A Probabilistic Formulation of Keyword Spotting

PhD Thesis

Joan Puigcerver

Supervised by Dr. Enrique Vidal
and Dr. Alejandro H. Toselli

November 19, 2018

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Work partially supported by the Spanish MEC under FPU grant FPU13/06281.

Abstract

Keyword Spotting, applied to handwritten text documents, aims to retrieve the documents, or parts of them, that are relevant for a query, given by the user, within a large collection of documents. The topic has gained a large interest in the last 20 years among Pattern Recognition researchers, as well as digital libraries and archives.

This thesis, first defines the goal of Keyword Spotting from a Decision Theory perspective. Then, the problem is tackled following a probabilistic formulation. More precisely, Keyword Spotting is presented as a particular instance of Information Retrieval, where the content of the documents is unknown, but can be modeled by a probability distribution. In addition, the thesis also proves that, under the correct probability distributions, the framework provides the optimal solution, under many of the evaluation measures traditionally used in the field.

Later, different statistical models are used to represent the probability distribution over the content of the documents. These models, Hidden Markov Models or Recurrent Neural Networks, are estimated from training data, and the corresponding distributions over the transcripts of the images can be efficiently represented using Weighted Finite State Transducers.

In order to make the framework practical for large collections of documents, this thesis presents several algorithms to build probabilistic word indexes, using both lexicon-based and lexicon-free models. These indexes are very similar to the ones used by traditional search engines.

Furthermore, we study the relationship between the presented formulation and other seminal approaches in the field of Keyword Spotting, highlighting some limitations of the latter.

Finally, all the contributions are evaluated experimentally, not only on standard academic benchmarks, but also on collections including tens of thousands of pages of historical manuscripts. The results show that the proposed framework and algorithms allow to build very accurate and very fast Keyword Spotting systems, with a solid underlying theory.

Resum

La detecció de paraules clau (*Keyword Spotting*, en anglès), aplicada a documents de text manuscrit, té com a objectiu recuperar els documents, o parts d'ells, que siguen rellevants per a una certa consulta (*query*, en anglès), indicada per l'usuari, dintre d'una gran col·lecció de documents. La temàtica ha recollit un gran interés en els últims 20 anys entre investigadors en Reconeixement de Formes (*Pattern Recognition*), així com biblioteques i arxius digitals.

Aquesta tesi defineix l'objectiu de la detecció de paraules claus a partir d'una perspectiva basada en la Teoria de la Decisió i una formulació probabilística adequada. Més concretament, la detecció de paraules clau es presenta com un cas concret de Recuperació de la Informació (*Information Retrieval*), on el contingut dels documents és desconegut, però pot ser modelat mitjançant una distribució de probabilitat. A més, la tesi també demostra que, sota les distribucions de probabilitat correctes, el marc de treball desenvolupat condueix a la solució òptima del problema, segons diverses mesures d'avaluació utilitzades tradicionalment en el camp.

Després, diferents models estadístics s'utilitzen per representar les distribucions necessàries: Xarxes Neuronal Recurrents i Models Ocults de Markov. Els paràmetres d'aquests són estimats a partir de dades d'entrenament, i les corresponents distribucions són representades mitjançant Transductors d'Estats Finits amb Pesos (*Weighted Finite State Transducers*).

Amb l'objectiu de fer el marc de treball útil per a grans col·leccions de documents, es presenten distints algorismes per construir índexs de paraules a partir dels models probabilístics, tan basats en un lèxic

tancat com en un obert. Aquests índexs són molt semblants als utilitzats per motors de cerca tradicionals.

A més a més, s'estudia la relació que hi ha entre la formulació probabilística presentada i altres mètodes de gran influència en el camp de la detecció de paraules clau, destacant algunes limitacions dels segons.

Finalment, totes les aportacions s'avaluen de forma experimental, no sols utilitzant proves acadèmics estàndard, sinó també en col·leccions amb desenes de milers de pàgines provinents de manuscrits històrics. Els resultats mostren que el marc de treball presentat permet construir sistemes de detecció de paraules clau molt acurats i ràpids, amb una sòlida base teòrica.

Resumen

La detección de palabras clave (*Keyword Spotting*, en inglés), aplicada a documentos de texto manuscrito, tiene como objetivo recuperar los documentos, o partes de ellos, que sean relevantes para una cierta consulta (*query*, en inglés), indicada por el usuario, entre una gran colección de documentos. La temática ha recogido un gran interés en los últimos 20 años entre investigadores en Reconocimiento de Formas (*Pattern Recognition*), así como bibliotecas y archivos digitales.

Esta tesis, en primer lugar, define el objetivo de la detección de palabras clave a partir de una perspectiva basada en la Teoría de la Decisión y una formulación probabilística adecuada. Más concretamente, la detección de palabras clave se presenta como un caso particular de Recuperación de la Información (*Information Retrieval*), donde el contenido de los documentos es desconocido, pero puede ser modelado mediante una distribución de probabilidad. Además, la tesis también demuestra que, bajo las distribuciones de probabilidad correctas, el marco de trabajo desarrollada conduce a la solución óptima del problema, según múltiples medidas de evaluación utilizadas tradicionalmente en el campo.

Más tarde, se utilizan distintos modelos estadísticos para representar las distribuciones necesarias: Redes Neuronales Recurrentes o Modelos Ocultos de Markov. Los parámetros de estos son estimados a partir de datos de entrenamiento, y las respectivas distribuciones son representadas mediante Transductores de Estados Finitos con Pesos (*Weighted Finite State Transducers*).

Con el objetivo de hacer que el marco de trabajo sea práctico en grandes colecciones de documentos, se presentan distintos algorit-

mos para construir índices de palabras a partir de modelos probabilísticos, basados tanto en un léxico cerrado como abierto. Estos índices son muy similares a los utilizados por los motores de búsqueda tradicionales.

Además, se estudia la relación que hay entre la formulación probabilística presentada y otros métodos de gran influencia en el campo de la detección de palabras clave, destacando cuáles son las limitaciones de los segundos.

Finalmente, todas la aportaciones se evalúan de forma experimental, no sólo utilizando pruebas académicas estándar, sino también en colecciones con decenas de miles de páginas provenientes de manuscritos históricos. Los resultados muestran que el marco de trabajo presentado permite construir sistemas de detección de palabras clave muy rápidos y precisos, con una sólida base teórica.

Agraïments (Acknowledgements)

Ha arribat l'hora d'escriure aquestes línies d'agraïment i em resulta difícil saber per on començar. Són moltes les persones que d'una forma o altra m'han donat suport, companyia i un bon grapat de bons moments durant aquests cinc anys de doctorat.

Si haig de començar per algú, però, aquells que més impacte han tingut en el contingut d'aquesta tesi, han sigut sens dubte els meus directors, Enrique i Alejandro. Ells són qui em van introduir al camp del *Keyword Spotting*, m'han ajudat a aclarir i tirar endavant moltes de les idees que ací es presenten. Alejandro ha realitzat alguns dels experiments que ací es presenten i ha sigut un *tester* immillorable a l'hora de trobar errades als algorismes que jo implementava. Enrique ha estat sempre pendent del correu electrònic, suggerint noves idees i comentant les meves, fins i tot en dies festius i en hores en les quals no deuria haver ningú treballant. A més, tots dos m'han donat sempre llibertat i m'han animat a explorar altres àrees de recerca i idees que em rondaren pel cap, encara que no estigueren estrictament relacionades amb l'objectiu d'aquesta tesi.

I will be always grateful to the reviewers for accepting their task in such a short period of time. They have provided valuable comments that have greatly improved this thesis. I must emphasize the work of Laurence Likforman-Sulem, who produced a very thoughtful review, after an unexpected problem with another reviewer.

I must also thank the creators and maintainers of the OpenFST and Kaldi toolkits. Without these tools, many of the experiments in this thesis would have not been possible, or would have made my life a nightmare.

No puc oblidar-me de tota la gent del PRHLT que m'ha ajudat d'una forma una altra en algun moment del doctorat, o que fan de l'esmorzar i el dinar un dels millors moments de la jornada, a més de la resta de professors i investigadors més experimentats, que s'han interessat per la meva recerca, i m'han fet sentir una peça significant del grup d'investigació. Haig de mencionar especialment a Dani i Mauricio, que van ser protagonistes en el desenvolupament de *Laia*, una eina que és la base de molts dels experiments d'aquesta tesi. No puc oblidar-me tampoc de Paco, Luís, Bea, Rubén i tots amb els qui qui he acabat algun que altre dissabte al XL, Nylon, o qualsevol altre *garito* de la ciutat.

Si ara toca parlar de festa, no puc oblidar-me tampoc dels meus amics de Pedreguer, els de "tota la vida". Encara que ja no ens veiem totes les setmanes, he gaudit molt (i seguisc gaudint) dels dinars i sopars en ocasions especials. I com no, menció especial al grup Verdolaguer, per les innumerables hores (i cerveses) que hem passat allà rient i vivint la (bona) vida.

Finalment, vull dedicar les últimes paraules a la meva família: a la de sang i a l'escollida.

A Camila, per suportar les meves ganes d'emprenyar-la i tenir el coratge de fer-me emprenyar, per fer-me riure i tirar endavant en els moments en els quals volia cremar-ho tot (tesi inclosa), i per recordar-me que res és important si no hi ha amb qui compartir-ho.

I com no, als meus pares i la meua germana, que sempre m'han fet costat, m'han animat a fer allò que m'apassionara, i m'han donat estima, tot i el meu caràcter, (a vegades) un tant distant amb ells. També als meus *uelos* (als quatre), i a la meva tia. En especial al *Pepe*, que, per fi, deixarà de preguntar-me "encara no has acabat?". Ara ja sí.

Allà on vaja, és sempre amb vosaltres.

Zürich, 19 de Novembre de 2018.

Preface

The main goal of the thesis is to provide a theoretically-sound, efficient and robust algorithms to create word indexes from documents with an unknown content (i.e. documents with stochastic content).

In order to explain the details to build such indexes, the thesis has been sequentially organized in 10 chapters. We encourage the reader to follow this sequential order. Nevertheless, some chapters can be skimmed or read in a different order, depending on the particular interests of the reader. Figure 1 shows the main dependencies between the different chapters of this thesis.

Chapter 1 covers the basic preliminaries and traditional solutions to the problem of Keyword Spotting. Then, chapter 2, presents the probabilistic framework that we have developed to tackle Keyword Spotting. These chapters should not be skipped in order to ensure a common vocabulary, and to understand the core of the theory behind this dissertation.

Chapter 3 presents several of the typically used evaluation measures in the field of Keyword Spotting (Recall, Precision, Average Precision, etc.) and we proof that our probabilistic perspective provides an optimal ranking for all of them, under certain conditions. If the reader is already familiar with the evaluation measures used in the field, and/or is not fond of mathematical proofs, one can skip this chapter. However, we encourage to read it, since the optimality results are referenced in posteriors chapters of the thesis.

Chapter 4 describes the models for handwritten text that we have used for the experimental evaluation of our framework, namely Hidden Markov Models and Recurrent Neural Networks. If the reader

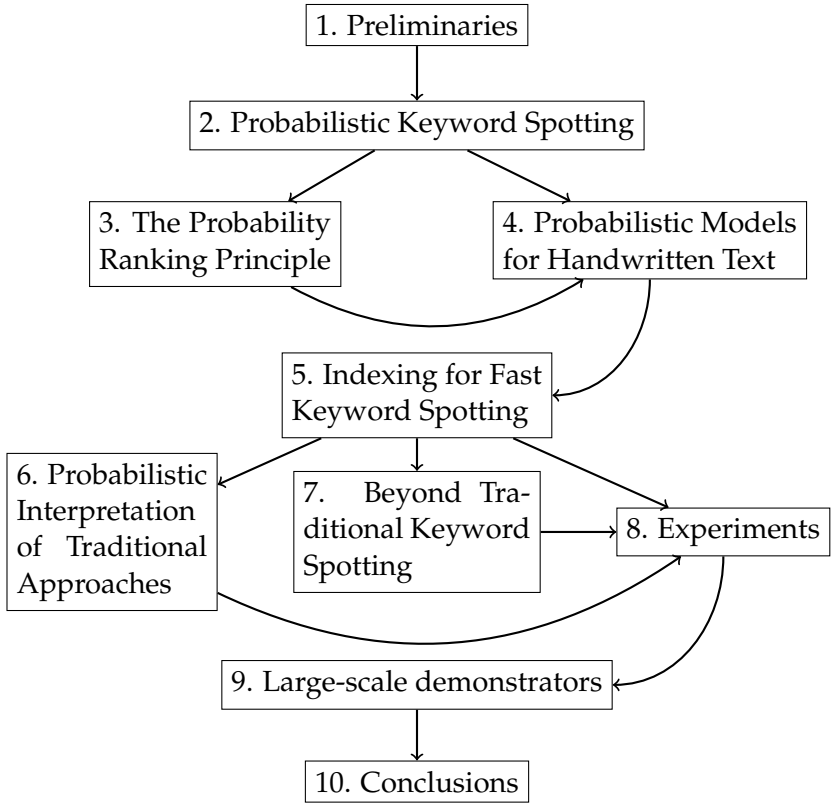


Figure 1. Illustration of the dependencies between the different chapters of this thesis.

is familiar with these models, then the first sections of this chapter may be skimmed. In any case, we highly encourage to read in detail section 4.6, where Weighted Finite State Transducers are presented. These are fundamental to understand the indexing algorithms.

