special skills and thus the presence of highly qualified and trained staff to fully exploit the potential of the technologies used. A further limitation arises from the size of the collections of interest to small institutions. Often these institutions keep collections of documents and manuscripts consisting of a few dozen to a few hundred pages at most, but despite their size, they can be of great interest to the community of scholars. Data limitation is one of the biggest obstacles to the application of techniques based on artificial intelligence, and this fact can

limit and hinder the digitisation process of these special collections, which in the case of collections of historical documents are anything but rare.

Under these conditions, it is obvious that systems that can adapt to small collections and therefore work satisfactorily with limited amounts of data, that do not require expensive and specific hardware, and that are easy to use even by non-highly qualified staff, could be of great interest and prove to be the necessary weapon for the digital conversion of small institutions, archives and libraries. The process should integrate the solutions by providing a simple, intuitive user experience with a quick learning curve. This could encourage organisations to choose one system over another, regardless of the quality and performance of the individual technology used. The whole process needs to be designed and conceptualised around the user experience, with the aim of making digital transition management processes simple and applicable to different, even very heterogeneous, document collections, always bearing in mind that an institution's goal is to simplify and speed up information management processes.

In this work, we will try to address problems related to the treatment of handwritten documents of historical and cultural interest, taking into account the considerations presented in this short introduction, and thus try to present solutions that can be applied in a simple way and provide valid and meaningful results to the user. We begin the discussion by presenting a model for evaluating the performance of a document transcription system based on the KeyWord Spotting (KWS) technique, focusing on the time required to obtain an error-free transcription of the entire document. Measuring performance in terms of time, or better in terms of time saved by using assisted transcription technologies compared to fully manual transcription, is entirely in the interest of institutions. A library that needs to provide the correct transcription of an ancient text may find it difficult to assess the effectiveness of an automated transcription system if it is rated on precision/recall or WER or CER indices. On the other hand, evaluating performance in terms of the time saved by using this system can

provide an immediate and simple interpretation.

Continuing the discussion, we present a semi-automatic procedure for creating a reference dataset that can be used to train ML models useful for processing small collections of handwritten documents. Small collections of historical documents often have characteristics typical of the collection itself, which can mean that stable solutions in other domains do not apply very well to the collection itself. A classic example of this behaviour is the poor performance of OCR solutions applied to cursive handwritten documents of historical interest. Although the OCR problem is considered solved for modern printed texts, this is absolutely not true in the case of handwritten texts and especially for historical documents. Historical handwritten documents can have very different characteristics depending on the time in which they were written or the language of the document, and the characteristics of the handwriting can make segmentation into characters an extremely complicated process. These problems make the use of OCR impossible and lead to the need for other technologies to solve the problem of handwriting recognition. In order to find solutions to such problems, it is often necessary to create reference data sets that are representative of the collection to be analysed, starting from the transcription of part of the collection itself. The process of labelling such data can be complicated and extremely time-consuming. We, therefore, propose a procedure that guides the user through the entire process of data labelling in order to obtain a correctly labelled dataset while minimising human effort.

Finally, we present a keyword spotting system designed for small collections of handwritten documents. KWS systems usually suffer from the problem of "Out-Of-Vocabulary" (OOV) words, i.e. query words that the system cannot spot because it does not know a representation for them. The problem of OOV words presents itself as overwhelming when managing collections with small dimensions, as there are few data available to create a set of reference words large enough to limit the probability of encountering OOV words. The advanced idea is to present a word spotting system that uses sequences of

two or three characters as recognition primitives instead of whole words or single characters. Compared to whole words, the use of character sequences reduces the probability of obtaining sequences outside the dictionary, assuming the same number of labelled pages for the construction of the reference set, while the consideration of sequences compared to single characters simplifies the problem of segmentation, which in some cases can be very complex when done at the character level. Moreover, the decomposition of words into sequences can allow the system to retrieve words that would have been OOV in a classical word-based system.

Below we summarize the main contributions that will be presented in detail in the next chapters:

- Chapter 3 - Performance Model

  This chapter presents a mathematical model that aims to estimate the time needed to obtain a complete and correct transcription of a collection of handwritten documents when a KWS system is used to support the process. The model makes it possible to calculate the time gain a user can expect when using an automatic system to support transcription compared to a completely manual process.

- Chapter 4 - Tools and Methodologies to Speed up the Labelling Data Process

  Labelling images of handwritten words with their transcription is a very time-consuming process for the user. Speeding up the process can be crucial for the application of automatic techniques in transcription. This chapter presents some methods aimed at speeding up the process, presenting following methods:

  - Line-Segmentation method:
    the proposed line segmentation method is a learning-free algorithm that allows us to manage lines of text with a curvilinear trend and is able to label all ink traces detected on the document.

  - Transcript alignment algorithm:
    automatically aligning a transcript to images of individual words on a page of a handwritten document can speed up the process of word segmentation and correct annotation. The alignment algorithm described in this section allows us to align the transcription of entire lines of text.

- Chapter 5 - KWS by N-gram Retrieval
  in this chapter, we describe a keyword spotting system based on an N-gram retrieval procedure. N-gram retrieval can help alleviate the problem of Out-Of-Vocabulary word recognition and improve the performance of word spotting in small collections of handwritten documents.

Finally, in Chapter 6 the experimental evaluations for the methods presented are reported, and Chapter 7 is dedicated to some discussions, conclusions, and future expansions.

Chapter 2

Historical Document Processing - An Overview

2.1 The Writing and the Digitization

Of all the achievements of mankind, writing is undoubtedly one of the most important, if not the most important. Without writing there would be no history, there would be no knowledge, and there would be no progress. We call "history" all the events and changes in the past that took place after the invention of writing and of which we have testimony through documents and engravings.

Writing is the representation of linguistic expressions by graphic signs. According to many scholars, the invention of writing lies precisely in the need to "store" purely linguistic expressions. A common opinion is that writing arose from the need to keep track of accounts, as evidenced by preserved examples of writing from ancient Egypt, China, and Central America. It is not possible, however, to fix an exact date for the invention of writing; rather, it must be seen as the result of a development over a long period of time. One theory that is particularly accepted by the community is that writing came into being

to replace a counting system that was based on clay "tokens". Following this idea, the earliest forms of writing replaced real objects with representations of those objects [98]. The transition from pictographic writing to complete writing begins with the introduction of ideograms , in this way the signs are no longer pictographic representations of objects but become ideograms that represent an identifying feature of the object or of the concept. Another step forward occurred thanks to the discovery of the "rebus principle", which envisages using a pictogram or an ideogram for its phonetic value and grouping different signs to obtain a representation of a complex sound. Based on this idea, the first alphabetic and syllabic symbols representing specific sounds are introduced [99].

Although the earliest examples of writing date back to the Sumerians around 3000 BC, the majority of scholars believe that the authorship of writing cannot be attributed to one population but that the idea of writing developed independently in the different civilisations wide spread across the planet. Despite the fact that the continuous development of writing enables it to adapt to every historical, cultural and social context that characterises each epoch and population, on the other hand, it limits its interpretability over time. It is therefore not surprising if ancient writings are completely incomprehensible, while others can be understood only by a small circle of scholars and experts.

The long history of writing goes hand in hand with the long history of writing media [89]. In the course of time, we find writings on stones and clay from the second millennium BC, stone slabs and wood were used from the third millennium BC until the tenth century, and finally fabrics, papyri, and papers were chosen over millennia. The transition from one material to another is slow and varies depending on the civilisation. In most cases, evolution is the result of economic and practical needs. For example, parchment replaces papyrus, which had to be imported from Egypt because it could be easily made almost anywhere. Then paper replaces parchment because it simplifies the printing process and allows books to be distributed more easily. Today, we are witnessing

a new development of the carrier material, in which paper is increasingly being reduced in favor of digital support.

The replacement of paper with digital support makes it possible not only to reduce the amount of paper used but also brings several advantages. The paper suffers from natural degradation processes, and as the paper deteriorates, the information stored is also lost. The digital format of information overcomes this problem by allowing more effective and efficient processes for securing and storing information. In addition, the paper medium limits access to information; to access the content of the document, the paper must be manipulated, thus inducing further deterioration and alterations that may hamper access to the content. Digital information, on the other hand, is designed to overcome transport barriers, does not interact physically with the individuals accessing them, and copies of a digital document can travel enormous distances in a few seconds. The use of digital documents has thus reduced the physical space needed to store documents and makes it easier to search for a particular document and, more generally, to access information.

The benefits of representing information digitally are obvious and justify the decision to create new information in digital form. However, humanity has been engaged in collecting information for thousands of years, and most of this knowledge is still kept on paper. Thus, the need arises to obtain a representation of this knowledge in a digital format to reap all the benefits it offers. In this view, the field of document processing and analysis acquires considerable importance, and the need for documents in digital format is becoming more and more widespread. Digitization, however, has a broader definition, it does not simply aim to photograph or scan a document to obtain a digital image but pushes itself to obtain a representation of the document that is digitally understandable and interpretable. The problems to be addressed involve layout analysis, semantic segmentation and object recognition, optical character recognition (OCR) or handwritten text recognition (HTR), and generally automatic or assisted transcription.

Libraries, museums, monasteries, and various archives have been engaged in scanning more and more historical documents over the last twenty years. The massive presence of document images has increased the need to perform document analysis and transcription in digital format, increasing the importance of document processing in the field of historical document analysis [34]. Ancient document images represent an extraordinary class of document images. These documents are characterized by certain aspects dictated by the times. These aspects affect specific issues that make the analysis and study of this class of documents a far more complex process than the analysis of modern documents. As a result, the processing of historical documents presents special challenges that are not usually present in processing other types of documents. A strong influence is dictated by the quality of the original documents, which is often far from optimal. The deterioration of the document, the ink or the support material, or the presence of holes in the paper make it difficult or even impossible to interpret part of the document. The historical nature of documents often manifests itself in extremely complex layouts, vocabulary, or rare languages. Moreover, the typical handwriting of archaic scripts is challenging to interpret due to rare fonts, specific symbols, irregular spacing, and lack of spaces between words and ligatures. A remarkably topical problem is the extreme variability that is typical of historical documents. Documents written at different times or in different geographical regions may have extremely different characteristics. This leads to a lack of annotated and validated data. The goal remains to get the transcription of even relatively small collections of documents from which little labelled data is available, and the only way to increase the labelled data is to obtain the transcription, and the circle keeps turning.

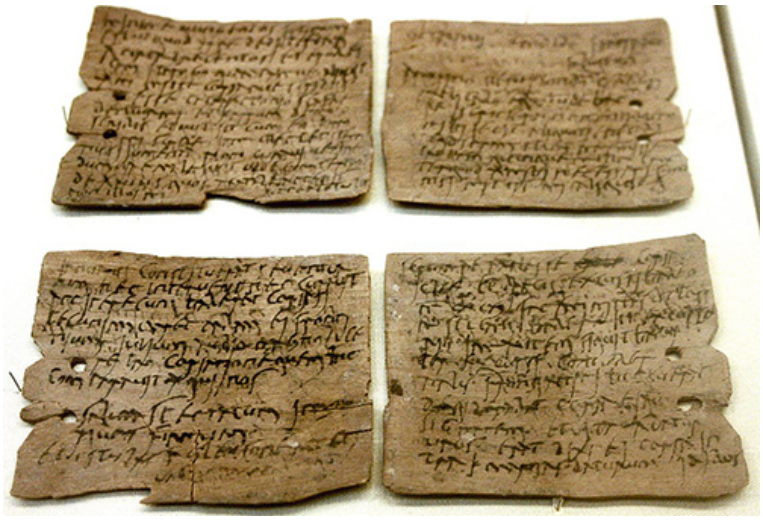
Automatic document processing is based on Artificial Intelligence and Machine Learning technologies and it is acquiring increasing popularity. The applicability of these techniques, mainly due to the increase in computing capacity, has made it possible to obtain exciting results even on types of documents that seemed highly complex until a few years ago. Towards the end of the first

decade of the 2000s, the term Deep Learning (DL) concealed the third wave of artificial intelligence, still ongoing today. This wave has also impacted the field of text recognition in printed or handwritten text. Lombardi and Marinai in [63] point out that since 2013, the number of scientific papers published in this field using DL approaches has steadily increased, reaching over 80% of published papers in recent years. The productive use of DL technique is based on the possibility of having large amounts of data on which to train its models. The impossibility of creating large datasets limits the application of these promising techniques to the field of transcription of small collections of historical documents. Specific solutions for dealing with collections of documents consisting of a few dozen pages then take on an important weight. Working with a small collection means that one needs to use methods capable of managing and coping with the scarcity of data while guaranteeing the success of operations. In this context, it is important never to forget a basic but extremely important rule: If we have to work with a few dozen documents, manually transcribing the contents of the entire collection can still be the cheapest and fastest solution to most of the problems.

2.2 A Classification of Historical Documents

Since man began to record his exploits, an enormous amount of documents has been collected. The first traces of writing can be dated back to around 3000 BC and from that time onwards the various civilisations that followed have always made writing a fundamental aspect. Over time, different techniques and media have developed and today we can benefit from a huge number of documents. The civilisations of different eras and geographical regions have produced very different written documents, and the differences are not only due to the use of different languages but also to different styles, rules, techniques, technologies and materials. To this day, all documents created on media or techniques that are no longer used are considered historical. The analysis of

these types of documents is therefore more difficult than that of contemporary documents [63]. Below is an overview of the main types of historical documents studied by scholars and researchers today.



Source: `https://en.wikipedia.org/wiki/Vindolanda_tablet`

Figure 2.1: Roman wooden writing tablet from the Roman fort of Vindolanda of Hadrian's Wall, Northumberland (1st-2nd century AD).

2.2.1 Earlier Documents

The first documents were produced on media different from paper. Various materials were used, such as dried leaves, stone tablets, bamboo scrolls, and papyrus. Among the oldest writings on leaves, we now find manuscripts from the 9th and 5th centuries BC from Southeast Asia and the Indian peninsula [53, 136]. The earliest records on wooden or bamboo scrolls were produced between the 5th and 3rd centuries BC in eastern China [124]. In Egypt, papyrus was widely used from the 4th century B.C. Among the oldest and best-preserved collections on papyrus, we find the Diary of Merer, dated 2550 B.C. [72].

2.2.2 Inscriptions

The intention of the text is often to pass on the memory of a historical event, figure or action. Media such as paper, parchment or papyrus were not suitable for this purpose because they are not very durable. Inscriptions made it possible to overcome this limitation by using durable materials such as marble, stone or metal. In Southeast Asia in the first millennium, important documents were engraved on soft metal sheets such as copper plates, which were softened by fire and engraved with a metal stylus. In Burma, the Kammavaca, the Buddhist manuscripts, were engraved on sheets of brass, copper or ivory. In Italy, some important Etruscan texts were similarly engraved on thin gold plates, and similar sheets were discovered in Bulgaria. Classical Greek and Roman civilisations made extensive use of inscriptions, typically to publicly display a text important to society. As a rule, inscriptions were made in capital letters and were characterised by the use of special linguistic registers that were embossed and solemn. In the Middle Ages, the use of inscriptions began to decline and gradually diminished until they disappeared.

2.2.3 Manuscripts

The term manuscript refers to handwritten texts that may be written in a book, on parchment or in codex format. Over time, different materials have been used as carriers. Commonly, vellum or other types of parchment, papyrus and paper were used, but there is much evidence of other materials. In Russia, for example, documents made of birch bark from the 11th century have survived, or in India manuscripts made of palm leaf can still be found, dating from antiquity to the 19th century. In the 14th century, paper spread from China to Europe via the Islamic world [6], and by the end of the 15th century, it had largely replaced parchment for many purposes, becoming over time the main medium for writing in all civilisations.

The golden age of manuscripts is typically the Middle Ages, but the pro-

Source: https://en.wikipedia.org/

Figure 2.2: The Nora Stone, the most ancient Phoenician inscription ever found (in 1773) outside Phoenicia proper.

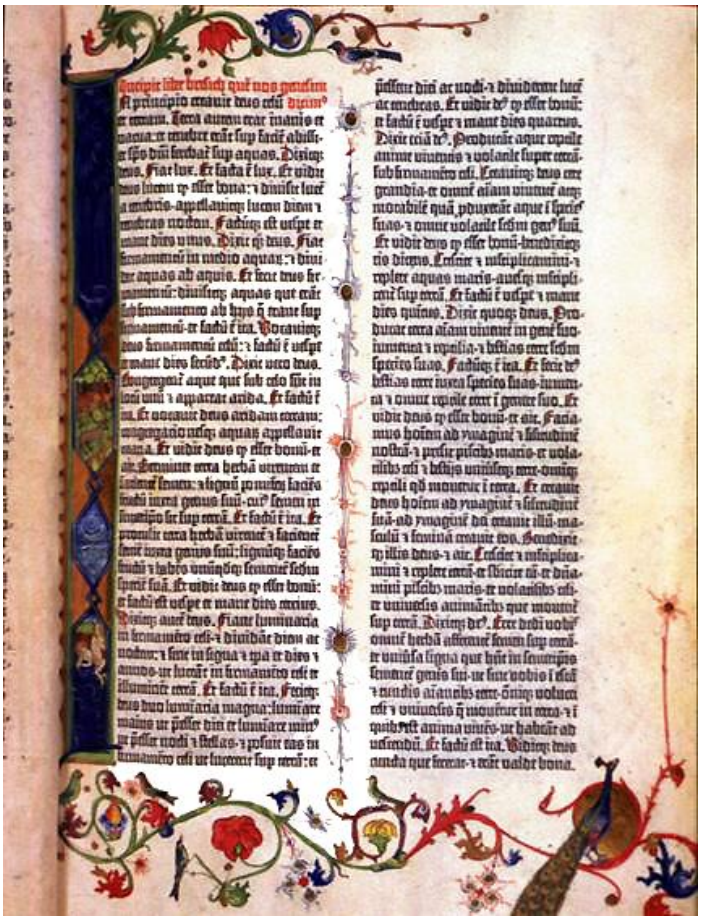
liferation of manuscripts extended into the early modern period. In the Western world, from the classical period to the first centuries of the Christian era, manuscripts were written without spaces between words (*scriptio continua*), often using a very difficult-to-understand style of writing, making these documents particularly difficult to read for untrained people. The fact that the pages contain very complex and structured ornaments and decorations also makes them difficult to understand. Various studies have analysed the structure of these documents, some of them to discern the text they contain [2, 45, 90, 134, 145]. Existing copies of these early manuscripts, written in Greek or Latin and usually dating from the fourth to eighth centuries, are classified according to their use of upper or lower case letters. There is no such distinction for Hebrew manuscripts, such as the Dead Sea Scrolls. Manuscripts

Source: https://en.wikipedia.org/

Figure 2.3: Manuscript of the medieval period with examples of rubrication.

that use only upper case letters are called *majuscule* letters, and manuscripts that use only lower case letters are called *minuscule* [80]. Majuscule, such as in the uncial script, are usually written much more carefully. The scribe raised his pen between strokes, creating a distinctive effect of regularity and formality. On the other hand, while lowercase letters may be written with the nib raised, they may also be italicised, which means that the nib is raised only slightly or not at all.

Even after the invention and spread of printing, handwritten documents continued to be very important. Private documents, letters or government documents remained handwritten until the invention of the typewriter in the late 19th century.



Source: http://commons.wikimedia.org/

Figure 2.4: Image of a page from Gutenberg's bible printed in the 1450s.

2.2.4 Incunabula

After the introduction of printing with movable type by Gutenberg in Europe, the first printed books began to spread and the first works printed with movable metal type until 1500 were called incunabula.

The first real book printed by Gutenberg was the Bible, and this was no accidental choice. European manuscripts were produced for religious purposes, and most of the books copied were in some way connected to the Catholic religion. Such works were not simply meant to be read, they also had spiritual

value. For this reason, the pages of the manuscripts were often decorated with complex decorative elements. For this reason, the "42-line Bible" printed by Gutenberg is not actually a fully printed work, but was designed so that the decorations could be added by hand after the text printing was completed. With this technique, the first printed books were produced in exactly the same style as the manuscripts, but printing made it possible to do the work more quickly, which led to a drastic reduction in the price of books. [47]

Since the incunabula were initially printed in exactly the same style as the manuscripts, they have all the typical features of earlier manuscripts. The use of different typographical signs depending on genre and region, the use of contractions and abbreviations in sentences, the frequent use of columns and marginal notes and the rubrication at the beginning of the chapter can thus be traced.

2.3 The Workflow of the Historical Document Processing

The workflow for processing historical documents involves several steps, which are shown in Figure 2.5. The first step shown in the figure refers to the image acquisition phase. This often requires special hardware and procedures and can be considered a separate topic before the actual document processing phase. The figure shows the process at a general level, but it should be noted that there is no single flow that applies to all types of documents. The architecture of a specific analysis system depends heavily on the intended application and the type of documents under consideration.

The first phase of the process is that of 'pre-processing', which involves improvement and enhancement techniques that can also be very demanding, as in the treatment of severely degraded documents. Degradation is one of the main obstacles to accurate analysis, and sometimes pre-processing can only improve the situation without being able to completely eliminate the problem.

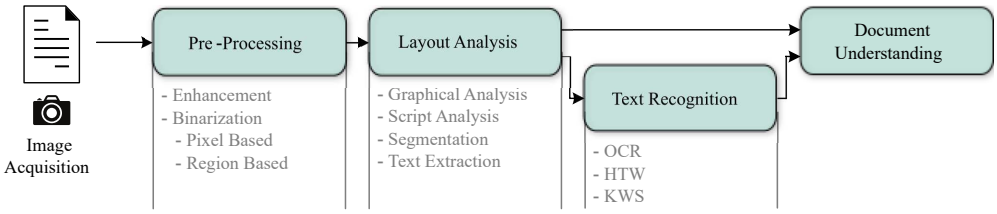


Figure 2.5: Typical Historical Document Processing Workflow.

For example, if there are large holes in the document, it may be impossible to recover the original information. In most historical documents, the writing is produced by depositing the ink on paper, so it seems natural to convert them into black-and-white before further processing to reduce their complexity. In the pre-processing phase, it may then be necessary to convert an image in a general colour space into a black-and-white image. To this end, binarization techniques allow black-and-white images to be obtained by stretching between background pixels and text pixels.

The pre-processing phase is followed by the layout analysis step, which consists of analysing the pages of the documents to obtain a description of the structure of the manuscript that allows the identification of regions of interest. An important aspect of this stage is the separation of text from non-text areas (such as ornaments, images or large initials). Once the textual areas are identified, it may be useful to divide the blocks into lines of text and again into individual words. Finally, the different areas identified can be related to each other by establishing the links that can connect them, defining, for example, the natural reading order or the alignment properties of complex layouts. For historical documents, layout analysis is much more challenging than for contemporary documents. In the Middle Age, for example, the text content is often mixed with ornaments and decorations that can take up a large part of the page. Many technologies focus their application on individual pieces of text or images of individual words. For this reason, once the text areas have been identified, only the images of the lines or words of interest need to be extracted. Segmentation methods aim to do just that, and can output images containing

a single line or word.

Once the images of text lines or words have been extracted, it is possible to apply text recognition techniques. At this point, it is important to distinguish between printed text and handwritten text. For the first case, OCR (Optical Character Recognition) technologies have proven their worth. These technologies are based on the ability to identify and recognise individual characters and then combine the results to obtain the interpretation of whole words or sentences. Printed text is characterised by a constant division of space and a low variability of the symbols of the alphabet, and it is mainly these characteristics that make the use of OCR effective. The recognition of handwritten text is a much more complex process and, despite the progress made in recent years, it is not yet ready for widespread use. One of the main reasons for the problems is that it is difficult to recognise and isolate characters to be able to recognise them, which is especially the case with cursive handwriting. Special techniques have been developed for recognising handwritten text, known as HTR (Handwritten Text Recognition) techniques, which are usually applied to whole words or to sequences of words. This leads to a significant increase in complexity and often requires contextual knowledge provided by linguistic models, dictionaries of reference terms or semantic knowledge such as domain-specific ontologies. For some applications, transcription of the whole test is not strictly necessary. In these cases, an interesting alternative to text recognition is the word spotting approach. In this case, the goal is to find the position of some keywords in the text without the need for explicit recognition. In this way, the complexity of the task is reduced, which can allow for an automated application. These techniques are known in the literature under the acronym KWS (KeyWord Spotting) systems.

The final phase of the document processing workflow is document understanding, although this is sometimes an optional stage. This phase usually uses both the results of layout analysis and text recognition. The aim is to analyse the document from the point of view of its logical structure and thus iden-

tify the different logical areas such as titles, headings, annotations, captions, images, paragraphs or chapters.

As mentioned earlier, machine learning (ML) and deep learning (DL) techniques are becoming increasingly important to solve problems at every stage of the document processing flow. In this perspective, great importance is given to data and the possibility of accessing databases with truth annotations. In this way, it is possible to quantitatively evaluate algorithms and ML or DL models. Obtaining high-quality annotated data is often a delicate and complicated task, often documents have to be annotated completely manually, and the historical nature of the documents makes this process possible only by a small number of experts. For this reason, several tools and solutions exist to automate the process of creating records and generating ground-truth as much as possible. However, today there are several publicly available datasets that can be used to test and implement techniques for each stage of the workflow.

In the remains of the chapter, we discuss some of the solutions available in the literature for each step of the workflow, Finally, the main datasets available are described later in this chapter.

2.4 Pre-Processing Techniques

The purpose of the pre-processing stage is to improve the quality of the document in order to facilitate processing and interpretation, and in the case of processing historical documents, this stage is crucial. Documents of historical interest may be in a poor state of preservation, which can make the processing of the document more challenging. Figure 2.6 shows examples of typical characteristics of the poor state of preservation of historical documents. There may be noticeable stains or damage to the paper or parchment, the ink may be faded or, on the contrary, it may be so pronounced that it is visible even on the opposite side of the page. In addition, it is possible that the scanning or photo-scanning process was not performed optimally. The most important operations

in the pre-processing phase are image enhancement, greyscale thresholding and binarization.

(a) stain (b) faded text

(c) permeated ink (d) paper damage

(e) character alternation (f) bad scan quality

Source: Rasyidi and Khan [93]

Figure 2.6: Historical documents affected by a variety of degradations.

2.4.1 Enhancement

Historical documents usually show damage, such as tears, scratches or holes in the paper or parchment. Many documents are in a poor state of preservation, and many are characterized by areas no longer legible or parts of the page being torn or damaged. It is often important to recognise such damage and correct it if possible. Cleaning techniques such as denoising can often help to improve

the condition of the images. Neji et al. [75] propose an end-to-end adversarial autoencoder (AAE) to clean documents from noise capable of cleaning the document from small damages. The proposed AAE uses a GAN (generative adversarial networks) so as to suit the aggregated posterior of the hidden code vector of the autoencoder with an arbitrary prior.

Lu and Dooms in [64] propose a damage detection system for document images that exploits information about the homogeneity patterns of the text both globally and allows the identification of image pixels associated with a damaged physical document. Nguyen et al. [76] propose a character attention generative adversarial network (CAGAN) to restore degraded characters in historical documents. The network is based on a U-Net architecture [102] and uses a loss function that considers common adversarial loss as a global loss and hierarchical character attentive loss as a local loss. The same problem is addressed in [135], where authors present a method to reconstruct the broken letters using an autoregressive generative model designed for image restoring called PixelCNN++ [105].

In the pre-processing stage, another important aspect regards the correction of distortion introduced during the image acquisition. Dewarping and skew reduction techniques allow to reduce or eliminate distortions that occur during the digitization of the document. Dewarping and skew reduction methods are usually applied to well-defined areas of the text. However, they have also been proposed in studies that apply corrections at the level of the entire document, allowing the correction of documents that are also characterized by non-textual content [92, 139]. Xi et al. [143] propose a framework based on the use of a convolutive neural network (CNN) for both the rectification of the distorted image of the document and the fine removal of the background. The network is trained with synthetically distorted documents, and experiments are conducted to test how the method can handle different types of distortions. Li et al. [62] propose a CNN-based method that can correct different types of distortions from a single input image. The method applies different corrections to different

areas of the image, performing a kind of local correction of distortions.

2.4.2 Binarization and Grayscale Thresholding

Binarization is the conversion of a colour or grayscale image to a black and white image. The importance of the binarization process in documents of historical interest lies in the fact that it is possible to minimize the impact of degradation and non-uniformity of the image background with this technique. An important approach to binarization is to classify at the pixel level. In [130], Tensmeyer and Martinez apply a Fully Convolutional Network to classify each pixel of an image. This approach has proven remarkably accurate leading to results with high performance. In [138], the authors propose a supervised binarization method based on a hierarchical Deep Supervised Network (DSN). Pixel classification is performed at different feature levels, this allows higher levels to distinguish text pixels from background noise so that it is possible to handle the severe degradation that occurs in document images. In binarization, the use of approaches based on deep networks has shown significant advantages, not only in terms of results but also in terms of processing speed. The classic solutions were based typically on sliding window approaches, and the sliding operation is an expensive operation, with deep approaches, it is possible to use a single network to process the entire document speeding up the entire process [63].

Among the notable classical methods, the Otsu threshold method [81] is worth mentioning. This type of thresholding uses techniques for analysing image histograms and is particularly suitable for bimodal images, i.e. images whose histograms show a clear separation between two main peaks. The acquisition of images of historical documents is not a standardised process and therefore the bimodal format of the images is not guaranteed. For this reason, in the cases we are interested in, it is better to use local thresholding techniques such as the methods of Niblack [77] and Sauvola [108]. These techniques are useful for images where the background is irregular and especially for images containing text. The classical methods were not developed with any particular

interest in documents of historical interest, but they can still be acceptable solutions in many cases.

2.5 Layout Analysis

Document layout analysis is an important task when creating collections of digitised documents. In the case of historical documents, this can be particularly challenging due to the complexity and variability of the layout, the intermingling of different elements and the decay of the documents. Typical layout analysis tasks include *page segmentation*, *line segmentation* and *baseline detection*. Page segmentation makes it possible to recognise within the images of a document the different regions, as for example the background, parts of text blocks, comments, embellishments, images, etc. Line segmentation consists of extracting only the text lines of the main text from the document. In the case of handwritten cursive writing, this process could be challenging because of the extremely skewed and curved trend of the text and inconsistent spacing between lines. Finally, baseline detection is the process by which we can identify the baselines of the text lines on the image of the document. By the baseline, we mean the ideal line on which the text line was written.

2.5.1 Page Segmentation

Layout analysis often refers to the task of dividing pages into the different sections that make them up. This involves dividing the document into semantic areas such as text areas, titles, paragraphs, annotations, images, and so on. Page segmentation is an active area of research and several competitions have been held recently [14, 37, 114].

2.5.2 Text Line Segmentation

Text line segmentation aims to identify the area corresponding to each line of text. While identifying lines of text is a relatively simple process in modern

printed documents, it is a highly complex task in handwritten documents. The complexity, especially in historical documents, lies in the irregular alignment of text lines, the complex management of space, and the presence of margin illustrations or notes. Chen and Seuret [12] propose a CNN with a single convolutional layer. Segmentation of text lines is achieved by super-pixel labelling, where each pixel is classified as background, main text, decoration, or comment. The authors report results characterized by an accuracy up to 90%. In [85], Pastor-Pellicer et al. use the Main Body Area (MBA), i.e., the area between the main idea and the line of continuous text, to detect lines of text. First, each image pixel is classified as background, text, or decoration using a CNN. Then, a second CNN classifies the pixels into text blocks by detecting the different MBAs. The method is among the few that use Deep Learning techniques and shows improvement over other techniques compared to it. Renton et al. [96] use a deep approach based on Fully Connected Networks that work on the entire document and avoid the approaches based on sliding windows. The comparison with the results obtained with classical methods highlights the goodness of the method and emphasises the improvement in computation time. Alberti et al [3] propose a segmentation method which utilises a pixel-level semantic segmentation step followed by a text-line extraction phase. The method achieves state-of-the-art performance in various datasets, proving particularly suitable for segmenting medieval documents.

2.5.3 Baseline Detection

Recently, the baseline detection method has been preferred to line segmentation [22, 24]. In this case, the task is to recognise the baselines of the text, which is a trade-off between the cost of annotation and descriptive performance.

Fink et al. [29][45] present a U-net based convolutional neural network for baseline detection in historical documents performing a sliding window dense prediction. In [44], Grüning et al. also use a U-net based architecture and focus on recognising even lines of text with a very pronounced curvature, adding an

attention mechanism and developing a sophisticated post-processing phase.

2.5.4 Multi-Purpose Solutions

The typical tasks of layout analysis vary, as we have seen so far. In recent years, some architectures have been introduced that allow several of these tasks to be handled simultaneously, thus simplifying the whole process and reducing the number of tools required.