Chapter 5 introduces the algorithms developed in this thesis to build such probabilistic indexes, in order to allow for fast Keyword Spotting solutions. This chapter, together with chapter 2, is one of the cornerstones of this dissertation.

Chapters 6 and 7 present, respectively, how our probabilistic framework relates with some traditional approaches, how to use it to tackle

other forms of Keyword Spotting, as well as other applications that have not been deeply studied by the community yet.

Chapter 8 presents the experimental evaluation of the proposed framework and algorithms, experiments showing the relationship with previous approaches, and non-traditional forms of Keyword Spotting.

Moreover, in chapter 9 we show that our solutions can be used to efficiently index collections of handwritten documents, much larger than traditional academic data sets.

Finally, chapter 10 summarizes the contributions of this work, including the scientific publications and open sourced software resulting from this work, and suggests interesting lines of future research.

Notation

Symbol(s)	Description
$A, B, \ldots, Z$	Random variables
$\boldsymbol{a}, \boldsymbol{b}, \ldots, \boldsymbol{x}$	Vectors
$\boldsymbol{A}, \boldsymbol{B}, \ldots, \boldsymbol{Z}$	Matrices or tensors
$[\boldsymbol{A}]_{i,j}$	Element of the matrix (or tensor) at row i and column j
$a_{1:T}$	A sequence of length T
$a_1, \ldots, a_T$	
$\boldsymbol{a}_{1:T}$	A sequence of vectors of length T
$\boldsymbol{a}_1, \ldots, \boldsymbol{a}_T$	
$\stackrel{\text{def}}{=}$	Used in equations to define a symbol or function
$\stackrel{\star}{=}$	The equality holds under some assumption
$a \leftarrow b$	In algorithms, the value of b is assigned to variable a
$P(\cdots)$	Probability mass function
$p(\cdots)$	Probability density function
$P(\cdots\,;\theta)$	Parametric probability mass function with respect to θ
$p(\cdots\,;\theta)$	Parametric probability density function w.r.t. θ
$\mathbb{E}[\cdots]$	Expected value of an expression
$\mathbb{E}[\cdots \mid \cdots]$	Expected value of an expression conditioned on some other

Contents

1 Preliminaries

During thousands of years the human kind used handwriting to preserve and share knowledge. However, its availability and speed of distribution was very limited, until the invention of the printing press, by Johannes Gutenberg (circa 1439). The printing press allowed an incredible acceleration in the distribution of information, and made possible that segments of the population that had never had access before, could start to gain it [McLuhan, 1962].

Moreover, handwriting has not always been the must reliable way of preserving information. Perhaps, the most iconic symbol of the destruction of human knowledge is the burning of the ancient library of Alexandria. Although its destruction is probably not due to a single event, it is certain that the library suffered several and important losses during its history, until its final collapse [MacLeod, 2004]. This is perhaps the most used example to illustrate how fragile are paper and ink, but it is not the only one in History.

In the current digital era, with the usage of computers and digital formats, information can be stored in a cheaper and more reliable way than ever before. In addition, any person around the world with access to a computer with Internet connection, can retrieve any piece of information, even if it is stored anywhere else in the globe. This has the potential to bring a true *democratization of human knowledge*, that was started with the invention of the printing press in the XVth century.

Given the previous reasons, it is no surprise that hundreds of archives and libraries have started digitizing their collections in order to protect their information from the passing of time, and also to allow

users to access their resources using the World Wide Web [Jimenez, 2007, D'Orazio, 2012, Madrigal, 2013, Paniagua, 2018].

During the 1950s and 1960s, the development of Handwritten Text Recognition (HTR) started, among other reasons, to make the digitized manuscripts more accessible. In the last 50 years, the field of HTR has improved significantly, and products that adopt its developments are present in the backpacks, pockets and desks of many people around the world. Nevertheless, when dealing with historical scanned documents, the current HTR solutions are still not accurate enough to deal with the large amount of variability that these documents present.

In the 1990s, researchers started working on Keyword Spotting (KWS) as an alternative to HTR. Regarding historical handwritten documents, the aim of KWS was to allow users to search for any *keyword* in a large collection of documents, without the need of a fully accurate transcription of the manuscripts.

Precisely, this has been one of the main motivations driving the this thesis. That is, to improve and develop new technology that allows libraries and archives to store the information contained in their manuscript collections, and that enables users to access this information in a efficient and robust manner.

In particular, this dissertation aims to do so by providing a formal probabilistic formulation of KWS. The core of the probabilistic view of the problem is the combination of seminal works from the Information Retrieval and Pattern Recognition fields, developed during the 1960s and 1970s. Surprisingly, this formulation has been practically neglected in the history of KWS applied to handwritten documents, although other application areas of KWS (i.e. in the speech community), and a few works in the handwritten domain, have drawn some connections between KWS and general Information Retrieval.

1.1 The field of a hundred names

The scientific literature is flooded with works that chase the aims described earlier. Nevertheless, depending on the authors' background or community, different names are used for tackling virtually the same

problem. One remarkable example is the name *Spoken Term Detection* (or STD), which is widely used by the speech processing community [Rose, 1995, Miller et al., 2007, Hazen et al., 2009]. Although the name *Keyword Spotting* (or KWS) has been used in the past by the speech processing community as well [Rohlicek et al., 1989, Weintraub, 1993], the former name has been broadly adopted in the recent years.

The contributions from the speech community are very significant, since they tackled the problem earlier than the researchers interested in historical handwritten documents [Khoubyari and Hull, 1993,Chen et al., 1993, Manmatha et al., 1996a, Manmatha et al., 1996b, Keaton et al., 1997]. As a matter of fact, some of the popular strategies to perform keyword spotting for handwritten documents were adopted from the speech community (see [Fischer et al., 2012], for example).

This should not come as a surprise to the reader, since the handwritten text recognition community has benefited for a long time from the developments made by their speech recognition colleagues: the use of Hidden Markov and Gaussian Mixture Models was first used for speech recognition [Jelinek, 1976], and then adopted by text recognition researchers [Kundu et al., 1989], and the same occurred with modern artificial neural network architectures, such as the Long-Short Term Memories (see [Graves et al., 2004] and [Graves and Schmidhuber, 2009]).

Sometimes, even within the same community, two different names are used to refer to the same problem. For instance, in the handwritten documents community, many researchers prefer the term *Word Spotting* [Manmatha et al., 1996a, Fischer et al., 2012]. And, just to make the things a bit more confusing, some researchers have used these names to refer to different problems (e.g. [Cambria et al., 2013]).

In this dissertation, we will use the term *Keyword Spotting* (abbreviated as KWS) which is the most popular in the literature nowadays. However, if the reader wants to investigate other works with the same or very similar aims, she should carefully review the literature related to all these topics:

- Keyword Spotting

- Word Spotting
- Spoken Term Detection (only for speech signals)

1.2 Taxonomy of Keyword Spotting systems

Different Keyword Spotting systems and publications can be classified using multiple criteria. In this section we aim to present the different categories that can be employed to classify a particular solution, which will be useful later to understand the assumptions and limitations of different approaches.

1.2.1 Segmentation assumptions

One of the most important practical distinctions between different Keyword Spotting systems is related to the assumptions that each system makes regarding the segmentation of the original document images. Collections of real handwritten document images have a large variability, and certain assumptions may be reasonable in some cases but not others. Any assumption that one system makes, is a real limitation if that assumption does not hold in reality.

1.2.1.1 Word segmentation

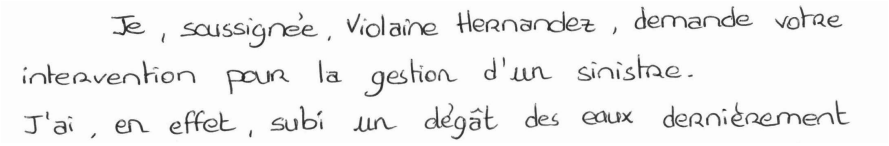
Many KWS approaches in the literature assume that is possible to have an accurate segmentation of the words present in the documents [Manmatha et al., 1996a, Almazán et al., 2012, Almazán et al., 2014, Sfikas et al., 2016, Sudholt and Fink, 2016, Mondal et al., 2016]. Generally, these approaches work directly with the cropped bounding boxes of the words in the document, and the query introduced by the user, which can be either a string or another image (see section 1.2.3). Many research assume that this allows to simplify the problem of KWS and helps to determine the best-case performance.

However, we can identify the following problems in these:

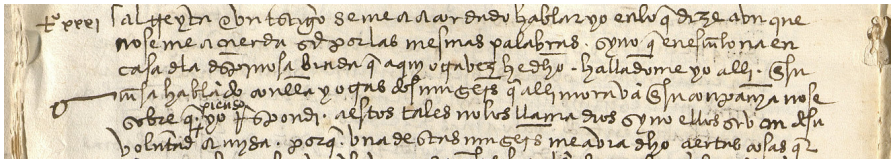
1. *Manual word segmentation is not practical.* This is quite obvious, since we aim to automate the processing of handwritten documents. It does not seem reasonable that one of the steps in-

volved needs of human labor to be completed. In addition, one has to realize that accurate *word* segmentation into bounding boxes or bounding polygons is a very tedious task (and thus, expensive).

2. *Automatic word segmentation is not good enough.* Some authors argue that, although manual word segmentation is not practical for obvious reasons, current automatic word segmentation approaches are good enough to perform word spotting. While this could be true for some academic data sets, actual data from real collections of historical documents clearly contradicts this hypothesis. See fig. 1.1, for example.



(a) RIMES database



(b) Alcaraz database

Figure 1.1. Accurate word segmentation is not always possible to perform. While isolated words are clearly identifiable in image (a), even expert human paleographers would face problems segmenting the individual words in image (b).

3. *Working with isolated word images discards useful information.* Notice that, if we try to identify whether a given query keyword is written in a particular cropped word region, we are assuming that the word contained in this particular region is independent of other words in the document. This assumption is obviously false, and it has been extensively shown in the literature that, using textual context information can substantially improve the results [Marti and Bunke, 2001b, Fischer et al., 2013, Toselli et al., 2015]. As an example of the importance of the context, see fig. 1.2. In theory, word segmentation does not nec-

essarily discard context information, but virtually all works assuming word segmentation ignore it.

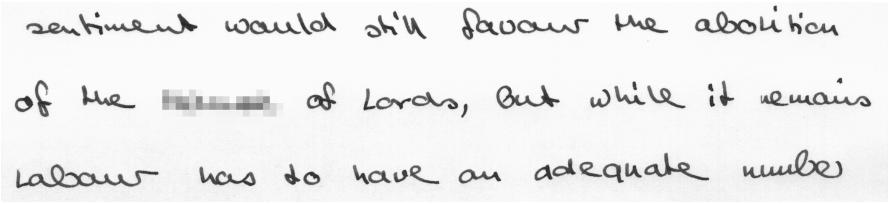


Figure 1.2. Textual context is a useful aid to identify words in handwritten text. Although the word "House" cannot be read in the image, any reader fairly formed in politics (or statistics), can infer it from the typical use of the English language or, at least, she would guess that words like "automobile" are very unlikely in this context.

1.2.1.2 Line segmentation

An important subset of papers in KWS assume that accurate line-segmentation is feasible in practice [Fischer et al., 2012]. These works typically use the machinery developed for Spoken Term Detection, since the x-axis in images (or y-axis for vertical-oriented text) can be interpreted as the time-axis in speech.

Of course, manual line segmentation is not feasible for real applications, for the same reasons as manual word segmentation. However, current automatic line segmentation approaches offer a very good accuracy in many historical collections, and thus this is a less restrictive segmentation assumption in practice [Likforman-Sulem et al., 2007, Louloudis et al., 2009, Grüning et al., 2017] .

One additional benefit that these systems offer, in front of most word segmentation-based approaches, is that they are able to take into account textual context information to improve the accuracy.

Nevertheless, the computation needed to take into account the context information is not negligible. The time needed to process a text line typically grows exponentially with the size of the textual context considered, unless some pruning and approximation strategies are used to accelerate the process. For instance, the number of states in a n-gram language models, which directly affects the running time of keyword spotting and text recognition systems using

these, increases exponentially with the context size n (more details will be discussed in section 4.5). Yet, this extra computational resources are only needed during a single step of the construction of the KWS index, as we will see in chapter 5.

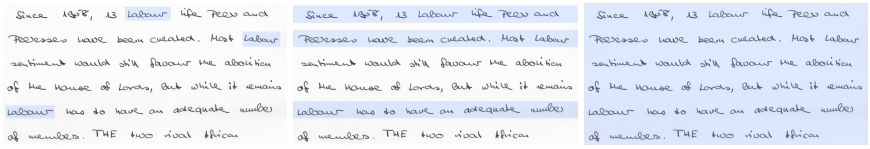
1.2.1.3 Segmentation-free

Finally, the less restrictive scenario is where no segmentation assumption is made whatsoever. There are many works that claim to follow a "segmentation-free" approach, however they actually are divided into two clearly separated steps: first, the location and segmentation of word-like areas of the image, and then deciding which of these areas are *relevant* for the given query [Rusiñol et al., 2011, Almazán et al., 2012, Papandreou et al., 2016, Rothacker et al., 2017].

Regardless of whether the system is implicitly or explicitly free of any segmentation assumption, we believe that all methods should be evaluated at some point under a fully automatic segmentation-free scenario. Mainly, because this will be the real operating scenario once the systems are deployed in libraries and archives, where millions of document pages have to be processed. Thus, despite the fact that the methods presented in this dissertation generally operate on segmented lines, we will carry on some experiments where no manual segmentation of the pages is given (see section 8.8).

1.2.2 RETRIEVED OBJECTS

Another important aspect of any KWS system is the type of the retrieved objects. In practice, this is usually related to the previous subsection: most works that operate under a line segmentation assumption retrieve "relevant lines" for a given query, and systems working under a word segmentation assumption, typically retrieve "relevant word instances". Nevertheless, although this is the usual practice, it does not mean that it is the only possible combination, nor the most recommended. Figure 1.3 illustrates the different types of retrieved objects described below.



(a) **(b)** **(c)**

Figure 1.3. Different types of objects to retrieve after a user query. For instance, if the user searches for "Labour", a keyword spotting system could choose to retrieve (a) individual instances of this keyword, (b) lines where this word was written, or (c) whole pages or paragraphs containing the keyword.

1.2.2.1 Word instances

As we mentioned before, a large portion of the keyword spotting systems described in the literature operate under the word segmentation assumption: that is, the system assumes that accurate word segmented regions have been extracted from the collection of images. Thus, when a user gives a query keyword to the system, it will provide a ranked list of the word regions that, according to the system, correspond to the given keyword [Manmatha et al., 1996a, Almazán et al., 2014, Sfikas et al., 2016, Sudholt and Fink, 2016, Gómez et al., 2017].

1.2.2.2 Lines

In a similar way than before, the system will retrieve full text line regions where it believes that the user's query keyword is written. This approach is followed also by many works [Fischer et al., 2012, Frinken et al., 2012, Toselli et al., 2013, Puigcerver et al., 2015c, Toselli et al., 2016b]. The benefit of this approach is twofold: first, it gives more context to the user to decide whether or not the retrieved object is actually of interest to the human labor required to produce the ground truth is much smaller. Also, it provides with performance measure values highly correlated with those systems retrieving word instances [Villegas et al., 2016b, Villegas et al., 2016a].

1.2.2.3 Pages

Last, but not least, the spotting system could choose to retrieve full pages or paragraphs containing multiple lines. This would give the user further context, and the measuring of the quality of the results would resemble more the traditional applications of Information Retrieval. Remember, that the user is trying to spot some keyword in our collection of documents because she needs to satisfy some need of *information*, and it is highly probable that the required information cannot be found in a single text line of our collection. Hence, it would make perfect sense that the system reported full text pages or paragraphs.

However, because most keyword spotting works are focused at a very low level (decide whether or not a word instance in a collection of images is the one that the user was searching for), they disregard this scenario. There are some works that evaluate their systems retrieving paragraphs, with more complex queries [Villegas et al., 2016a, Toselli et al., 2018b].

1.2.3 QUERY REPRESENTATION

The user may interact with the system in different ways, in order to present the query keyword. The two classical alternatives are the *Query-by-String* and *Query-by-Example* paradigms. Figure 1.4 shows a representation of the two paradigms, which are explained next.

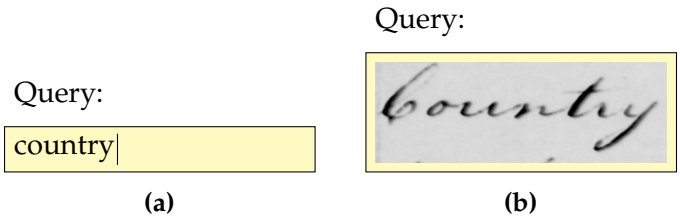


Figure 1.4. Illustration of the different query paradigms used in the Keyword Spotting literature. The yellow box represents the user input. In figure (a) the user types a query string using her keyboard, while in figure (b) she uses an exemplar image containing the keyword to search for.

1.2.3.1 *Query-by-String*

On the one hand, the Query-by-String (QbS) paradigm assumes that the query keyword is presented to the system as an individual symbol part of a vocabulary lexicon or, alternatively, as a sequence of characters of a given alphabet. This paradigm is typically adopted in the speech community [Rohlicek et al., 1989, Weintraub, 1993, Rose, 1995], and many of the handwriting community works influenced by the former [Fischer et al., 2012, Frinken et al., 2012, Toselli et al., 2013, Toselli et al., 2015].

1.2.3.2 *Query-by-Example*

On the other hand, the Query-by-Example (QbE) paradigm assumes that an exemplar image, containing the query keyword of interest, is given to the system, and it has to find the instances of the same keyword within the collection of document images [Manmatha et al., 1996a, Almazán et al., 2012, Tarafdar et al., 2013, Retsinas et al., 2016, Sfikas et al., 2016, Mondal et al., 2016].

Historically, in this case, KWS is seen as a particular instance of Content-based Image Retrieval (CBIR) [Toshikazu et al., 1991, Bird et al., 1996, Smeulders et al., 2000], since most researchers focusing on this paradigm have a Computer Vision background, where CBIR has a long tradition.

In this thesis, we will focus mainly on the QbS paradigm, however our probabilistic framework will also be applied to the QbE case. Without giving much further details, it will be shown in successive chapters (see section 2.3) that the QbE case only introduces one additional hidden variable with respect to the QbS case, and this only requires minor modifications to the algorithms.

One clear advantage of QbS in front of QbE is that the user only needs her keyboard to search for any imaginable concept that she wishes, and a broader set of queries (such as Boolean queries or arbitrary regular expressions) can be used. Notice that the QbE is more restricted in this sense, since, in principle, we would need at least one exemplar image for each of the keywords forming the query.

Nevertheless, the QbE paradigm also offers some advantages in front of some QbS approaches. In particular, as we will discuss later (see section 5.2), QbS approaches relying on a closed lexicon are prone to the out-of-vocabulary problem, while QbE approaches are essentially resistant to it.

1.2.4 TRAINING DATA

Finally, a fundamental distinction among different KWS systems is whether they need human annotated training data to be built. This is a very important distinction, since systems that do not need annotated data (i.e. unsupervised methods) are much cheaper to build than those requiring large quantities of annotated samples (also know as supervised methods).

1.2.4.1 Unsupervised

Initial works on KWS for historical documents typically fall into this category [Khoubyari and Hull, 1993, Manmatha et al., 1996a, Keaton et al., 1997]. Virtually all the unsupervised approaches have been restricted to the query-by-example paradigm, explained before. Typically, researchers apply some feature extraction mechanism on the images (typically, pre-segmented word-shaped boxes) in order to extract visual features that could discriminate *similar* images. Then, they compare the features extracted from the collection of images against the features obtained from the query, in order to rank them by some similarity (or dissimilarity) measure. Recently, there are still works being published under this paradigm [Tarafdar et al., 2013, Papandreou et al., 2016, Retsinas et al., 2016, Dey et al., 2016, Zagoris et al., 2017], although the popular trend is to use supervised methods.

1.2.4.2 Supervised

Because visually similar or dissimilar images do not necessary mean relevant or irrelevant pairs of objects, researchers soon noticed that better quality results could be obtained by using supervised methods. In fact, most recent successful KWS methods are supervised algorithms [Fischer et al., 2012, Frinken et al., 2012, Almazán et al., 2014, Sudholt and Fink, 2016, Toselli et al., 2013].

This thesis focuses on supervised methods. As it will be shown, our framework involves the probability distribution of the content of the text images (i.e. the posterior probability over transcripts), and supervised methods excel in the task of learning parametric models for these kind of distributions. However, experimental evaluation with different amounts of supervised training data will be carried on, in order to measure the amount of needed annotation.

1.3 Information Retrieval

In this dissertation, KWS is interpreted as a particular scenario of a (very) simple *information need*. Precisely, this is the goal of Information Retrieval (IR), as an academic field: to build information systems that enable users to find useful information among large collections of unstructured documents [Manning et al., 2008]. The academic field was originated almost in parallel to computers, back in the 1940s and 1950s, mainly in order to organize companies and libraries catalogs [Sanderson and Croft, 2012]. With the popularization and spread of the Internet, the field gained significant importance and has been the central business of various companies (such as Yahoo, Google or the extinct Altavista or Lycos).

The term *information need* is very fuzzy and can be ambiguous in many cases. For instance, when a user searches for "Paris" while she is planning her holiday trip, she probably wishes to find nice hotels, the best flight fares and interesting attractions. However, when some local Frenchwoman searches from "Paris" on the Internet, she is probably expecting to find other types of information (e.g. the web address of the City Hall, hospitals, etc.). The task of the IR system is to provide the users with *relevant* answers for their information need.

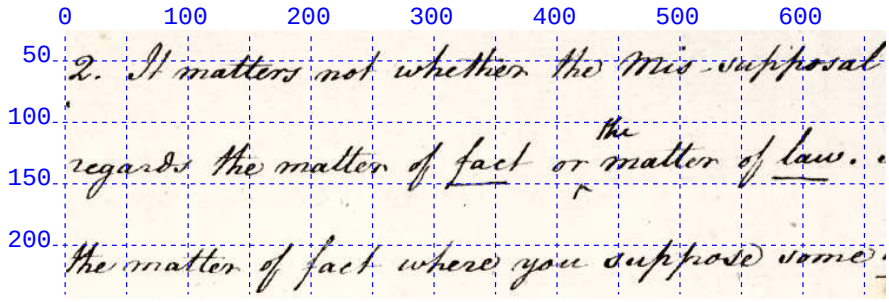
When we place traditional KWS systems and literature under the IR goggles, the definition of *relevant* information is trivial: a document (i.e. full page, text line or individual word instance) is *relevant* if, and only if, it contains an instance of the queried keyword. However, these systems have to cope with different sources of ambiguity: in particular, the textual content of the documents is unknown. This contrasts with the early web search example, where the very defini-

tion of *relevance* is ambiguous, but the content of the documents is not.

Regardless of the source of uncertainty, probability is the standard way of measuring it. Thus, the systems presented in this dissertation can be interpreted as instances of Probabilistic IR systems [Manning et al., 2008] applied to KWS in handwritten documents.

Finally, we want to emphasize the fact that aiming at building "indexes" from text images was one of the foundational goals of KWS [Manmatha et al., 1996a, Manmatha et al., 1996b], as well as interpreting KWS as an instance of Information Retrieval. However, the indexes that we build are more closely related to the ones used by traditional search engines, which make them very easy to use and integrate with existing IR systems.

In order to better illustrate our indexing goals, fig. 1.5 shows an example of one of the probabilistic KWS indexes that we intend to build, for a given (segment of a) page containing handwritten text.



Keyword	Prob.	Bounding box
2	0.93	1 – 36 – 20 – 31
21	0.07	1 – 36 – 24 – 31
It	0.98	33 – 36 – 27 – 31
If	0.01	33 – 36 – 26 – 31
	...	
some	0.83	570 – 198 – 78 – 31
soner	0.02	576 – 198 – 83 – 31

Figure 1.5. Illustration of a probabilistic index similar to the ones that we intend to build as the outcome of this thesis.

In this example, the index contains the list of candidate words written in the image, with its location (bounding box) and the confidence (probability) of the system in that entry. For example, our KWS system is very confident that the word "It" and "some" are written in the image, although it is less certain about the latter. At the end of this thesis, the reader will be able to understand the algorithms used to build this index and the theory supporting them.