In [144], Xu et al. propose a segmentation method for historical manuscript documents that allows the different areas of the document to be segmented by classifying each pixel of the image and distinguishing between background, main text, comments and embellishments. The method is based on a fully convolutional neural network (FCNN) architecture. After the pixel classification performed by the network, some post-processing steps allow for improving the results by reducing noise and correcting segmentation errors. Oliveria et al. [79] present *dhSegment*, a pixel-wise predictor based on CNN that allows us to handle different activities such as text area extraction, baseline identification or image extraction. The network architecture used an encoder-decoder approach that exploits the description capability of ResNet [46] and the localisation capability of U-net [102]. Monnier and Aubry present in [73] the tool *docExtractor*, which is presented as a generic approach to extract lines of text and embellishments without the need for annotation of real data. The tool is based on a neural network architecture very similar to that of dhSegment [79] but advances the idea of being able to perform the initial training with a fast methodology for synthetically generating correctly annotated documents.

While the methods also show interesting results by training architectures on general or synthetically generated datasets, they often require a phase of fine-tuning on the collections of interest to achieve optimal performance, which could be an obstacle for some processes when dealing with collections of documents of limited size.

2.6 Text Recognition

The text recognition phase attempts to obtain a literal transcription of the document content, or of part of it. The main techniques used for this purpose are optical character recognition (OCR) and handwriting text recognition (HTR). Optical character recognition is based on the possibility of subdividing words into characters and the following classification for their recognition. The character is then used as a fundamental recognition unit. Typically, cursive handwriting cannot rely on regular spacing because of the variability of handwriting. Therefore, handwriting recognition usually relies on recognition methodologies that are free from character-level segmentation.

2.6.1 Historical OCR

Different techniques allow the optical recognition of historical characters, however, techniques based on neural networks are currently the most used. Optical character recognition is applicable when it is possible to obtain a good segmentation at the character level. For that reason, this technique is widely used with incunabula documents (i.e., a document printed with the movable type technique between the 15th and 16th centuries.). These documents have been modelled on manuscripts and are characterized by extensive use of ligatures, complex layouts, and often typographical abbreviations that do not have a corresponding equivalent in Unicode [104]. For all these reasons, optical recognition of historical characters is drastically more complex than modern optical character recognition [120].

2.6.2 HTR Techniques

With the introduction of hidden Markov models (HMM) in the field of text recognition, there has been a breakthrough in offline recognition techniques [87]. With these models, it was possible to focus not on single characters but on strings and then on words or whole sentences. In the last decades, neural net-

works have attracted more and more attention and over the years different techniques and solutions have been proposed [63]. Among the first promising deep approaches, methods based on Connectionist Temporal Classification (CTC) [40] have been proposed. CTC is a type of neural network output with an associated scoring function, typically used for training recurrent neural networks (RNNs) such as LSTM networks. Graves et al. were among the first to propose this approach for handwritten text recognition. With the works [41,42], they propose a BLSTM recurrent network followed by CTC to recognise handwritten text in English, which shows an improvement over HMM-based approaches. The same idea is further developed in several subsequent works. Pham et al. [86] present a multimodal LSTM (MDLSTM), followed by CTC. Shi et al. [113] combine CNN and BLSTM, followed by CTC, and introduce the CRNN model. Bluche et al. [8] propose a variation of this model by introducing a gated version, GCRNN, which is used for multilingual recognition.

An alternative approach is based on the use of seq2seq models with the attention mechanism. Sueiras et al. [127] present a seq2seq model that uses a sliding window for text recognition. Bluche et al. [7, 15] present an attention-based end-to-end model using an MDLSTM network. Similarly, Ly et al. [65,67] propose an attention-based seq2seq model using a residual LSTM network for text recognition in Japanese historical documents. Zhang et al. [146] present an attention-based seq2seq model with a CNN encoder and a GRU decoder.

With the success of the self-attention and transformer [137] architecture, researchers have recently started to evaluate this architecture for text recognition problems. In this direction, Kang et al. [52] presented a CNN transformer model for handwritten text recognition. Finally, Ly et al. [66] propose a recurrent network with a self-attention mechanism for recognising handwritten Japanese text.

2.6.3 Keyword Spotting - An alternative to the HTR

An alternative to full-text recognition is to identify some keywords within one or more documents. By keyword spotting (KWS) we mean the task of finding the positions in the images of documents that correspond to a word in the search query. Keyword identification provides a less restrictive framework than traditional handwritten text recognition, so this technique has good potential in cases with less reliable or smaller data. With the KWS technique, it is possible to index, search and analyse historical documents, even if normal text recognition techniques are not very effective.

Among the first works to explore the KWS in the context of historical documents, we find the works of Rath and Manmatha [94], where they used a dynamic time-warping algorithm to compute image similarity and compared the performance of Ward linkage and k-means algorithms on the George Washington dataset. Fischer et al. [32] present a word-spotting system based on character Hidden Markov Models. The proposed system is tested on several datasets: the IAM database for modern writing, the George Washington database, and the Parzival database for historical writing. The method outperforms the previous KWSs, suggesting that the improvement is due to the use of HMM.

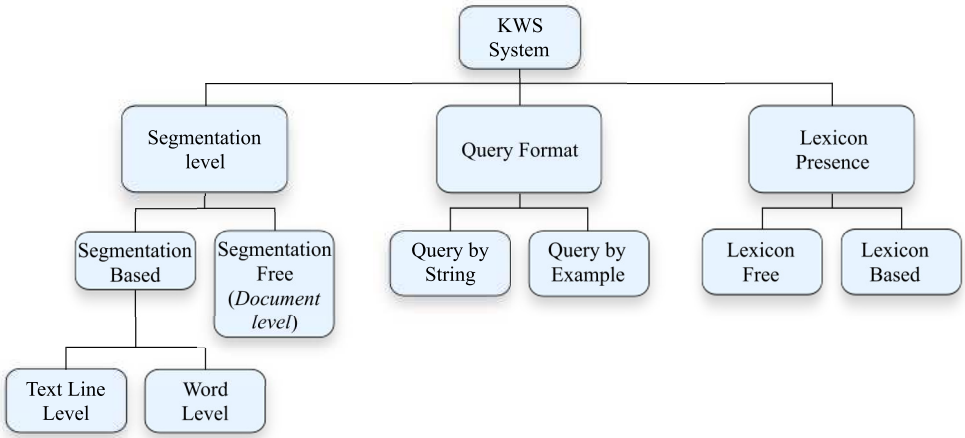


Figure 2.7: Categorization for KWS systems.

From these first simple applications, the KWS systems have evolved, with different variants and features. In Figure 2.7 we show the possible categorization for a KWS system. The first differentiation concerns the level of segmentation required by the spotting method. Systems that do not include a segmentation phase are referred to as segmentation-free systems. On the contrary, when a segmentation phase is envisaged, which aims to detect areas of interest to search for keywords. In this case, these systems are referred to as segmentation-based. Segmentation can be done at the text line level or the word level. The first involves extracting every single line of text present in the document, while the second extracts all the words. A further grouping of KWS systems can take into account how the system is queried. The query can consist of a word image or a text string representing the transcription of the word to search. When the query is in the form of an image, it is referred to as Query-by-Example (QbE) system. These systems search within the document for zones that have visual features similar to the features of the query image. If the query is in the form of a text string, it is referred to as Query-by-String (QbS) system. In this case, a region is searched within the document containing the query word's image. In this case, it is necessary to link the space of representation of the images with that of the textual transcriptions. A further distinction is made between lexicon-based and lexicon-free systems. The former relies on the presence of a predefined keyword list, fixed during the training phase, while the latter does not rely on such a list and it can search for a generic word.

A problem with the KWS technique is that it can only recognize words for which at least one reference image is known. Words that are contained in a document but not known by the KWS system, called Out Of Vocabulary (OOV) words, cannot be identified and transcribed. To overcome this limitation Fisher et al. [33] apply a character-based recognition with hidden Markov models to the KWS. Authors compare the character-based system to the dynamic time-warping system, founding that the Hidden Markov models outperform the Dynamic Time Warping system in terms of the mean aver-

age precision. Frinken et al. [36] use a Recurrent Neural Network (RNN) as the basis for the KWS system. In this way, they demonstrate that the use of RNN surpasses a classic dynamic approach based on time warping and a system based on hidden Markov models. In [126], Sudholt and Fink present a KWS system based on a CNN network. By taking a probabilistic perspective on training CNNs, the authors derive two loss functions for binary and real-valued word string embeddings. They also propose two different CNN architectures designed explicitly for word spotting. Granell et al. [39] evaluate the use of deep techniques such as Bi-directional Long-Short Term Memory (BLSTM) and Convolutional Recurrent Neural Nets (CRNN). Results on the Rodrigo dataset show that CRNN outperforms HMM and BLSTM, achieving the lowest Word Error Rate (WER) and Character Error Rate (CER) for this dataset, significantly improving OOV recognition. The trade-off underlined is that for deep neural network architectures to be competitive in time efficiency with other techniques, they require significant computing power.

CNN networks, due to performance, are the networks that are most establishing themselves in the field of document analysis. These networks work by producing in their output a suitable descriptor of the image of the word query. In [4], Almazan et al. propose an attributes-based approach that leads to a low-dimensional, fixed-length representation of the word images. The approach proposes to embed the feature of a word encoding the word strings as a Pyramidal Histogram of Characters PHOC. The histogram encodes whether a character appears in the word or not. The spatial pyramid adds a rough localization, defining in which part of the word the character appears. The PHOC representation is computed for each level of the pyramid, and the final PHOC histogram is the concatenation of these partial histograms. In [125], Sudholt and Fink present a CNN architecture trained with PHOC descriptors known as PHOCNet. The network is a CNN architecture designed for word spotting. PHOCNet can process images with generic dimensions and it can predict the PHOC representation of the word image. The authors show the proposed archi-

tecture is capable of surpassing cutting-edge results by exhibiting short training and testing times.

2.7 Dataset

This paragraph gives an overview of the datasets of historical documents available in the literature. Figure 2.8 shows a diagram in which the different datasets are ordered on the basis of the historical reference period and the main use case. Distinguishing the different datasets based on the use case is important because the annotated truth needs to be provided in different ways depending on the problem to address. For example, if the goal is layout analysis, we need to know the structure of the document. On the other hand, if we are interested in text recognition, we need to know the transcription of the words contained in the images. The importance of truth annotations is reflected both in the fact that ground truth is necessary to provide learning samples for machine learning algorithms and in the fact that truth enables the performance of proposed algorithms to be automatically evaluated. In this discussion, we focus on collections of Western historical documents in various languages such as mediaeval Latin, mediaeval German and Spanish, various early modern European languages, and eighteenth-century English.

2.7.1 The IAM-HistDB

The IAM-HistDB comprises three different databases, each with a different style and language. The three databases are the Saint Gall database, the Parzival database and the George Washington database. The collection was created as part of the HisDoc project at the IAM Institute of the University of Bern in Switzerland and is one of the collections first made available to the research community [34].

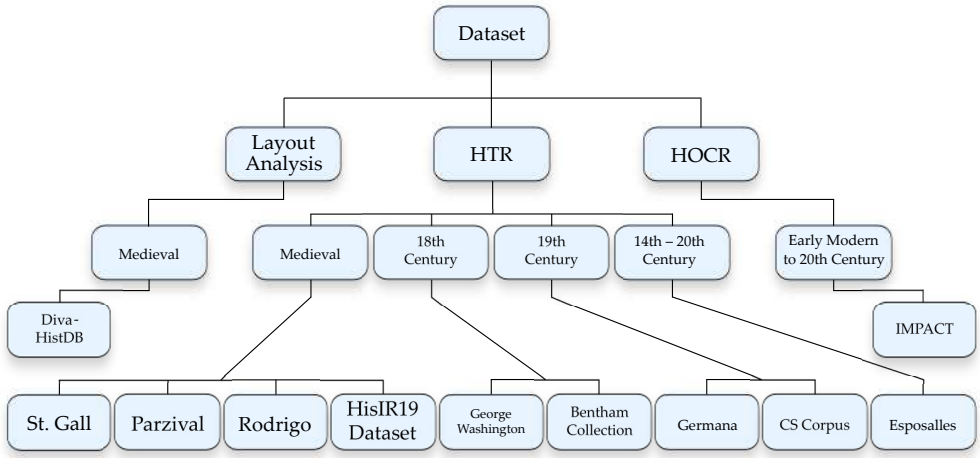


Figure 2.8: Main data sets of historical documents organized by use case and time period.

2.7.2 Saint Gall

The Saint Gall dataset[1] contains images of a 9th-century Latin manuscript written by a single scribe in the Carolingian script. The original manuscript, a hagiography of Saint Gall, comes from the Abbey of Saint Gall in Switzerland, where the original version of the documents is still preserved.

In addition to the page images and the transcript, the record contains extensive ground truth. The lines of text are annotated by drawing their position on the pages as closed polygons, and there is a corresponding transcription for each line. The transcription is provided in two ways: The first faithfully reproduces the content of the pages, while an adapted version of the modern text edition is also available, often deviating from the manuscript text by slightly changing abbreviations and punctuation for better readability. The St. Gall database is characterised by a particularly regular page layout. All pages have a single-column text with 24 lines for each page. However, the pages have marginal notes, coloured and decorated initial letters, holes in the parchment, ink smears and missing spaces between words. First, the dataset was proposed

[1] https://fki.tic.heia-fr.ch/databases/saint-gall-database

with the aim of matching the transcription with the images of the words with the work of Fischer et al. [31].

Figure 2.9: Some examples of the Saint Gall dataset.

2.7.3 Parzival

The Parzival dataset[2] contains handwritten pages of an Arthurian epic in Old German from the 13th and 15th centuries. The forty-seven images of the Parzival come from three different manuscripts, created by three scribes using a tiny Gothic script in a multi-column layout. The Parzival has a two-column layout with rhymed lines in pairs. The presence of ornaments surrounding and embellishing the areas with the main text is striking. The signs of decay are very present on the parchment, and stains, holes and faded ink are visible. The Parzival collection contains pre-processed page illustrations and associ-

[2]https://fki.tic.heia-fr.ch/databases/parzival-database

ated transcripts at the line and word level. There are no annotations on the positions of text zones or lines of text, let alone decorations. Pre-processing consists of a binarisation step and the removal of distortions.

Figure 2.10: Some examples of the Parzival database.

The database was introduced by Fischer et al. in [35] with the aim of transcribing the lines and images of words.

2.7.4 George Washington

The George Washington database[3] was one of the first collections shared with the community. It consists of 20 handwritten page images of letters written by George Washington and his associates during the American Revolutionary War. The original letters are stored at the United States Library of Congress. Writing is a continuous cursive script in English used in the 18th century. The dataset

[3]https://fki.tic.heia-fr.ch/databases/washington-database

was provided by Rath et al. [94] to be then used mainly for text recognition and keyword identification. Truthful annotations include transcriptions of images of lines of text and individual words. The layout of the documents is not very regular and there are often page numbers, lines, stamps and signatures next to the main text. In addition, the suboptimal state of preservation manifests itself in faded ink and stains on the paper, which makes the task of the recognition systems more difficult.

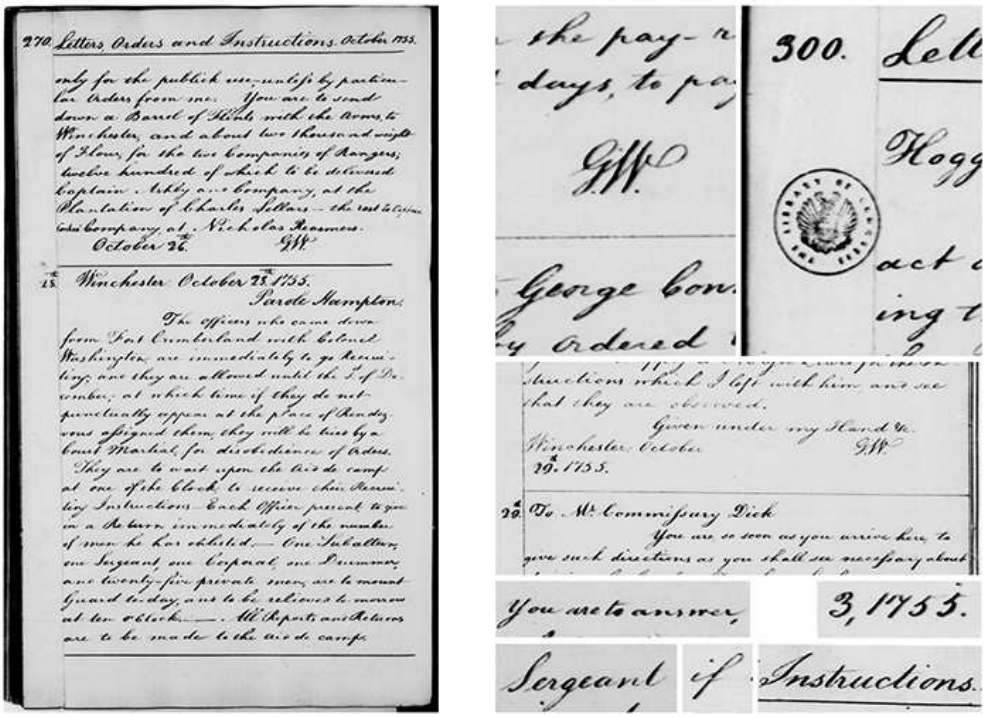


Figure 2.11: Some examples of the George Washington dataset.

2.7.5 The HisIR19

The HisIR19 dataset[4] [13] is the data set used in the ICDAR 2019 Competition on Image Retrieval for Historical Handwritten Documents. The dataset focuses on writers from the European Middle Ages, from the 9th to the 15th

[4]https://tc11.cvc.uab.es/datasets/HisIR19_1

centuries. Most of the material is anonymous. It is made up of 1200 images by 520 different writers. Three hundred authors contribute with one page, 100 authors with three pages, and finally, 120 contribute with five pages. The test dataset contains 20,000 images: approximately 7,500 pages are from isolated documents, and approximately 12,500 are from authors who contributed three or five pages.

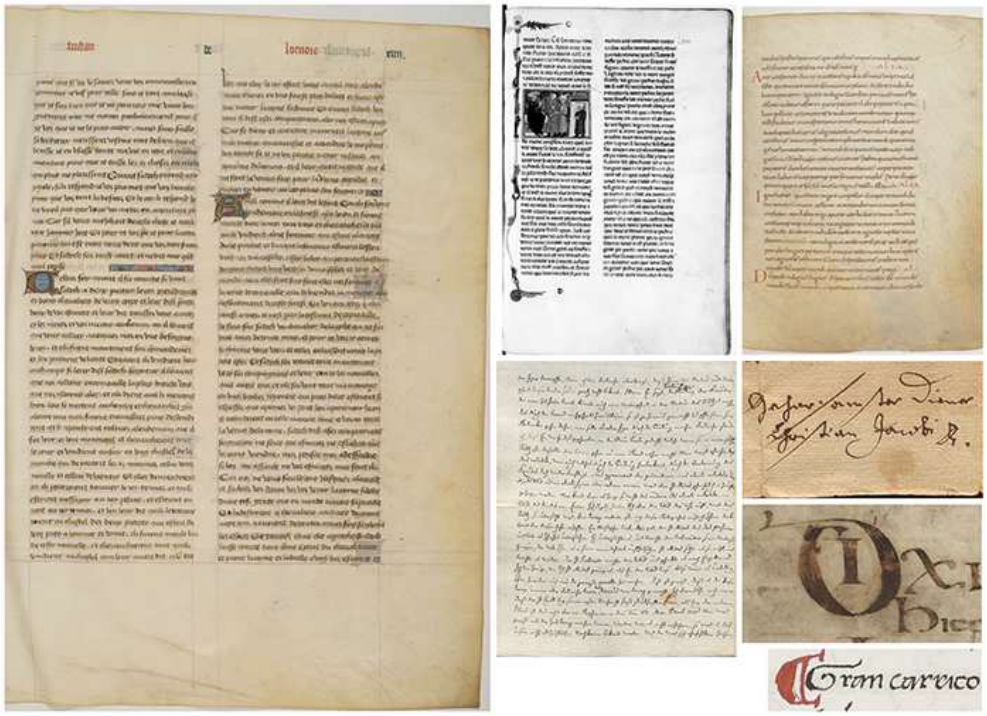


Figure 2.12: Some examples of the HisIR19 dataset.

2.7.6 Bentham Collection

Bentham collection[5] is a large set of documents that were written by the English philosopher and economist Jeremy Bentham (1748-1832). The transcription of this collection is currently being carried out by amateur volunteers participating in the award-winning crowd-sourced initiative Transcribe Bentham. Currently,

[5]https://zenodo.org/record/44519

more than 6,000 documents have been transcribed. The dataset was proposed in the ICFHR competition in 2014 [107]. The dataset provides images of whole pages, and the ground truth includes information about the layout and the transcription at the line level of each image. The Bentham dataset is part of the tranScriptorium project [106]. This project aims to develop innovative solutions for the transcription of images of historical handwritten documents. The project focuses on the English, Spanish, German and Dutch languages, and the Bentham collection is the dataset relating to the English language.

Figure 2.13: Some examples of the Bentham Collection dataset.

2.7.7 Germana

Germana [91] was obtained by digitizing and annotating a 764-page Spanish manuscript entitled "Noticias y documentos relativos a Doña Germana de Foix, ultima Reina de Aragon", written in 1891 by Vicent Salvador. It contains about

21,000 lines of text manually transcribed by paleographers. Most of the pages consist of only cursive text written on sheets with well-separated lines. Due to its sequential book structure, this dataset is particularly suitable for the realistic evaluation of interactive handwriting recognition systems.

2.7.8 Rodrigo

The Rodrigo corpus[6] is an old Spanish dataset, and it is composed of 853 pages. Created like the preceding datasets for handwriting recognition and line extraction research, the researchers based at the Universitat Politecnica de Valencia used the digitized images of an Old Spanish historical chronicle, the "Historia de España del arçobispo Don Rodrigo." The layout of the documents is quite linear, with most pages containing a single block of almost calligraphic script on well-separated lines. However, the old Gothic style of writing can make recognition difficult.

2.7.9 Cristo-Salvador Corpus

The CS Corpus ("Cristo-Salvador") is a database derived from the 19th-century manuscript known as "Cristo-Salvador", made available by the Biblioteca Valenciana Digital (BIVALDI). The manuscript entitled "Noticia histórica de las fiestas con que Valencia celebró el siglo sexto de la venida a esta capital de la milagrosa imagen del Salvador" by Vicente Boix is a small document consisting of 53 coloured images of pages compiled by a single author. The corpus is a single book by one author with a single column layout and pictures in the text. Some of the pages show smudging, background variations and differences in brightness.

[6] https://zenodo.org/record/1490009

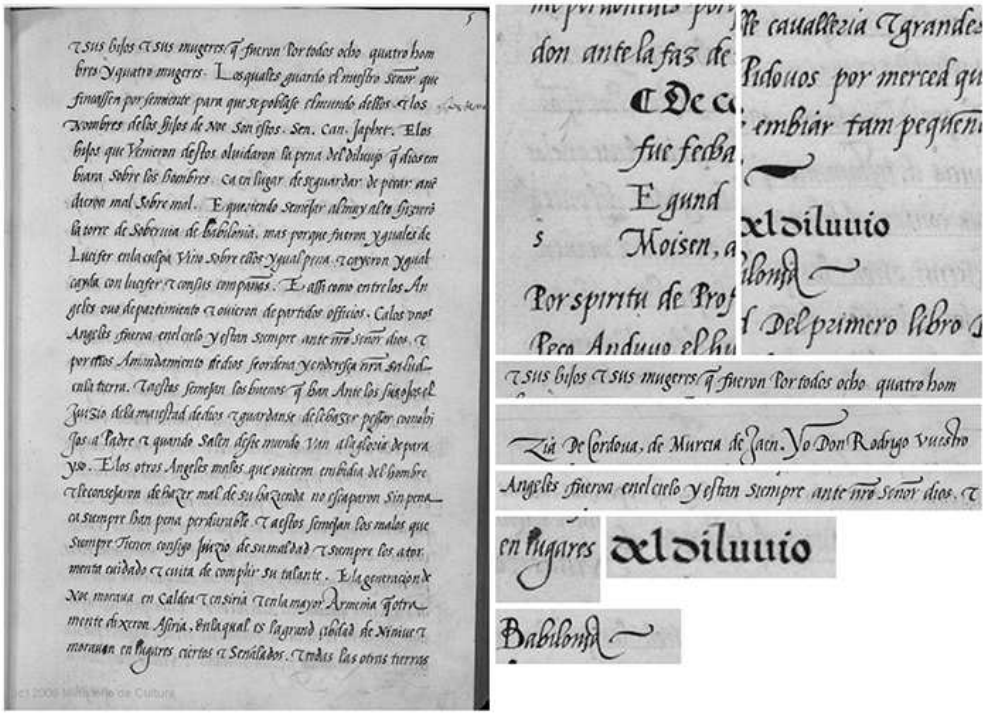


Figure 2.14: Some examples of the Rodrigo dataset.

2.7.10 Esposalles

The Esposalles dataset[7] [100] consists of handwritten pages of marriage license books kept in the archives of the Barcelona Cathedral. These documents span around 500 years, up to the beginning of the 20th century. The dataset consists of a volume that collects the licenses comprising 173 pages, 1,747 registers, 5,447 lines. The whole image of each page is available, as well as the images of the page lines. The full transcript is also available. There is another set consisting of indexes of two volumes. It consists of 29 pages. Each page is divided vertically into columns, and each column is divided into rows, for a total of 1,563 rows. For each column of the page, there is an image for each row together with its transcription.

[7] http://dag.cvc.uab.es/the-esposalles-database/

Figure 2.15: Some examples of the Esposalles dataset.

2.7.11 Diva-HistDB

As a benchmark for evaluating pre-processing performance on mediaeval documents, the HistDoc project has created Diva-HistDB[8]. This dataset contains 150 pages of images from three different manuscripts with associated background truths for binarization, layout analysis and line segmentation [115]. The three manuscripts were selected for their complexity in layout. Two of the manuscripts are written in Carolingian characters and date from the 11th century while the other one is from the 14th century. All three manuscripts consist of a single column of text surrounded by very large marginal notes, furthermore, some pages have decorative initials. The ground truth focuses on the identification of spatial and colour-based features and consists of four different annotated classes that may overlap: Main Text, Decorations, Notes

[8]https://www.unifr.ch/inf/diva/en/research/software-data/diva-hisdb.html

and Background. Ground Truth is encoded in the format PAGE XML. The dataset is available free of charge on the HistDoc project website.

Figure 2.16: Some examples of the Diva-HisDB dataset.

2.7.12 IMPACT

While the preceding datasets for medieval handwriting recognition were limited to a handful of manuscripts written in Latin and early vernacular languages from only a few countries, researchers and archivists working with early modern optical character recognition have a much more comprehensive resource in the IMPACT[9] dataset [82]. Created by a consortium of European libraries, the IMPACT dataset provides a genuinely diverse, pan-European collection that includes texts from the fifteenth through the twentieth centuries in eighteen different European languages and a variety of distinct scripts. Printed books

[9] https://www.digitisation.eu/resources/impact-dataset/

comprise the majority of the collection, including newspaper pages, legal documents, journals, and an assortment of miscellaneous documents. In addition to document metadata, the extensive set of ground truth for this collection includes both the full document text in Unicode along with layout analysis annotation and reading order specified with the XML-based PAGE format.

2.7.13 Codice Ratio

The Codice Ratio[10] is a research project that aims to propose new methods and tools for document processing. The project focuses on the collections of the Vatican Secret Archives, one of the world's largest and most important historical archives. Researchers built their dataset to train their classifier for the recognition of handwritten medieval documents [30]. The dataset is composed of single characters and the corresponding annotation. The characters were chosen from several manuscripts, and synthetically generated data was added. In total, the dataset contains examples of 1,000 characters for each character class. Character images are included in a dataset, while the ground truth consisting of PNG images of words and transcripts of text files are included in a separate dataset.

2.8 Tools and Software Platforms

In this section, we will give a quick overview of the main tools and software platforms available in the field of handwriting recognition useful for the application of the processing of documents of historical and cultural interest.

2.8.1 Tools

The most important tools for historical optical recognition are Abbey FineReader, Tesseract, OCRopus and AnyOCR. There are also a few other generic tools for the historical recognition of handwriting. Tesseract was initially proposed

[10] http://www.inf.uniroma3.it/db/icr/index.html

for the recognition and transcription of present-day documents. It was later broadly utilized also with historical documents [43, 71, 78]. Tesseract does not include any built-in pre-processing tools. The tool at first recognizes the lines of text, also handling slatted or skewed lines. Later it performs the recognition of words by separating the characters [117]. In this way, a segmentation stage is then performed. Next, a list of potential character matches is made, computing the similitude of the bit vector between the unknown character and every possible match. OCRopus, recently renamed OCRopy, was also created for text recognition in modern documents and subsequently applied to historical ones [121]. The pre-processing step involves binarization, noise removal, skew correction, page segmentation, and layout analysis to detect columnar layouts. Finally, the tool identifies the lines of text, and segments them at the character level [112]. Individual characters are classified, and hypothetical interpretations are represented in the form of graphs. The classifier bases its decision on traversing the graph with the lowest cost. Thanks to this method, the OCRopus classification is very accurate in the case of anomalous instances and can better distinguish upper- and lower-case characters. The software includes the integration of linguistic models. These can improve the classification of characters. However, few linguistic models are available for ancient languages [121]. OCRoRACT aims to improve accuracy by incorporating Tesseract and OCRopus into one system. A significant limitation of these systems is the need to use a massive amount of data for the training that experts must generate with a manual transcript [88]. Furthermore, the collections of documents are often not very large. A small set of training is used to train Tesseract. Using the data trained with Tesseract, the OCRopus classifier is then trained. This process is repeated until the improvement rate between iterations is less than 1%. Following this approach, Ul-Hasan et al. [88] obtained a character error rate (CER) on par with that obtained using OCRopus trained on a huge training set. AnyOCR follows the same hybrid approach as OCRoRact [10, 49]. However, the Tesseract classifier is not used and is replaced with an unsupervised

segmentation-based classifier. The idea of the unsupervised classifier was taken because, in this way, the manual annotation step is not necessary. The classifier uses a variant of the k-means clustering algorithm and groups images based on a "blurriness" metric. The results in terms of error rate on a printed Latin text of the fifteenth century are little better than the results obtained by OCRopus. However, there is a considerable advantage in the interaction time required by expert users.

2.8.2 Software Platforms

A few important software systems include several features, such as preprocessing, machine learning training, and transcription. Among these systems, there are DIVA-Services [142], Transkribus [50], and eScriptorium [54]. DIVA-Service is a web-based service built on a RESTful architecture that provides an API for each stage of historical document processing. The service provides document image analysis algorithms, machine learning algorithms, and text recognition tools. The choice of a RESTful architecture is justified by the desire to create a service that is easy to use and manage, allowing simply the integration of DIVA services in different software and tools. DIVA-Services offers a suite of standard tools. There are layout analysis methods that allow the extraction of points of interest, lines of text, and images. Moreover, it has extraction techniques such as scale-invariant transformation (SIFT), or Gabor filters for features extraction are available. DIVA-Services also integrates the OCRopus and Tesseract libraries for optical character recognition. Finally, different machine learning classification algorithms are implemented, such as support vector machine (SVM), Gaussian mixture methods, and K-Nearest Neighbours (KNN). DIVA-Services is an open-source project, which is why it is very suitable and used in research projects involving customized software. Transkribus provides all the functionalities of the document transcription process. The software has a client-server architecture. A software client communicates with a central server via a RESTful API. The software allows the user to work with

document images or with pre-existing layouts or transcription data that have been stored in PAGE XML or ALTO formats. It allows handling document segmentation manually or with the application of layout analysis techniques. It is possible to use both HTR and OCR capabilities. ABBY Fine Reader SDK is used for OCR, and two methods are available for HTR. The first is based on HMM and allows to incorporate a linguistic model, while the second is a tool based on RNN, which allows facing the problem of OOV words. eScriptorium is an open-source web-based platform for the analysis and annotation of historical documents. It allows you to upload document collections, segment them and transcribe them manually or automatically with the help of an OCR engine. The purpose of eScriptorium is to provide a framework for organizing and transcribing handwritten documents and make transcription results available for academic and research use. The platform aims to serve as an open-source alternative to Transkribus, and its web-based nature makes it easier to use than a RESTful solution. DIVA-Services, Transkribus, and eScriptorium offer similar functionality, but they better suit different needs. Transkribus offers a complete toolchain for the transcription process, and it allows to handle everything via the client interface. Furthermore, it uses different important used standards. These features make Transkribus the tool preferred by researchers and users looking for comprehensive document processing software. The most significant limitation of Transkribus is its open-source hybrid nature, which can result in severe usage limitations. eScriptorium wants to overcome this limitation and provides an entirely open-source framework with a simply usable web-based interface. DIVA-Services is instead focused on a RESTful architecture. This makes it perfect for users who want to integrate the services offered within customizable software. Furthermore, DIVA-service can boast a completely open-source approach. Therefore, the approaches of these software can be considered complementary, as they are addressed to different use cases and different target users.