1.4 Pattern Recognition

As we discussed in the previous section, KWS can be interpreted as a form of Information Retrieval where the content of the documents is unknown. Recall that we are dealing with scanned document images containing text. Thus, under a probabilistic formulation, we need to *guess* which text is likely to actually be written in such images. Precisely, Pattern Recognition (PR) is the academic field that focuses on the recognition of patterns from arbitrary data [Duda et al., 2000, Bishop, 2006]. PR is sometimes considered a branch of Machine Learning (ML), but many authors consider them synonyms [Bishop, 2006].

As the title of this thesis suggest, it focuses on probabilistic (or statistical) PR algorithms. In fact, although many PR algorithms or tasks may not involve the explicit computation of any probability or distribution, many of them can be reinterpreted in these terms, and all of them use statistical methods to (1) *learn* the parameters of the algorithm from a given *training data*, and (2) *decide* (or predict) some set of optimal actions or outcomes from new *evaluation data*.

Pattern Recognition has proven to be a phenomenal approach to find *general* rules to solve problems from examples. In practice, once we have *trained* our model from data, we need to check that it actually *generalizes* well to previously unseen data (produces the correct answer). This is an essential step to ensure that our algorithm is not just memorizing the data supplied during training. Different algorithms and models have different generalization properties. The number of parameters of model, the dimensionality of the data, the independence assumptions taken into consideration, etc. are just a few

aspects that affect the ability of a PR method to be successful [Duda et al., 2000, Bishop, 2006, Murphy, 2012].

1.5 Decision Theory

Decision Theory is the study of the reasoning underlying an *agent*'s choices. In particular, Decision Theory is used to *decide* the optimal strategy with respect some utility or loss function. Decision Theory is deeply linked to Pattern Recognition. Notice that during the training phase of any PR method, the system has to *decide* the optimal set of parameters, according to some optimization criterion (e.g. maximizing the likelihood function, or minimizing the least squares error). Similarly, during the evaluation phase, the system hast to *decide* the optimal values of the desired variables, according to some criterion (e.g. minimize the expected classification error).

In this thesis, we will use Decision Theory to prove that the developed probabilistic framework is optimal for a broad set of criteria typically used in KWS and many other IR application.

The simplest application example of Decision Theory is to solve a binary classification problem. That is, given some object and two classes (sometimes, also known as categories or labels), we need to *decide* to which class this object belongs to. In any binary classification problem there are four possible outcomes, which can be represented by a 2×2 *cost matrix* (or, equivalently, a utility matrix).

In the context of KWS, given a query and some arbitrary object to retrieve, the system needs to decide whether that object is relevant for the query or not (i.e. whether it should be retrieved). Table 1.1 shows the cost matrix, Δ, involved in the decision.

Because the *truth* about the relevance of the object for the given query is unknown, we want to make a decision that minimizes the expected cost of the outcome. Thus, we denote the relevance of an object with a binary random variable, R. When the object is "Relevant", we assume that the value is $R = 1$. When the object is "Not relevant", the value of the variable is $R = 0$.

Table 1.1. Cost matrix for the binary decision problem involved in Keyword Spotting. For instance, Δ_{10} is the cost of classifying as "Not relevant" an object which actually is "Relevant".

		Decision	
		Not relevant	Relevant
Truth	Not relevant	Δ_{00}	Δ_{01}
	Relevant	Δ_{10}	Δ_{11}

Since the definition of *relevance* depends on the object and the query, in order to make the optimal choice we need to know the *true* conditional probability $P(R \mid X = \boldsymbol{x}, V = v)$, where $\boldsymbol{x}$ and v represent the given object and query, respectively. If the relevance depended on other variables, we would need to condition its value also on them. Nevertheless, the following reasoning is independent on the variables on which the value of R depends, thus we will drop the conditions from the next notation.

The optimal value of the variable R (denoted with r^*) that minimizes the expected cost, is given by the Bayes decision rule:

$$\begin{aligned} r^* &= \arg\min_{r' \in \{0,1\}} \mathop{\mathbb{E}}_{r} [\Delta_{rr'}] \\ &= \arg\min_{r' \in \{0,1\}} P(R=0)\Delta_{0r'} + P(R=1)\Delta_{1r'} \end{aligned} \tag{1.1}$$

Here, r denotes the *true* value of the relevance variable, while r' represents an hypothetical decision. Performing simple operations on the previous expression we can derive a decision rule that depends only on $P(R=1)$ and a single threshold value η:

$$\begin{aligned} r^* &= \begin{cases} 0 & P(R=0)\Delta_{00} + P(R=1)\Delta_{10} < P(R=0)\Delta_{01} + P(R=1)\Delta_{11} \\ 1 & \text{otherwise} \end{cases} \\ &= \begin{cases} 0 & P(R=1) < \frac{\Delta_{01} - \Delta_{00}}{\Delta_{10} + \Delta_{01} - \Delta_{11} - \Delta_{00}} = \eta \\ 1 & \text{otherwise} \end{cases} \end{aligned} \tag{1.2}$$

A low threshold will likely rise the number of false positives (or Type-I errors), since we will increase the number of objects assigned

to the "Relevant" class. On the contrary, a high value of the threshold will increase the number of false negatives (or Type-II errors), since more objects will be marked as "Not relevant". These two types of errors play a crucial role in many quality measures used to evaluate Information Retrieval systems (see chapter 3).

2 *Probabilistic Keyword Spotting*

In section 1.5 we briefly related KWS (and many other Information Retrieval scenarios) to a binary classification problem: to decide whether a particular image is relevant for a given query. Note that even in the segmentation-free scenario where the system has to retrieve all the relevant *positions* within a candidate image, the problem can still be formulated as a binary classification problem. The only difference is that the concept of *relevance* is conditioned on the image $\boldsymbol{x}$, the query v, and a particular position, $\boldsymbol{p}$, within the image.

Next, we will present in detail a probabilistic formulation of KWS, which is one of the main contributions of this thesis. First, we will study the concept of *position-independent relevance*, where we do not care about the exact position of the spotted words within the image regions considered. Then, the concept of *position-dependent relevance* will be introduced for different types of *positions*. This will later be used to tackle segmentation-free KWS, in such scenarios where the system has to retrieve all the instances of the query words within a unsegmented collection of images.

2.1 Position-independent Keyword Spotting

In this section, we assume that we aim to determine whether or not the text specified by a given query, v, is written somewhere in a given image $\boldsymbol{x}$. For instance, the candidate image could be a segmented word image, a text line containing several words, or a full text page containing multiple text lines. Notice that we are not concerned on the particular location of the queried text within the image, or whether v is written more than once in the image. This definition of *relevance*

can be expressed as:

$$R \mid \boldsymbol{x}, v \stackrel{\text{def}}{=} \begin{cases} 1 & v \text{ is written somewhere in } \boldsymbol{x} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Notice that in the previous definition, the condition that needs to be met to determine the relevance is not rigorously (mathematically) formulated. Let w be the actual text depicted in the image $\boldsymbol{x}$ (i.e. its unknown *true* transcript). When we say that "the text v is written somewhere in $\boldsymbol{x}$" we just understand that v is included in w. Observe that, if the transcript of the image (w) is given, the concept of *relevance* does not depend on the image itself, only on its transcript. Now, we are able to rigorously define the concept of *relevance*.

$$R \mid \boldsymbol{x}, v \;=\; R \mid \boldsymbol{x}, w, v \;=\; R \mid w, v \stackrel{\text{def}}{=} \begin{cases} 1 & w = w'vw'' \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where $v, w', w'' \in \Sigma^*$ are assumed to be sequences of symbols over an adequate set, Σ, of linguistic units such as characters, syllables, words, etc. For the sake of clarity, rather than writing $w = w'vw''$ we will use $w \in L(v) \stackrel{\text{def}}{=} \Sigma^* v \Sigma^*$, where $L(v)$ will denote the *language* (i.e. set of sequences) over Σ such that v is contained in all the sequences of $L(v)$:

$$R \mid \boldsymbol{x}, v \;=\; R \mid w, v \stackrel{\text{def}}{=} \begin{cases} 1 & w \in L(v) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Following the decision theory framework described in section 1.5, we aim to compute $P(R = 1 \mid X = \boldsymbol{x}, V = v)$ in order to decide whether or not the image region $\boldsymbol{x}$ is relevant for the query v. Since the relevance is defined in terms of the text written in the image, we will need to marginalize over the variable W, representing the transcripts of the image region $\boldsymbol{x}$.

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) = \sum_{w \in \Sigma^*} P(R = 1, W = w \mid X = \boldsymbol{x}, V = v) = \sum_{w \in \Sigma^*} P(R = 1 \mid X = \boldsymbol{x}, V = v, W = w) P(W = w \mid X = \boldsymbol{x}, V = v) \tag{2.4}$$

Recall that our definition of *relevance*[1] in eq. (2.3) is conditionally independent on the image itself when the transcript is given. This simply means that $P(R \mid X, V, W) = P(R \mid V, W)$. In addition, we can also assume that the transcript of an image is conditionally independent on the query, when the image is given (i.e. $P(W \mid X, V) = P(W \mid X)$). This assumption is actually true in practice, since users can perform any kind of query, regardless of the text rendered in a particular image. Using these two facts, the previous equation simplifies to:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) = \sum_{w \in \Sigma^*} P(R = 1 \mid V = v, W = w) P(W = w \mid X = \boldsymbol{x}) \tag{2.5}$$

Now, we can split the sum over all possible $w \in \Sigma^*$ into two disjoint sets, $L(v)$ and $\Sigma^* - L(v) = \{w : w \notin L(v)\}$:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) = \sum_{w \in L(v)} P(R = 1 \mid V = v, W = w) P(W = w \mid X = \boldsymbol{x}) + \sum_{w \notin L(v)} P(R = 1 \mid V = v, W = w) P(W = w \mid X = \boldsymbol{x}) \tag{2.6}$$

Finally, observe that from the definition of *relevance* in eq. (2.3) $P(R = 1 \mid V = v, W = w) = 1$ for all the transcripts including the queried text v (i.e. $w \in L(v)$). Likewise, for all the transcripts *not* including v this probability will be exactly zero. Thus, the relevance probability is equal to:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) = \sum_{w \in L(v)} P(W = w \mid X = \boldsymbol{x}) \tag{2.7}$$

Up to this point we did not need to specify which is the concrete set of linguistic units, Σ, used to represent both query text and transcripts. A discussion about possible choices will appear in chapter 5, along with corresponding developments and specific techniques for each choice. Also, as will be discussed in chapter 7, all the above formulation applies also for more complex queries where v is not just

[1] Also used by virtually all KWS benchmarks.

a single sequence of characters or words, but a set of sequences defined by Boolean operators (AND/OR/NOT) or even general regular expressions.

In the remaining of this chapter, however, we will assume that Σ is a set of *words* and v consists in just a single word. This will permit a more focused (and traditional) presentation, which allows for an easier reading without entailing any loss of generality. Under this assumption, fig. 2.1 illustrates the application of eq. (2.7) to a rather extreme case where the underlying transcript of a text image is somehow ambiguous.

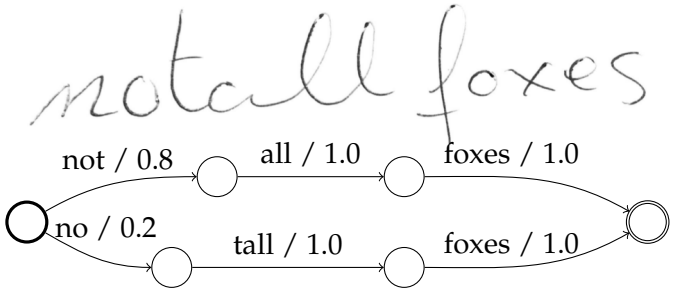


Figure 2.1. An image with two possible transcripts. The two are: "not all foxes" (probability 0.8), and "no tall foxes" (probability 0.2). They are represented by a complete path through a weighted directed acyclic graph, depicted below the image, and their posteriors are the product of the weights through the path.

On the one hand, the person that wrote the text may have intended to write "not all foxes", but she did not put enough space after the word "not". On the other hand, she perhaps intended to write "no tall foxes" and forgot to leave enough space after the word "no". In any case, one might argue that the first transcript is more likely than the second one, since the adjective "tall" usually refers to people, and foxes are animals not particularly tall after all.

Given this premise, we can now compute the probability of relevance of any given query word. For instance, for the word "tall" there is only one transcript including this word, with probability equal to 0.2. However, notice that the word "foxes" is included in all possible transcripts, thus the probability that the image is relevant for this keyword is 1.0, which is equal to the sum of the posteriors of all possible

transcripts of the image.

$$P(R = 1 \mid X = \boldsymbol{x}, V = \text{“tall”}) = 0.2$$
$$P(R = 1 \mid X = \boldsymbol{x}, V = \text{“foxes”}) = 1.0$$

2.1.1 WORD-SEGMENTED IMAGE REGIONS

In the particular case that the image region $\boldsymbol{x}$ represents a segmented word image, then eq. (2.7) reduces to the posterior of the query keyword v, given the word image.

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) \stackrel{\star}{=} P(W = v \mid X = \boldsymbol{x}) \tag{2.8}$$

This is due to the fact that $L(v) = \{v\}$, when the image regions are assumed to contain a single word.

In this particular case, there is a direct equivalence between the probability distributions used in word spotting and text recognition. However, this is not true in the general scenario where $\boldsymbol{x}$ may contain an arbitrarily large number of words.

2.2 Position-dependent Keyword Spotting

In this section, we will study the keyword spotting scenarios when we wish to determine whether or not a given query keyword, v, is written at a specific *position of interest*, $\boldsymbol{p}$, within the image region $\boldsymbol{x}$. This is especially useful when the image region $\boldsymbol{x}$ contains a significant amount of text, such as a line or full page. Then, the user typically would expect the location of the spotted word within the image, and not only to answer whether the image was relevant or not. As we mentioned earlier, this is particularly required in the case of segmentation-free KWS.

Notice that we are using the vector notation for the position, since we could represent an arbitrary position. For instance, a position could be just a pixel coordinate within the image $\boldsymbol{x}$, or we could represent the left and right columns of a text line image (in these cases, $\boldsymbol{p} \in \mathbb{N}^2$). If the text image $\boldsymbol{x}$ contains multiple text lines we could want to use word bounding box coordinates as positions ($\boldsymbol{p} \in \mathbb{N}^4$).

Finally, we could want to represent not a physical position within the image, but a logical position within the transcript of the image ($\boldsymbol{p} \in \mathbb{N}$).

In particular, we will study three cases for the meaning of a position $\boldsymbol{p}$:

1. The position $\boldsymbol{p} \in \mathbb{N}$ represents a column within a text line image. In this case, we will just denote the position as c, instead of using $\boldsymbol{p}$. L_τ will denote the random variable representing the column of interest.

2. The position $\boldsymbol{p} \in \mathbb{N}^2$ represents a segment (or interval) within a text line image. In this case, we will assume that we work with horizontally-oriented text and that (c_0, c_1) represent the left and right columns of the interval, respectively. L_σ will denote the random variable over the segment of interest.

3. The position $\boldsymbol{p} \in \mathbb{N}$ represents the *logical* position within the transcript of a text line image. In this case, we will simply refer to the position using the variable k. L_κ will denote the random variable over the position of interest.

The reader should notice that we are assuming that the image $\boldsymbol{x}$ represents a text line in all three cases. It is worth emphasizing that these definitions could be easily extended to fully segmentation-free scenarios ($\boldsymbol{x}$ representing a full page), but we decided to rely on the line-segmentation assumption for practical reasons: modern methods for text line detection and segmentation are quite robust and it is very common to work at this level when processing handwritten documents.

2.2.1 Relevance of an image column

In this scenario, we want to determine whether or not the query keyword, v, is written in a particular column, c, of the text line image, $\boldsymbol{x}$. We will say that the word is written in a column of a text line, if the bounding box of the keyword representation includes the column. Figure 2.2 gives a clear example to better understand the idea.

Figure 2.2. Example showing two relevant columns (c_2 and c_3) of the given image and the query keyword "foxes", and a non-relevant column (c_1). Intuitively, any column of the image belonging to the bounding box of the keyword within the image will be relevant.

We can capture this intuition in a more formal statement:

$$R \mid \boldsymbol{x}, v, c \stackrel{\text{def}}{=} \begin{cases} 1 & v \text{ is written at column } c \text{ of the line image } \boldsymbol{x} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

As we did in the previous section, we need to formalize slightly better this definition, in order to express in mathematical notation what we mean by "a word is written at a column".

In the previous section, where we did not care about the location of the words depicted in the image, we only needed to introduce a random variable W to represent the actual text written in the image. Nevertheless, notice that the transcript can be *aligned* in many different ways to each column of the image.

Alignments can be represented in many different ways, depending on the constraints over them. For instance with one additional random variable A, over sequences of natural numbers, we can represent the initial column of each word in the transcript $w = w_1, \ldots, w_n$ of the text line image.

Suppose that the transcript of the given text line $\boldsymbol{x}$ is given by the sequence $w = w_{1:n} = w_1, \ldots, w_n$, and the initial column of each word in the transcript is given by the sequence $a_{1:n+1} = a_1, \ldots, a_n, C$. Notice that in order to simplify the definitions and equations, we are assuming that the sequence of columns representing the alignment, has an extra element at the end which is always equal to the total number of columns, C. This allows us to fully formalize the definition of *rele-*

vance of a given query keyword, text line image, and column:

$$R \mid \boldsymbol{x}, v, c =$$
$$R \mid v, c, w_{1:n}, a_{1:n+1} = \begin{cases} 1 & \exists k : w_k = v \wedge a_k \leq c < a_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

Now, we are able to give an expression for the relevance probability of a column, by marginalizing over the random variables W and A. In the following equations, the random variables will be dropped in order to simplify the notation. As an illustration, we would simplify the probability $P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c)$ as $P(R = 1 \mid \boldsymbol{x}, v, c)$, since the symbols representing the values that the random variables take are enough to identify the corresponding random variables.

$$P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c) =$$
$$\sum_{w_{1:n} \in \Sigma^*} \sum_{a_{1:n+1} \in \mathbb{N}^{n+1}} P(R = 1 \mid \boldsymbol{x}, v, c, w_{1:n}, a_{1:n+1}) P(w_{1:n}, a_{1:n+1} \mid \boldsymbol{x}, v, c) \tag{2.11}$$

First, as we did in the development of the previous section, we must notice that the relevance does not depend on the image itself when the transcript of the text line and its alignment are given. Likewise, the transcript and its alignment only depend on the image itself, and not on the query keyword and column that the user may be interested in. Thus, the equation above simplifies to:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c) =$$
$$\sum_{w_{1:n} \in \Sigma^*} \sum_{a_{1:n+1} \in \mathbb{N}^{n+1}} P(R = 1 \mid v, c, w_{1:n}, a_{1:n+1}) P(w_{1:n}, a_{1:n+1} \mid \boldsymbol{x}) \tag{2.12}$$

Finally, we can split the sums into two subsets: the set of transcripts and alignments for which eq. (2.10) is equal to 1, and those for which it is equal to 0. Thus, the expression is simplified to:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c) =$$
$$\sum_{\substack{w_{1:n} \in \Sigma^*, \exists k: \\ w_k = v}} \sum_{\substack{a_{1:n+1} \in \mathbb{N}^{n+1}: \\ a_k \leq c < a_{k+1}}} P(W = w_{1:n}, A = a_{1:n+1} \mid X = \boldsymbol{x}) \tag{2.13}$$

In short, this is the sum of the probability of all transcripts and alignments such that the transcript includes the word v at some position, and the alignment of that word includes the column c.

Now, we can use this equation to plot a "heat-map" on top of the text line image, representing the probability that the given keyword is written in each column of the image. For instance, let's consider again the example depicted in fig. 2.1. However, this time, the weighted directed acyclic graph will not only represent the transcripts of the text line image, but also their possible alignments.

Figure 2.3 shows the same example image as before, with the heat-map for the query keyword "all" superimposed, and the new graph with the transcripts, their probabilities (words and weights in the edges), and (some of) the alignments of these (represented by the start column, depicted at each node of the graph). Notice that this is a very visual way to report the location of the spotted word to the user.

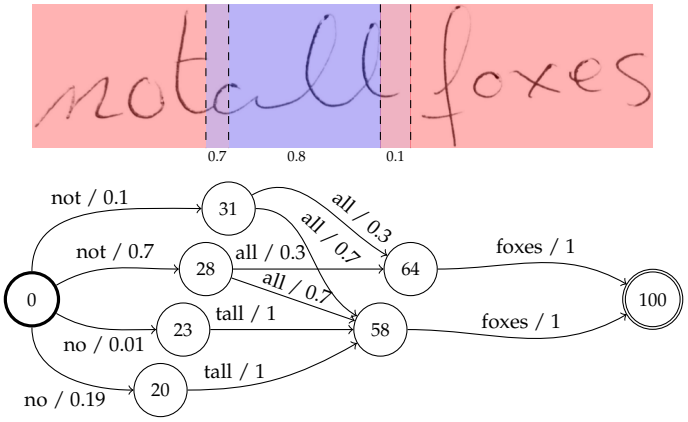


Figure 2.3. Heat map representing the relevance probability of the image columns for the query keyword "all". The heat-map was computed using the probability distribution over the transcripts and alignments of the image, $P(W, A \mid X = \boldsymbol{x})$, represented by the weighted directed acyclic graph (words and numbers in the edges represent transcripts and probabilities, and the numbers in the nodes represent the alignment). Columns with high probability are filled with blue, and those with a low probability are filled with red. For instance, all columns between the 31st and the 58th have a relevance probability equal to 0.8.

According to eq. (2.13), in order to compute the relevance probability of an image column, c, for a given keyword v, we need to sum the posterior probability of all alignments (i.e. paths in the graph) where the word written on top of column c is equal to v. In the figure above, let's consider the 45th column (the procedure for any column in the range $[31, 58)$ would be identical and yield the same result), and let's consider the keyword "all".

There are four edges in the graph that traverse the 45th column. We will denote each edge by the tuple (start state, end state, word, probability). For each of them we will compute the total posterior probability mass due to this edge, that is the sum of all paths through each of them.

- $(28, 58, \text{all}, 0.7)$, with total probability $0.7 \cdot 0.7 \cdot 1.0 = 0.49$.
- $(28, 64, \text{all}, 0.3)$, with total probability $0.7 \cdot 0.3 \cdot 1.0 = 0.21$.
- $(31, 64, \text{all}, 0.3)$, with total probability $0.1 \cdot 0.3 \cdot 1.0 = 0.03$.
- $(31, 58, \text{all}, 0.7)$, with total probability $0.1 \cdot 0.7 \cdot 1.0 = 0.07$.

Thus, the relevance probability of this column, for the keyword "all" is 0.8 (the sum of the four paths), as shown in the figure.

2.2.2 RELEVANCE OF AN IMAGE SEGMENT

In the previous section, we presented a probability distribution that allowed us to draw a "heat-map" in order to represent the location of the spotted words, for a given query. In this section, we will present an alternative distribution which is more useful in practice, when one wants to build a search index of the terms written in a collection of text lines.

In this section, we aim to determine whether or not the query keyword, v, is written *exactly* at a particular segment of the text line $\boldsymbol{x}$, delimited by columns c_0 and c_1 (i.e. it starts at column c_0 and ends at c_1 of the text line image). Intuitively, we want our definition of

relevant to be:

$$R \mid \boldsymbol{x}, v, c_0, c_1 \stackrel{\text{def}}{=} \begin{cases} 1 & v \text{ starts at column } c_0 \text{ and ends at } c_1 \text{ of } \boldsymbol{x} \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

In order to properly define this concept, we will need to represent the alignments of the transcripts with the text line image, as we did in the previous section. Thus, we will use again the random variable A to represent the alignment of the transcript. A is a random variable over sequences of columns, $a_{1:n+1} = a_1, \ldots, a_n, C$.

$$\begin{aligned} R \mid \boldsymbol{x}, v, c_0, c_1 = \\ R \mid v, c_0, c_1, w_{1:n}, a_{1:n+1} \stackrel{\text{def}}{=} \begin{cases} 1 & \exists k : w_k = v \wedge a_k = c_0 \wedge a_{k+1} = c_1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.15}$$

Observe that we are imposing the constraint in the definition that the k-th word in the transcript must be equal to the query keyword, and it's alignment must start and end *exactly* at columns c_0 and c_1 of the text line image.

Following the same steps as in the previous section, one arrives to the next expression, which is used to compute the relevance probability of a text line segment.

$$\begin{aligned} P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\sigma = (c_0, c_1)) = \\ \sum_{\substack{w_{1:n} \in \Sigma^*, \exists k: \\ w_k = v}} \sum_{\substack{a_{1:n+1} \in \mathbb{N}^{n+1}: \\ a_k = c_0 \wedge a_{k+1} = c_1}} P(W = w_{1:n}, A = c_{1:n+1} \mid X = \boldsymbol{x}) \end{aligned} \tag{2.16}$$

Notice that we can compute eq. (2.13) (i.e. $P(R = 1 \mid \boldsymbol{x}, v, c)$), from eq. (2.16) (i.e. $P(R = 1 \mid \boldsymbol{x}, v, c_0, c_1)$) by summing the relevance probability of all segments that contain the column c. In addition, the relevance probability introduced in this section allows us to extract other useful information like how many instances of a given query keyword are written in the given text line.