Chapter 3

Performance Model

Digital libraries have evolved from a way to store and preserve documents to an integrated information processing platform and web applications that enable the storage, creation and manipulation of information and knowledge. This has required the development of specialised software tools for processing the digital images of the document to extract its textual content into a machine-readable format.

Much of the cultural heritage is in the form of small collections of handwritten documents, usually consisting of up to a few hundred pages written by a few different scribes and these collections are of particular interest to scholars. The accessibility of their contents has become an increasingly urgent requirement, and so librarians and public administrators have turned their attention to computer-assisted transcription because it promises to be faster and cheaper than classical methods involving only humans.

Assisted transcription systems are based at their core on handwriting recognition techniques. Since these tools have to deal with the enormous variability of handwriting found in the different collections of many scribes, they rely on complex tools that draw heavily on the techniques ML and DL [38]. However, to achieve high accuracy they require huge training sets, usually in the order of thousands of pages, and have been used successfully for large collections, of-

ten using crowd-sourcing for the training set labelling to make the overall cost acceptable. The technique of keyword spotting (KWS) promises to circumvent some of the disadvantages of explicit recognition by allowing a transcript of a word to be obtained without the need for recognition. This is essentially a match between the images of a training set, of which the transcription is also known, and the images of the document to be transcribed. Because of these features, KWS-based systems are better suited for processing small collections of documents.

Regardless of the technology used to implement the transcription system, the intervention of an expert user is mandatory to validate and/or correct the system output in order to obtain a complete and error-free transcription of the entire content of the document. This means that with the current state of the art, it is not possible to obtain a correct transcription of a document without human intervention, regardless of the system used and its performance. It is therefore interesting to assess whether the use of a KWS-type text recognition system can provide an effective advantage in the transcription of documents. We are therefore interested in determining the minimum performance requirements of the KWS so that the use of the system is beneficial in reducing the time taken by the user to complete the transcription of the document. In other words, we would like to answer the following questions: Is the KWS system so good that the time taken by the user to validate the output, in order to obtain a complete and correct transcription of the document content, is less than the time taken by the same user to transcribe it manually? If so, can we estimate by how much the user's time is reduced?

In this chapter, we present a model to define the conditions under which the use of the system is profitable, and we describe a procedure for estimating both the performance improvements and the accuracy of the estimate in terms of the actual improvement. For the discussion and definition of the model, we have chosen to use a multi-output KWS system, i.e. a system that for each word image provides an ordered list of its possible transcriptions. The proposed

model assumes that no information is available other than that obtained from the samples and the performance of the KWS system on the training set.

3.1 The Model

To introduce the proposed model, let us assume that the collection of documents to be transcribed has been segmented at the word level in order to extract the number n_{DC} of example images from the entire data collection. Let us further assume that n_{TS} samples from the collection were transcribed manually and used to define the training set and the reference dictionary of the KWS system. In this way, the dataset of samples for which a transcript needs to be provided to complete the activity consists of $n_{DS} = n_{DC} - n_{TS}$ samples. We also denote by N_{DC} and N_{TS} the number of keywords, i.e. the number of individual entries in the vocabulary associated with the data collection and the training set, respectively. Using a KWS system to support the transcription, it returns for each of the nDS instances an ordered list of a few possible transcriptions. To obtain the transcription, the user needs to choose the correct transcript from the list when it is available or type it if not. In this way, the number of actions required from the user to achieve the complete and error-free transcription is n_{DS}. According to our hypothesis that only information from the training set is available to the system, N_{TS} is known because the samples in the training set were labelled, while N_{DC} is not known.

We can note that TS and DS represent the two subsets of words to be transcribed from the entire collection and that the two sets generally partially overlap. This means that the system only knows a representation of the words that belong to the intersection for the set DS, but that in order to obtain a complete transcript, a transcript must also be provided for the words that belong only to DS. These words belong to the set of *Out-Of-Vocabulary* (OOV) words, which can represent a non-negligible percentage of the words in small document collections. If the system is not able to provide a transcription for

the OOV words, the user must transcribe them manually, and this process may involve a significant effort.

We can express the time needed to transcribe the content of the document collection using the KWS system according to the following expression:

$$T_u = T_{TS} + T_{out} + T_{miss} + T_{oov} \tag{3.1}$$

Here, T_{TS} is the time required to create the training set and transcribe the images that make it up, T_{out} is the time required to validate and/or correct the output of the KWS system, T_{miss} is the time required to transcribe the instances of words not retrieved by the system and T_{oov} is the time required to transcribe the OOV words manually. We can state that using the system to perform the transcription is beneficial if the time T_u is less than the time T_{man} required to complete the manual transcription of the data collection. Therefore, the condition must be fulfilled:

$$T_u < T_{man} \tag{3.2}$$

The manual time T_{man} can be expressed as the composition of two times:

$$T_{man} = T_{TS} + T_{DS} \tag{3.3}$$

We can then rewrite the condition (3.2) as:

$$T_{out} + T_{miss} + T_{oov} < T_{DS} \tag{3.4}$$

The time T_u is affected by the use of the KWS, and it depends to some extent on the performance of the system. Let us now focus on the behaviour of the KWS system while spotting the keyword w_i of the training set, by denoting with n_i^{DS} the number of instances of w_i in DS, with n_i^c the number of *correct* instances, i.e. those that the system correctly considered to be instances of w_i, with n_i^w the number of *wrong* instances, i.e. those that the system wrongly

considered to be instances of w_i, and finally with n_i^m the number of *missing* instances of w_i (Figure 3.1).

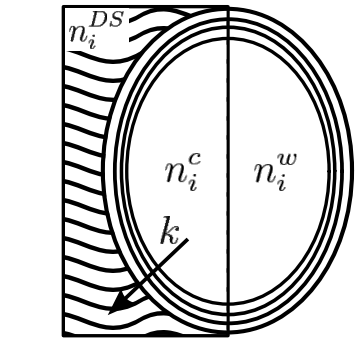


Figure 3.1: As the size k of the KWS system output list grows, the overall system performance changes allowing more words to be fetched.

When the KWS spots the word w_i, it returns the ranked list of k possible transcriptions. The measures used in the literature to evaluate the performance of an information retrieval system that provides k alternatives as output are the *recall*@k and the *precision*@k:

$$recall@k = R^k = \frac{\#RetrievedRelevantImagesAtk}{\#RelevantImages} \tag{3.5}$$

$$precision@k = P^k = \frac{\#RetrievedRelevantImagesAtk}{\#RetrievedImages} \tag{3.6}$$

The recall and precision indices of this KWS system for the keyword w_i can be expressed as:

$$r_i = \frac{n_i^c}{n_i^{DS}} \tag{3.7}$$

$$p_i = \frac{n_i^c}{n_i^c + n_i^w} \tag{3.8}$$

We can than reformulate the expressions of n_i^c, n_i^w and n_i^m in terms of recall and precision as k varies:

$$n_i^c = r_i^k \cdot n_i^{DS} \tag{3.9}$$

$$n_i^w = r_i^k \cdot (\frac{1}{p_i^k} - 1) \cdot n_i^{DS} \tag{3.10}$$

$$n_i^m = (1 - r_i^k) \cdot n_i^{DS} \tag{3.11}$$

If we denote by t_i^c the time required to validate a correct sample of the i-th keyword, by t_i^w the time required to produce a correct transcript for an incorrect sample of the i-th keyword, by t_i^m the time required to produce a transcript for a missing sample of the i-th keyword, and with t_i^M the time required to manually transcribe image of a generic word (i.e. a word of the traing set or an OOV words), we can write the expression for each of the four terms in equation (3.1) as follows:

$$T_{TS} = \sum_{i=1}^{N_{TS}} (t_i^M \cdot n_i^{TS}) \tag{3.12}$$

$$T_{out} = \sum_{i=1}^{N_{TS}} [(t_i^c \cdot n_i^c) + (t_i^w \cdot n_i^w)] \tag{3.13}$$

$$T_{miss} = \sum_{i=1}^{N_{TS}} (t_i^m \cdot n_i^m) \tag{3.14}$$

$$T_{oov} = \sum_{i=N_{TS}+1}^{N_{DS}} (t_i^M \cdot n_i^{oov}) \tag{3.15}$$

In the same way we can rewrite the term T_{DS} of equation (3.3):

$$T_{DS} = \sum_{i=1}^{N_{TS}} (t_i^M \cdot n_i^{DS}) + \sum_{i=N_{TS}+1}^{N_{DS}} (t_i^M \cdot n_i^{DS}) \tag{3.16}$$

We can now rewrite equations (3.13) and (3.14) as a function of the recall precision indices of the system using equations (3.9), (3.10) and (3.11):

$$T_{out} = \sum_{i=1}^{N_{TS}} ((t_i^v \cdot r_i^k) + (t_i^w \cdot (\frac{1}{p_i^k} - 1) \cdot n_i^{DS})) \tag{3.17}$$

$$T_{miss} = \sum_{i=1}^{N_{TS}} (t_i^m \cdot (1 - r_i^k) \cdot n_i^{DS}) \tag{3.18}$$

Equations (3.17) and (3.18) show that T_{out} and T_{miss} express a dependence on the performance indices of the system and are the only components of T_u that introduce this dependence. The equations also show that the effects of system performance on user time t_u are modulated by the times for validation, correction, and transcription of correct, wrong, and missing samples, which depend on the user interface of the system. Given the performance of the system in terms of r_i^k, p_i^k, and the mean value t^M of the times t_i^M required to transcribe the instances of the training set, equation (3.4) therefore allows us to determine the maximum values of t_i^v, t_i^w, and t_i^m that satisfy the condition for profitable use of the system. Conversely, given the user interface properties of t_i^v, t_i^w, and t_i^m, the equation allows us to calculate the minimum values of r_i^k and p_i^k that the KWS system must have in order to be profitable for the writer. In the following subsections, we will derive such conditions for lexicon-based and lexicon-free KWS systems.

3.1.1 Lexicon-Based Systems

In this case, as the KWS system is not able to find OOV words, the use of the system is profitable when

$$T_{out} + T_{miss} < T_{DS} - T_{oov} \tag{3.19}$$

which can be written in terms of the KWS performance as:

$$\sum_{i=1}^{N_{TS}} [(t_i^v \cdot r_i^k + t_i^w \cdot r_i^k \cdot (\frac{1}{p_i^k} - 1) + t_i^m \cdot (1 - r_i^k)) \cdot n_i^{DS}] < \sum_{i=1}^{N_{TS}} (t_i^M \cdot n_i^{DS}) \tag{3.20}$$

In the case of a perfect KWS system, i.e., a system for whom $r_i^k = p_i^k = 1, \forall i$, the inequality above becomes

$$\sum_{i=1}^{N_{TS}} (t_i^v \cdot n_i^{DS}) < \sum_{i=1}^{N_{TS}} (t_i^M \cdot n_i^{DS}) \tag{3.21}$$

which holds with certainty if $t_i^v < t_i^M, \forall i$. In the case of a real system, both r_i^k and p_i^k are less than 1 and the first and last terms of the sum on the left-hand side of the inequality (3.20) show an opposite tendency as r_i^k increases: the former becomes larger whilst the latter becomes smaller. The second term behaves complexly, but since $\frac{1}{p_i^k} > 1$ and in any information retrieval system recall and precision are such that when p_i^k increases r_i^k does not increase (and usually decreases), it becomes smaller as p_i^k increases.

3.1.2 Lexicon-Free Systems

Let us now consider the case where the KWS system is able to recognise words that are not included in the query list. To estimate the extent to which the system is able to spot OOV words, we assume that a test set (TSS) containing $n_{TSS} \approx n_{TS}$ samples of the data collection is provided to the KWS system trained on TS. We can divide the OOV found by the KWS system in TTS into two parts; These include the correct OOV, which consists of the OOV words that have an empty output list, and the wrong OOV, which consists of the OOV words that have a non-empty output list. Under the same assumptions as in the previous subsection, and denoting by $n_i^{oov^c}$ and $n_i^{oov^w}$ respectively the number of correct OOV and wrong OOV word images that are instances of the N_{oov} keywords, we can estimate T_{oov} as follows:

$$T_{oov} = \sum_{i=N_{TS+1}}^{N_{oov}} (t_i^M \cdot n_i^{oov^c} + t_i^{M_w} \cdot n_i^{oov^w}) \tag{3.22}$$

It is worth noting that $t_i^{M_w} > t_i^M$ because in the case of a wrong OOV, the user has to read the output list to find the transcription and only then start transcribing the word. In contrast, the time for transcribing the correct OOV is only the time for transcription because the output list is empty.

At this point, after interacting with the system, the user has used up the time:

$$T_u = T_{out} + T_{miss} + T_{oov} \tag{3.23}$$

where the times appearing on the right side are estimated using the Equations (3.20) and (3.22), respectively, and the query list is composed of $N_{TTS} = N_{TS} + N_{OOV}$ keywords. Thus, to achieve the transcription of the remaining samples of the data set, the user will spend the time T'_{out} for processing the output of the system and the time T'_{miss} for transcribing the missed words when spotting the N_{TTS} keywords, plus the time T'_{oov} for transcribing the OOV word, i.e., the word images that are instances of the $N_{DS} - N_{TTS}$ keywords. We can express these times as follows:

$$T'_{out} = \sum_{i=N_{TS}+1}^{N_{TTS}} (t_i^v + t_i^w \cdot (\frac{1}{p_i^k} - 1)) \cdot r_i^k \cdot n_i^{DS} \tag{3.24}$$

$$T'_{miss} = \sum_{i=N_{TS}+1}^{N_{TTS}} (t_i^m \cdot (1 - r_i^k) \cdot n_i^{DS}) \tag{3.25}$$

$$T'_{oov} = \sum_{i=N_{TTS}+1}^{N_{DS}} (t_i^M \cdot n_i^{oov^c} + t_i^{M_w} \cdot n_i^{oov^w}) \tag{3.26}$$

and thus:

$$T'_u = T'_{out} + T'_{miss} + T'_{oov} \tag{3.27}$$

To estimate T'_{oov} we need the value of N_{DS} and the values of $n_i^{oov^c}$ and $n_i^{oov^w}$. We can estimate them by assuming that the coverage of the query list computed on TTS with respect to the actual list, i.e., the ratio N_{TTS}/N_{DS}, is the same as the coverage of the query list computed on TS with respect to that of TTS,

i.e. the ratio N_{TS}/N_{TTS}, and thus $n_{DS} = \frac{N_{TTS}^2}{N_{TS}}$. Similarly, we estimate $n_i^{oov^c}$ and $n_i^{oov^w}$ by assuming that the distribution of OOV words between correct and wrong in DS is the same as in TTS.

Under these assumptions, we can divide the time T_{DS} for the manual transcription of DS into the time for transcribing the word images that are instances of the N_{TS} keywords obtained from the manual transcription of TS, the time for transcribing the word images that are instances of the N_{TTS} keywords obtained from the data in TTS, and the time for transcribing the remaining OOV words:

$$T_{DS} = \sum_{i=1}^{N_{TS}} t_i^M \cdot n_i^{DS} + \sum_{i=N_{TS}+1}^{N_{TTS}} t_i^M \cdot n_i^{DS} + \sum_{i=N_{TTS}+1}^{N_{DS}} t_i^M \cdot n_i^{DS} \tag{3.28}$$

We can now establish the condition for the profitable use of a lexicon-free KWS as follows:

$$T_u + T'_u < T_{DS} \tag{3.29}$$

This expression shows that the use of the KWS system can be profitable compared to manual transcription if the inequality (3.29) for the word images that are instances of the query list keywords holds to such an extent that the extra time incurred by transcribing the incorrect OOV words detected by the system is compensated, i.e. if the following occurs:

$$\left(\sum_{i=1}^{N_{TTS}} t_i^M \cdot n_i^{DS} - \sum_{i=1}^{N_T TS} [\star] \cdot n_i^{DS} \right) < T_{oov} + T'_{oov} \tag{3.30}$$

where the expression between the square bracket (the $\star$) is the same as in equation (3.20).

Table 3.1: The table reports an overview of all the element of all the components.

	Description
DC	Data Collection - All the words of the collection.
TS	Training Set - The words in the pages selected for the training.
DS	Data Set - All the word of the collection that do not belong to the training pages.
TTS	Test Set - Pages different from the training set used to estimate OOV ratio.
n_{DC}	Number of word images from the entire Data Collection.
n_{TS}	Number of word images from the Training Set.
n_{DS}	Number of word images from the Data Set.
N_{DC}	Number of different words in the entire Data Collection.
N_{TS}	Number of different words in the Training Set.
N_{DS}	Number of different words in the Data Set.
T_{man}	Time required to transcribe the entire collection manually.
T_u	Time required to transcribe the entire collection using a KWS to support the process.
T_{TS}	Time required to transcribe the training set words and to prepare the keyword list.
T_{out}	Time to validate or correct the out of the KWS system.
T_{miss}	Time to transcribe the in vocabulary words that the system can not recognize.
T_{oov}	Time to transcribe OOV words.
n_i^c	Number of instances of the word w_i correct recognized by the KWS.
n_i^w	Number of instances of the word w_i wrongly recognized by the KWS. i.e the KWS proposes a transcription that it is not correct.
n_i^m	Number of instances of the word w_i for which KWS does not propose any transcription.
$n_i^{oov^c}$	In Lexicon-Free KWS, the number of instances of OOV word correct recognized
$n_i^{oov^w}$	In Lexicon-Free KWS, the number of instances of OOV word wrong recognized. In this case the KWS can provide a wrong transcript or no transcript.
t_i^c	Time required to validate a correct out of the KWS.
t_i^w	Time required to correct a wrong out of the KWS
t_i^m	Time required to transcribe a missed word.
t_i^M	Time to manually transcribe a generic word without using the KWS

3.2 The Model at Work

To show how to use the model in practice, let us define the gain G achievable while using the keyword spotting system as:

$$G = 1 - \frac{T_{user}}{T_{man}} \tag{3.31}$$

where $T_{user} = T_u$ in the case of a lexicon-based KWS or $T_{user} = T_u + T'_u$ in the case of a lexicon-free KWS, and T_{man} is defined as in the previous section. The parameters of the models described in the previous section can be calculated or estimated by the following steps:

1. Transcription of the training data;
2. Training of the system and feasibility check;
3. Keyword spotting on the Test set;
4. Computing the gain G

3.2.1 Transcription of the Training Data

This step requires the manual transcription of the training set's word images and the recording of time to achieve a complete and correct transcription. After manually transcribing the training set and recording the time spent by the user, we know the values of n_{TS}, N_{TS} and t_i^M for each keyword.

3.2.2 Training of the System and Feasibility Check

After training the system, it is possible to obtain the values r_i^k and p_i^k calculated on TS for each keyword and check whether the condition (3.29) is satisfied or not. If it is not, and considering that the values r_i^k and p_i^k on DS are very likely to be smaller than those calculated on TS, the performance of the KWS system may not be good enough to profitably use the assisted transcription instead of

the manual one. At this point, one may consider increasing the training set or, if possible, reconfiguring the KWS system with a larger value of k and repeating the test. The first approach requires more user time, while the second approach depends on the architecture of the KWS.

3.2.3 Keyword Spotting on the Test Set

Once the KWS system has been trained and passed the feasibility test, it is used to recognise the words of TTS. After validating the system outputs, we reach the transcription of the test set, obtain the values of t_i^v, t_i^w and t_i^m and can calculate the values of r_i^k and p_i^k. In the case of a lexicon-free system, we can also obtain the values of N_{TTS}, $n_i^{oov^c}$ and $n_i^{oov^w}$ and calculate the values of t_i^M, $t_i^{M_w}$, r_i^k and p_i^k for the N_{TTS} keywords.

Estimating the User Time: Lexicon-Based System

The application of the model requires the values of its parameters as well as the values of n_{DS} obtained from the data set, which are unknown. Considering that TS, TTS and DS were extracted from the data collection we want to transcribe, it is reasonable to assume the following:

1. the distribution of the values of r_i^k and p_i^k computed on TTS and DS is similar;

2. the distribution of the length of the keywords is similar on each set, and because t_i^M depends mostly on the number of characters rather than on the actual character of the keyword, it is independent of the actual keyword;

3. the values of the model parameters are normally distributed;

4. all the samples of the data set are instances of the keywords obtained from the training set, i.e. than $N_{TS} = N_{DS}$

According to these assumptions, we use for t_i^M the mean value computed on TS, while for t_i^v, t_i^w, t_i^m, r_i^k, and p_i^k we use the mean values computed on TTS, so that the time for processing the system outputs can be written as follows:

$$\left(t^v \cdot r^k + t^w \cdot r^k \cdot (\frac{1}{p^k} - 1) + t^m \cdot (1 - r^k)\right) \cdot n^{DS} \tag{3.32}$$

where all the parameters assume the respective mean values and use this equation for estimating T_u.

Estimating the User Time: Lexicon-Free System

In this case, we follow the same line of thought as before for estimating T'_{out} and T'_{miss}, while the value of N_{DS} as well as those of $n_i^{oov^c}$ and $n_i^{oov^w}$ can be estimated as in section 3.1.2. Under these assumptions, we can estimate T'_{out}, T'_{miss} and T'_{oov} as follow:

$$T'_{out} = \left(t^v + t^w \cdot (\frac{1}{p^k} - 1)\right) \cdot r^k \cdot n^{DS} \tag{3.33}$$

$$T'_{miss} = \left(t^m \cdot (1 - r^k) \cdot n^{DS}\right) \tag{3.34}$$

$$T'_{oov} = t^M \cdot n^{oov^c} + t^{M_w} \cdot n^{oov^w} \tag{3.35}$$

where the values of the parameters are as in the previous case and use them to compute T'_u.

3.2.4 Computing the Gain

Under the same assumptions as before, we can estimate T_{man} as follows:

$$T_{man} = t^M \cdot (n_{TS} + n_{DS}) \tag{3.36}$$

and eventually derive the estimated value of G using equation (3.31), which represents the reduction of the user time achieved using the KWS system with respect to the manual transcription.

Chapter 4

Tools and Methodologies to Speed up the Labelling Data Process

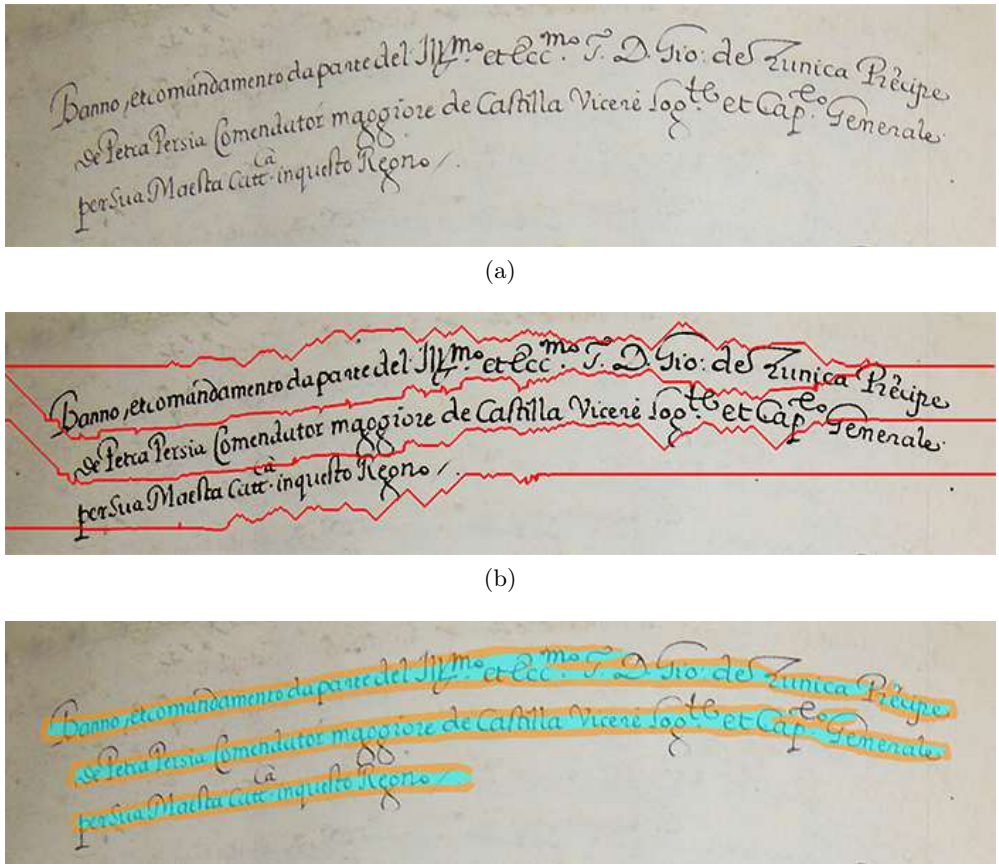
4.1 Text Line-Segmentation

The problem of line segmentation is well known in the literature and has been addressed by various proposed solutions, including depth-based methods, which have shown increasingly better performance in recent years, as explained in section 2.5.2. Despite the significant advances, the methods available suffer from some limitations, and the problem of segmenting lines of text in handwritten documents is still considered an open issue. The methods in the literature can be divided into two broad categories: learning-free and learning-based.

The advantage of learning-free methods becomes clear when segmentation has to be applied to documents or collections of documents with special features, which may consist of a few pages. In this case, the scarcity of available documents may make it impossible to perform the training phase due to the lack of labelled data. In this context, learning-free methods may prove to be

a convenient solution, but as far as we know, they do not provide a general solution that works satisfactorily under all conditions. An example of a feature that complicates segmentation is certainly the presence of lines of text that have a curved and irregular shape. Indeed, most methods assume that the document pages have a fairly regular layout and that the text lines follow an ideal horizontal line. However, it is not very difficult to find handwritten historical documents that have text lines with a curved course and this feature complicates the segmentation process and the system performance cannot handle it satisfactorily. Figure 4.1 b) shows some examples of segmentation of documents with curved lines by the learning-free method proposed by Alberti et al. [3], which performs among the bests in this category. The picture shows that this method has problems in segmenting documents with curved lines and the segmentation does not turn out satisfactorily.

Since learning-based methods require a learning or fine-tuning phase for optimal performance, which should be performed with similar documents close to the documents of interest, some pre-trained solutions have been proposed that promise considerable performance even without a specific training phase. Among these solutions, we can mention the deep-based solutions docExtractor [73] and dhSegment [79] which are proposed as ready-to-use solutions that have already undergone an extensive learning phase on very large datasets. Although the performance of these systems in recognising lines of text is very high even in the case of curved trends, they are subject to behaviours that can be problematic for some applications. A well-representative example of such behaviour is that these solutions are able to recognise the position of the centre-line of text lines well, but have problems recognising the base-line and the top-line. This behaviour leads to the exclusion of ink strokes in the segmentation of the line, in particular the exclusion of the typical and characteristic ascenders and descenders of many cursive manuscripts, as shown in Figure 4.1. These strokes, however, represent descriptive and discriminative features of cursive manuscripts, and the loss of this information during the process of segmentation



(a)

(b)

(c)

Figure 4.1: (a)Examples of text area with curve baselines. Example of segmentation performed with Alberti et al. [3]. (c) Example of segmentation performed with Oliveria et al. [79].

can be a serious problem.

In this chapter, we propose a solution to the problem of segmenting lines of text by presenting a learning-free method that is able to process handwritten lines of text with a curved shape and that tends to avoid truncating the ascending and descending strokes of cursive writing. Among the learning-free algorithms proposed in the literature, the algorithm presented by Surinta et al. in [128] separates consecutive lines of text even when they partially overlap. It is relatively simple to implement and robust for different types of handwritten

documents. However, it cannot process text lines with curved baselines. To overcome these limitations, we kept the idea of formulating text line segmentation as a path-planning problem to be solved by the A* algorithm. However, we have reformulated it for the problem at hand and used the information from the available transcriptions as described below.

4.1.1 The Proposed Line-Segmentation Solution

The line segmentation method proposed in this chapter presupposes a phase of preparation of the images to be processed. In the continuation of the discussion, we will hypothesize that during the pre-processing phase, a binarization process of the image is applied and that therefore the input to the segmentation process is a black-and-white image.

As a preliminary step to the segmentation process, the text area present in the document is recognised in such a way that the segmentation focuses only on the area of the image that actually contains the text to be extracted. This is done by calculating the horizontal projection profile (HPP) and vertical projection profile (VPP) of the foreground pixels and finding the histogram regions that correspond to the predominantly black rows/columns of the image. Figure 4.2 shows an example of the detection of the text area highlighted within the red rectangle.

In the method originally proposed by Surinta et al. [128] the authors use the A* search algorithm to determine the boundaries of each line of text. To determine the position of the text lines, the method computes the horizontal projection profile (HPP) of the entire text area and locates the centres of the text lines in accordance with the local HPP maxima. The part of the image between two successive maxima of the HPP is thus the search space to find the cut boundary separating the two text lines. The cut boundary is defined by applying the A* algorithm to find the shortest path between the leftmost and rightmost white pixels of each image line within the search space, using the squared Euclidean distance between two nodes of the path to weight the

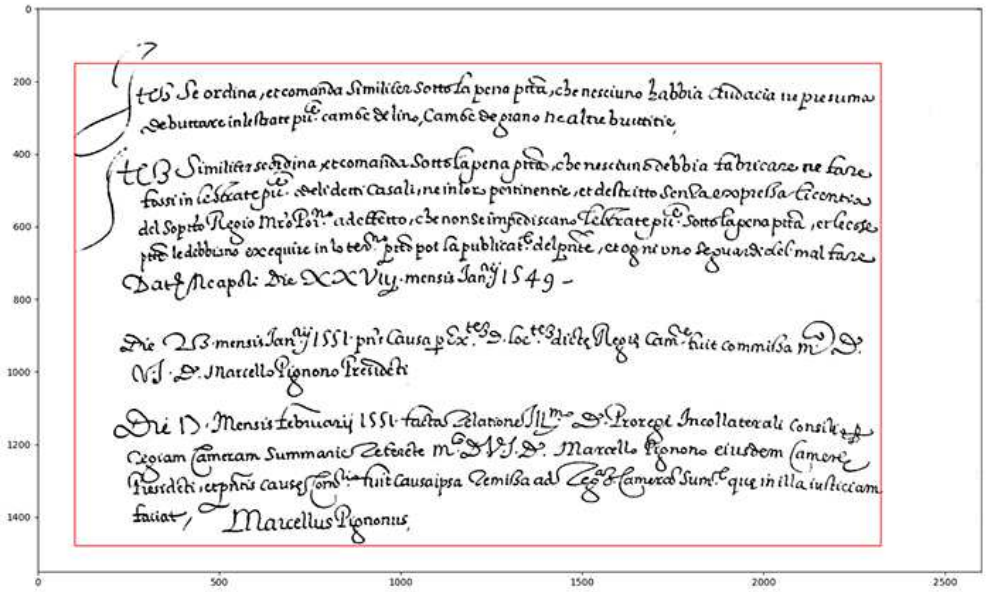
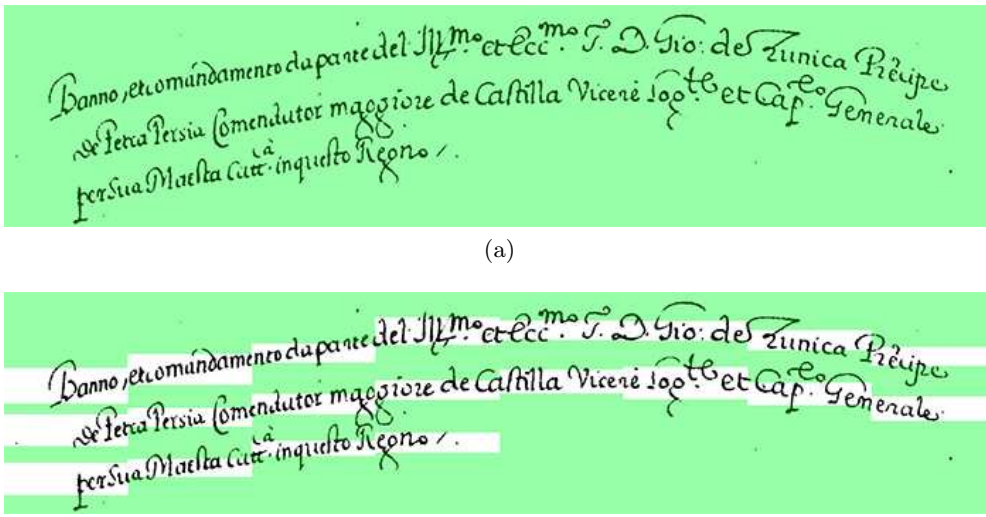


Figure 4.2: The figure shows in the red rectangle the text area detected in a document image.

different paths. Since the goal is to segment the page into lines of text, the black ink pixels are considered obstacles and the path search tries to work around them to connect the origin and destination points.