Following the example in fig. 2.3, we could retrieve the segments where the word "all" is likely to be written, and their probabilities.

Table 2.1. Possible horizontal segments where the word "all" is written in the example depicted in fig. 2.3.

Term	Segment	Probability
all	28–58	0.49
all	28–64	0.21
all	31–58	0.07
all	31–64	0.03

2.2.3 RELEVANCE OF A TRANSCRIPT POSITION

In the previous section we introduced a relevance probability of a text line segment, which we could use to tell whether or not a given segment of a given text line is relevant for the given query keyword. Nonetheless, from table 2.1 one can notice that multiple alignment possibilities for the same word instance can dilute the total probability mass.

Intuitively, we would like to provide the user with the multiple instances of the keyword within the text line, with their best (or expected) alignment (i.e. segment).

In order to do so, one could try to sum the probability of all the overlapping segments. And then, keep the interval with the highest probability Nevertheless, this may become problematic since it is possible that multiple instances of the same word appear in the text line with some overlapping segments. Figure 2.4 highlights this problem using (part of) a famous proverb from the Greek philosopher Parmenides of Elea (born circa 515 BC).

As a better alternative, in this section we propose a new relevance probability conditioned on the text line $\boldsymbol{x}$, the query keyword v, and the *logical position* within the transcript sequence. Intuitively, we want to decide whether or not the k-th word (word at position k) of the text line depicted in the image $\boldsymbol{x}$ is relevant for the query v. Thus, intuitively, we could define this concept of relevance as:

$$R \mid \boldsymbol{x}, v, k \stackrel{\text{def}}{=} \begin{cases} 1 & v \text{ is the } k\text{-th word in image } \boldsymbol{x} \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

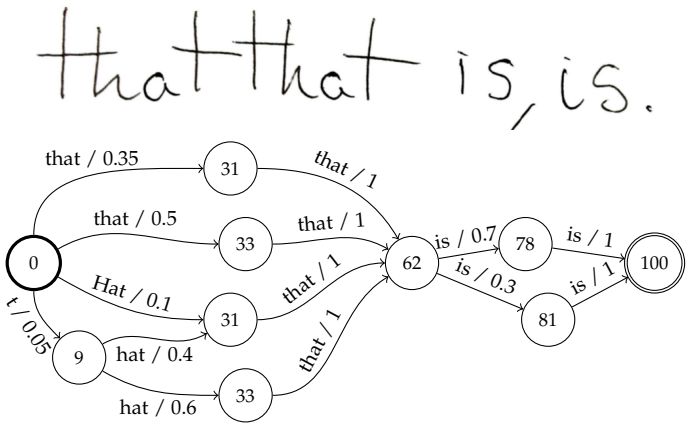


Figure 2.4. Example showing that multiple instances of the same keyword (e.g. "that") may have overlapping segments. Notice that the instance of the word "that" in the interval 0–33 overlaps with another instance of the same word in the interval 31–62. The set of hypotheses includes also other transcripts like "Hat that is is" or "t hat that is is". Punctuation marks are excluded to simplify the graph. Although the example seems artificial, it comes from Parmenides' proverb: "That that is, is. That that is not, is not.".

Yet, in order to properly define it we need to use the transcript of the text line w, represented as a sequence of words $w = w_{1:n} = w_1, \ldots, w_n$, as we did in the previous sections.

$$R \mid \boldsymbol{x}, v, k = R \mid v, k, w_{1:n} \stackrel{\text{def}}{=} \begin{cases} 1 & \exists k : w_k = v \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

In order to compute $P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\kappa = k)$, we marginalize over the variable W, representing the transcript of the text line $\boldsymbol{x}$, and we obtain:

$$\begin{aligned} P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\kappa = k) = \\ \sum_{w_{1:n} \in \Sigma^*} P(R = 1 \mid \boldsymbol{x}, v, k, w_{1:n}) P(w_{1:n} \mid \boldsymbol{x}, v, k) = \\ \sum_{w_{1:n} \in \Sigma^*} P(R = 1 \mid v, k, w_{1:n}) P(w_{1:n} \mid \boldsymbol{x}) = \\ \sum_{\substack{w_{1:n} \in \Sigma^*, \exists k: \\ w_k = v}} P(W = w_{1:n} \mid X = \boldsymbol{x}) \end{aligned} \tag{2.19}$$

In short, we need to compute the sum of the posterior probability of all transcripts of the text line $\boldsymbol{x}$, $w_{1:n}$ such that the query keyword, v, is present at the k-th position of the transcript.

As an illustration of the differences between the relevance probability of the segments, presented in section 2.2.2, and the one presented above, refer to table 2.2. This example shows some of the relevance probabilities that would be computed from fig. 2.4, using both eq. (2.16) and eq. (2.18), for the query keywords "that" and "is". In the case of the relevance probability for word positions, the expected value of the segment is also shown. Here, the differences are significant. However, it will be shown later that they are much smaller in practice. This is due to the fact that the statistical models used and the algorithms that adjust their parameters, tend to assign most of the probability mass to the most likely alignment.

Table 2.2. Example highlighting the differences between the relevance probabilities computed according to eq. (2.16) (left) and eq. (2.18) (right). Observe that the probabilities of the word segments are much sparse than that of the word positions. In addition, notice that one could not directly sum the probabilities of overlapping segments, since the possible intervals of different segments may overlap. The expected value of the interval is also shown for the word positions probabilities. Observe that all paths contribute to the pair ("is", 4) since all possible transcripts contain the word in that position.

Term	Segment	Prob.
that	0–31	0.35
that	0–33	0.5
that	31–62	0.47
that	33–62	0.53
is	62–78	0.7
is	62–81	0.3
is	78–100	0.7
is	81–100	0.3

Term	k	Prob.	$\mathbb{E}$[Segment]
that	1	0.85	0.0–32.2
that	2	0.95	32.1–62.0
that	3	0.05	32.2–62.0
is	3	0.95	62.0–78.9
is	4	1.0	78.1–98.9
is	5	0.05	78.9–100.0

2.3 Query-by-example paradigm

Traditionally, as we already mentioned in section 1.2.3, many researchers in the field focused on the query-by-example (QbE) paradigm, where

the query is not presented to the system as a given string, but as an example image. In the previous probabilistic formulation, however, we assumed that the keyword v was given as a discrete symbol in a vocabulary (or a sequence of discrete symbols in an alphabet).

Nevertheless, we can still derive a proper probabilistic formulation for query-by-example. The key step is to introduce an additional random variable, V, which represents the actual keyword written in the query image, $\boldsymbol{y}$. In the QbS case, the value of V is given, but in the QbE its value is hidden.

2.3.1 POSITION-INDEPENDENT KEYWORD SPOTTING FOR QBE

Next, we will derive a form for the relevance probability when a text region image, $\boldsymbol{x}$, and a query image, $\boldsymbol{y}$ are given. That is, we will derive an expression for $P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y})$. In particular, this expression will make use of marginalization over all possible values of the hidden random variable V.

$$P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}) = \sum_{v \in \Sigma^*} P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}, V = v) P(V = v \mid X = \boldsymbol{x}, Y = \boldsymbol{y}) \quad (2.20)$$

Now, we can safely assume that transcript (v) of the query image does not depend on the text image region $\boldsymbol{x}$, only on the query image itself, $\boldsymbol{y}$. Likewise, the relevance of a pair of images only depends on the transcript of these images, and not on the images, themselves. Thus, by marginalizing $P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}, V = v)$ among all possible transcripts w of the text region $\boldsymbol{x}$, we get:

$$P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}) = \sum_{v \in \Sigma^*} \sum_{w \in \Sigma^*} P(R=1 \mid V=v, W=w) P(W=w \mid X=\boldsymbol{x}) P(V=v \mid Y=\boldsymbol{y}) \quad (2.21)$$

As discussed earlier, observe that, applying the definition of relevance in eq. (2.3), $P(R = 1 \mid V = v, W = w) = 1$ for all the transcripts that include the query keyword v (i.e. $w \in L(v)$). Likewise, for all

the transcripts that *not* include the keyword this probability will be exactly zero. Thus, the relevance probability is equal to:

$$\begin{aligned} P(R=1 \mid X=\boldsymbol{x}, Y=\boldsymbol{y}) = \\ \sum_{v\in\Sigma^*} \sum_{w\in L(v)} P(W=w \mid X=\boldsymbol{x}) P(V=v \mid Y=\boldsymbol{y}) = \\ \sum_{v\in\Sigma^*} P(V=v \mid Y=\boldsymbol{y}) \sum_{w\in L(v)} P(W=w \mid X=\boldsymbol{x}) = \\ \sum_{v\in\Sigma^*} P(V=v \mid Y=\boldsymbol{y}) P(R=1 \mid X=\boldsymbol{x}, V=v) \end{aligned} \tag{2.22}$$

This equation implies that in order to compute the relevance probability in the query-by-example scenario (i.e. given a text image region $\boldsymbol{x}$ and a query image $\boldsymbol{y}$), we simply need to compute the sum of the query-by-string relevance probabilities, for all possible transcripts of the query image, weighted by their posteriors.

2.3.1.1 Word-segmented image regions

The previous equation takes an even simpler expression when both $\boldsymbol{x}$ and $\boldsymbol{y}$ contain a single word, which is the case for traditional word-segmented query-by-example KWS. Recall from eq. (2.8) that in a word-segmented *query-by-string* KWS scenario, the relevance probability is equivalent to the word posterior of the given query string. On the other hand, in the *query-by-example* scenario, we marginalize over all possible values of V:

$$P(R=1 \mid X=\boldsymbol{x}, Y=\boldsymbol{y}) \stackrel{\star}{=} \sum_{v\in\Sigma^*} P(V=v \mid Y=\boldsymbol{y}) P(W=v \mid X=\boldsymbol{x}) \tag{2.23}$$

This is because, since we assume that the images contain individual segmented words, $L(v)$ from eq. (2.22) can only contain one element: v (i.e. $L(v) = \{v\}$).

2.3.2 Position-dependent Keyword Spotting for QbE

The other relevance probabilities conditioned on some position that were presented in section 2.2 can also be (trivially) adapted to the

query-by-example scenario using the same marginalization approach. In summary, the position-dependent relevance probabilities under the query-by-example paradigm are:

- Relevance of an image column:

$$P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}, L_\tau = c) = \sum_{v \in \Sigma^*} P(V = v \mid Y = \boldsymbol{y}) \underbrace{P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c)}_{\text{(A)}} \tag{2.24}$$

- Relevance of an image segment:

$$P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}, L_\sigma = (c_0, c_1)) = \sum_{v \in \Sigma^*} P(V = v \mid Y = \boldsymbol{y}) \underbrace{P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\sigma = (c_0, c_1))}_{\text{(B)}} \tag{2.25}$$

- Relevance of a transcript position:

$$P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}, L_\kappa = k) = \sum_{v \in \Sigma^*} P(V = v \mid Y = \boldsymbol{y}) \underbrace{P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\kappa = k)}_{\text{(C)}} \tag{2.26}$$

Where (A), (B) and (C) are defined as in eqs. (2.13), (2.16) and (2.19), respectively.

2.4 Segmentation-free spotting using position-dependent relevance

So far we have assumed that the text region $\boldsymbol{x}$ encloses a segmented word, or more generally, a segmented text line. As we mentioned in the introduction, most publications in the field of KWS for handwritten documents actually fall in these two scenarios. However, one must not forget that our final goal is to be able to search through entire collections of documents with no manual segmentation of either the individual words or text line instances. Thus, the remaining question is: how can we address the problem of segmentation-free spotting?

Typically, two approaches are followed to address this: (a) one could use a sliding window approach with word-shaped or text line-shaped windows and then apply the previous methods in each of the windows, or (b) one could try to automatically detect and segment the interesting text regions from the page images.

Observe that the first approach, although more general, entails an enormous amount of computation, since the number of potential text regions to consider quickly explodes, particularly when considering large documents. Instead, we propose to use automatic (text line) segmentation of the pages, and then use our probabilistic methods in the segmented lines.

Most publications aiming at segmentation-free KWS want to obtain the bounding boxes of all the instances of a given keyword within the pages of our collection of documents. That is, the *retrieved objects* are individual word instances of the given query keyword. Notice that, once the text line has been segmented, the methods described in sections 2.2.2 and 2.2.3 can extract the location of the keyword instance(s) within the text line. With this information, and the location of the text line within the page image, it is trivial to obtain the bounding box of the spotted word within the page image. Actually, we have performed experiments using this approach (see section 8.8), in combination with our probabilistic framework.