The main disadvantage of the original method is certainly its inability to segment documents containing lines of text with a curved baseline that cannot be separated by a horizontal line. This is because the first phase of the method is based on identifying the search spaces by analysing the horizontal black pixel projection profile on the entire image. When the text lines have highly curved baselines, it is not possible with the projection technique to satisfactorily identify the different spaces between the text lines. To solve this problem, we conjectured that identifying the search spaces for each part is possible by looking at the projections not on the whole image, but on different consecutive vertical regions called *stripes*. By applying the A* search to each strip, we approximate the curved baseline with a step function whose step size is equal to the horizontal size of the strips. Figure 4.3 shows an example where the

computation of the HPP does not allow to identify a search space and thus does not segment the text space (a), while the computation of the histogram on each strip results in as many search spaces as the number of text lines in each strip (b).



(a)

(b)

Figure 4.3: Examples of detection of search areas for text line segmentation. Green areas represent the search areas detected by computing the histogram on the whole image (a) or on each stripe (b).

Below, we report the procedural steps of our text line segmentation algorithm:

1. **Divide the image into stripes** The image of the text area is divided into S stripes of the same width, in such a way that the sum of the widths of the stripes equals the width of the document image.

2. **Find the horizontal projection profile** The HPP is calculated for each strip. The profile analysis makes it possible to identify the position of the line of text in the region studied, assuming that each peak corresponds to one line of text. If the number of lines of the document is known, this information can be used to determine the maximum number of peaks

that can be recovered from the histogram of the region under study. The peaks, starting with the peak with the highest value, are then extracted from the HPP until the maximum number is reached if they are present in the histogram. If the histogram contains a number of peaks that is below the threshold, this means that not all lines are present in the strip under consideration, such as in the case where the ink of the handwritten text does not extend over the entire line. In these cases, all detected peaks are selected, even if they are fewer than expected. if the number of rows is unknown, all the detected peaks are taken into account and considered as a different row. Once the peaks are identified, the space between the different peaks represents the search area that separates the two lines.

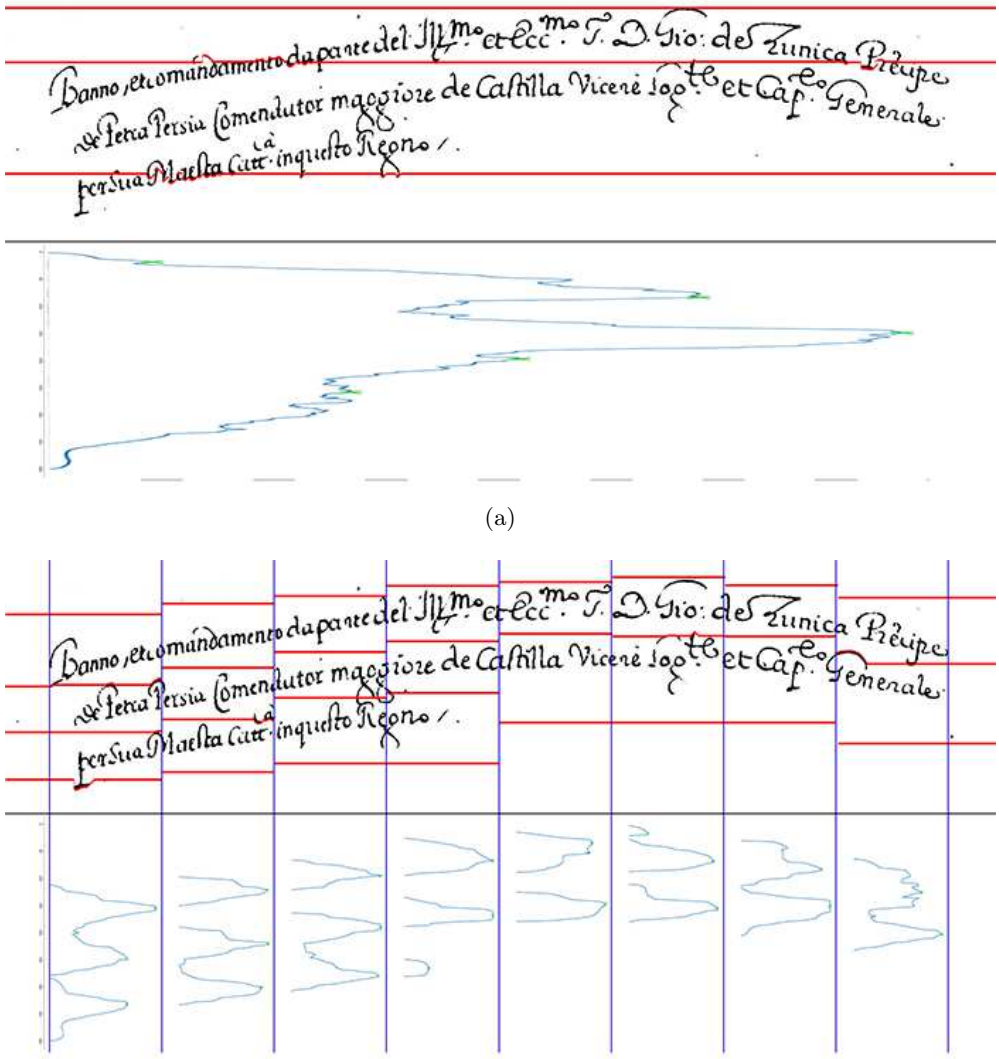
3. **Carry out A* path planning along the search areas in each stripe** Once the different search areas for each strip are identified, the A* algorithm is run to find the cut boundaries for each search area. In the original algorithm, the presence of overlapping ascenders and descenders, as shown in Figure 4.4(a), poses an *insurmountable* obstacle to the path search, since it is impossible to define a path without crossing them. To solve this problem, Surinta et al. [128] modify algorithm A* to allow obstacles to be crossed, leaving it to the cost function to model whether an obstacle should be crossed or not. However, this makes the algorithm more complex and increases the difficulty of computing the cost function. In our implementation, we preferred to add a preliminary phase to the path planning, which consists of identifying the obstacles as the area of HPP between two local maxima, i.e., the search space for the A* algorithm, that does not contain a white row, and then *tunnel* them by inserting a path of white pixels in the row of the image corresponding to the centre of the search area so that the A* path planning algorithm can find a valid path. Figure 4.4(b) shows an insurmountable obstacle modified by a sequence of white pixels in the centre of the ink trace.

(a) (b)

Figure 4.4: The image shows an example of an insurmountable obstacle. (a) An example of an insurmountable obstacle, (b) An example of a modified insurmountable obstacle: note the path of white pixels inserted in the middle of the ink track for allowing crossing the obstacle.

4. **Connect the cutting boundaries between adjacent stripes.** The final step is to combine the results of the different A* for each stripe with the results of the algorithm in the immediately adjacent one, and so on. In this way, cutting boundaries are obtained that traverse the entire text area.

Figure 4.5 shows an example of the HPP calculated for the entire text area (a), and for the case where it is calculated for 8 different strips (b). In the latter case, it is much easier than in the former to identify the local maxima in each stripe. It is also worth noting that only two local maxima are detected in the four rightmost stripes since the handwriting of the last line does not extend over the entire line.



(a)

(b)

Figure 4.5: Text-line segmentation in case of lines with curved baseline: (a) the result of the original algorithm ($S = 1$); (b) the result of our algorithm (in the case $S = 8$).

4.2 Transcript Alignment

The process of transcript alignment aims to match the words present in the image of a handwritten line of text with their corresponding digital transcription.

Figure 4.6 shows the workflow of the process. It starts with a colour image of a document and its line-by-line transcription and ends with a data structure that links the transcriptions to the bounding boxes of the corresponding word images in each line of text.

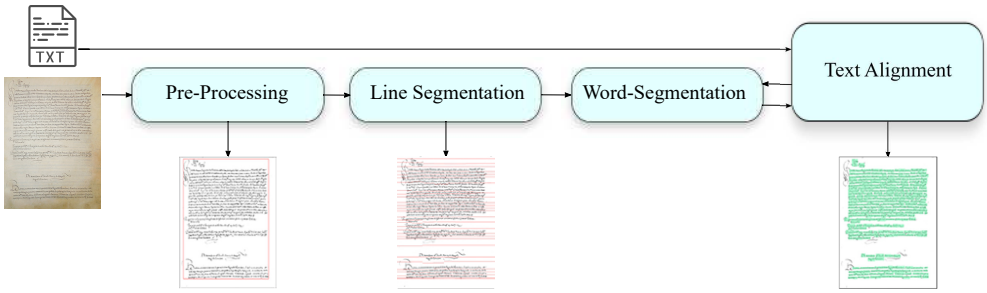


Figure 4.6: Workflow of the transcript alignment process. The input consists of the colour image of a document and its transcription. The document is pre-processed by creating its black-and-white version and identifying the text area. Then follows the segmentation into lines of text and finally the alignment of the transcription.

The first stage is a pre-processing phase that prepares the image for the subsequent stages. Here the image is binarized and then a Gaussian filter is applied to reduce noise and blur the ink strokes. The second stage includes a line segmentation method that allows for the extraction of images from the original document, each containing only one of the text lines of the entire document. For each line a preliminary phase of word segmentation is then carried out, this segmentation is then corrected and improved during the transcript alignment phase until the system is able to provide an alignment of all the words present in the line transcription. The preliminary stage of segmentation into words is achieved by calculating the vertical projection profile (VPP) of the black pixels of the text line, and the edges of each bounding box correspond to the columns of white pixels. In this way, different bounding boxes are identified for each of the continuous ink components on the line, as shown in Figure 4.7.

The word segmentation provides the sequence of bounding boxes W:

$$W =< w_1, w_2, \ldots, w_m >$$

Figure 4.7: Examples of preliminary word segmentation. The bounding boxes are represented as grey areas of the text line.

where m is the number of identified boxes. For the digital transcription of the handwritten row it is possible to construct the sequence of transcripts T:

$$T =< t_1, t_2, \ldots, t_n >$$

where n is the total number of words that make up the transcription of the line of text. Aligning each bounding box in W with its corresponding transcript in T would be a trivial task in case $m = n$ and each bounding box includes just a one-word image, i.e., if an error-free word segmentation would be available. As such an ideal word segmentation is not available, there may be both over- and under-segmentation errors. To deal with them, the alignment algorithm involves a correction process that, in short, attempts at either merging adjacent bounding boxes or splitting a bounding box to delineate whole word images. The method consists of scanning the sequences W and T for analysing each ordered pair (w_{curr}, t_{curr}) and performing a consistency test to decide whether it is possible to align the transcript with the bounding box.

To perform the consistency test, the algorithm computes the *Average Character Width (ACW)* for each line of text according to the following equation:

$$ACW = \frac{\sum_W (Word\ image\ width\ (pixels))}{\sum_T (Number\ of\ characters)}$$

It also records the minimum and maximum value of ACW calculated over all the text-lines extracted from the document respectively denoted in the following with mth and Mth.
Then, assuming that the transcript to be linked with w_i is t_j, it estimates the

ACW for the box w_i as:

$$ACW_{box} = \frac{width\ of\ w_i}{number\ of character\ of\ t_j}$$

The value of ACW_{box} is then compared with the values of mth and Mth, so to distinguish three different cases:

1. **Correct segmentation:** $mth < ACW_{box} < Mth$ The size of the box w_i matches the number of characters in the transcript t_j. In this case, the transcript t_j is aligned with the bounding box w_i, and the next unmatched pair (w_{i+1}, t_{j+1}) is considered.

2. **Over-segmentation:** $ACW_{box} < mth$ The box size w_i is too small to accommodate the number of characters in the transcription t_i. In this case, the algorithm assumes that an over-segmentation error has occurred, and the tentative word segmentation is modified by merging w_i with w_{i+1}. This way, the box size increases, and the consistency test can be repeated. If it is passed the merged bounding boxes are associated with the transcript t_j, and the next unmatched pair (w_{i+2}, t_{j+1}) is considered.

3. **Under-segmentation:** $ACW_{box} > Mth$. The box size w_i is too large to accommodate the number of characters of the transcript t_j. In this case, the algorithm assumes that an under-segmentation error has occurred. In this case, the method first attempts to split the bounding box to try to obtain a smaller box that can be aligned with the current transcription t_j. For this purpose, the box w_i is tentatively split in correspondence to the minimum of its VPP so that it is possible to segment within the ink trace. The leftmost part of w_i and the transcript t_j are then considered for the consistency test. If it is passed, the split and the corresponding alignment are validated and the remaining part of the box w_i and the transcript t_{j+1} are considered for the next consistency test. Otherwise, the transcription t_j is merged with the adjacent transcription t_{j+1}, the

ACW_{box} is then computed using the total number of characters of t_j and t_{j+1} and then a new consistency test is performed. If successfully passed, the bounding box w_i was associated with the two transcripts t_j and t_{j+1}.

The alignment, thus, proceeds by going through the sequences W and T and performing a consistency test between a potential box and a transcript from time to time. The method provides two options for the order in which pairs are to be selected for consistency testing: the first method is called *Forward* and consists of selecting the boxes and the transcripts from left to right in the sequences, the second method is called *MiM* (Meet in the Middle), where we alternately select the text line sides. In short, we start with the leftmost box/transcript and perform the consistency test. Once this is done, instead of continuing with the next box/transcript, we go to the rightmost box/transcript and move one step backwards, from right to left. We then return to the leftmost box/transcript among those still waiting to be aligned and so on. In this way, we try to limit the spread of alignment errors to the entire line of text and avoid a possible "snowball effect" [17]. Figure 4.8 shows an example of the entire process of alignment performed on an Italian document of the IV century written in Latin.

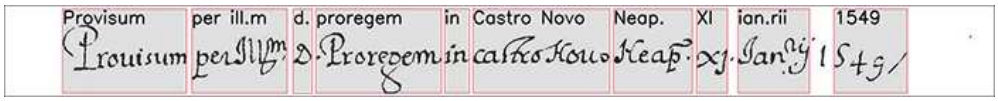


Figure 4.8: Examples of alignment method. The result of the MiM algorithm performed on an example of the handwritten text of the IV century.

This page intentionally left blank.

Chapter 5

KWS by N-gram Retrieval

5.1 The Rationale of the N-gram Retrieval Solution

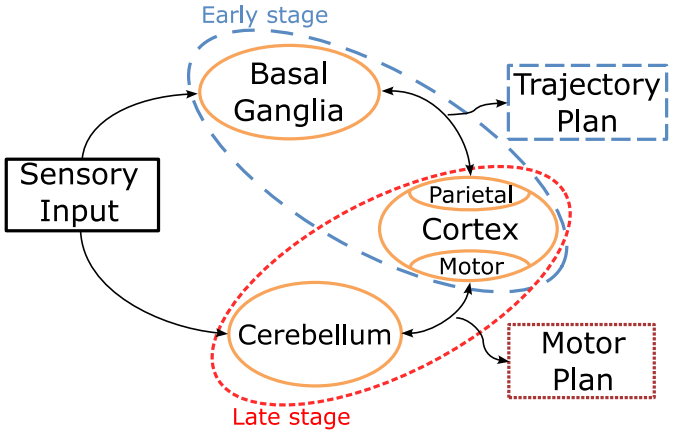
For handwriting, it is complicated and time-consuming to address character-level segmentation and apply OCR techniques for recognition. For this reason, specific HTR techniques, such as KWS, are preferred. Typically, KWS focuses its attention on whole words and furthers its application to handwriting but introduces the problem of OOV words that cannot be easily spotted. If on the one hand, we find the single characters and on the other the entire words, one wonders: "*what if we put ourselves in the middle?*" Segmenting cursive handwriting into sequences of characters can be much easier than segmenting it at the character level, and recovering sequences of a few characters (N-grams) could allow for the identification of entire words, even those that would have been OOV at the word level. Hence our idea of building a keyword spotting system based on retrieving N-grams instead of whole words.

5.1.1 Writing as a Complex Motor Act

Writing by hand is not only a cognitive skill but also a complex motor act. It is therefore subject to a learning phase that enables the writer to memorise and

acquire the necessary motor activations through the repetition of movements. Like any other motor activity, at the beginning of learning, writing requires special attention and the movements result slow and not very accurate, but with practice, one gains more and more speed and confidence until finally, a writer can show fluent movements that can be performed without any special effort. From this observation, two different phases of the learning to write process can be derived: an early phase followed by a late phase [69].

Studies on motor learning [51, 84, 97, 110, 129] show how the execution of complex movements requires interaction between different structures, such as cortical and subcortical structures, basal ganglia, cerebellum and cortical motor regions, up to the motor circuits of the spinal cord. Figure 5.1 shows a neural scheme for the learning process of handwriting, which illustrates which areas are involved in the two phases of the early and late stages. According to this scheme, in the early phase, the trajectory or sequence of dots to be joined to produce the handwriting is learned. In the late phase, the sequence of motor commands for a particular trajectory is considered acquired and executed as a single behaviour. This results in the desired movement being executed quickly and accurately, or in other words, the movement to produce the desired trajectory has been automated.



Source: Marcelli et Al. [69]

Figure 5.1: Neural scheme of the model for procedural motor learning.

According to this model, sensory information is transmitted to the basal ganglia and cerebellum via an input module. The basal ganglia and parietal cortex then interact with each other to define the desired trajectory and select the points in the spatial sequence to be reached in order to perform the motor task. The parietal cortex in turn sends this information to the cerebellum by interacting with the motor cortex and selecting the appropriate motor commands. The goal of learning handwriting is thus to develop a repertoire of automated movements that correspond to the trajectories most often used to write words, i.e. the pathways that are most often repeated during the learning phase and with which the writer is most familiar. In principle, each person develops different motor programmes for different processes. Just think about the shapes and forms of handwriting are characteristic of each individual, and it is these differentiations that provide a scientific basis for assessing the authorship of written documents in legal proceedings [26, 27, 141]. On the other hand, this means that it is not easy to identify and define the trajectories for which motor programmes are typically developed.

We can imagine, however, that a trajectory for which a motor programme develops must observe some characteristics. Indeed, it is difficult to imagine that a trajectory that is too complex and long would prove to be a good candidate, which would lead to excluding the development of motor automatisms for whole words. On the other hand, in this case, it would be necessary to store a large number of motor programmes that could be associated with each learned word, which would be inconvenient for words that are written with little regularity or for new words that have never been written before and that cannot benefit from an automatism. It is, therefore, necessary to look for automatable trajectory in pathways associated with subsequences of words that can then be used again and again when they appear in a word, regardless of how familiar the writer is with the word in question.

When we think of the most natural discretisation of a word, we immediately think of the division of the word into the characters of which it is composed.

The set of characters is a much more limited set than the set of all the words of a language, and by defining an automatism for each character it would be possible to compose any word without any limit. However, the distinction in characters alone might make little sense if we consider them as basic trajectories for handwriting, especially if we think of cursive writing. It would be more likely to focus on sequences of a few characters, or sequences consisting of two or three characters that can be repeated many times in different words of a language. The development of automaticity related to these sequences of characters could prove useful in the process of automating writing motor programmes. The tendency of an individual to develop motor patterns in the writing process also emerges in the study of Sparrow and Newell [119], who reasoned that this aspect is a consequence of the organism's tendency to conserve metabolic energy. Motor learning thus suggests that the size of the motor pattern is related to the ability to memorise a certain number of elementary actions, while linguistics suggests that there are sequences of signs in a language that occur more frequently than others. This aspect seems to support the hypothesis that the most automated motor sequences are associated with short sequences of a few signs that occur frequently in the writer's language.

Developing motor programmes for character sequences that can be triggered each time an author wants to write a particular sequence may mean that the variability of the trajectory is reduced for successive repetitions. For example, if a sequence of two characters is automated and always triggers the activation of the same motor programme, this may mean that the sequence of ink on paper always appears with very similar features, even if the sequence of characters belongs to different words. Figure 5.2, for example, shows how the sequence 'th' appears very similar in the two words 'other' and 'then'. This could mean that considering sequences of two or three characters as primitives for recognising handwriting could be a correct and functional choice.

Figure 5.2: The 2-gram 'th' appears very similar in both the execution in the words 'other' and 'than'.

5.1.2 The OOV Problem

As discussed in section 2.6.2, the KWS technique has proven to be a good alternative for recognising handwritten text in documents of historical interest [1], with solutions based on a reference dictionary being of particular interest. However, this class of KWS systems suffers from the major problem that recognition is limited to the words contained in the reference dictionary, which effectively prevents the recovery of out-of-vocabulary (OOV) words. This problem is more relevant the smaller the reference dictionary of keywords. Small dictionaries, by definition, contain only a few words, so the likelihood of having to handle OOV words increases. On the other hand, too large a dictionary increases the complexity of the system considerably, as there are more and more classes of possible interpretations.

Limited dictionaries, however, can be useful for problems with a well-defined domain, such as postal addresses [56] or bank cheques [60]. For similar problems, it is easy to narrow the range of possible words, thus creating a small reference dictionary that reduces the likelihood that the system will have to deal with OOV words. However, if it is not possible to define a range of possible words, increasing the cardinality of the reference dictionary seems to be the only solution. However, this would lead to the system having to evaluate many more possible words, effectively increasing the processing time, and also introducing many similar words that may only differ in a few letters, which could worsen the retrieval performance.

The definition of large dictionaries requires the ability to access large data

sets, which cannot always be guaranteed. This observation seems to be particularly true for historical documents. The amount of data available for particular documents or collections of documents is often limited by the lack of such large manuscript collections. Collections of historical documents may even consist of less than a hundred pages, but with features typical and characteristic only of the collection itself. In these contexts, it is therefore difficult to define a dictionary large enough to severely limit the problem of OOV words.

To limit the OOV problem, some promising solutions have used the introduction of language models based on N-grams. Kozielski et al [58] investigated the use of linguistic character models using character models of 10 grams estimated by the WittenBell method. Brakensiek et al. in [9] present a recognition system based on Markov models hidden in linked mixtures and investigate the use of N-grams of characters as a linguistic model. In these cases, the introduction of linguistic models based on N-grams has led to an improvement in the recovery of OOV words, although the performance remains far from that of words in the vocabulary.

The idea of drawing attention to N-grams could therefore be interesting to manage the recovery of OOV words. As discussed in section 5.1.1, the writing of certain character sequences by a scribe can produce similar trajectories, suggesting the use of such character sequences as building blocks of an OOV system. This could therefore enable recognition of a word once the sequences that make it up have been identified, whether it is a dictionary word or an OOV. To illustrate the concept, we can refer to Figure 5.3. Let us imagine that the word "violent" is a word belonging to the group of OOV words. The word then never appears in training instances, but focusing on the N-grams, it is possible that the N-grams composing the word are known by the system as they are present in different words of the training set. Therefore, the system may be able to recover, for example, the two-character (2-gram) sequences "vi", "io", "le", "en" and "nt". If it is possible to define a relationship between the recognised 2-grams, for example through a spatial superposition relationship, the

composition of the different sequences can provide the "violent" transcription.

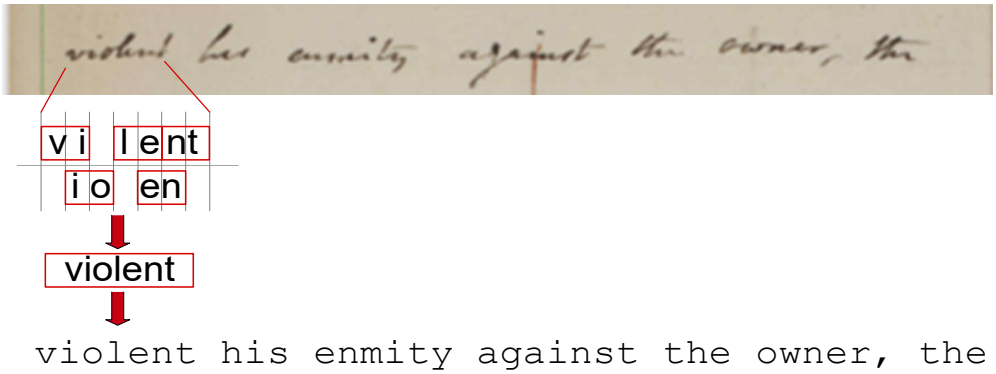


Figure 5.3: Example of spotting an OOV word based on 2-gram detection.

5.2 The N-gram Spotting Idea

The general idea behind the proposed N-gram spotting system is to search for a query word within the image of a document page by decomposing the word into the N-grams for which the system can perform a search. Figure 5.4 shows the general workflow of the system. The first step is to decompose the query word into its N-grams, thus defining the set of query N-grams that contains all

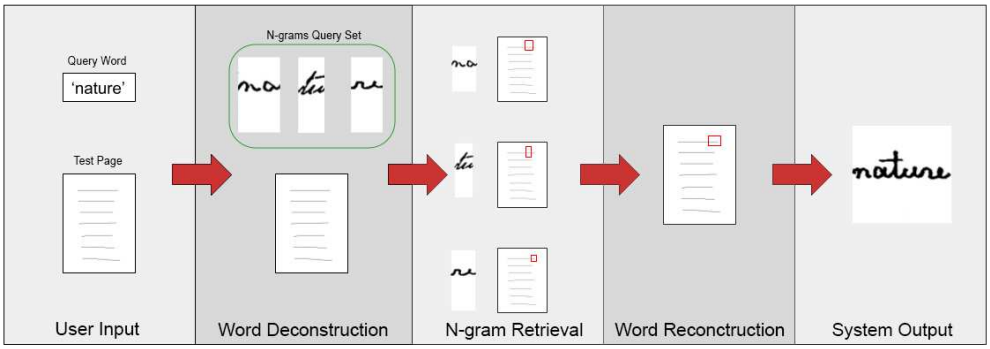


Figure 5.4: The system receives the query word "nature" and a page on which to search. The query word is decomposed into the query N-grams that are searched for. At the end of the process, the N-grams found are combined to provide the position of the word within the document page.

the N-grams occurring in the word for which the spotting system can perform a search. Once this set is determined, the step of spotting the N-grams allows to identify the positions of the different N-grams within the document. In this phase, only the N-grams are searched, ignoring the original query word entirely. In the final step of the process, the focus is returned to the word level by analyzing the detected N-grams and determining the position of the original query word.

The system as a whole receives as input the image of a page of the document and a string of the query word to be searched so that the QbS search paradigm is fully applied at this level. In the N-gram retrieval phase, the problem is to find the image areas of the page that contain the N-grams of the query set. The query set consists of N-gram reference images, and this set can be considered as a dictionary of the N-grams to search. From this point of view, the N-gram spotting system is equivalent to the QbE search paradigm but still accepts string-level whole-word searches. In its overall view, the system combines the QbS and QbE search paradigms by presenting itself as a hybrid search system that combines a QbS system at the word level with a QbE at the N-gram level.

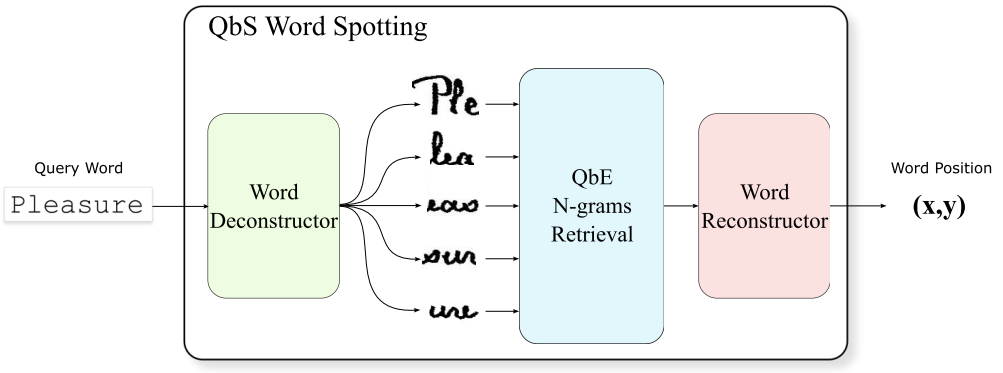


Figure 5.5: The system combines the QbS search approach at the word level with the QbE search at the N-gram level.

In the next sections, the different phases of the workflow are described in detail, starting with the deconstruction of the word query and the definition

of the N-gram query set, then the analysis of the N-gram retrieval phase, and finally the process of combining the identified N-grams to find the original query word.

5.3 Word Deconstruction

The first stage of the keyword spotting process is to define the "N-grams query set" starting from the string of the original keyword. This set is thus composed of the N-grams contained in the word for which the spotting system can provide an output.

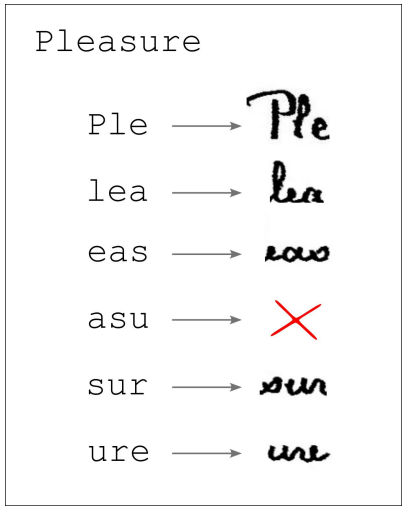


Figure 5.6: The image shows the process of the construction of the N-gram Query Set. From a query word, all the N-grams are extracted and then image examples are selected from the system dictionary if the N-gram class is available.

The process of set definition is very simple. The word sequence of the query is decomposed into all its reference N-grams, including overlapping N-grams. To illustrate the concept, consider the word "Pleasure" shown in Figure 5.6. Suppose we want to extract all the 3-grams that make up the word. Starting with the first character "P", the first 3-gram "Ple" is extracted, then continue with the following character "l" and extract the sequence "lea", and so on until all available characters are processed. Once the N-grams present in the word are

defined, the N-grams query set must be defined, which consists of the images of the N-grams in the reference dictionary of the QWS system. Of course, the set consists only of the N-grams that are present in the query word and in the reference dictionary at the same time. Indeed, it may happen that the query word contains an N-gram that is not present in the dictionary (see the case of "asu" in Figure 5.6) and for which the QWS system cannot provide an answer. Therefore, only N-gram images that the system can use for the search may appear in the N-grams query set.

5.4 N-gram Retrieval

In the following, two alternatives for the QbE system to retrieve N-grams within a handwritten line are proposed. The first is based on the implementation of a Siamese Neural Network capable of learning a measure of the distance between N-gram images and on the definition of a Sliding-Window architecture to carry out the search. The second is still based on a Siamese paradigm but it avoids any sliding-windows operators and takes advantage of the benefits that an attention mechanism can bring.

5.4.1 Sliding-Window Architecture

A Siamese Network is a type of network architecture that contains two or more identical sub-networks that are used to generate feature vectors for each input and compare them to obtain a measure of similarity between them [21,55]. This architectural scheme can be used for our purpose by providing the network with two images of N-grams to be compared, one belonging to the N-gram query set and the other cropped from the text line to be analysed. The architecture of a branch of the network consists of a convolutional backbone for feature extraction, followed by a fully connected layer for refining the encoding of the final embedding. As envisaged in the Siamese paradigm, the weights of the two branches are shared to ensure coherent feature extraction for both images for

the purpose of computing the similarity measure. Figure 5.7 shows the network architecture on the top right. The two images of the N-grams to be compared are fed to the network and a measure of similarity between the two images is provided, which is greater the more similar the input images are to each other.

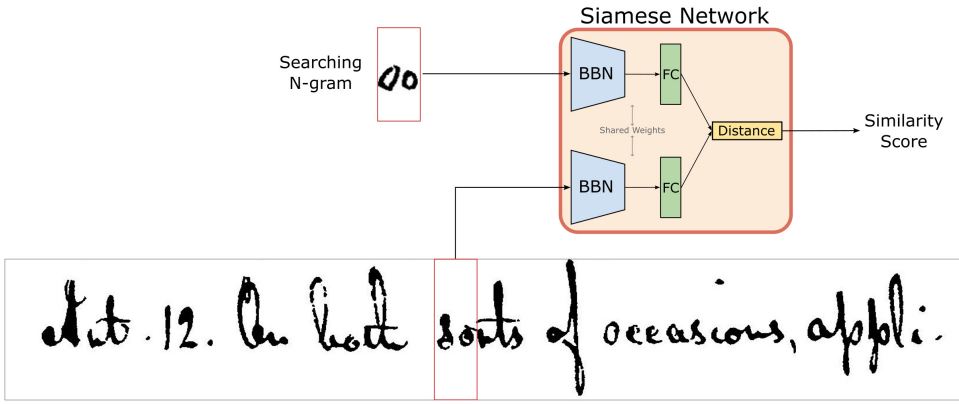


Figure 5.7: A crop of images of a text line is fed into the Siamese Network together with the image of the reference N-gram to obtain the similarity score.

The system described so far only provides a measure of the similarity between two images and does not allow locating the reference N-gram in the line of text. There is therefore a need to measure the distance of the reference N-gram along the entire line of text in order to assess whether and where the N-gram is present. A sliding window system can be used for this purpose; a selection window crops out a part of the image of the text line that is used for comparison with the reference N-gram image. The selection window is then scrolled over the entire line to obtain the trend of the similarity score over the entire line. As can be seen in Figure 5.8, when analysing the similarity score trend, the peaks should correspond to an instance of the N-grams we are looking for. The N-gram spotting step is then repeated for all N-gram classes contained in the N-gram query set, so as to obtain a trend score for each possible N-gram of the query word.

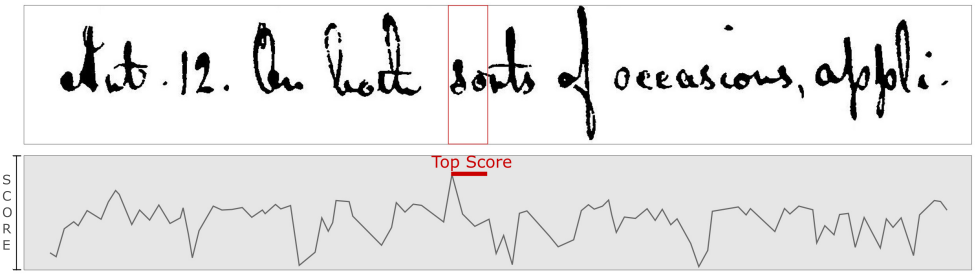


Figure 5.8: The sliding window cuts out the image portions of a line of text and the similarity score calculated with the reference N-gram is reported for the entire line. The score trend helps identify where the reference N-gram might be.

5.4.2 Attention-Based Architecture

In this section we describe an alternative to the solution described in 5.4.1, taking inspiration from the work of Souibgui et al. [118]. We propose a few-shot learning-based model to tackle the task of spotting N-grams in a line of handwritten text. The proposed architecture is based on a faster R-CNN [95] detector, which still uses a Siamese architecture as a basis with the aim to provide a similarity score between the reference N-gram images and the crop found in the handwritten text line.

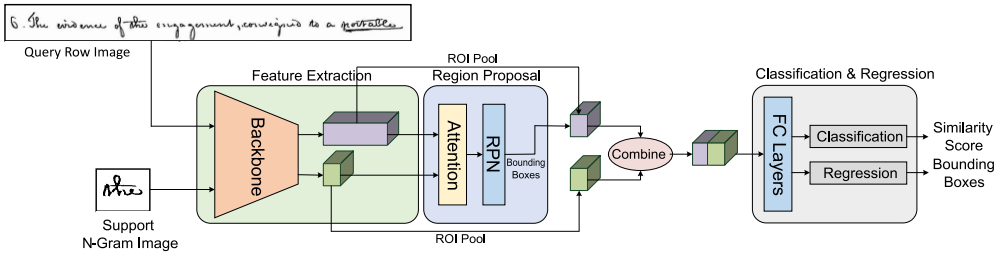


Figure 5.9: The architecture of the Attention-Based approach. The backbone is used in feature extraction of the query and support images, as predicted by the Siamese model. .

Figure 5.9 shows the attention-based architecture of the proposed system. The system receives as input a line of handwritten text and an image of a reference N-gram to be spotted. The first stage is a feature extraction stage in which the CNN network extracts features from text images. It should be

noted that the network used to extract features from the N-gram and the line crop images is the same as required for a Siamese architecture. The choice of backbone for feature extraction is not fixed, and by changing the choice of network for feature extraction, a different search mode can be applied, allowing identification to be performed in a different search space each time. The feature extraction phase is followed by the region proposal phase. In this phase, the feature map of both the query image and the reference N-gram image is sent to an attention module and furthermore to a Region Proposal Network (RPN). An upstream attention mechanism of the RPN network strengthens the regional suggestion phase by exploring the link between the supporting image and the query image [28]. After this step, ROI-pooling is applied to the regions proposed by the RPN and the feature map of the supporting N-gram image. The feature maps are then combined and sent to a classification and regression module. This module consists of a collection of fully connected layers divided into two heads. The first is a classifier head and the second is a regression head. The output of the classifier uses a sigmoid activation function that can decide whether the proposed region belongs to the supporting image class or not. At the same time, the regression model generates the coordinates of the bounding boxes within the handwritten line image with respect to the classified parts of the image.

The backbone for feature extraction can be chosen from different types so that different search modes can be combined with the aim of evaluating the results obtained in different feature spaces. We now propose an architecture that combines two different search modes, as shown in Figure 5.10, in such a way that we can perform N-gram spotting in different feature spaces. The two independent branches operate simultaneously and obtain the two solutions y_1 and y_2 using the backbones BB_1 and BB_2, respectively. Finally, the two solutions are combined into the new solution Y using weighted concatenation:

$$Y = (w_1 \cdot Y_1)||(w_2 \cdot Y_2) \tag{5.1}$$

Through the chosen weights of each backbone w_1 and w_2, it is possible to define the relative importance of the solution y_1 compared to y_2.

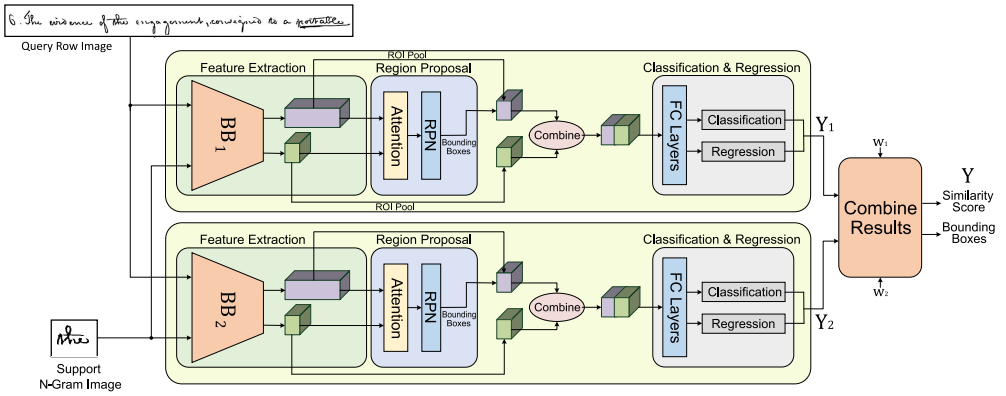


Figure 5.10: The architecture performs the N-gram spotting in different feature spaces. At last, the solution are fused together to provide a single spotting result.

Fusion and Rescore Solutions

The N-gram spotting solutions provide the option to select the α number of samples for each class of N-grams to be used in the search. The N-gram query set is composed of the N-gram classes that make up the original query word, but each of these classes is filled by all N-gram instances from the training set. This means that for each class there are multiple images of the same N-gram, each with its own features. Indeed, one characteristic of the manuscript text is precisely variability, even if the scribe remains the same, repetition of the same N-gram will produce ink traces that are never perfectly identical. Repeating the search for the same class of N-grams, but using a different example image, could result in different instances being found within the analysed line, thus recovering a larger number of N-grams. For example, Figure 5.11 a) shows how the 3-gram '*the*' was recognised by two different searches.

To obtain a single solution for each class of N-grams, the different ones are unified. If two overlapping solutions with values s_1 and s_2 are found, it is possible to combine them into a new solution with a higher similarity value s'.

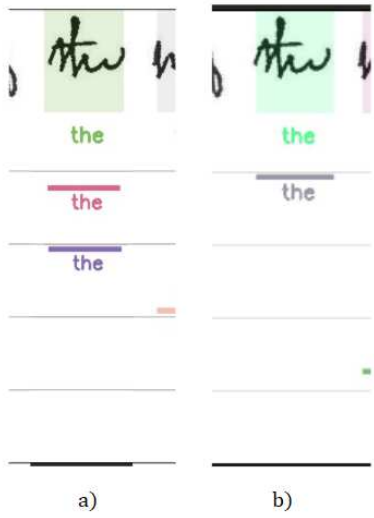


Figure 5.11: The figure shows an example of the combination of two similar solutions proposed for the same area. a) shows two proposals of the 3-gram "the" with two different scores; b) shows the result of the fusion. It results in a new interpretation for the 3-gram "the" whose score is higher than that of the initial interpretations.

The gain γ is then defined as:

$$\gamma = \delta \cdot \left(1 - \frac{|s_1 - s_2|}{max(s_1, s_2)}\right) \tag{5.2}$$

where δ is the maximum possible increment. The score s' of the new unified solution s_3 is then given by:

$$s' = max(s_1, s_2) + \gamma \tag{5.3}$$

Thus, the score s' is directly proportional to the maximum score between s_1 and s_2 and the difference between the two scores. In this way, the cases where the similarity values of the initial solutions s_2 and s_2 are high and close to each other are rewarded. Figure 5.11 b) shows how the similarity value of the new solution is greater than the similarity values of the initial solutions shown in Figure 5.11 a).

Once the solution fusion and the rescoring mode are defined, the whole N-gram retrieval phase can be schematised as in Figure 5.12. The first phase deals with the detection of the N-grams in a line of the reference text. This phase receives as input the line to be analysed, the N-grams query set and the number α of searches to be repeated for each N-gram class. The subsequent phase of fusing and reordering modifies the results of the previous phase by providing the positions within the text line for each class of N-grams.

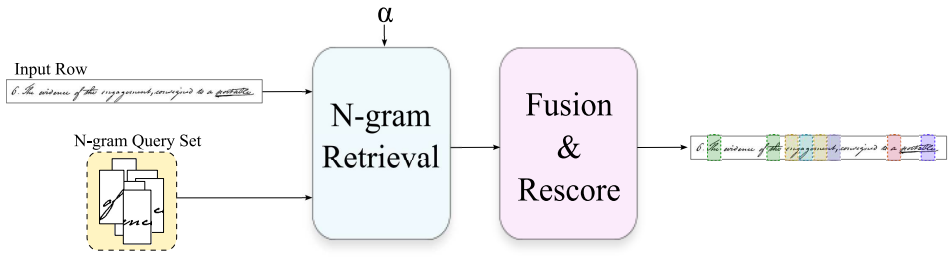


Figure 5.12: Overall schema for the N-gram retrieval stage.

5.5 Word Reconstructor

Once the positions of the N-grams of interest within the handwritten line of text have been determined, the position of the original search word can be retrieved. In this section, we present a method to reconstruct the position of the searched word starting from the analysis of the results of the previous phase of N-gram spotting.

The N-gram spotting phase provides the position of the N-grams within the analysed line of text. Since the goal is to find the word that consists of all the N-grams in the search set, we look for "*high-density areas*", i.e. areas of the text line where there are overlaps of searched N-grams. If the N-gram retrieval phase was successful for all classes of N-grams, a density area could correspond to an area where the searched word occurs. It is important to note, however, that the detection of density areas alone is not sufficient to confirm that these ranges correspond to the position of the words sought. This is because the

density zone does not take into account the position of the N-grams and could therefore contain different anagrams of the search word. Also, the N-gram detection phase is not error-free and could result in density zones that do not contain all the N-grams of the word. Therefore, a density area evaluation phase is required in which a confidence measure is assigned to each area.

A confidence measure was defined with a value in the range $(0, 100)$, where 100 represents the maximum confidence estimated considering three criteria:

1. number of retrieved N-grams;

2. mean similarity score of retrieved N-grams;

3. position of retrieved N-grams.

For the first criterion, the more N-grams detected in the density area, the greater the confidence measure. For a maximum confidence measure, the density area must have exactly the number of expected N-grams. More generally, the confidence level sc varies linearly with the number of detected N-grams. For example, if the number of detected N-grams equals half of the expected number, the confidence measure is halved:

$$sc = sc \cdot \frac{\#Ngram\_in\_area}{\#Ngram\_expected} \tag{5.4}$$

When applying the second criterion, note that each N-gram was recognised with a similarity score. The lower the similarity score, the more reliable the prediction. The confidence measure can then be reshaped by subtracting the average of the similarity scores of all N-grams belonging to the density area. In the optimal case, each detected N-gram has a similarity score of zero, resulting in a maximum likelihood for each N-gram. In this case, the confidence would not change because every N-gram in the area is safe:

$$sc = sc - avg(Ngrams\_distance\_score) \tag{5.5}$$

To evaluate the position of the N-grams in the density area, we can calculate the pyramidal decomposition of the query word and consider the different sets of N-grams at the different levels of the representation. To calculate the representation, the word for each level must be divided into different parts corresponding to the depth of the level. In other words, level two of the representation consists of the first and the second half of the query word, the third level consists of the word divided into three parts, and so on. We can assign the set of N-grams that make up each sequence of the representation and thus build the pyramidal N-gram sets representation (PNGR) of the query word. Figure 5.13 shows an example of the PNGR of the query word word "action".

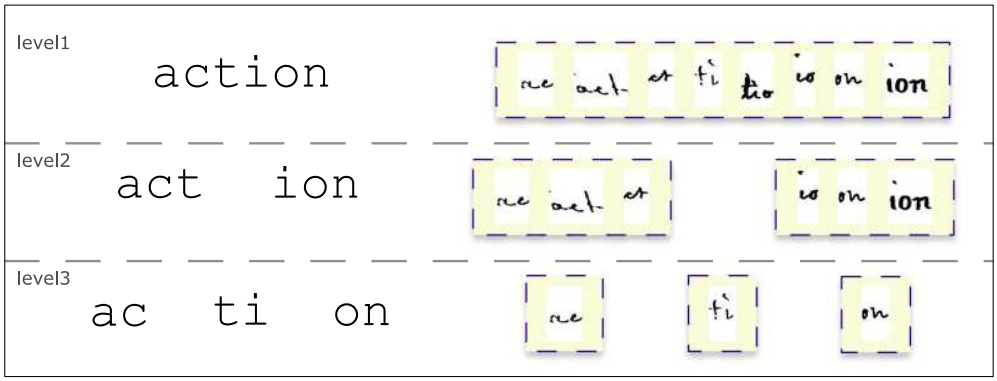


Figure 5.13: Example of Pyramidal N-grams sets Representation of the word word "action". On the left, there is the PNRG at string level, while on the right the PNRG on N-gram images level.

Similarly, the PNGR of a density area can be computed. The density area is divided into an increasing number of contiguous zones from time to time and the sets of N-grams belonging to the different zones are constructed. When the density area is consistent with the query word, consistency between the two PNGRs must be maintained at all levels. To assess the consistency, the number of N-grammes of the PNGR of the density area that does not match the PNGR of the word query is counted. An N-gram of the density area is inconsistent with the word query representation if it is present in a set of N-grams at a particular level of the density area representation but is not present in the relative set

of the query word representation at the same level. The confidence value can then be reshaped based on the ratio between the inconsistent and consistent N-grams. If all N-grams from level 2 of the representation are inconsistent, the confidence value is reduced by 100%. If, on the other hand, all N-grams are consistent, the restructuring has no effect on the confidence value:

$$sc = sc \cdot \frac{\#Correct\_in\_PNGR}{\#Total\_in\_PNGR} \tag{5.6}$$

Computing the *sc* score, to each detected density area is assigned a confidence measure, the higher the more likely it is to contain an instance of the query word. In this way, once the system receives a query word, it can return a list of all the areas that may contain the word, each with its confidence measure.

This page intentionally left blank.

Chapter 6

Experimental Evaluations

In this chapter, we will present an experimental evaluation of the methods proposed in the previous chapters. We start with a validation of the model presented in chapter 3. We then present the methodology for defining labelled words, focusing in particular on testing the line segmentation method and the transcript alignment method proposed in chapter 4. Finally, we analyse the results of a KWS system based on N-gram retrieval method described in chapter 5.

6.1 The Performance Model Validation

In Chapter 3, we made several assumptions that allowed us to define an estimate of the gain G of the transcription time obtained using a KWS system to support the process. In order to assess whether and to what extent the assumptions made allow a reliable estimate of the effective value of G, we conducted a series of experiments to compare the estimated value of G with the actual value. The experiments involved three experts who transcribed the pages of the set DS. The pages were transcribed manually by alternating 20-minute transcription sessions with 10-minute rest periods, as it is common practice to avoid fatigue effects. The transcription sessions were conducted by two ex-

perts, and during the break session, another expert checked for inconsistencies between the two transcripts to obtain an error-free transcription. The experts were palaeographers with more than 10 years of experience and basic knowledge in information technology, especially in word processing, spell checking and annotation tools. Before the transcription, all experts were trained in the use of the tool's graphical user interface (GUI). There were three training sessions. In the first session, which lasted 60 minutes, they were familiarised with the main functions of the GUI for both transcription and validation, while in the remaining two sessions they were able to practice using the GUI for transcription and validation until they felt comfortable with it. It took the experts less than 30 minutes to become familiar with the functioning of transcription, while it took them a little longer - a few minutes - to master the GUI for validation. The experiments were conducted with 50 pages of the Bentham Collection data set. We used 5 pages of the dataset as TS, 5 pages as TTS and the remaining 40 pages as DS. Table 6.1 shows the composition of each set, where with n are counted *all* the words in a set and with N are counted the number of *different* words into a set.

Table 6.1: The composition of the dataset used in the experimental work. DC, data collection; TS, training set; TTS, test set; DS, data set

n_{DC}	n_{TS}	N_{TS}	n_{TTS}	N_{TTS}	n_{DS}
10733	1089	354	942	391	8702

During each session, we recorded the expert's activity via the user interface and calculated the value of t_i^M, i.e. the time taken to transcribe an instance of the i-th word in the data collection. From the recorded data, we calculated the mean μ and the standard deviation σ of these values over the entire data set. The result was $\mu = 5.81$ and $\sigma = 1.237$ sec for the first expert, and $\mu = 5.65$ and $\sigma = 1.251$ sec for the second. No statistically significant differences were found between the samples of the two experts. This allows us to say that the

experts were both familiar with the use of the interface and procedures and that the values recorded are statistically significant.

The mean value $\mu = 5.73$ was then selected as the actual value for estimating the user time. There were only 12 words on the 10 pages for which the two experts provided different transcriptions. After assessing the performance of the two experts, each transcribed half of the pages from DS. The sum of the time taken to transcribe the pages of DS and the minimum time taken to transcribe TS and TTS was taken as the T_{man} time for the manual transcription of DS. We considered minimum times instead of average times to analyze the worst case and make the estimation in a conservative way. Their values are given in Table 6.2

Table 6.2: Times to manually transcribe the training set, the test set, the dataset, and the whole collection. The times are in seconds.

T_{TS}	T_{TTS}	T_{DS}	T_{man}
6240	5472	52459	61534

6.1.1 The Validation Tool

To evaluate the performance of the KWS system in supporting transcription, we have developed a validation tool to process the system output. In this section, we briefly describe the use of the graphical tool. As we have mentioned in chapter 3, the values of the times t_i^v, t_i^w, t_i^m and $t_i^{M_w}$ depend on the user interface of the validation tool. In our case, the user interface appears as in Figure 6.1 as soon as a page of the collection has been opened for validation. The top of the user interface displays the current line of text in the document, with each word enclosed by its bounding box provided by the segmentation step. In the middle of the user interface, the main field displays the current word, i.e. the word being validated, and directly below it the field for its manual transcription. The rightmost field displays the output list containing all the

proposed transcriptions for the current word and the lowest part of the user interface contains a text area that displays the transcription of the entire page line by line and is updated as the transcription progresses.

Figure 6.1: The user interface of the validation tool.

In the case of a correct sample, the output list contains the correct interpretation. The user thus searches for the correct transcription and confirms the word as soon as he has found it by clicking on the correct transcription. The interface then displays the transcription on the corresponding word image in the text line as well as in the lowest field, as shown in Figure 6.2, and moves on to the next word to be transcribed.

In the case of an incorrect example, the correct transcription is not present in the output list, as shown in Figure 6.3, but the word is an instance of a keyword included in the query list. In this case, the user must enter the transcription. To speed up this process, the interface provides an auto-complete mode, i.e. as soon as the user enters the first characters, the system updates the output list by displaying all entries in the query list that match the characters entered so far. As soon as the correct transcription is displayed on the user interface (see

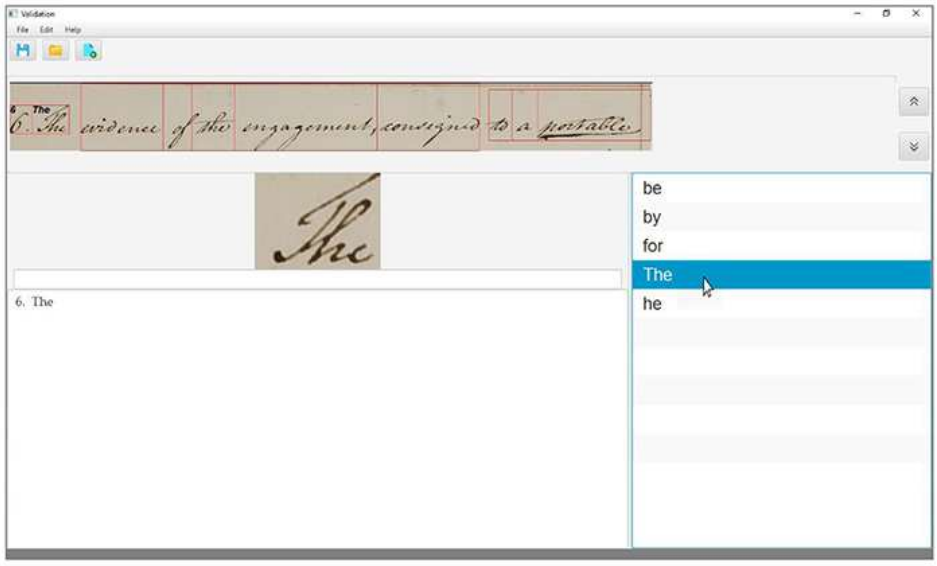


Figure 6.2: The user interface when transcribing a correct sample.

figure 6.4), the user can confirm it by clicking on it, as in the previous case, or confirming using the *enter* key.

In the case of a missing word, the output list is empty, so the user has to enter the transcription manually. As in the previous case, the auto-complete mode in the output list shows all entries in the query list that match the string entered so far, and as soon as the correct entry appears, the user can continue by simply clicking on it.

If the current word is an OOV word, the system can display either an empty or a non-empty output list, depending on whether the OOV is correct or incorrect. In both cases, the user must type in the entire transcription, but in the case of a wrong OOV, the user will first search the output list for the transcription and only then start transcribing, as shown in Figure 6.5.

During validation sessions, the tool logs all user actions and the time spent by the user on each action, making it possible to calculate the number of correct, incorrect, missed, OOV-correct and OOV-wrong words, as well as the times taken to reach their transcriptions.

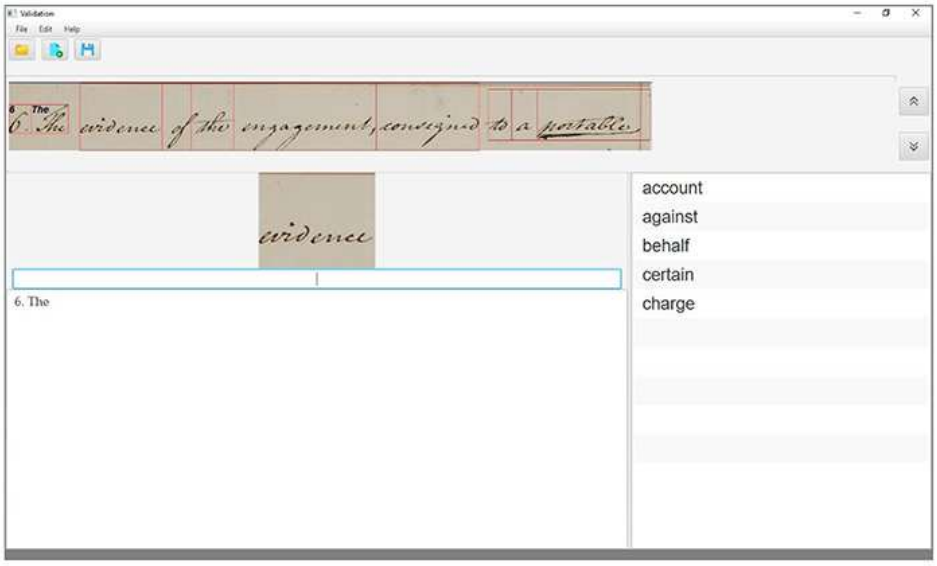


Figure 6.3: The user interface when transcribing a wrong sample.

6.1.2 Experimental Results

In the experimentation, we used a KWS system whose architecture and operation can be summarised as follows. Essentially, it is built on two main blocks: the reference set (RS) and the knowledge base (KB). RS is built by processing the word image from TS in such a way as to recover the trajectory [16], decomposing it into elementary parts called *strokes* [19], and finally labelling each stroke with the character to which it belongs [111]. Thus, each word is represented by a string consisting of as many characters as strokes extracted from the ink trace. Then, each word from DS is decomposed into strokes as before, but the labelling of the strokes is determined by matching each word from DS with all words from RS: whenever a sequence of strokes is found whose shape is similar, the labels of these strokes found in the words from RS are copied to the matching stroke of the word from DS [18]. Thus, each word from DS is connected to a graph whose number of nodes is equal to the number of strokes, and each node is labelled with a character if the corresponding stroke matches one of the strokes from RS. When searching for a query, the KWS

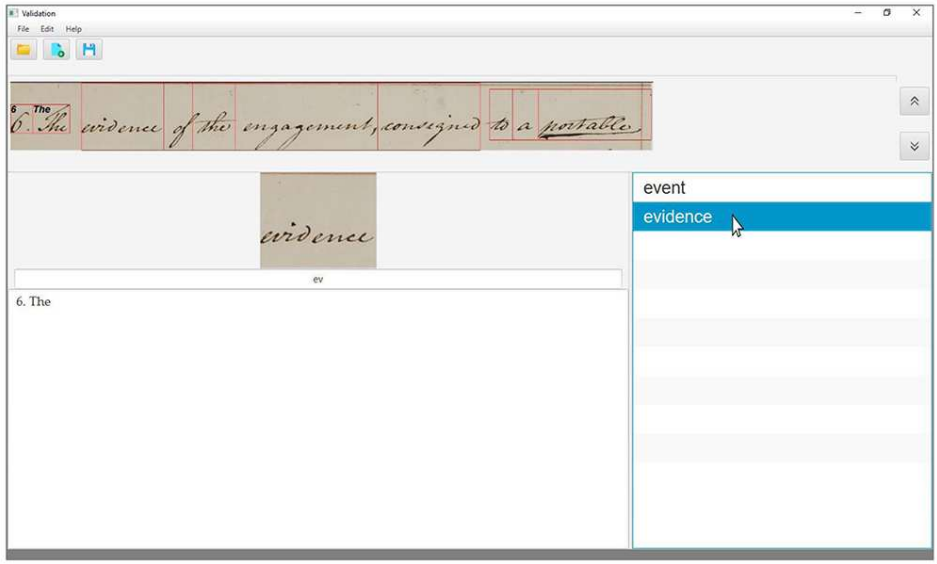


Figure 6.4: The user interface after a few characters of the word to transcribe have been entered. The output list is updated and, once the correct transcription appears in the box, the user validates the correct transcription by just a click or using the enter key.

system searches within the graph of each word of DS for a path whose nodes correspond to the characters of the query [20]. When such a path is found, the word is returned in response to the query. Due to the multiple labelling of the strokes, the same word image may be returned in response to different queries and a word may be discovered that is not an instance of a keyword, allowing the system to detect OOV words.

An initial experiment was conducted to determine whether and to what extent the length of the output list affects the time taken to validate the system outputs. The validation tool was configured to return the top 5, top 10, and top 15 interpretations for each word image. For each configuration, a different expert performed the transcription of a batch of five pages DS. The total time to complete the task as k varies is reported in Table 6.3. These results suggest that the value of $k = 10$ is the best compromise for our user interface because for a longer list, the time to find the correct transcription in the list counterbalances

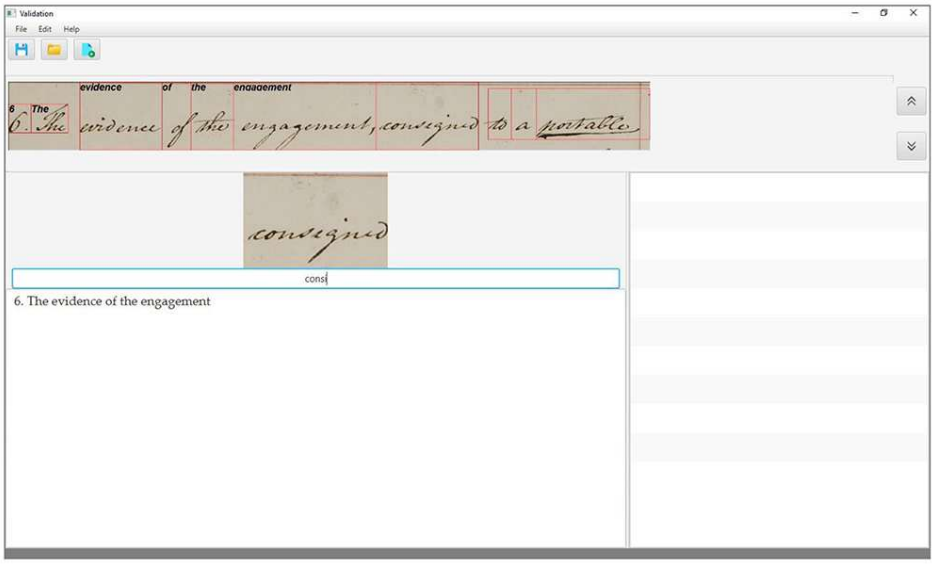


Figure 6.5: The word "*consigned*" is a correct out-of-vocabulary (OOV) word: the output list is empty and the user enters the entire transcription.

the improvement in recall. This is consistent with the observation that people can search at a glance within a list of about five items, but as the list grows longer, search time grows almost linearly with the number of items in the list.

Table 6.3: Time to complete the transcription of DS as k varies.

k	**Time in sec**
5	2996.062
10	2835.130
15	3314.240

In the second experiment, we performed keyword spotting in the set TTS and calculated the mean and standard deviation of the model parameters by recording user activity, as shown in Table 6.4.

Since the KWS is a lexicon-free system, to simulate a lexicon-based system, we have disabled the output list display interface if the words are OOV words

Table 6.4: The mean and the standard deviation of the model parameters estimated on TTS. The times are in milliseconds.

t^v		t^w		t^m		t^{M_w}		r^k	p^k	n^{oov^c}	n^{oov^w}
μ	σ	μ	σ	μ	σ	μ	σ				
1024	359	3152	1045	2543	682	5903	6211	0.65	0.71	39	247

(we know this because we have transcribed DS manually for performance evaluation) and do not update the query list. In contrast, we enable the display of the output list in the case of a lexicon-free system and add the unique words obtained by transcribing the OOV to the query list. So, using the values in Table 6.4, we calculated the sum on the left-hand side of inequality (15), replacing n_{DS} with n_{TTS}. Then, by adding T_{TS} and T_{OOV} (using t^M and t^{M_w} for the lexicon-based and lexicon-free cases, respectively), we calculated T_{user} and, finally, the gain G. We then calculated G using the actual user time recorded by the tool to complete the task, and then added T_{TS} to calculate T_{user} and the gain G. Table 6.5 shows the value of T_{user} and G estimated using our model and the actual value for both the lexicon-based and lexicon-free configurations of the KWS system. As shown in Table 6.5, the model provides a reliable estimate of G in both cases. However, in the case of a lexicon-free system, the model shows the largest difference between the estimated and the actual value of G.

Table 6.5: Comparison between the values provided by our model and the actual ones on TTS. Times are expressed in the format hh:mm:ss.

Values	**Lexicon-Based**		**Lexicon-Free**	
on TTS	T_{user}	$G(\%)$	T_{user}	$G(\%)$
estimated	01:02:12	14.86	00:51:21	20.41
actual	01:04:48	12.14	01:01:30	16.61

In the last experiment, the expert who transcribed the first 20 pages of DS

performed the validation of the system output on the remaining 20 pages, while the second expert who transcribed the last 20 pages validated the system output on the first 20 pages. This procedure was chosen to avoid the memory effect that could have altered the time spent if they had performed the validation on the same pages they had already transcribed. Table 6.6 shows the results of the experiment. They show that, as in the case of TTS, the estimated values on DS are an upper bound for the actual values, and that the difference in percentage between the estimated and the actual values is similar to the one in the case of TTS, confirming that it is possible to derive a reliable estimate of the actual value of G for the entire data collection from the model's estimates on TTS.

Table 6.6: Comparison between the values provided by our model and the actual ones on DS. Times are expressed in the format hh:mm:ss.

Values	**Lexicon-Based**		**Lexicon-Free**	
on DS	T_{user}	$G(\%)$	T_{user}	$G(\%)$
estimated	11:27:37	13.91	10:30:30	19.25
actual	11:31:02	12.38	11:10:14	15.02

6.1.3 Discussion

In this paragraph, we addressed the problem of estimating the time required by the user to obtain a complete and correct transcription of small collections of historical documents when using a KWS system that can provide multiple possible transcriptions for each image word in the collection, compared to the time required by the user for manual transcription. The model shows that the reduction in user time depends on both the performance of the KWS system and the user interface of the validator. In particular, it is shown that, for the same precision and recall, i.e. for a *given* KWS system, the effective time saving depends on the time required to process the different types of output (correct, incorrect, missing, and OOV) compared to the time required to man-

ually transcribe the corresponding word, so that the smaller the ratio between the processing time of the outputs and the time required to manually transcribe it, the greater the time savings for the user. Conversely, *given* a user interface to be used for validation, the KWS system must have a minimum level of performance to be useful for assisting the transcription. This interaction between the performance of the KWS system and the time efficiency of the user interface should be carefully considered when designing a system that aims at minimizing human efforts when dealing with small collections of handwritten historical documents.