Observe, that this is not the only possible retrieval goal. For instance, in [Villegas et al., 2016b], the goal was to retrieve *relevant segments* from a collection of documents, where a segment is a collection of 5 sequential text lines. Retrieving relevant segments was important because the queries could be formed by multiple keywords, similar to the popular Boolean or "phrase queries" in traditional Information Retrieval, and it is unusual that all words are written in the same text line. Nevertheless, we could still automatically segment the text lines and then compute the value of $P(R=1 \mid X=\boldsymbol{x}, V=v)$, where $\boldsymbol{x}$ represents the full segment.

Finally, it is worth mentioning that the relevance probability described in section 2.2.1 can be used to obtain a "heat-map" over the full text page. The idea is essentially the same as in fig. 2.3, but we obtain the text lines using a sliding window approach, which gives

visually appealing results as shown in fig. 2.5. This figure shows a two-dimensional heat-map over a page segment, for the query keyword "matter".

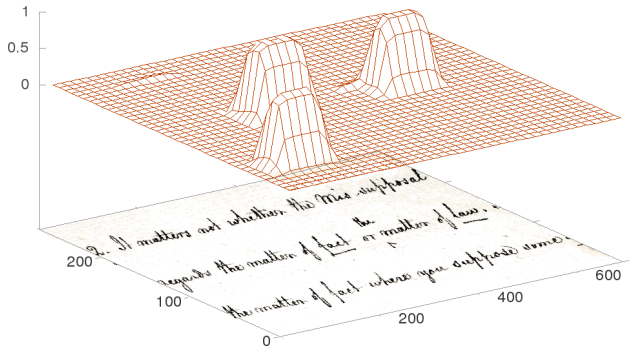


Figure 2.5. Relevance probability heat-map over a page fragment. The heat-map was computed using eq. (2.13) and a sliding window over the page image. Pixels where the keyword "matter" is written have a higher relevance probability than others. Observe that the model can distinguish properly similar words to the query (such as the plural "matters").

In short, the probabilistic framework presented in this chapter can be used to tackle different scenarios where Keyword Spotting is applied: from word-based segmentation to fully segmentation-free KWS, for both query-by-string and query-by-example paradigms.

2.5 Relationship among position-dependent and independent relevance

We have introduced several relevance probabilities concerning different types of objects that a KWS system may find useful to retrieve as results of a query.

1. $P(R = 1 \mid X = \boldsymbol{x}, V = v)$ could be used to retrieve any type of text image region, depending on what $\boldsymbol{x}$ represents: individual segmented words, text lines, paragraphs or full pages.

2. $P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c)$ could be used to retrieve individual columns within a text line image.

3. $P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\sigma = (c_0, c_1))$ could retrieve word segments within a text line image.

4. $P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\kappa = k)$ could retrieve word positions within a text transcript from an arbitrary text image region (either a text line, or a paragraph, as long as its transcript admits a sequential representation).

One might wonder what is the relationship between the different probabilities presented. Or, even more practical, how can we use one of them to approximate or bound another relevance probability.

For instance, suppose that we are designing a KWS system that has to retrieve relevant text lines given a query keyword. Then, the obvious choice would be to use $P(R=1 \mid X=\boldsymbol{x}, V=v)$, with $\boldsymbol{x}$ representing an individual text line. However, if we want to show the user *where* the keyword was spotted within the text line, we need to use a different probability to get, for instance, the segment with the highest relevance probability (i.e. $P(R=1 \mid X = \boldsymbol{x}, V = v, L_\sigma = (c_0, c_1))$). However, these two probabilities answer two different questions. But, can we use one to address the other with some guarantees?

In this section we will study the relationship between the relevance probabilities introduced in the previous section. We shall see that several upper and lower bounds can be established among them.

2.5.1 FRÉCHET INEQUALITIES

As we mentioned in section 2.1, $P(R=1 \mid X=\boldsymbol{x}, V=v)$ represents the relevance probability independently of any *position of interest*, while the rest of relevance probabilities are conditioned on different types of locations (columns, column intervals, or word positions).

Now, let's assume that we are dealing with an arbitrary enumerable type of location[2], represented by the random variable L. Then, we can interpret the position-independent relevance probability as a

[2]Notice that all the types of positions described in section 2.2 are actually enumerable and finite for a given image (although the set of possible locations may be very large).

disjunctive probability over all possible locations. That is, the position-independent probability is equal to the position-dependent probability of the first location, *or* the second, *or* the third location, etc.

Indeed, recall that the position-independent relevance definition was:

$$R \mid \boldsymbol{x}, v \stackrel{\text{def}}{=} \begin{cases} 1 & v \text{ is written } \textit{somewhere} \text{ in } \boldsymbol{x} \\ 0 & \text{otherwise} \end{cases}$$

which is equivalent to:

$$R \mid \boldsymbol{x}, v \stackrel{\text{def}}{=} \begin{cases} 1 & \begin{cases} v \text{ is written at position } l_1 \text{ of } \boldsymbol{x} \\ v \text{ is written at position } l_2 \text{ of } \boldsymbol{x} \\ \cdots \\ v \text{ is written at position } l_{n-1} \text{ of } \boldsymbol{x} \\ v \text{ is written at position } l_n \text{ of } \boldsymbol{x} \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Figure 2.6 shows a simplification of the generic position-independent and position-dependent KWS scenarios. In the figure, the big box represents the whole image region, $\boldsymbol{x}$, which contains three different locations l_1, l_2 and l_3. In each location, the possible transcript of each location is represented by the colored circles and their probability is represented by the fraction of circles of a given color. For instance, in the first location the probability of the word "blue" is equal to $\frac{1}{6}$.

For the sake of simplicity, in the example we will assume that the transcript of each location is independent of the others. Thus, the transcript of each location would be given by selecting a circle in each box. However, it is extremely important to realize that this independence assumptions is obviously false in real scenarios. For instance, when the positions represent columns of an image, the word aligned to neighbor columns is obviously not independent. Likewise, when the positions represent segments of the image, the text aligned to overlapping and contiguous segments is also not independent.

Now, back to fig. 2.6, we might wonder what is the relevance probability of the word "blue" in a specific location. For instance, $P(R=1 \mid X=\boldsymbol{x}, V=\text{"blue"}, L=l_1) = P(W=\text{"blue"} \mid X=\boldsymbol{x}, L=l_1) = \frac{1}{6}$.

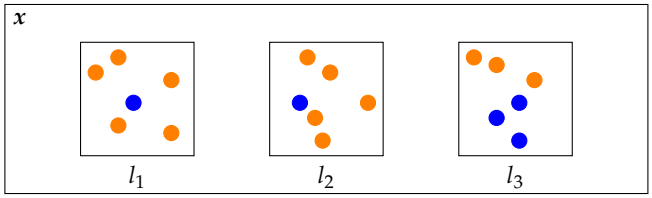


Figure 2.6. Example of the relationship between position-independent and position-dependent relevance probabilities. A text region is represented by the biggest box, $\boldsymbol{x}$, and several positions within the region are presented by the smallest boxes (i.e. l_1, l_2, and l_3). The circles inside each box represent the possible transcripts of the corresponding position. Thus, the probability that the first position is relevant for the query "blue" is equal to $P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"},L=l_1)=\frac{1}{6}$. However, the probability that the whole region is relevant for the query is equal to $P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"})=\frac{47}{72}\approx 0.7$. The Fréchet inequalities tell us that $\frac{3}{6}=0.5\leq P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"})\leq\frac{5}{6}\approx 0.8$.

Likewise, the probabilities $P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"},L=l_2)=\frac{1}{6}$ and $P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"},L=l_3)=\frac{3}{6}$.

On the other hand, according to our position-independent relevance, $P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"})$ is the probability that the transcript of at least one box is equal to "blue". This is equal to one minus the probability that *none* of the boxes' transcripts is equal to "blue". Thus, $P(R=1\,|\,X=\boldsymbol{x},V=\text{"blue"})=1-\frac{5}{6}\cdot\frac{5}{6}\cdot\frac{3}{6}=\frac{47}{72}\approx 0.7$.

In our particular example, the position-independent relevance probability can be computed easily from the position-dependent ones, because we assumed that the transcript of each position was independent from the others. When this assumption does not hold, one can use the *Fréchet inequalities of the logical disjunction* [Fréchet, 1935] to find bounds for the position-independent relevance.

In general, given a set of logical propositions $\{A_i : 1\leq i\leq n\}$ (e.g. the i-th position of $\boldsymbol{x}$ is relevant for the keyword v), the Fréchet inequalities [Fréchet, 1935] are:

- Inequalities of the logical disjunction:

$$\max_i P(A_i)\leq P(A_i\vee\cdots\vee A_n)\leq\min\{1,\sum_i P(A_i)\} \qquad (2.27)$$

- Inequalities of the logical conjunction:

$$\max\{0, \sum_i P(A_i) - (n-1)\} \leq P(A_i \wedge \cdots \wedge A_n) \leq \min_i P(A_i) \tag{2.28}$$

In our particular scenario, for any position-dependent and position-independent probability distributions, eq. (2.27) implies that:

$$\begin{aligned} \max_l P(R = 1 \mid X = \boldsymbol{x}, V = v, L = l) \leq \\ P(R = 1 \mid X = \boldsymbol{x}, V = v) \leq \\ \min\left\{1, \sum_l P(R = 1 \mid X = \boldsymbol{x}, V = v, L = l)\right\} \end{aligned} \tag{2.29}$$

In fact, the lower bound on the relevance probability has already been used in previous Keyword Spotting systems, as an heuristic to compute relevance *scores* for text lines [Toselli et al., 2013, Toselli et al., 2016b, Toselli et al., 2016a].

Figure 2.7 shows the relationship among the different relevance probabilities introduced earlier. Fréchet inequalities give different lower bounds for $P(R = 1 | X = \boldsymbol{x}, V = v)$. In addition, theorem 2.1 sketches a proof for the relationship between the column and the segment probabilities.

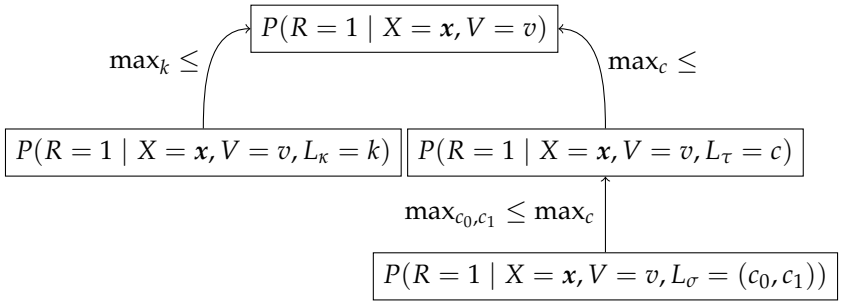


Figure 2.7. Diagram of the relationship among the different relevance probabilities for a fixed text image $\boldsymbol{x}$ and keyword v. The inequalities should be read in the direction of the arrows, e.g. $\max_k P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\kappa = k) \leq P(R = 1 \mid X = \boldsymbol{x}, V = v)$.

Theorem 2.1. *For a given text image $\boldsymbol{x}$ and keyword v, the highest relevance probability among all segments is lower than or equal to the highest relevance probability among all columns.*

$$\max_{c_0,c_1} P(R=1 \,|\, X=\boldsymbol{x}, V=v, L_\sigma=(c_0,c_1)) \leq \max_{c} P(R=1 \,|\, X=\boldsymbol{x}, V=v, L_\tau=c)$$

Proof sketch. Let's consider the segment (c_0, c_1) with the highest relevance probability for a fixed text image $\boldsymbol{x}$ and query keyword v.

According to eq. (2.16), $P(R=1 \,|\, X=\boldsymbol{x}, V=v, L_\sigma=(c_0,c_1))$ is equal to the sum of the probability of all possible transcripts of $\boldsymbol{x}$ such that the keyword v starts at column c_0 and ends at c_1.

Similarly, $P(R=1 \,|\, X=\boldsymbol{x}, V=v, L_\tau=c)$ is equal to the sum of the probability of all possible transcripts of $\boldsymbol{x}$ where the keyword v is written in a segment containing the column c (see eq. (2.13)).

Now, suppose that a segment (c_0', c_1') overlaps with (c_0, c_1), where the keyword v is also written. For all columns in the overlapping region , their relevance probability will be greater than that of the individual segments (see eq. (2.13)). And, obviously, their relevance probability is lower than or equal to the maximum among all columns. □

3 The Probability Ranking Principle

3.1 Ranking multiple relevant images

In the previous chapter we introduced different relevance probabilities that we used to decide whether or not a given image region (or a position within an image region), is relevant for the given query. As we already saw in section 1.5, these probabilities allow a KWS system to make the optimal decision under some binary classification loss.