In the case of a lexicon-based system, the model shows that the performance benefits of the KWS are limited to the word images in the dataset that are instances of the lexicon keywords; that is, the more keywords we target, the greater the potential benefit. Therefore, a multi-step procedure could be adopted to create a training set with as many keywords as possible and split the dataset into batches. Then the results are validated and if the user enters new keywords in response to an incorrect output, they are added to the keyword list and the next batch is processed.

For lexicon-free systems, the model shows that the more wrong OOVs the system finds, the greater the disadvantage of the KWS system compared to lexicon-based KWS and manual transcription, but also that these advantages can be mitigated by updating the query list. Although the mitigation mainly depends on the distribution of keyword patterns in the test set and data set, the higher the performance of the KWS system in recognizing OOV words, the lower the number of false OOVs found, and this can ensure the viability of lexicon-free KWS compared to lexicon-based or manual transcription.

in conclusion, it is interesting to read table 6.6 explaining directly the relation with time T_{man}, i.e., the time for the complete manual transcription of the collection. In the case of the set DS, the recorded time T_{man} is equal to 13:08:42. Looking at the values given in the table for the *actual* line, considering for example a lexicon-free system, we find that the transcription can be

completed in a time of 11:10:14. Thus, the 15% gain represents a time savings of almost 2 hours, time actually saved by the user, which may translate into a reduction in transcription costs. However, given the time gain estimated by the model, we would have expected a time saving of about two hours and forty minutes. This means that in reality, it took the experts just under one minute less per page to transcribe the collection than the time estimated by the model. This illustrates that the model can provide a realistic estimate of how the time required for transcription can be reduced by choosing an appropriate KWS system to support the transcription process.

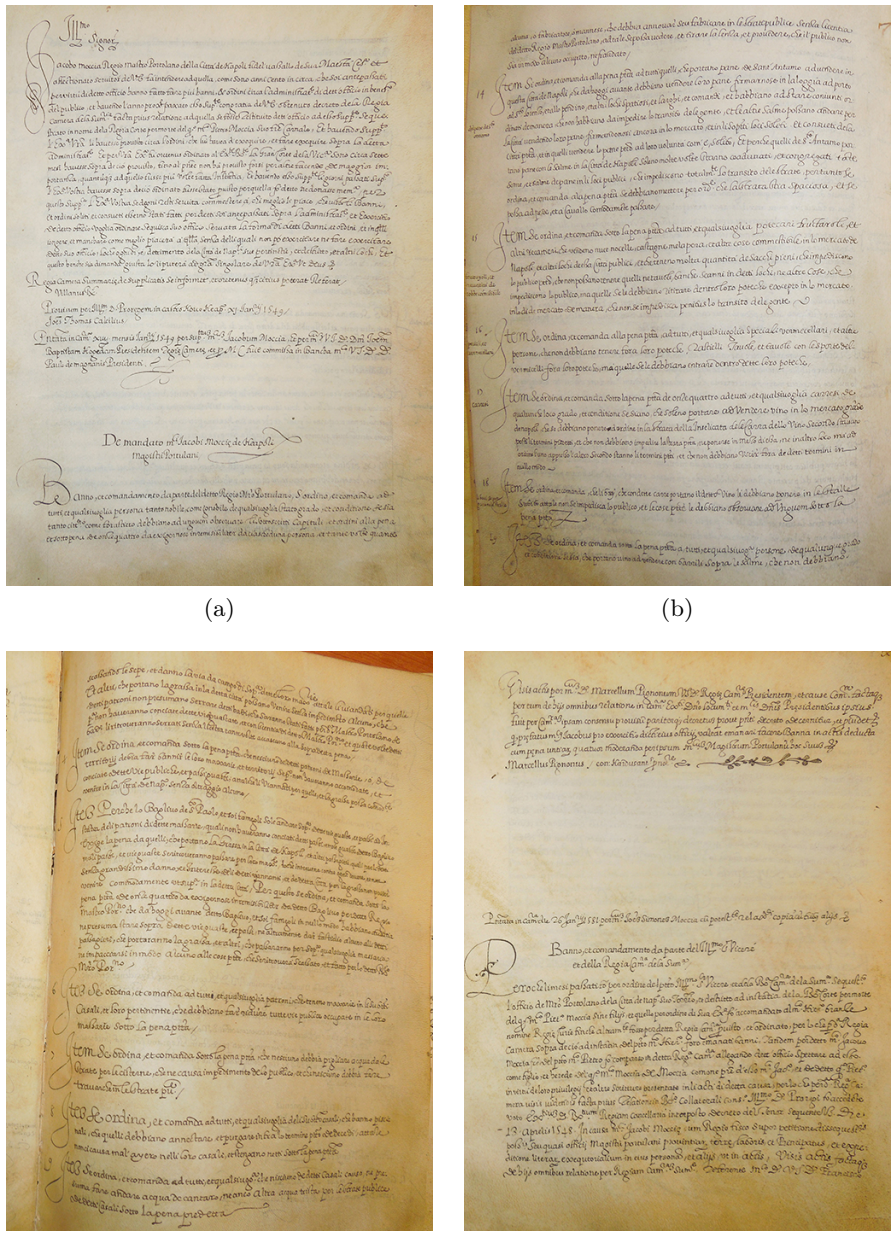
6.2 Tools and Methodologies to Labelling Data

in Chapter 4 we have proposed some methodologies to speed up the process of data labelling for building a dataset composed of images of handwritten text labelled with its transcription. In this chapter, we present experimental validation for the handwritten line segmentation method and for the transcript alignment.

6.2.1 The Moccia-Code Dataset

To evaluate the performance of the line segmentation and transcript alignment methods, we used images from a seventeenth-century documentary manuscript. The script belongs to the bastard Italian typologies widely used in chancery in the sixteenth to seventeenth centuries, executed in brown ink, sometimes lighter, mainly by a single hand. The pages are easily readable, and their layout is fairly regular. The manuscript entitled *Code of the Moccia family. Privileges, Investitures, Announcements and Decrees, 1449-1610*, is currently preserved in the archives of the Salerno-Lucan Province of Friars Minor, based in the Monastery of the Holy Trinity in Baronissi (Italy, Salerno) [11]. From now on we will refer to this collection as *Moccia Code*.

As part of the typology of the book document and, in particular, the car-



(a) (b) (c) (d)

Figure 6.6: **Examples of pages from the Moccia Code:** (a) f. 6r; (b) f. 7v; (c) f. 9r; (d) f. 10r.

tulary, in which, in addition to structure and content, are taken up purposes and functions, especially those of "I remember for the future" and documentary preservation, it formed a kind of family 'archive casket' and *munimen*, as a collection of documents that certified the rights and privileges of the family, to be presented in court in case of possible summons, or to verify the "legitimacy of the titles held by the holders of the offices", which, in short, could be useful for the defense of a representative of a Neapolitan family, the Moccias, who held the office of Portolania in the Kingdom of Naples since the fifteenth century and throughout the modern era, in the inquisitorial and judicial phase. In addition to three of Alfonso il Magnanimo, Ferrante I. and Federico d'Aragona, which refer to the granting, confirmation and re-granting of the privilege of "mastro portolano" and procurator of the province of Terra di Lavoro to members of the Moccia family, there are copies, mainly in the form of a simple copy, administrative documents (instructions to officials, orders, regulations) and judicial communications (*acta*, summonses, decrees, execution of judgments) issued by the main magistracy and provincial offices of the Kingdom of Naples between the 15th century and the early 17th century. and the early 17th century. Probably the transcription was made directly from the originals or, in any case, from loose documents in the possession of the principal (or recipients) or temporarily available for recording. About thirty documents contain the registration data of the original document in the Royal Chamber of Sommaria, in the Archives of the Aragonese Kings and Viceroy, and in the office of the "mastro portolano"

The book was probably commissioned by one of the last representatives of the family, perhaps a certain Giovanni Simone Moccia, whose titles of ownership were contested and who was accused of abuse in the exercise of the Portolania office. Here the transcription does not provide for the completion of the abbreviated words, although occasionally the omitted nasals and the development of special graphic signs related to the enclitic *-ue*, to the ending *us*, *q* and *p* were indicated in round brackets

As Figure 6.6 shows, the Moccia-Code contains mainly text documents,

which simplifies the segmentation of text lines, but presents a very challenging testbed for transcription alignment, due to irregular line spacing (6.6a), uneven illumination conditions (6.6.b), curved text lines (6.6.c), uneven background due to ageing (6.6.d). In addition, there are overlaps of ascenders and descenders, handwriting is small, spacing between words is very irregular, and abbreviations are frequently used in the documents

For our experiments, we selected images corresponding to the front/back of five paper documents of the *Moccia Code*, referred to as *ff. 6r-10r*, whose line-by-line transcription is available in [11].

6.2.2 The Line Segmentation Method

To test the proposed segmentation method, we conducted experiments on different datasets showing the performance of the method on the Bentham Collection [107], George Washington [94], Jefferson's letter [131], Sant Gall [31] and finally the Moccia-Code. For a proper comparison with the state of the art, segmentation methods available in the literature were selected and segmentation was performed for all selected methods on all available datasets. The methods selected for comparison are those of Surinta et al. [128], which served as an inspiration for our method, the method proposed by Alberti et al. [3], which was chosen because its performance is comparable to the best methods in the two recent text line segmentation competitions [23, 24], and the deep methods docExtractor [73] and dhSegment [79] because trained models (on a large dataset) are proposed as "off-the-shelf" tools that can be used without further training and can thus be adapted to the application scenario we are dealing with.

The table 6.7 contains the results of the experiments on the segmentation of text lines on all selected datasets.

The row segmentation results listed in Table 6.7 show that the proposed method outperforms both the learning-free method proposed in [128] and the learning-based method recently proposed in [3], albeit to different degrees, and

Table 6.7: Line segmentation results. The table reports, for each data set, the actual number and the number of correctly segmented text lines provided by each method.

		our	**Surinta et al.**	**Alberti et al.**	**docExtractor**	**dhSegment**
Dataset	*N Lines*		[128]	[3]	[73]	[79]
Moccia	275	253	153	144	274	267
Code		92.00%	55.64%	52.36%	99.64%	97.09%
Bentham	1056	1004	924	1003	1040	967
Collection		95.08%	87.50%	94.98%	98.48%	92.45%
George	653	600	585	587	632	635
Washington		91.88%	89.59%	89.89%	96.78%	97.24%
Jefferson	23	22	19	19	22	23
Letter		95.65%	82.61%	82.61%	95.65%	100.00%
Saint Gall	1430	1415	1351	1387	1419	1420
		98.95%	94.48%	96.99%	99.23%	99.30%

that in both cases the greatest improvements are achieved on the Moccia-Code. This is due to the curved baselines in this collection, which neither method can deal with effectively, as they tend to merge multiple lines when the baseline of a line bends along the writing direction. The results also show that the deep docExtractor and dhSegment methods perform the best, as they are still effective when the other methods, including those presented in this work, fail. It is interesting to note, however, that docExtractor returns only the region encompassing the central region of the handwriting, thus ignoring both ascenders and descenders, while dhSegment returns only the baseline. In the case of applications where the entire ink of the text line needs to be available for subsequent steps, as in our case, it is necessary to develop and integrate some ad hoc post-processors into the application workflow, so acquisition may not be the best option.

6.2.3 The Transcript Alignment

The performance of an alignment method is usually expressed in terms of accuracy, i.e. the ratio between the number of correct alignments between the image word and the corresponding transcript, and the total number of transcripts to be aligned. To implement this definition, the basic idea is to evaluate, for each transcription, the overlap between the word image in the ground truth and the image produced by the system, and to hypothesise that the word image matches its transcription if the overlap with the ground truth image is above a threshold. However, this type of accuracy assessment does not take into account the actual size and position of the bounding boxes. Conversely, depending on the application, whether or not the non-overlapping area contains a significant amount of ink may be important. Therefore, the accuracy given may not reflect the actual accuracy.

To get around this, we implemented a performance evaluation by visual inspection, where we assume that a transcript has been correctly aligned to a bounding box if the image word within the bounding box contains ink corre-

sponding to the characters of the transcript. In the following, this condition is referred to as *perfect* alignment. This is a very restrictive condition, as it results in the cases shown in Figure 6.7(a) being considered misalignments, even if the missing part/added ink does not prevent the transcript from being read. We have therefore considered a less restrictive alignment, which we refer to below as *acceptable* Alignment. Here we assume that a transcript is correctly aligned even if the *maximal* bounding box lacks the ink of the initial/final character of the transcript or if it contains the ink of *maximal* the initial/final character of the neighbouring transcripts in addition to the ink corresponding to the transcripts. Under this condition, the case shown in Figure 6.7(b) can be considered as a correct alignment where the bounding boxes are missing or contain the ink corresponding to a character. Finally, it should be noted that this condition allows two or more transcripts to be considered aligned even if they are aligned within the same bounding box, as shown in Fig. 6.7(c), i.e. when a partial segmentation has been treated by merging the transcripts instead of dividing the bounding box.

To enable reviewers to calculate the performance of alignment methods, a graphical interface for validating and correcting method output was designed and implemented. Figure 6.8 shows the interface in use. From time to time, the interface displays all detected alignments and highlights the position of the current word within the text line with a green box. If a user detects an incorrect alignment, they can correct it with the mouse in a simple and intuitive way. Right-clicking on the image of the line allows the user to change the right vertex of the detected box, while left-clicking on the text allows the user to change the left vertex of the box. An example of a correction is shown in figure 6.8. At the end of the validation process, the interface allows to count the number of human interventions and thus the performance of the alignment method. Trivially, if the user does not make any corrections, this means that the alignment method was able to correctly align the entire data collection. Using the interface's log files, we can also track the time the user takes to validate and thus fully align

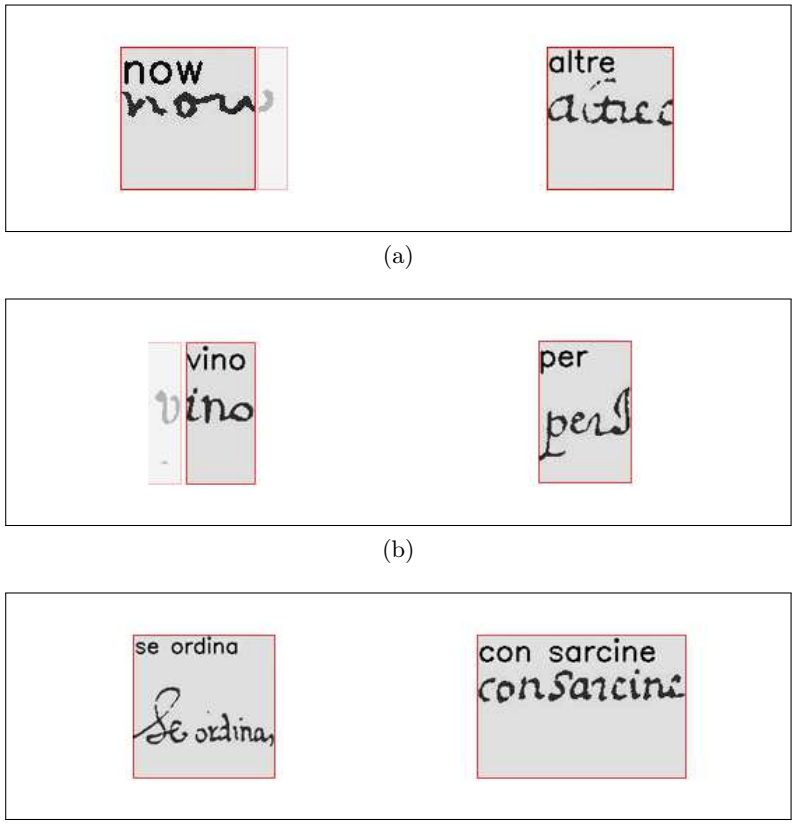
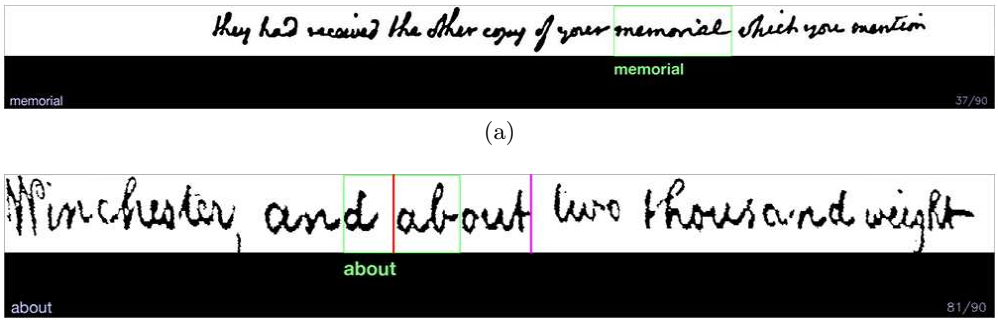


Figure 6.7: Examples of acceptable alignments on the Moccia-Code: (a) missing/added strokes; (b) missing/added characters; (c) multiple-to-one alignment.

the dataset, i.e. segment the entire collection at the word level without errors.



(a)

(b)

Figure 6.8: Graphical User Interface to assist the validation process of the alignment transcription method. (a) Sows the case of a correct alignment. (b)shows the case of an incorrect alignment, in this case, the new vertices for the box are visualized in red (the left one) and purple (the right one).

In order to mitigate as much as possible the bias that arises from the subjective assessment of our definition, the assessment was carried out independently by two people. If the experts assessed the same alignment differently, the final decision was left to a third party. Table 6.8 shows the performance obtained with the Moccia Code when a perfect or acceptable alignment was adopted.

Table 6.8: Performance of the alignment method and its variants on the Moccia Code. The results are given in terms of accuracy. The best results are in boldface.

Alignment	**Forward**	**MiM**
Perfect	**45.05 %**	42.47%
Acceptable	67.59 %	**68.39%**

Table 6.8 shows that there is no clear indication of which alignment strategy works best between the Forward and Mim variants we have implemented. The former is superior to the latter in terms of perfect alignment, while the opposite is true, albeit to a limited extent, in terms of acceptable alignment. However, the best results of both methods in terms of acceptable alignments show that word segmentation errors are mainly due to bounding boxes missing

(or containing) at most one character (belonging to adjacent transcripts) and that the MiM strategy performs slightly better than the Forward strategy for segmentation errors occurring simultaneously on both sides of the text line.

To compare our method with the state of the art, we list in Table 6.9 the alignment methods proposed in the literature over the last twenty years. For each method, we indicate the dataset they use, the type of writing of the documents, the historical period of creation, whether it is publicly available or available on request from collectors, the method used for text alignment and the reported performance. As we can note from the table, some collections are not available, and others contain digital images of the documents but the transcript of the content is not available (and we were unable to provide it ourselves because of the language). Two of them contain old printed or encrypted documents. Moreover, no implementations of the methods are available, and the relevant documents do not contain all the details for a re-implementation. Finally, in the case of Corpus Cristo Salvador, the composition of the test set used in the performance assessment is neither specified in the document nor identified in the data set. All these problems limit the possibility of a direct comparison. However, it was possible to evaluate our alignment method on some of the data sets used by the various previous methods. In particular, we evaluated our method on the Bentham Collection, George Washington and Jefferson Letter datasets, as the test sets are publicly available in the repository. Table 6.10 shows the results of our method and those of its competitors. We note that for a fair comparison with the metrics used in the other studies, we report the results of our method in terms of *perfect* alignment precision.

Error analysis has shown that most alignment errors are due to irregularities in both the spacing between words and the size of the bounding boxes containing the word image of the same transcript. Indeed, we have found that most errors occur along lines of text which, as the type approaches the right-hand margin, show a narrowing/widening of both the spaces between words and the horizontal width of the type, which will most likely allow you to write a whole word before

Table 6.9: Overview of alignment methods in the literature. The period of production is expressed by the ordinal of the century AC. Results are reported in terms of accuracy.

	Type	Dataset	Period	Available	Method	Result
[131]	Handwritten	Thomas Jefferson Letter	XVIII	Yes	Dynamic Programming	72.00%
[57]	Handwritten	George Washington	XVIII	Yes	Dynamic Time Warping	75.40%
[103]	Handwritten	George Washington	XVIII	Yes	HMM	72.80%
[133]	Handwritten	Corpus Cristo Salvador	XIX	Yes	HMM	92.80%
[48]	Handwritten	The Swiss Literary Archives	XX	No	HMM	94.66%
[147]	Handwritten	Kabinet van de Koningin (KdK) collection	XIX	Only images Transcription not available	Ink Projection Segmentation	69.00%
[123]	Handwritten	ICDAR2009 test set	XXI	No	Word Segmentation	97.04%
[122]	Handwritten	ICDAR2009 test set	XXI	No	Word Segmentation	99.48%
[61]	Handwritten	Queste del Saint Graal	IX	Yes	Segmentation Free	72.90%
[101]	Handwritten	C5 Hattem Manuscript	XVI	Only images Transcription not available	HMM - Dynamic programming	75.50%
[17]	Handwritten	Bentham Collection	XVIII	Yes	Ink Projection	75.93%
[148]	Early Printed	Gutenberg Bible	XV	Yes	CNN-based	90.00%
[132]	Chipered	Copiale ciphered manuscript	XVIII	Yes	Attention based	90.00%

Table 6.10: Comparison of alignment results with the state of art. The results are given in terms of accuracy. The best results are in boldface.

Database	Method	Result
Bentham Collection	[17]	75.93%
	our	**77.20%**
George Washington	[57]	75.40%
	[103]	72.80%
	our	**79.76%**
Jefferson Letter	[131]	72.00%
	our	**88.80%**

the margin. For abbreviations where the last part of the word is displayed superscript, the method tends to misalign. This affects the pixel size of the abbreviated word image and makes alignment more complex for the method. Another feature that is common in historical manuscripts is that the first or last letter of a paragraph is often capitalised and/or flowery. This change in writing style also complicates the automatic alignment of the method. Whenever a line of text has these features, the method fails to align the transcript to the leftmost and/or rightmost edge of the line, and the errors spread to both sides of the line.

Time Required to Achieve Proper Word-Segmentation

After running the alignment algorithm and correcting the output of the method via the interface shown in Figure 6.8, we get the correct word-level segmentation of the document images. The correction is a process that must be performed by an expert user who can read and interpret the script to judge correct alignment. However, an expert user is able to correctly segment a text into its words without first having to perform alignment using the suggested methods, and once in possession of correct word-level segmentation, alignment with transcription

is a trivial and immediate process. Both approaches require the intervention of a user, it is, therefore, legitimate to ask which of the two methods is more convenient from the user's point of view. Table 6.11 shows the times recorded by the interface for validating and correcting the alignment on different datasets: 5 pages of the Bentham collection (*Bentham*5), 20 pages of the Bentham collection (*Bentham*20), 50 pages of the George Washington, and the Jefferson's letter.

Table 6.11: Time required to perform the validation of the transcript alignment method.

Dataset	**Total Time (hh:mm:ss)**
*Bentham*5	00:11:36
*Bentham*20	00:57:25
George Washington	01:03:44
Jefferson's letter	00:03:08

An experienced user performed a full manual segmentation of the *Bentham*5 and *Bentham*20 datasets using a dedicated word segmentation tool. The tool allows the user to work on images of the documents and draw bounding boxes for words providing special features to speed up the segmentation process. However, segmenting in this way does not take advantage of any automated process, as is the case with our alignment method. Table 6.12 shows the times recorded for manual segmentation.

Table 6.12: Time required for fully manual word segmentation.

Dataset	**Total Time (hh:mm:ss)**
*Bentham*5	00:41:40
*Bentham*20	02:52:40

As can be seen from the comparison between the times shown in the two

tables 6.11 and 6.12, the use of the proposed procedure makes it possible to achieve the objective of word segmentation by saving up to about 70% of the time necessary in the case of manual segmentation.

6.2.4 Discussion

In this section, we presented an experiment for the line segmentation and transcript alignment techniques presented in Chapter 4. The performance of the proposed solution was evaluated using the Moccia-Code, a small collection of handwritten historical documents that proved to be very challenging compared to some of the datasets currently used as benchmarks for performance evaluation.

The performance of the line segmentation algorithm shows that it outperforms the state-of-the-art, with the sole exception of *docExtractor* and *dhSegment*, but none of them outputs the entire handwriting of the text line, which limits its usability without further ad-hoc determinations as in our case, as well as for keyword spotting, writer identification, and handwriting recognition. The segmentation method divides the document into a fixed number of stripes, which was set to 8 in the experimental phase. This value has shown the best performance on the datasets considered. However, there is no indication that this value should be considered the best in more general cases. This default may prove to be a weakness of the system, and an automatic estimation system for the optimal value may prove to be essential to obtain the best performance on additional data collections different from those tested.

As for alignment performance, the results of a direct comparison with state-of-the-art learning-free methods have shown that the proposed method outperforms its competitors on every dataset tested. They also show that they are at least qualitatively consistent with the performance of other state-of-the-art methods on datasets that were not available for direct comparison. Finally, it is worth noting that in the case of the Moccia-Code, the performances of our method are lower than those obtained on the datasets currently used in the

literature as benchmarks for performance evaluation. This confirms that the Moccia-Code is much more challenging for text alignment of mostly text-based handwritten documents.

The experimental results obtained so far not only provide convincing evidence that our methods are an effective fully automatic solution for text alignment of mostly text-based historical handwritten documents, but also give hints for possible future developments. If the results of the consistency test indicate that an under- or over-segmentation error has occurred, the alignment algorithm attempts to correct this error by using information from the neighbouring frame/transcript. Even if at the end of the process it turns out that something went wrong because a transcript or a bounding box is misaligned, the algorithm does not allow to undo previous decisions and trying an alternative word segmentation. Furthermore, we assumed that the spatial dimension in pixels of a character is constant regardless of the single character. This is a very restrictive assumption that can be relaxed considering that each character class can have its own size.

6.3 KWS by N-gram Retrieval Validation

In chapter 5 we presented a system for spotting OOV words that bases the search core on the detection of N-grams within a line of handwritten text. We proposed two solutions for the N-gram QbE system. The first one is based on a sliding window architecture, while the second one uses the attention mechanism to search for N-grams. In this section, we will evaluate the performance of the two methods by highlighting the retrieval capabilities of the proposed solutions.

6.3.1 N-gram Retrieval

Under the premises of the whole work, the definition of our application scenario as a collection of very small handwritten documents will always remain. To simulate the conditions of the application scenario as well as possible, we

considered a small subset of the Bentham Collection for the experiments. 20 pages were selected, 5 of which were used as a training set. The images of the documents were first binarised using the method of Sauvola et al. [108]. Then the documents are segmented into lines of text, and the transcript of the training set was used to align the individual transcripts with the images of the words present in the documents, using the methods described in the chapter 4.

From the set of correctly labelled images, the images of the N-grams present in the entire training set could be extracted, and using these images to build the set of reference N-grams of the system. To perform the extraction of the correctly labelled N-grams, a graphical interface was designed and implemented to facilitate the process. Figure 6.9 shows an example of this interface. Starting from a dataset consisting of correctly labelled words, the software automatically extracts the images of the N-grams. The tool allows the user to view all the N-grams extracted starting from the original word and correct any extraction errors by manually changing the N-gram crop box.

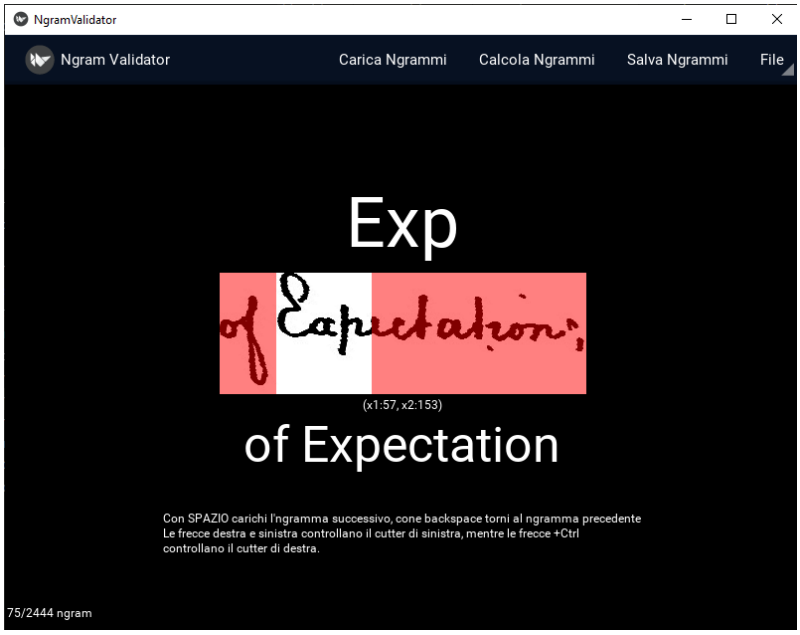


Figure 6.9: Graphical User Interface to extract and validate the N-gram images from a set of labelled word images).

Using the interface, all the N-grams with N equal to 2 and 3 were extracted. The decision to limit N to 3 is due to the fact that considering character sequences consisting of more than 3 characters would, in our opinion, contradict the premises of this work. We want to use N-grams as recognition primitives and keep the sequences large enough to facilitate segmentation but small enough to remain effective recognition primitives. After extracting the N-grams from the training set, we obtain a set of N-grams consisting of 1044 different classes. However, the dataset is very imbalanced as the cardinality of each class varies from 1 to 117, with 615 classes consisting of less than 3 elements. To reduce the imbalance, the minimum cardinality of the classes was increased to 3 by simple image transformations and adding noise. In this way, the training set is composed as in table 6.13.

Table 6.13: Dimension of the training set composed of N-grams obtained from 5 fully labelled pages of the Bentham collection

	Pages	**N-grams Class**	**N-grams Items**
Bentham Collection	5	1044	5440

The training set thus defined turns out to be a set of small data. Consequently, it is possible to test and explore scenarios that allow the use of the N-gram spotting system even on data sets with these characteristics. For this reason, the architectures presented are based on a Siamese architecture paradigm, which makes it possible to obtain similarity measures between images even with few exemplars. The basic frameworks for feature extraction of the images used are all pre-trained on large datasets. In other words, we propose a management with "few-shot" learning methods. Such a learning scenario allows a system to learn discriminative patterns even when only a small amount of training data is available so that the model can adapt to the data provided. The model primarily learns a similarity function between a search image and a support image and provides high similarity values for very similar

images.

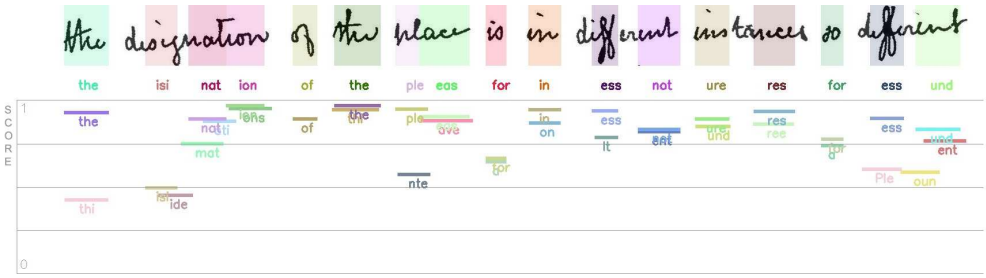


Figure 6.10: Example of the result of the N-gram spotting on a query text line image. The image highlights all the boxes detected in correspondence with each N-gram reporting the scores associated with each box at the bottom.

To discuss which metric should be used to evaluate system performance, we show a sample output in Figure 6.10. The system provides different interpretations for each text area of the text line, each with a different similarity score. However, the system provides a unique interpretation for each N-gram class for each area of the text line, due to the merging of options by the module "*Fusion & Rescore*". The problem we face can be considered as a retrieval problem and the system as a multi-element recommender system, with the peculiarity that the system cannot suggest multiple options belonging to the same class and therefore only one element can be interpreted as relevant. On the other hand, our interest lies in the fact that at least one of the top k options proposed by the system is the correct one, in order to be able to claim that the correct transcription of the N-gram has been identified. We, therefore, propose a modification of the *precision@k* and *recall@k* metrics used in similar problems:

$$recall@k = r^k = \frac{true\_relevant\_n\text{-}gramsAtk}{relevantn\text{-}grams} \tag{6.1}$$

$$precision@k = p^k = \frac{true\_relevant\_n\text{-}gramsAtk}{retrievedn\text{-}grams} \tag{6.2}$$

where $true\_relevant\_n\text{-}gramsAtk$ is the number of N-grams successfully detected given the first k options for each area of the text line, $retrievedn\text{-}grams$ is the number of all N-grams detected within the text line, while $relevantn\text{-}$

grams is the total number of N-grams that make up the text line. It should be noted, however, that the system is not able to recognise N-grams outside the lexicon of N-grams defined in the training phase, but suggests an interpretation for all areas of the queried text line in any case, even if it has a low score. It is therefore interesting to limit the analysis to the N-grams that the system can actually recognise ad retrieve. We will refer to these N-grams as *in-vocabulary*. It is proposed to use as a metric the reference to k, which is restricted only to the N-grams in the vocabulary, defined as:

$$r@k_{InVoc} = \frac{true\_relevant\_n\text{-}gramsAtk}{relevantn\text{-}grams_{InVoc}} \tag{6.3}$$

where $relevantn\text{-}grams_{InVoc}$ is the number of N-grams in the vocabulary that are in the text line.

Sliding-Window Solution

The core of the proposed solution based on the sliding window is the simple Siamese Neural Network presented in section 5.4.1. One branch of the network essentially consists of a convolutional backbone for feature extraction, followed by fully connected layers for refining the embedding of the input image. The PHOCNet network backbone [125] was used for the experiments. The purpose of this network is to compute the PHOC [4] representation of a handwritten text image. It is therefore designed to work with images from the same domain as the images of our N-grams. The PHOCNet backbone used was pre-trained with the IAM handwriting dataset [70], a large handwriting dataset composed of images of handwritten words. Once the backbone architecture was selected, the entire branch of the Siamese network was trained with N-gram images. To train the Siamese architecture, a triplet loss [109] was used. To train the Siamese network branch with a triplet loss, it is necessary to design an architecture with three branches that all have common weights and receive three images as input:

- Anchor sample image A: taken as reference;

- Positive sample P: which belongs to the same class as the anchor;
- Negative sample N: belonging to a different class than the anchor.

Equation 6.4 gives the triple loss:

$$L(A,P,N) = Max\left(\|f(A) - f(P)))\|^2 - \|f(A) - f(N)))\|^2 + \alpha, 0\right) \quad (6.4)$$

where $f(A)$, $f(P)$ and $f(N)$ are the embeddings of A, P and N and α is a hyperparameter to be tuned. By minimising this loss function, the learning process will tend to modify the embeddings so as to reduce the difference between similar images ($\|f(A) - f(P))\|^2$) and maximise the difference between different images ($-\|f(A) - f(N))\|^2$).

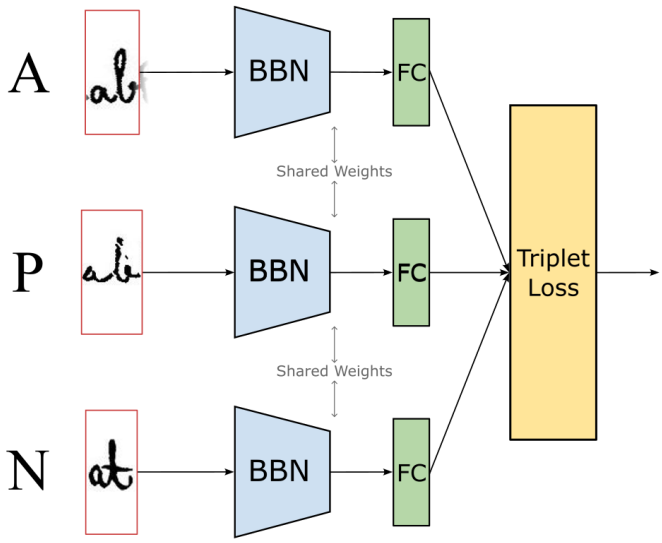
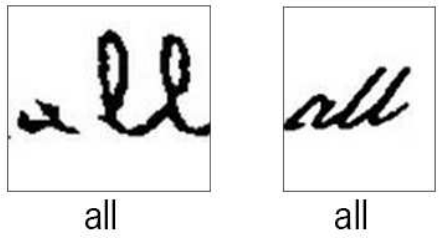


Figure 6.11: Three-branch Siamese Network used for the training phase.).

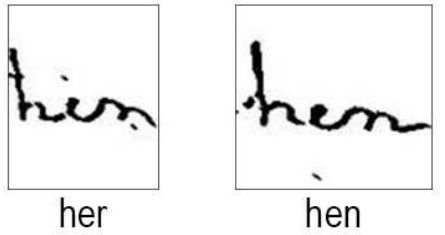
Since the three branches share the weights, it is possible to train a three-branch net and then use a two-branch net, as shown in section 5.4.1. The goal of the network is to calculate "similar" embeddings for images that belong to the same class. Once a network has been trained, it is also possible to use a single branch to calculate the embeddings of different images and then evaluate the

distance between the embeddings to assess the similarity between the images. A two-branch Siamese network simply restricts the assessment of similarity to two images at a time.

Choosing to train the network with a triplet loss, the choice of how the training triplets are constructed is very important. As mentioned earlier, the goal is to minimise the distance between images of the same class and maximise the distance between images of different classes. In the case of manuscript writing, the elements of a class are characterised by high variability. One can easily imagine that all N-grams belonging to the same class and written at different times, even by the same author, and perhaps originating from different words, will always differ from each other. Moreover, N-grams that differ by only one letter can look very similar, as shown in Figure 6.12. For this reason, it is important to define *hard-to-train* triplets that can imprint these characteristics as much as possible.



(a)



(b)

Figure 6.12: The image shows how two N-grams belonging to the same class could appear dissimilar (a), or two N-grams belonging to different classes could appear similar (b).

For these reasons, after randomly selecting an anchor A from a class, the image of the positive element P is chosen among the "most distant" elements of the same class. It may seem strange to speak of *distance* before we have trained the network to provide that distance. However, the network performs feature extraction over a pre-trained convolutional backbone. We can then use the embedding provided by the pre-trained backbone to evaluate the "*a priori*" distance between images of the same class of N-grams and then select an image for the positive element. To define the negative element N, the 10 classes closest to the anchor class are selected based on the Levenshtein distance calculated from the N-gram labels. The element N is then randomly selected from this set with a probability of 80%, otherwise, it is randomly selected from the entire dataset. To construct the training set of hard triplets, a maximum of 10 anchor elements are then selected for each class around which to construct the triplets. Any N-gram images that were not used to construct the triplets and are still present in the training set are used to define the validation set.

Once the training process for the Siamese network is complete, the Sliding-Window architecture can be evaluated on the 15 test pages of the Bentham Collection. For this purpose, all N-grams belonging to the reference dictionary created on the 5 training pages were searched in the test pages. The results are shown in Table 6.14. The table shows the indices of *precision*, *recall* and $recall_{InVoc}$ at k, where k is equal to 1 and 5. $k = 1$ practically means that for each part of the entered text line, the system only considers the N-gram recognised with the highest score in the output. If one brings $k = 5$, instead a maximum of 5 N-grams are considered for each text section. Three experiments were repeated with three different numbers of *shots*, i.e. 1, 2 or 3. The number of *shots* indicates how many N-gram samples for each class are selected as samples for the N-gram retrieval phase. If we select the number of shots equal to 3, three searches will be repeated for each class of N-grams with three different images of the N-gram randomly selected from the available images.

Table 6.14: Results for the N-gram retrieval with the Sliding-Window architecture

	#shot	p@1	r@1	r@1 InVoc	p@5	r@5	r@5 InVoc
Siding Window	1	0.0042	0.0045	0.0333	0.0108	0.0467	0.1956
	3	0.0067	0.0080	0.0178	0.0059	0.0404	0.1250
	5	**0.0068**	**0.0082**	**0.0300**	**0.0092**	**0.0815**	**0.2956**

Attention-Based Architecture Solution

The greater complexity of the attention-based architecture compared to the sliding-window architecture allows further decisions to be made to expand the datasets for the training phase of the model and to test different pre-trained network architectures for feature extraction.

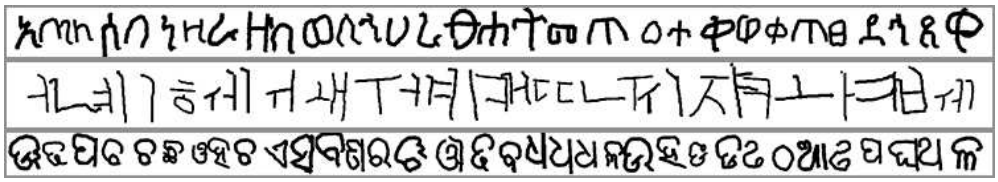
As a first experiment, we decided to select different backbones for feature extraction to test the basic architecture and evaluate its performance when the backbone varies. We chose two architecture backbones that performed well on different image processing tasks: VGG16 [116] and Resnet18 [46], and we also decided to test the PHOCNet [125] backbone as was done for the previous sliding window architecture. Table 6.15 shows the results for the single branch architecture described in section 5.4.2 with different backbones. The results are given for the cases $k = 1$ and $k = 5$ and for the number of hits of 1, 3 and 5. The different systems were trained with the set of n-grams extracted from the training data of the Bentham collection. From the table, we can see how the performance of the model changes depending on the backbone network used for the feature extraction phase. It is noticeable that the VGG16 network backbone achieves the best results in terms of all performance indices.

The performance of the model seems to be unsatisfactory in most cases. To achieve better performance, it is essential to perform a pre-training phase with synthetic data. We then extended the pre-training dataset by synthetically generating two datasets much larger than our actual data. It should be noted that

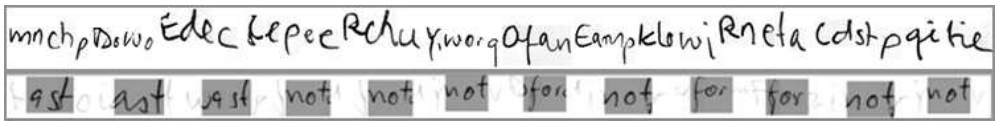
Table 6.15: Results for the N-gram retrieval with the Attention-Base architecture evaluating different backbones (BB) for feature extraction:

	#shot	p@1	r@1	r@1 InVoc	p@5	r@5	r@5 InVoc
BB_{VGG16}	1	**0.0701**	0.0373	**0.0942**	0.1747	0.0854	0.2114
	3	0.0579	**0.0339**	0.0638	0.1947	0.1006	0.2302
	5	0.0465	0.0274	0.0513	**0.1980**	**0.1062**	**0.2732**
$BB_{RESNET18}$	1	0.0000	0.0000	0.0000	0.0533	0.0061	0.0150
	3	0.0333	0.0042	0.0167	0.0333	0.0042	0.0167
	5	0.0167	0.0026	0.0059	0.0333	0.0061	0.0150
$BB_{PHOCNet}$	1	0.0000	0.0000	0.0000	0.0167	0.0048	0.0083
	3	0.0310	0.0071	0.0392	0.0671	0.0153	0.0566
	5	0.0111	0.0048	0.0083	0.0954	0.0233	0.0780

this process of expanding the training dataset does not impose any further burden on the user, as no further transcriptions or manual validations are required. To create the first synthetic dataset, we chose the Omniglot dataset [59]. Omniglot consists of 1623 different characters handwritten by different scribes from 50 different alphabets. There are 20 examples of each character in this dataset. We generated 2000 lines of text with 964 different symbols by randomly arranging the symbols in the array with a high probability of overlapping symbols. For the second set of synthetic training data, the lines of handwritten text were instead generated using a generative network capable of generating words from the handwritten text. The network was trained on the IAM dataset [70] and then used to generate random words (composed into 2000 lines of text). In this case, it was possible to use these lines to mark the N-grams they contained and use these N-grams as marked elements for the training set. Figure 6.13 shows some examples of lines that were synthetically generated using the two methods.



(a)



(b)

Figure 6.13: Synthetic generated handwritten data. (a) shows examples generated starting from the Omniglot dataset; the lines are a collection of handwritten symbols. (b) the handwritten lines generated are composed of N-grams; in the second image N-grams "not", "for", and "ast" are highlighted.

A second experiment was run to assess the contribution of synthetic datasets. Two backbones were used in this instance for feature extraction: the VGG16 backbone because it performed better than the Resnet18 backbone architecture, and the PHOCNet backbone because it was more appropriate for the application domain. The models were then pre-trained on the synthetic datasets, and only at the end of a fine-tuning phase were performed on the actual data from the Bentham Collection.

The results are summarized in the table 6.16, which also shows how the pre-training phase on synthetic data enables the system to significantly improve its performance in all relevant metrics. This effectively emphasizes the fact that the contribution of synthetic data brings significant advantages. It is important to note that using the Omniglot dataset's synthetic training data with the PHOCNet network was not viable since the PHOC representation of a word made up of symbols rather than letters has no meaning.

For the experiments conducted performed so far, we have always used the single-branch architecture. As discussed in the section 5.4.2, the proposed model allows the use of different branches, each using a different backbone for

Table 6.16: Results for the N-gram retrieval using the Attention-based architecture evaluating different backbones (BB) for feature extraction and training on synthetic data and fine-tune on real ones.

	#shot	p@1	r@1	r@1 InVoc	p@5	r@5	r@5 InVoc
BB_{VGG16} Omniglot	1	0.1823	0.1291	0.3376	0.3126	0.2171	0.6092
	3	0.2060	0.1405	0.4279	0.3462	0.2349	0.7170
	5	**0.2216**	**0.1470**	**0.4436**	**0.3725**	**0.2450**	**0.7594**
BB_{VGG16} Synth	1	0.1555	0.0778	0.2298	0.2940	0.1511	0.4908
	3	0.1823	0.1036	0.3357	0.3019	0.1644	0.5346
	5	0.1878	0.1117	0.3292	0.3336	0.1856	0.5832
$BB_{PHOCNet}$ Synth	1	0.1084	0.0612	0.1669	0.2682	0.1543	0.4156
	3	0.1369	0.0831	0.2105	0.2734	0.1638	0.4781
	5	0.1333	0.0791	0.1976	0.2861	0.1700	0.4809

feature extraction. In the third experiment conducted, we want to evaluate whether using the simultaneous use of different branches and performing the search in different feature spaces simultaneously can lead to an improvement in performance. We still choose to decide to use the VGG16 and PHOCNet backbones, respectively pre-trained with the dataset generated from the Omniglot dataset and with the synthetically generated dataset containing n-grams, in perfect continuity with the previous experiment. The outcome result of the model depends on the values assigned to the parameters w_1 and w_2 of equation 5.1, values that allow to weight the obtained solutions with respect to the search space. We evaluated the model by varying the two weights from the minimum value of 0 to the maximum value of 1 with a step of 0.1 and testing all combinations of weights by performing a grid search [5].

The search's outcomes are shown in Figure 6.14 for the case of a 5-shot scenario, and it is possible to note that merging the two separate branches

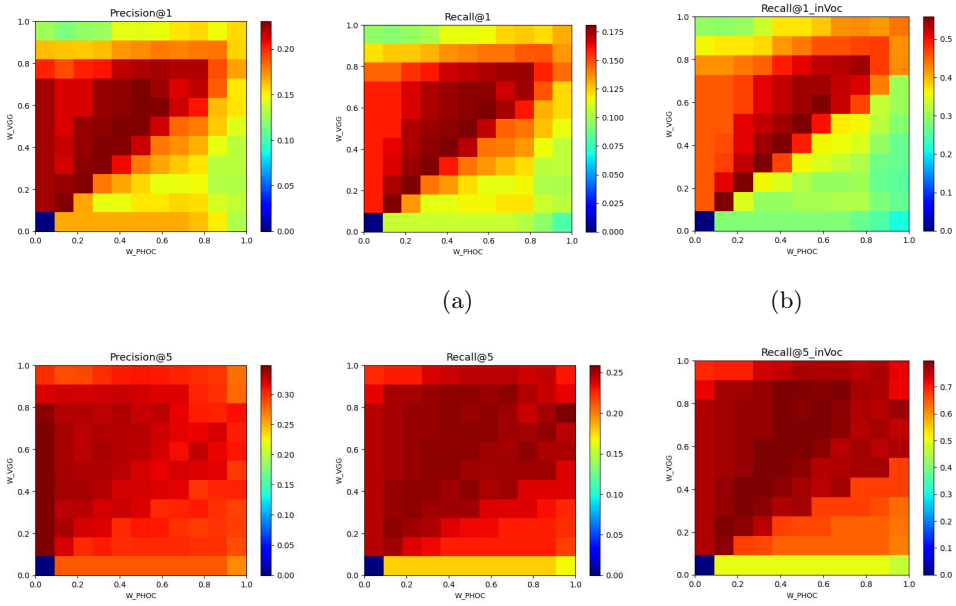


Figure 6.14: Visualizing feature weights for multi-space architecture. Variation of the (a)Precision@1 (b)Recall@1 (c)Recall@1_InVoc (d)Precision@5 (e)Recall@5 (f)Recall@5_InVoc according to the fusion weights for visual and PHOC features in case of 5-shot scenario.

yields the best outcomes. Except in the instance of r@5, the weight of the branch with the VGG16 backbone is consistently larger than or equal to the weight of the branch with the PHOCNet backbone. We may infer from the weight values, supporting our intuition, that the more discriminating feature spaces should be more significant for the purposes of the combination. The best result for each statistic in the case of 1, 3, and 5-shot scenarios are shown in the table 6.17. Additionally, compared to the prior performance with the single backbone model, the performance of all indices is better. This suggests that searching multiple feature spaces may contribute to better performance.

Table 6.17: Best result for the N-gram retrieval reached with the Attention-Based multi-space architecture.

	#shot	p@1	r@1	r@1 InVoc	p@5	r@5	r@5 InVoc
BB_{VGG16} + $BB_{PHOCNet}$	1	0.2140	0.1424	0.4505	0.3437	0.2392	0.7195
	3	0.2045	0.1495	0.4783	0.3536	0.2519	0.7498
	5	**0.2303**	**0.1808**	**0.5582**	**0.3747**	**0.2588**	**0.7975**

Comparison to Existing SOTA Methods

To the best of our knowledge, there are no articles that address the issue of finding N-grams inside a line of handwritten text, which is a challenge that has received little attention in the literature. However, Soubgui et al. [118] have encountered a very similar issue. They reflect the state of the art in identifying encrypted symbols inside a line of handwritten text. Despite the fact that the two issues' domains are different, they both offer solutions based on a few-shot setup and search for a symbol inside a line of handwritten text (whether it be an encrypted symbol or a Latin N-gram). The model developed by Souibgui et al. was pre-trained using a synthetic dataset derived from the Omniglot dataset and after we ran a fine-tuning phase of the model using real data from our training set from the Bantam Collection to ensure an accurate comparison with the proposed strategy.

The performance of the Sliding-Window system is lower than that of the other solutions, while the results of the model developed by Souibgui et al. [118] are comparable to those of our Attention-Based model, as shown in Table 6.18. It is noteworthy to notice that when k grows and when the number of shots is big, the Attention-Based proposed system performs better than [118]. This implies that the "*Fusion & Rescore*" module's contribution to rescoring overlapping solutions really enhances system performance.

Table 6.18: Comparison width SOTA method for the N-gram retrieval problem.

	#shot	p@1	r@1	r@1 InVoc	p@5	r@5	r@5 InVoc
Souibgui et al. [118]	1	0.1613	0.1121	0.3408	0.3007	0.2042	0.6465
	3	0.2088	0.1427	0.4391	0.3436	0.2301	0.6948
	5	0.2241	0.1602	0.4703	0.3342	0.2364	0.7314
Sliding Window Model	1	0.0042	0.0045	0.0333	0.0108	0.0467	0.1956
	3	0.0067	0.0080	0.0178	0.0059	0.0404	0.1250
	5	0.0068	0.0082	0.0300	0.0092	0.0815	0.2956
Attention Based Model	1	0.2140	0.1424	0.4505	0.3437	0.2392	0.7195
	3	0.2045	0.1495	0.4783	0.3536	0.2519	0.7498
	5	**0.2303**	**0.1808**	**0.5582**	**0.3747**	**0.2588**	**0.7975**

6.3.2 Discussion

In this section, we have presented an experimental validation for the N-gram retrieval methods proposed in chapter 5. Based on the experimental results, we have demonstrated the adequacy of our approach and shown that it is possible to recognise N-grams of references within a handwritten line of text. The proposed attention-based solution proved to be not only more powerful than the Sliding-Window proposal but also competitive with the state of the art. We limited the analysis to some of the available feature spaces, but the results show that using more than one representation can improve performance. In this direction, a comprehensive study of the different available representations could be conducted by analysing in detail different architectures of backbones with different depths to understand their impact on the model.

The methods do not limit the search for N-grams to the words that appear in the training pages from which the classical recognition systems build their reference lexicons but offer the possibility to recognise N-grams also in the

OOV words that appear only in the test pages. This encourages considering this solution as a possible way to solve the problem of OOV word spotting.

6.4 From N-gram Retrieval to Word Spotting

In this section, we present a use case of the whole-word spotting system to analyse the capabilities of the architecture in OOV word retrieval. Finally, we evaluate if and how the use of the proposed KWS system can be useful for the final user to build a transcription of small data collections, analyzing the time that the use of the system can save.

6.4.1 Can we Spot OOV words?

The main aim of the system proposed and described in chapter 5 is to overcome the distinction between the words belonging to the words present in the training pages and the OOV words of the whole collection, and to allow a search for any word. In this section, we try to verify whether the proposed architecture is able to overcome this distinction and actually recognise OOV words by evaluating its performance on the 20 pages of the Bentham Collection used in the previous sections.

The KWS system uses the QbS search paradigm, which allows us to search for a word in an image of a document by using a text string as input. This makes querying the system very easy, as we do not have to worry about retrieving sample images of the queried words. To find out if and how searching for OOV words affects the search, we decided to define two groups of query words. The first group consists of in-vocabulary words ($InVoc$), i.e. words that appear on the 5 training pages of the collection, and the second group consists of OOV words only:

- W_{InVoc} - made of 55 In-Vocabulary words;
- W_{OOV} - made of 85 Out-Of-Vocabulary words.

We then decided to evaluate the performance of the system for the two sets of query words by first using the set W_{InVoc} to define a performance baseline and then moving on to evaluate the words of the set W_{OOV}. In this way, the comparison between the results of the experiments can reveal any difficulties in recognising OOV words.

The KWS QbS system is based on a QbE N-gram retrieval system, which is the core element for search. In the previous section 6.3.1, we evaluated the two system proposals for solving the N-gram retrieval problem, the first one based on a sliding-window solution and the second one on an attention-based solution. The results show that the attention-based solution outperforms the sliding-window one by far. However, in this experiment, we chose the sliding-window solution as a precautionary measure to evaluate the performance of the KWS system. The sliding-window solution proves to be very easy to implement and particularly fast in training. However, in our opinion, it is interesting to evaluate the possibility of OOV spotting even if a non-optimal N-gram retrieval system is available. In other words, we somehow set a lower bound on the performance measure. If it is possible to detect OOV words even with the least penetrating core system, using a better system can only improve the overall performance.

When queried with a search string, the system returns a confidence-ordered list of images that are likely to contain instances of the searched word. Since the number of occurrences of the words in the collection is unknown, setting the maximum length of the k output list may result in underestimating performance in terms of both recall and precision: Given the value of k, this leads to an underestimation of Recall if there are more than k instances of the queried word in the collections, while it leads to an underestimation of Precision in the case of words with fewer than k instances. From an application's point of view, a user wants to retrieve instances of the searched word, preferably without having to read the entire text. If all occurrences of the searched word are contained in a reasonably large list k, which from the user's point of view

means that he or she can search in a short time, the user might be satisfied. In other words, an index that can satisfactorily measure the performance of the system may be the recall that varies with the length of the output list k. Thus, the system works satisfactorily if it can achieve high recall values while keeping the value of k low.

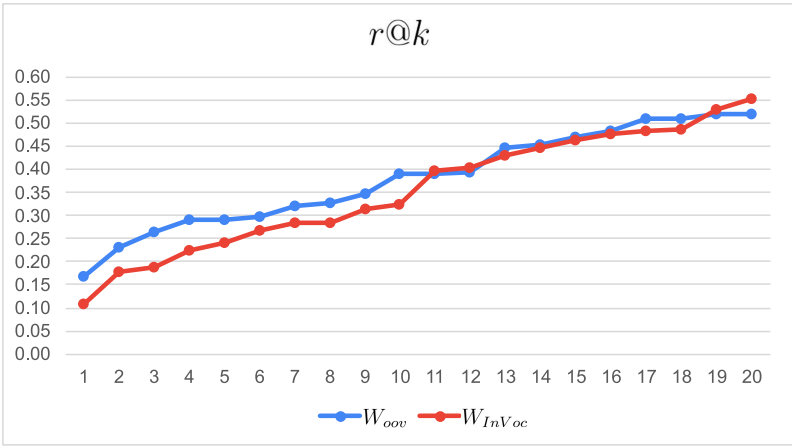


Figure 6.15: Recall at k reached by the system performing search on W_{InVoc} and on W_{OOV} sets.

Figure 6.15 shows the results in terms of recall for different values of k ($r@k$) for the two sets of query words. If we look at the trends of the two lines, we can immediately notice that both show a similar trend. The system finds words from both the W_{InVoc} set and the W_{OOV} set with a similar retrieval rate, regardless of the value of k. This is because the N-gram recognition phase takes place in the N-gram search space, and as long as all N-grams are available to search for the queried word, it makes no difference to the system whether it is an InVoc word or an OOV word. Although the sliding-window N-gram retrieval QbE system has shown unsatisfactory performance, its use within the overall KWQ QbE system still enables the retrieval of InVoc and OOV words and achieves retrieval rates around 50% with a still quite small output list length k.

However, this system has limitations in terms of suggestion accuracy. Figure

6.16 shows the evolution of accuracy for the sets W_{InVoc} and W_{OOV}. It can be seen that the accuracy of the system tends to be low regardless of the value of k. In fact, even with a rather low confidence value, the system tends to suggest solutions to fill the list of suggestions and, of course, the longer the list, the lower the accuracy.

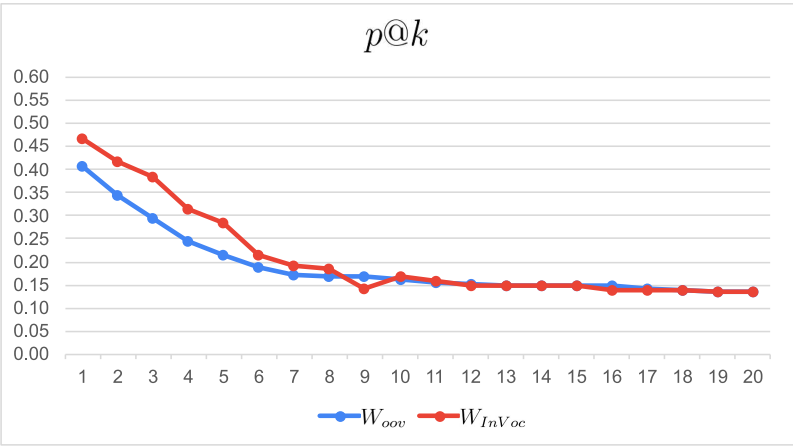


Figure 6.16: Recall at k reached by the system performing search on W_{InVOc} and on W_{OOV} sets.

6.4.2 Can the System Support a Transcription?

Once a word-spotting system is defined that can recognise both InVoc and OOV words, the only general question that remains to be answered is: "*Is it convenient to use such a system?*" To answer this question, we can imagine using our KWS system to obtain the transcription of the collection composed of 20 pages of the Bentham Collection and evaluate the time gain that can be obtained by using this system, estimated by the model presented in Chapter 3.

To this end, we need to make the necessary clarifications to be able to use this KWS system for the purpose of transcription. Remember that the KWS system is a query-by-strings system, i.e. it can search within the collection for words belonging to a list of query words. The system is in fact a lexicon-based system, but this lexicon may or may not consist of words belonging to the

training set. We then consider the first 5 pages of the collection as the training set, and we build the reference lexicon as the list of words included in the 5 training pages by adding to them the additional 100 most common words in the English language in the 1700s [68, 74].

We also need to specify the value of k to be used to indicate an appropriate number of transcript suggestions to propose to the user who has to correct and validate the transcription. As we saw in section 6.1, and as intuition suggests, a small number of options to be evaluated makes the use of the system productive. We, therefore, decide to use $k = 5$, also taking into account that the values for precision and recall for $k = 5$ have similar values and are therefore a good compromise, considering the graphs in Figure 6.15 and Figure 6.16:

- $r@k = 0.2668$
- $r@k = 0.2501$

The model used to calculate the time gain assumes that the time T_{TS} needed to transcribe the training pages and define the keyword list is given. We then considered the time it takes a user to transcribe the words and then the time it takes to build the N-gram dataset using the tools presented in the previous chapters.

At this point, we can estimate the gain obtained by using the KWS system to transcribe all 20 selected pages from the Bentham Collection. Table 6.19 shows the times estimated by the model and the achievable gain calculated for transcribing the entire collection DC, or the times that can be achieved for transcribing the 15 pages of the DS set (i.e. excluding the training pages).

When looking at the values in table 6.19 it is interesting to note that for the complete transcription of the 20 pages, it makes no difference whether the user uses the KWS system or transcribes the whole collection manually. However, considering that the gain is not negligible for only 15 test pages of the DS collection, we can conclude that the KWS system does indeed help in transcribing the pages without transcribing the DS and that the effort the user

Table 6.19: Comparison width time required to transcribe 20 pages of the Bentham Collection in a fully manual way or using the KWS system as a support tool.

Collection	Manual Time	KWS Time	Gain
15 test pages	4h 26m 15s	2h 47m 6s	37.24%
Entire collection	5h 35m 58s	5h 44m 45s	-2.61%

has to put in to create the keyword list is not negligible at all.

We have so far considered the case where the pages of the training set have to be transcribed from scratch, and we have found that the manual transcription of these pages has a strong impact on the time needed to transcribe the whole collection. We can now imagine using the system when the transcript of the training pages is already available. In this case, the time saved by the system is not negligible, as we can see from table 6.20.

Table 6.20: Comparison width time required to transcribe 20 pages of the Bentham Collection in a fully manual way or using the KWS system as a support tool in the case the transcription of the training set is already available.

Collection	Manual Time	KWS Time	Gain
Entire collection	5h 35m 58s	4h 0m 45s	28.34%

6.4.3 Discussion

In this section, we measured the potential of the KWS system by testing it on a small collection consisting of 20 pages from the Bentham Collection. The results show that the system can retrieve words from both the $InVoc$ word set and the OOV word set with similar recall and precision rates, regardless of the value of k. Searching for words in the N-gram search space effectively eliminates the difference between $InVoc$ words and OOV words, provided we have enough N-grams available to cover a large number of words. An important feature of the proposed system is that it is segmentation-free, i.e., it avoids

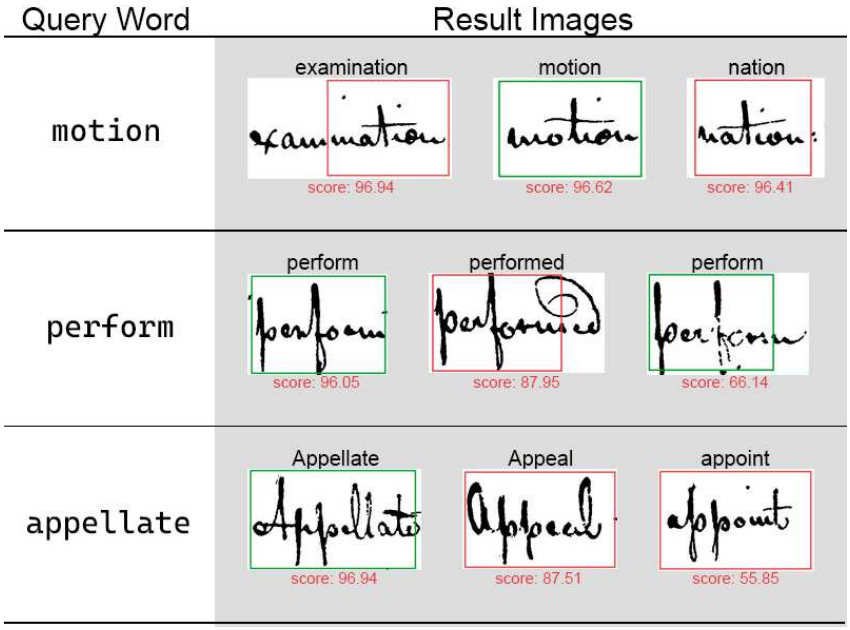


Figure 6.17: **Example of system results:** Examples of images retrieved by the system for a few query words. The correct outputs are highlighted with green boxes, while the incorrect ones are in red

both character-level and word-level segmentation, which is far from easy with cursive writing. The price for this is that the system also retrieves parts of words that are similar to the searched word. In other words: If the retrieved word is part of a longer word, the system can retrieve those parts, e.g., in the case of the retrieved word "perform", the system can retrieve part of the word "performed", as can be seen in Figure 6.17, with a fairly high confidence value, since the transcript of the selected excerpt is in fact the same or very similar to the retrieved word. As evidence of this, the figure shows the case of the query word "motion", and the system retrieves different words but which manuscript versions look very similar as happens for "nation". Similarly, there are also cases where not all the required N-grams of the queried word can be found, but most of them can, as in the case of the queried word "appellate" and the instance of the word "appeal" in the figure.

Experimental results have shown that the system can recognise *OOV* words

with similar performance to $InVoc$. They also show that the overall performance is very encouraging considering that the training set is small, with many classes and few samples per class, and that we chose the Sliding-Windows solution as the core system for the N-gram retrieval phase, which shows the lowest performance. In examining the time spent transcribing the entire collection, it is also interesting to note that we have reached a lower limit for the use of the system. Using a more powerful N-gram retrieval system or increasing the pages of the set DS to be transcribed could lead to a non-negligible gain in time, making the use of the system convenient for transcription from the user's point of view.

Chapter 7

Conclusions

In this thesis, we have addressed the problem of keyword spotting in images of documents of historical interest, focusing on the particular case of manuscript collections that are at most a few hundred pages long. The digitisation process of libraries, which is affecting more and more organisations today, is a process that involves not only information technologies, but more generally the structure of organisations, hardware resources, facilities, human resources and knowledge, as well as the overall organisational processes. The complexity of the process can be an obstacle to the digital transformation of smaller organisations, which, however, keep archives and document collections that often consist of a few dozen pages and have special and unique characteristics that are of particular interest to researchers and scholars for this reason. However, limitations in resources often mean data scarcity, and this limitation makes IT technology experts disdain, but their objective still remains to provide innovative artificial intelligence-based solutions that promise to simplify and speed up the management of document collections. For this reason, an AI system capable of adapting its operation to a limited amount of data may be critical to the use of this solution by scholars interested in studying collections.

One of the mantras of the AI community is the "80/20" rule. This means that once a dataset is created, it is often split into two subsets. The first

subset consists of 80% of the data used for training, while the remaining 20% is used as a test set for the trained methods. AI systems, especially deep learning solutions, are very data-hungry and learn well when an abundance of data is available to them. However, when the available datasets are limited and the process of labelling, and thus defining training sets, is very tedious, the idea of "80/20" might not be the best choice. In these cases, it might be useful to refer to another famous "80/20" rule, namely, the well-known Pareto principle [25,83], which states, in short, that "*most of the effects (80%) depend on a limited number of causes (20%)*". This view would lead to a total reversal: If a scholar has to hand-label 80% of the collection to train a method, he or she might decide that the effort required is too high and choose to abandon AI tools. It is therefore necessary to combine efforts to obtain methods and systems that allow users to minimize their effort while obtaining satisfactory results, which can lead to solutions that are convenient for the user.

We then talked about the "convenience" of using an AI system to support the processing of historical documents, in our case a KWS system to support the transcription of a collection. But what does "*convenience*" mean? What can be *convenient* for the end user? To answer these questions, in chapter 3 we proposed a model aimed at estimating the time savings that the use of a KWS system can bring to the transcription process of a manuscript collection. By focusing on time and providing a tangible measure of gain, we believe that it is possible to provide the user with a clear and simple feature that can define the goodness of the AI system and its ease of use. Focusing on the time that can be saved by using a system allows the user, even the less trained and experienced, to get an idea of the contribution the technology can make to their work. If the estimate of time saved is accurate, it also allows the user to understand well the potential of the method to support transcription and avoid false promises of AI solutions that can completely replace the presence of the experienced human user.

From the analysis of the time estimation model discussed in Chapter 6, it

is clear that the preparation of the training data and the relative keywords list has a large impact on the time the user has to spend on the collection. For this reason, it is crucial for the positive benefits of the systems to propose solutions that allow for speeding up the image word labelling processes. In this direction, in Chapter 4 we proposed solutions for training a keyword spotting system designed for small collections of handwritten documents. We have proposed techniques and methods to speed up the process of labelling handwritten words starting from images and transcripts of entire documents by proposing a technique of segmentation into text lines and transcript alignment. The presented approach is learning-free and represents an end-to-end solution that includes a line segmentation algorithm and an alignment algorithm. The former can extract text lines with a curved baseline, while the latter allows us to align the transcript to the image part of the text line that contains the handwritten word. Therefore, the proposed technique allows us to easily and quickly obtain a set of images of words correctly labelled with the correct transcription. The proposed graphical validation and correction tools are intended to be simple and intuitive, allowing the user to speed up the labelling process.

However, the proposed methods have limitations that we believe can be addressed through further development. The main limitation of the line segmentation method is that it is based on a subdivision into a fixed number of vertical stripes, which was set to $S = 8$ in the experimental phase. This value has been shown to be optimal for the datasets tested, but there is no evidence that it is the best choice in general. To overcome this inconvenience, a self-adaptive algorithm can determine the optimal value for each collection to which the method is applied. Starting from a value $S = 1$, the method can gradually increase the value until the best performance is achieved. On the other hand, the alignment method may not align all the words in the transcripts of a line. This behaviour can be easily caught and solutions can be implemented to deal with such anomalies. For example, a backtracking mechanism that takes into account that the number of bounding boxes to be aligned should be equal to the

number of words in the line's transcript can correct such errors and improve the overall performance of the system. It may be possible to make the consistency check itself more effective by estimating the average character width for each character class and then using that value to estimate the bounding box width to use in the consistency check.

The analysis of the transcription time estimated by the model has shown that, in addition to the time required to create the keyword list and label the data, another important factor is the presence of Out-Of-Vocabulary words that must be transcribed throughout the collection. The ability to handle OOV words can therefore be extremely important for the transcription process of small data collections, where the limited size of keyword lists increases the likelihood of encountering OOV words. To address this problem, in Chapter 5 we presented a KWS system that can adapt to small collections and recognize both $InVoc$ and OOV words. The basic idea is to build the keyword recognition system around an N-gram retrieval system, assuming that N-grams can represent good search primitives for words. Our hypothesis is based on the notion that N-grams are the result of the execution of a particular motor program. If a subject develops motor programs to control the movements necessary for writing, he is unlikely to automate whole words or single letters. On the other hand, the automation of character sequences implies that the N-grams are similar, even if they belong to different words. The experiments performed seem to support this hypothesis because the N-grams detected are not always sequences belonging to words present in the set of $InVoc$ words. Moreover, the recall and precision rates between whole $InVoc$ and OOV words show similar trends. These behaviours suggest that the hypothesis is confirmed.

Analysis of transcription times with the proposed KWS system has shown that even when a non-optimal KWS system is used, transcription time can be reduced by about 30% compared to fully manual transcription. When the collections to be transcribed are very small, as in our experiment that included only 20 manuscript pages, we have seen that the ability to recognize OOV

words, but especially the ability to quickly generate keyword lists, are fundamental aspects for the use of a profitable transcription support system. In fact, we have seen how the system allows a significant time saving during its use, i.e., in the transcription of the 15 pages of the collection that are not used for the creation of the keyword list and the knowledge base for the KWS system, which in our case consists of the set of reference N-gram examples. The analyzed system allowed to save almost 40% of the time required for the transcription of the 15 pages and still provide a complete and correct transcription of the collection. It is therefore important to point out that if techniques are not used to optimize and speed up the preparation process of the KWS system (i.e. the definition of the keyword list from the training pages), the effort required at this stage may cancel out the time saved by using the KWS during its use. In our experiment, labelling only 5 of the 20 pages of the collection using classical and manual techniques was sufficient to negate the positive effects of the system for assisted transcription. However, it is useful to underline how speeding up the labelling process of the same 5 pages with the methods proposed in chapter 4, based on the automatic alignment of the transcription, allowed the whole system to have a positive outcome and to record a time saving of almost 30%.

The experiments then demonstrated and confirmed what the model used to estimate the time gain had suggested, namely that the performance of the KWS system alone is not sufficient to measure the contribution that the system can make to the transcription process. Operations such as the labelling and knowledge base building phase, or the ability to handle OOV words, are therefore essential features for evaluating the performance of the system on a par with the performance indexes of the KWS system.

Finally, it is interesting to analyze the behaviour of N-gram retrieval systems in a bit more detail, as it represents the core element of our KWS proposal. To address the few data scenario, we proposed a Sliding-Window solution, which is characterized by a very simple architecture and therefore does not require an extremely large amount of data for training, as opposed to a much more

complex Attention-Based architecture, which does require more data. We then decided to use synthetic data generation techniques for building a training set large enough to allow the system to significantly improve its performance. It should be emphasized that the data generation techniques allowed us to expand the data set for the Attention-Based model rather than for the Sliding-Window model. Although we focused on state-of-the-art handwriting image generation techniques, these techniques are not able to properly represent the variability of a subject's handwriting. This behaviour can be observed in the examples shown in Figure 6.13, where all N-grams belonging to the same class appear extremely similar to each other. This behaviour did not allow us to constructively increase the size of the training set for the Sliding-Window solution, as the application would have been limited to the addition of a single N-gram image per class, rather than the generation of different handwriting inks associated with the same N-gram, as it would happen when different subjects write the same N-grams. These observations lead to an interesting consideration: To constrain the hand-made process of labelling images of handwritten words, we need automatic image generation systems capable of modelling handwriting variability typical of individual human subjects. In other words, to have a performing system capable of reading human handwriting, we would need to have a system capable of writing like a human.

Appendix A

This appendix contains a list of repositories with tools, code, datasets, and additional material used during the experimental phases.

List of Repositories

- **A* Line Segmenter**
  `https://github.com/Natural-Computation-Lab/AStarSegmenter`
  The repository contains the code for the line segmentation algorithm presented in chapter 4

- **Transcripts Alignment - MiM Algorithm**
  `https://github.com/Natural-Computation-Lab/MiMtranscriptAligner`
  The repository contains the code for the transcript alignment algorithm presented in chapter 4. The repository also contains the user interface tool for the data validation and correction of the alignment showed in 6.8

- **The Moccia Code**
  `https://github.com/Natural-Computation-Lab/MocciaCode`
  The repository contains examples of images of the *Moccia-Code*. In the repository, it is possible to find entire pages or single-line images, and all the ground truth files consisting of the correct transcription.

- **Validation N-Grams Tool**
  `https://github.com/ZeePPe/Ngram-Validator-Tool`
  The repository contains the user interface tool used to validate and extract N-gram images shown in figure 6.9.

- **Keyword Spotting based on N-gram retrieva**
  `https://github.com/ZeePPe/PHOCsiameseNet`
  The repository contains the code for the N-gram retrieval system based on the Sliding-Window architecture described in section 5.4.1.
  The repository also contains the code for the KWS system to spot OOV words described in Chapter 5

Bibliography

[1] R. Ahmed, W. G. Al-Khatib, and S. Mahmoud. A survey on handwritten documents word spotting. *International Journal of Multimedia Information Retrieval*, 6(1):31–47, 2017.

[2] R. Alaasam, B. Kurar, and J. El-Sana. Layout analysis on challenging historical arabic manuscripts using siamese network. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 738–742. IEEE, 2019.

[3] M. Alberti, L. Vögtlin, V. Pondenkandath, M. Seuret, R. Ingold, and M. Liwicki. Labeling, cutting, grouping: an efficient text line segmentation method for medieval manuscripts. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1200–1206. IEEE, 2019.

[4] J. Almazan, A. Gordo, A. Fornés, and E. Valveny. Handwritten word spotting with corrected attributes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1017–1024, 2013.

[5] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 2012.

[6] J. Bloom. *Paper before print: the history and impact of paper in the Islamic world*. Yale University Press, 2001.

[7] T. Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. *Advances in neural information processing systems*, 29, 2016.

[8] T. Bluche and R. Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 646–651. IEEE, 2017.

[9] A. Brakensiek, J. Rottland, and G. Rigoll. Handwritten address recognition with open vocabulary using character n-grams. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 357–362. IEEE, 2002.

[10] S. S. Bukhari, A. Kadi, M. A. Jouneh, F. M. Mir, and A. Dengel. anyocr: An open-source ocr system for historical archives. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 305–310. IEEE, 2017.

[11] G. Capriolo. *Paternas literas confirmamus: Il libro dei privilegi e delle facoltà del mastro portolano di Terra di Lavoro (secc. XV-XVII)*, volume 2. FedOA-Federico II University Press, 2017.

[12] K. Chen, M. Seuret, J. Hennebert, and R. Ingold. Convolutional neural networks for page segmentation of historical document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 965–970. IEEE, 2017.

[13] V. Christlein, A. Nicolaou, M. Seuret, D. Stutzmann, and A. Maier. Icdar 2019 competition on image retrieval for historical handwritten documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1505–1509. IEEE, 2019.

[14] C. Clausner, A. Antonacopoulos, and S. Pletschacher. Icdar2019 competition on recognition of documents with complex layouts-rdcl2019. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1521–1526. IEEE, 2019.

[15] D. Coquenet, C. Chatelain, and T. Paquet. End-to-end handwritten paragraph text recognition using a vertical attention network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[16] L. P. Cordella, C. De Stefano, A. Marcelli, and A. Santoro. Writing order recovery from off-line handwriting by graph traversal. In *2010 20th International Conference on Pattern Recognition*, pages 1896–1899. IEEE, 2010.

[17] G. De Gregorio, I. Citro, and A. Marcelli. Transcript alignment for historical handwritten documents: The mim algorithm. In *20th International Graphonomics Society Conference, 7-9 June*, page in press, 2022.

[18] C. De Stefano, M. Garruto, L. Lapresa, and A. Marcelli. Detecting handwriting primitives in cursive words by stroke sequence matching. *Advances*

in Graphonomics, A. Marcelli and C. De Stefano (eds.), Arezzo: Zona Editrice, pages 281–285, 2005.

[19] C. De Stefano, G. Guadagno, and A. Marcelli. A saliency-based segmentation method for online cursive handwriting. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(07):1139–1156, 2004.

[20] C. De Stefano, A. Marcelli, A. Parziale, and R. Senatore. Reading cursive handwriting. In *2010 12th International Conference on Frontiers in Handwriting Recognition*, pages 95–100. IEEE, 2010.

[21] S. Dey, A. Dutta, J. I. Toledo, S. K. Ghosh, J. Lladós, and U. Pal. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*, 2017.

[22] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos. cbad: Icdar2017 competition on baseline detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1355–1360. IEEE, 2017.

[23] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos. cbad: Icdar2017 competition on baseline detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1355–1360. IEEE, 2017.

[24] M. Diem, F. Kleber, R. Sablatnig, and B. Gatos. cbad: Icdar2019 competition on baseline detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1494–1498, 2019.

[25] R. Dunford, Q. Su, and E. Tamang. The pareto principle. *The Plymouth Student Scientist*, 2014.

[26] M. Eldridge, I. Nimmo-Smith, A. Wing, and R. Totty. The variability of selected features in cursive handwriting: Categorical measures. *Journal of the Forensic Science Society*, 24(3):179–219, 1984.

[27] M. Eldridge, I. Nimmo-Smith, A. Wing, and R. Totty. The dependence between selected categorical measures of cursive handwriting. *Journal of the Forensic Science Society*, 25(3):217–231, 1985.

[28] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *CVPR*, 2020.

[29] M. Fink, T. Layer, G. Mackenbrock, and M. Sprinzl. Baseline detection in historical documents using convolutional u-nets. In *2018 13th IAPR*

International Workshop on Document Analysis Systems (DAS), pages 37–42. IEEE, 2018.

[30] D. Firmani, M. Maiorino, P. Merialdo, and E. Nieddu. 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*, pages 263–272, 2018.

[31] A. Fischer, V. Frinken, A. Fornés, and H. Bunke. Transcription alignment of latin manuscripts using hidden markov models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, pages 29–36, 2011.

[32] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Hmm-based word spotting in handwritten documents using subword models. In *2010 20th International Conference on Pattern Recognition*, pages 3416–3419. IEEE, 2010.

[33] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern recognition letters*, 33(7):934–942, 2012.

[34] A. Fischer, M. Liwicki, and R. J. Ingold. *Handwritten Historical Document Analysis, Recognition, And Retrieval-State Of The Art And Future Trends*. World Scientific, 2020.

[35] A. Fischer, M. Wuthrich, M. Liwicki, V. Frinken, H. Bunke, G. Viehhauser, and M. Stolz. Automatic transcription of handwritten medieval documents. In *2009 15th International Conference on Virtual Systems and Multimedia*, pages 137–142. IEEE, 2009.

[36] 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, 2011.

[37] L. Gao, X. Yi, Z. Jiang, L. Hao, and Z. Tang. Icdar2017 competition on page object detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1417–1422. IEEE, 2017.

[38] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou. A survey of document image word spotting techniques. *Pattern recognition*, 68:310–332, 2017.

[39] E. Granell, E. Chammas, L. Likforman-Sulem, C.-D. Martínez-Hinarejos, C. Mokbel, and B.-I. Cîrstea. Transcription of spanish historical handwritten documents with deep neural networks. *Journal of Imaging*, 4(1):15, 2018.

[40] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[41] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2008.

[42] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems*, 21, 2008.

[43] I. Gruber, M. Hrúz, P. Ircing, P. Neduchal, T. Zítka, M. Hlaváč, Z. Zajíc, J. Švec, and M. Bulín. Ocr improvements for images of multi-page historical documents. In *International Conference on Speech and Computer*, pages 226–237. Springer, 2021.

[44] T. Grüning, G. Leifert, T. Strauß, J. Michael, and R. Labahn. A two-stage method for text line detection in historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(3):285–302, 2019.

[45] A. Hamid, M. Bibi, M. Moetesum, and I. Siddiqi. Deep learning based approach for historical manuscript dating. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 967–972. IEEE, 2019.

[46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[47] O. Hiroharu. *Inkyunabura no Sekai (The World of Incunabula)*. Japan Library Association, 2000.

[48] E. Indermühle, M. Liwicki, and H. Bunke. Combining alignment results for historical handwritten document analysis. In *2009 10th International Conference on Document Analysis and Recognition*, pages 1186–1190. IEEE, 2009.

[49] M. Jenckel, S. S. Bukhari, and A. Dengel. anyocr: A sequence learning based ocr system for unlabeled historical documents. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 4035–4040. IEEE, 2016.

[50] P. Kahle, S. Colutto, G. Hackl, and G. Mühlberger. Transkribus-a service platform for transcription, recognition and retrieval of historical documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 4, pages 19–24. IEEE, 2017.

[51] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, S. Mack, et al. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

[52] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas. Pay attention to what you read: non-recurrent handwritten text-line recognition. *Pattern Recognition*, 129:108766, 2022.

[53] M. W. A. Kesiman, J.-C. Burie, G. N. M. A. Wibawantara, I. M. G. Sunarya, and J.-M. Ogier. Amadi_lontarset: The first handwritten balinese palm leaf manuscripts dataset. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 168–173. IEEE, 2016.

[54] B. Kiessling, R. Tissot, P. Stokes, and D. S. B. Ezra. escriptorium: an open source platform for historical document analysis. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 19–19. IEEE, 2019.

[55] G. Koch, R. Zemel, R. Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015.

[56] A. Kornai. An experimental hmm-based postal ocr system. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 3177–3180. IEEE, 1997.

[57] E. Kornfield, R. Manmatha, and J. Allan. Text alignment with handwritten documents. In *First International Workshop on Document Image Analysis for Libraries, 2004. Proceedings.*, pages 195–209, 2004.

[58] M. Kozielski, M. Matysiak, P. Doetsch, R. Schlöter, and H. Ney. Open-lexicon language modeling combining word and character levels. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 343–348. IEEE, 2014.

[59] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

[60] L. L. Lee, M. G. Lizarraga, N. R. Gomes, and A. L. Koerich. A prototype for brazilian bankcheck recognition. In *Automatic Bankcheck Processing*, pages 87–107. World Scientific, 1997.

[61] Y. Leydier, V. Églin, S. Brès, and D. Stutzmann. Learning-free text-image alignment for medieval manuscripts. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 363–368, 2014.

[62] X. Li, B. Zhang, J. Liao, and P. V. Sander. Document rectification and illumination correction using a patch-based cnn. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019.

[63] F. Lombardi and S. Marinai. Deep learning for historical document analysis and recognition—a survey. *Journal of Imaging*, 6(10):110, 2020.

[64] T. Lu and A. Dooms. Bayesian damage recognition in document images based on a joint global and local homogeneity model. *Pattern Recognition*, 118:108034, 2021.

[65] N. T. Ly, C. T. Nguyen, and M. Nakagawa. An attention-based end-to-end model for multiple text lines recognition in japanese historical documents. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–634. IEEE, 2019.

[66] N. T. Ly, C. T. Nguyen, and M. Nakagawa. Attention augmented convolutional recurrent network for handwritten japanese text recognition. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 163–168. IEEE, 2020.

[67] N. T. Ly, C. T. Nguyen, and M. Nakagawa. An attention-based row-column encoder-decoder model for text recognition in japanese historical documents. *Pattern Recognition Letters*, 136:134–141, 2020.

[68] J. Lynch. Guide to eighteenth century english vocabulary. *URL http://andromeda. rutgers. edu/˜ jlynch/C18Guide. pdf (2017-06-12)*, 2006.

[69] A. Marcelli, A. Parziale, and R. Senatore. Some observations on handwriting from a motor learning perspective. In *AFHA*, volume 1022, pages 6–10, 2013.

[70] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *IJDAR*, 2002.

[71] J. Martínek, L. Lenc, and P. Král. Building an efficient ocr system for historical documents with little training data. *Neural Computing and Applications*, 32(23):17209–17227, 2020.

[72] H. Mohammed, I. Marthot-Santaniello, and V. Märgner. Grk-papyri: a dataset of greek handwriting on papyri for the task of writer identification. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 726–731. IEEE, 2019.

[73] T. Monnier and M. Aubry. docextractor: An off-the-shelf historical document element extraction. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 91–96. IEEE, 2020.

[74] I. Nation. How many high frequency words are there in english. *Language, Learning and Literature: Studies presented to Hakan Ringbom*, 4:167–181, 2001.

[75] H. Neji, J. Nogueras-Iso, J. Lacasta, M. B. Halima, and A. M. Alimi. Adversarial autoencoders for denoising digitized historical documents: the use case of incunabula. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 6, pages 31–34. IEEE, 2019.

[76] K. C. Nguyen, C. T. Nguyen, S. Hotta, and M. Nakagawa. A character attention generative adversarial network for degraded historical document restoration. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 420–425. IEEE, 2019.

[77] W. Niblack. *An introduction to digital image processing.* Strandberg Publishing Company, 1985.

[78] B. Nunamaker, S. S. Bukhari, D. Borth, and A. Dengel. A tesseract-based ocr framework for historical documents lacking ground-truth text. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3269–3273. IEEE, 2016.

[79] S. A. Oliveira, B. Seguin, and F. Kaplan. dhsegment: A generic deep-learning approach for document segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12. IEEE, 2018.

[80] P. Orsini. *Studies on Greek and Coptic Majuscule Scripts and Books*. De Gruyter, 2018.

[81] N. Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[82] C. Papadopoulos, S. Pletschacher, C. Clausner, and A. Antonacopoulos. The impact dataset of historical document images. In *Proceedings of the 2Nd international workshop on historical document imaging and processing*, pages 123–130, 2013.

[83] V. Pareto. Cours conomie politique profess a l'universit de lausanne, vol. i, 1896. *Vol. II*, 1897.

[84] A. Parziale. A neurocomputational model of reaching movements. *PhD Thesis*, 2016.

[85] J. Pastor-Pellicer, M. Z. Afzal, M. Liwicki, and M. J. Castro-Bleda. Complete system for text line extraction using convolutional neural networks and watershed transform. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 30–35. IEEE, 2016.

[86] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th international conference on frontiers in handwriting recognition*, pages 285–290. IEEE, 2014.

[87] T. Plötz and G. A. Fink. Markov models for offline handwriting recognition: a survey. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(4):269–298, 2009.

[88] V. Pondenkandath, M. Alberti, M. Diatta, R. Ingold, and M. Liwicki. Historical document synthesis with generative adversarial networks. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 5, pages 146–151. IEEE, 2019.

[89] B. B. Powell. *Writing: theory and history of the technology of civilization*. John Wiley & Sons, 2012.

[90] A. Prusty, S. Aitha, A. Trivedi, and R. K. Sarvadevabhatla. Indiscapes: Instance segmentation networks for layout parsing of historical indic manuscripts. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 999–1006. IEEE, 2019.

[91] D. Pérez, L. Tarazón, N. Serrano, F. Castro, O. R. Terrades, and A. Juan. The germana database. In *2009 10th International Conference on Document Analysis and Recognition*, pages 301–305, 2009.

[92] V. K. B. Ramanna, S. S. Bukhari, and A. Dengel. Document image dewarping using deep learning. In *ICPRAM*, pages 524–531, 2019.

[93] H. Rasyidi and S. Khan. Historical document image binarization via style augmentation and atrous convolutions. *Neural Computing and Applications*, 33(13):7339–7352, 2021.

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

[95] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[96] G. Renton, Y. Soullard, C. Chatelain, S. Adam, C. Kermorvant, and T. Paquet. Fully convolutional network with dilated convolutions for handwritten text line segmentation. *International Journal on Document Analysis and Recognition (IJDAR)*, 21(3):177–186, 2018.

[97] M. Rijntjes, C. Dettmers, C. Büchel, S. Kiebel, R. S. Frackowiak, and C. Weiller. A blueprint for movement: functional and anatomical representations in the human motor system. *Journal of Neuroscience*, 19(18):8043–8048, 1999.

[98] A. Robinson. *The Story of Writing*. Thames & Hudson, 2007.

[99] A. Robinson. *Writing and Script: A Very Short Introduction*. OUP Oxford, 2009.

[100] V. Romero, A. Fornés, N. Serrano, J. A. Sánchez, A. H. Toselli, V. Frinken, E. Vidal, and J. Lladós. The esposalles database: An ancient marriage license corpus for off-line handwriting recognition. *Pattern Recognition*, 46(6):1658–1669, 2013.

[101] V. Romero-Gómez, A. H. Toselli, V. Bosch, J. A. Sánchez, and E. Vidal. Automatic alignment of handwritten images and transcripts for training handwritten text recognition systems. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 328–333. IEEE, 2018.

[102] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[103] J. Rothfeder, R. Manmatha, and T. M. Rath. Aligning transcripts to automatically segmented handwritten manuscripts. In H. Bunke and A. L. Spitz, editors, *Document Analysis Systems VII*, pages 84–95, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[104] J. A. Rydberg-Cox. Digitizing latin incunabula: Challenges, methods, and possibilities. *Digital Humanities Quarterly*, 3(1), 2009.

[105] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[106] 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, 2013.

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

[108] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[109] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[110] R. Senatore. The role of basal ganglia and cerebellum in motor learning. a computational model. *PhD Thesis*, 2012.

[111] R. Senatore and A. Marcelli. Where are the characters? characters segmentation in annotated cursive handwriting. In *Proc. 16th IGS Conference*, pages 171–174, 2013.

[112] F. Shafait. Document image analysis with ocropus. In *2009 IEEE 13th International Multitopic Conference*, pages 1–6. IEEE, 2009.

[113] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[114] F. Simistira, M. Bouillon, M. Seuret, M. Würsch, M. Alberti, R. Ingold, and M. Liwicki. Icdar2017 competition on layout analysis for challenging medieval manuscripts. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1361–1370. IEEE, 2017.

[115] F. Simistira, M. Seuret, N. Eichenberger, A. Garz, M. Liwicki, and R. Ingold. Diva-hisdb: A precisely annotated large dataset of challenging medieval manuscripts. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 471–476. IEEE, 2016.

[116] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[117] R. Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[118] M. A. Souibgui, A. Fornés, Y. Kessentini, and C. Tudor. A few-shot learning approach for historical ciphered manuscript recognition. In *ICPR*, 2021.

[119] W. A. Sparrow and K. M. Newell. Metabolic energy expenditure and the regulation of movement economy. *Psychonomic Bulletin & Review*, 5(2):173–196, 1998.

[120] U. Springmann and A. Lüdeling. Ocr of historical printings with an application to building diachronic corpora: A case study using the ridges herbal corpus. *arXiv preprint arXiv:1608.02153*, 2016.

[121] U. Springmann, D. Najock, H. Morgenroth, H. Schmid, A. Gotscharek, and F. Fink. Ocr of historical printings of latin texts: problems, prospects, progress. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pages 71–75, 2014.

[122] N. Stamatopoulos, B. Gatos, and G. Louloudis. A novel transcript mapping technique for handwritten document images. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 41–46, 2014.

[123] N. Stamatopoulos, G. Louloudis, and B. Gatos. Efficient transcript mapping to ease the creation of document image segmentation ground truth with text-image alignment. In *2010 12th International Conference on Frontiers in Handwriting Recognition*, pages 226–231. IEEE, 2010.

[124] D. Stromer, V. Christlein, A. Maier, P. Zippert, E. Helmecke, T. Hausotte, and X. Huang. Non-destructive digitization of soiled historical chinese bamboo scrolls. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 55–60. IEEE, 2018.

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

[126] S. Sudholt and G. A. Fink. Attribute cnns for word spotting in handwritten documents. *International journal on document analysis and recognition (ijdar)*, 21(3):199–218, 2018.

[127] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing*, 289:119–128, 2018.

[128] O. Surinta, M. Holtkamp, F. Karabaa, J.-P. Van Oosten, L. Schomaker, and M. Wiering. A path planning for line segmentation of handwritten documents. In *2014 14th international conference on frontiers in handwriting recognition*, pages 175–180. IEEE, 2014.

[129] B. A. Swett, J. L. Contreras-Vidal, R. Birn, and A. Braun. Neural substrates of graphomotor sequence learning: A combined fmri and kinematic study. *Journal of neurophysiology*, 103(6):3366–3377, 2010.

[130] C. Tensmeyer and T. Martinez. Document image binarization with fully convolutional neural networks. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 99–104. IEEE, 2017.

[131] C. I. Tomai, B. Zhang, and V. Govindaraju. Transcript mapping for historic handwritten document images. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 413–418. IEEE, 2002.

[132] P. Torras, M. A. Souibgui, J. Chen, and A. Fornés. A transcription is all you need: Learning to align through attention. In *International Conference on Document Analysis and Recognition*, pages 141–146. Springer, 2021.

[133] A. H. Toselli, V. Romero, and E. Vidal. Viterbi based alignment between text images and their transcripts. In *Proceedings of the Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2007).*, pages 9–16, 2007.

[134] A. Trivedi and R. K. Sarvadevabhatla. Hindola: a unified cloud-based platform for annotation, visualization and machine learning-based layout analysis of historical manuscripts. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 31–35. IEEE, 2019.

[135] L. Uzan, N. Dershowitz, and L. Wolf. Qumran letter restoration by rotation and reflection modified pixelcnn. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 23–29. IEEE, 2017.

[136] D. Valy, M. Verleysen, S. Chhun, and J.-C. Burie. A new khmer palm leaf manuscript dataset for document analysis and recognition: Sleukrith set. In *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing*, pages 1–6, 2017.

[137] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[138] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee. Binarization of degraded document images based on hierarchical deep supervised network. *Pattern Recognition*, 74:568–586, 2018.

[139] M. Wagdy, K. Amin, and M. Ibrahim. Dewarping document image techniques: Survey and comparative study. *International Journal of Image and Graphics*, 21(03):2150031, 2021.

[140] D. J. Waters. What are digital libraries. *CLIR issues*, 4(1):5–6, 1998.

[141] A. Wing and I. Nimmo-Smith. The variability of cursive handwriting measure defined along a continuum: letter specificity. *Journal of the Forensic Science Society*, 27(5):297–306, 1987.

[142] M. Würsch, R. Ingold, and M. Liwicki. Divaservices—a restful web service for document image analysis methods. *Digital Scholarship in the Humanities*, 32(suppl_1):i150–i156, 2017.

[143] G.-W. Xie, F. Yin, X.-Y. Zhang, and C.-L. Liu. Dewarping document image by displacement flow estimation with fully convolutional network. In *International Workshop on Document Analysis Systems*, pages 131–144. Springer, 2020.

[144] Y. Xu, W. He, F. Yin, and C.-L. Liu. Page segmentation for historical handwritten documents using fully convolutional networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 541–546. IEEE, 2017.

[145] X. Yin, N. Aldarrab, B. Megyesi, and K. Knight. Decipherment of historical manuscript images. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 78–85. IEEE, 2019.

[146] Y. Zhang, S. Nie, W. Liu, X. Xu, D. Zhang, and H. T. Shen. Sequence-to-sequence domain adaptation network for robust text image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2740–2749, 2019.

[147] S. Zinger, J. Nerbonne, and L. Schomaker. Text-image alignment for historical handwritten documents. In *Document recognition and retrieval XVI*, volume 7247, pages 14–21. SPIE, 2009.

[148] Z. Ziran, X. Pic, S. U. Innocenti, D. Mugnai, and S. Marinai. Text alignment in early printed books combining deep learning and dynamic programming. *Pattern Recognition Letters*, 133:109–115, 2020.

This page intentionally left blank.