In reality, the actual loss function depends on the user querying the system. Thus, a given system should be able to perform well under multiple of these losses. Thankfully, as we saw in section 1.5, a binary classification loss can be expressed simply as a threshold value, that the user would set according to her preferences.

This means that evaluating the performance of a given KWS under multiple binary classification losses, is equivalent to evaluating the quality of the ranking that such system produces for a given list of documents and queries. In fact, this is how users actually interact with traditional Information Retrieval systems, such as web search engines.

Thus, we need to decide what is the optimal strategy to rank the set of documents for a given query (or set of queries), so that some criterion is optimized (maximized, or minimized).

A very well known strategy in the field of Information Retrieval to tackle this problem is the so-called *Probability Ranking Principle*, originally formulated by William S. Cooper, and published in [Robertson, 1977]:

> If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.

In the following section, we shall see that this principle is actually optimal for a wide range of quality measures as long as the following assumptions are met:

1. The relevance of a given object and query is independent of other objects and other queries. That is, the joint distribution of conditional relevance probabilities factorizes in:

$$P(R_1, \ldots, R_N \mid X_1, \ldots, X_N, V_1, \ldots, V_N) = \prod_{i=1}^{N} P(R_i \mid X_i, V_i)$$

2. The *true* relevance probability is known for all objects and queries in the evaluation set. That is, given an evaluation set $\{(r_i, x_i, v_i) : 1 \leq i \leq N\}$, then:

$$r_i \mid x_i, \boldsymbol{v}_i \sim P(R_i \mid X = x_i, V = \boldsymbol{v}_i)$$

Notice that position-dependent KWS and other IR instances, where the relevance is conditioned on more variables, are also represented by the previous assumptions, if one considers the random variable V to be defined over the Cartesian product of all the underlying variables.

Luckily, the first assumption holds in virtually all the existing Keyword Spotting benchmarks. Yet, it may be false in other IR applications. The second assumption, however, is never correct, since the *true* distributions are unknown, and have to be estimated from data.

During the development of this chapter we will use the term "retrieved objects" to emphasize that our results apply to any IR system where the aforementioned assumptions hold.

3.2 Evaluation measures and optimality

When stating that a certain strategy (i.e. decision rule) is optimal, one needs to state with respect to which criterion that optimality holds. And, of course, then one has to *prove* it. However, all papers that we could find that refer to the "optimality" of the PRP cite the original publication [Robertson, 1977]. Yet, this work only proves that the PRP is optimal under the binary classification scenario (0–1 loss) presented in section 1.5, and for two of the most basic retrieval quality measures: the Precision-at-k and Recall-at-k.

In this section, several popular measures to evaluate the quality of a ranking system are introduced, and we prove that the PRP is the optimal strategy with respect to all of them. More specifically, we present alternative proofs regarding the Precision-at-k and Recall-at-k, and, more importantly, we present proofs of the optimality with the very popular Average Precision and Discounted Cumulative Gain (for both unnormalized and normalized versions).

In all cases, the proofs show that the PRP *maximizes the expected value* of the corresponding measure. Following the principles of Decision Theory, we need to optimize this value since the *true* relevance of each retrieved object for the respective query is unknown.

Given an ordered list of N elements, we will denote the relevance of the element at position i with the random variable R_i. Thus, the sequence of random variables for the N elements is $R_1, \ldots, R_N$.

Note that each element refers to some retrieved object for some particular query. Thus, in the context of KWS, the relevance probability of the i-th element in the list would be conditioned on the corresponding image, keyword (and position within the image, in position-dependent scenarios). Despite that, we will drop the conditioned variables from the notation, for a clearer presentation.

The procedure to complete all proofs is almost identical: first, we will derive an expression for the expected value of the corresponding quality measure, and then we will show that such expected value can increase as long as the ranked list is not ordered in decreasing order of the relevance probability.

3.2.1 PRECISION-AT-k

Definition 3.1. Precision-at-k of a ranked list, sometimes abbreviated as P@k, measures the fraction of the k-top retrieved elements that are actually relevant.

$$\pi_k(R_1, \ldots, R_N) \stackrel{\text{def}}{=} \frac{1}{k} \sum_{i=1}^{k} R_i \tag{3.1}$$

One can easily compute the expected value of the P@k, taking into account the independence assumption on the relevance probabilities, i.e. $P(R_1, \ldots, R_N) = \prod_i P(R_i)$:

$$\mathbb{E}\left[\pi_k(R_1, \ldots, R_N)\right] \stackrel{\star}{=} \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}\left[R_i\right] = \frac{1}{k} \sum_{i=1}^{k} P(R_i = 1) \tag{3.2}$$

Theorem 3.1. *A sequence of elements ordered by decreasing relevance probability (i.e. $P(R_i = 1) \geq P(R_j = 1), 1 \leq i < j \leq N$) maximizes the expected value of the Precision-at-k, for any cut-off $1 \leq k \leq N$.*

Proof. Notice that for any given k, any set of k results with the largest sum of $P(R_i = 1)$ maximizes eq. (3.2). Thus, by sorting the N results according to its decreasing relevance probability, the k-top results maximize eq. (3.2) for any possible value of k. □

3.2.2 RECALL-AT-k

Definition 3.2. Recall-at-k of a ranked list, sometimes abbreviated as R@k, measures the fraction of all relevant elements that are found in the k-top of the retrieved list.

$$\rho_k(R_1, \ldots, R_N) \stackrel{\text{def}}{=} \begin{cases} 0 & T(R_1, \ldots, R_N) = 0 \\ \frac{1}{T(R_1, \ldots, R_N)} \sum_{i=1}^{k} R_i & \text{otherwise} \end{cases} \tag{3.3}$$

where $T(R_1, \ldots, R_N)$ is the total number of relevant elements in the sequence $R_1, \ldots, R_N$.

$$T(R_1, \ldots, R_N) \stackrel{\text{def}}{=} \sum_{i=1}^{N} R_i \tag{3.4}$$

In the previous definition, we chose to express T as a function of $R_1, \ldots, R_N$ to highlight the fact that T is also a random variable whose value depends on those. For the sake of clarity, we will just write T to refer to $T(R_1, \ldots, R_N)$. and T_k to refer to the number of relevant elements among the top-k of the retrieved list. Following this notation, an alternative definition of Recall-at-k and Precision-at-k would be:

$$\rho_k \stackrel{\text{def}}{=} \begin{cases} 0 & T = 0 \\ \frac{T_k}{T} & \text{otherwise} \end{cases} \quad ; \quad \pi_k \stackrel{\text{def}}{=} \frac{T_k}{k} \quad ; \quad T_k \stackrel{\text{def}}{=} \sum_{i=1}^{k} R_i \tag{3.5}$$

The calculation of the expected value of the Recall-at-k is slightly more tedious than for the Precision-at-k. The reason is that the denominator does not factorize with respect to the expectation, since it is not a constant in this case. However, we can use the law of total expectation to marginalize the expected value into a sum of conditional expectations.

$$\mathbb{E}\left[\rho_k(R_1, \ldots, R_N)\right] = \sum_{t=0}^{N} P(T = t)\, \mathbb{E}\left[\rho_k(R_1, \ldots, R_N) \mid T = t\right] \tag{3.6}$$

Now, the value of T is constant within each conditional expectation. Also, by definition, when $T = 0$, the Recall-at-k is always 0 (hence, it does not depend on the ranking). Finally, using the linearity of the conditional expectation, the previous equality can be rewritten as:

$$\begin{aligned} \mathbb{E}\left[\rho_k(R_1, \ldots, R_N)\right] &= \sum_{t=1}^{N} \frac{P(T = t)}{t}\, \mathbb{E}\left[\sum_{i=1}^{k} R_i \mid T = t\right] = \\ &\sum_{t=1}^{N} \frac{P(T = t)}{t} \sum_{i=1}^{k} \mathbb{E}[R_i \mid T = t] = \\ &\sum_{t=1}^{N} \frac{P(T = t)}{t} T_k(R_1, \ldots, R_N \mid T = t) \end{aligned} \tag{3.7}$$

With $T_k(R_1, \ldots, R_N \mid T = t)$ defined as the expected number of relevant elements within the top-k positions when the total number of

relevant events is t.

$$T_k(R_1, \ldots, R_N \mid T = t) \stackrel{\text{def}}{=} \sum_{i=1}^{k} \mathbb{E}[R_i \mid T = t] \tag{3.8}$$

When the sequence of relevant events can be inferred from the context, we will simply write it as $T_k(t)$. It is important to emphasize that this quantity is different from T_k, which is not an expected value, but a random variable.

Lemma 3.2. *Given a ranked list with relevance variables $R_1, \ldots, R_N$, and a cut-off position k, if we swap any pair of elements at positions l and m, for any value of $1 \leq l \leq k$ and $k < m \leq N$, the difference in $T_k(t)$ is equal to:*

$$P(R_l = 1) - P(R_m = 1)$$

For any value of $1 \leq k \leq N$ and $1 \leq t \leq N$.

Proof. Let's suppose that the relevance variables of the original and the swapped sequences are, respectively, $R_1, \ldots, R_N$ and $R'_1, \ldots, R'_N$, such that $R_l = R'_m$, $R_m = R'_l$ and $R_i = R'_i$, $\forall i \notin \{l, m\}$. We will use $\overline{T_k}$ and $\overline{T_k}'$ to denote the expected value of $T_k(t)$ and $T'_k(t)$, respectively.

Thus, for the first sequence we have:

$$\overline{T_k} = T_k(R_1, \ldots, R_N \mid T = t) = \sum_{i=1}^{k} \mathbb{E}[R_i \mid T = t] =$$
$$\sum_{i=1}^{l-1} \mathbb{E}[R_i \mid T = t] + \mathbb{E}[R_l \mid T = t] + \sum_{i=l+1}^{k} \mathbb{E}[R_i \mid T = t] \tag{3.9}$$

Taking into account that the two sequences only differ at positions l and m, and that $T = T'$ (i.e. the total number of relevant elements is identical), we have the following expression for the second sequence:

$$\overline{T_k}' = T_k(R'_1, \ldots, R'_N \mid T' = t) =$$
$$\sum_{i=1}^{l-1} \mathbb{E}[R'_i \mid T' = t] + \mathbb{E}[R'_l \mid T' = t] + \sum_{i=l+1}^{k} \mathbb{E}[R'_i \mid T' = t] =$$
$$\sum_{i=1}^{l-1} \mathbb{E}[R_i \mid T = t] + \mathbb{E}[R_m \mid T = t] + \sum_{i=l+1}^{k} \mathbb{E}[R_i \mid T = t] \tag{3.10}$$

Note that the expressions for $\overline{T_k}$ and $\overline{T_k}'$ are almost identical, except for the l-th position. The difference of the two is:

$$\begin{aligned}
\overline{T_k} - \overline{T_k}' = \\
\mathbb{E}[R_l \mid T = t] - \mathbb{E}[R_m \mid T = t] = \\
P(R_l = 1)\,\mathbb{E}[1 \mid T = t, R_l = 1] - P(R_m = 1)\,\mathbb{E}[1 \mid T = t, R_m = 1] \stackrel{\star}{=} \qquad (3.11) \\
P(R_l = 1) - P(R_m = 1) \qquad (3.12)
\end{aligned}$$

The equality in eq. (3.11) holds under the assumption that the relevance probability of an object and query is independent of the others. □

Theorem 3.3. *A sequence of elements ordered by decreasing relevance probability (i.e. $P(R_i = 1) \geq P(R_j = 1), 1 \leq i < j \leq N$) maximizes the expected value of the Recall-at-k, for any cut-off $1 \leq k \leq N$.*

Proof. Let's suppose that the relevance variables of the original sequence are $R_1, \ldots, R_N$, and those of the modified sequence are $R'_1, \ldots, R'_N$, such that $R_l = R'_m$, $R_m = R'_l$ and $R_i = R'_i, \forall i \notin \{l, m\}$.

Following eq. (3.7), the expected values of the R@k, for both sequences, are:

$$\overline{\rho_k} = \sum_{t=1}^{N} \frac{P(T = t)}{t} T_k(R_1, \ldots, R_N \mid T = t) \qquad (3.13)$$

$$\overline{\rho_k}' = \sum_{t=1}^{N} \frac{P(T' = t)}{t} T_k(R'_1, \ldots, R'_N \mid T' = t) \qquad (3.14)$$

Taking into account that $T = T'$ and lemma 3.2, the difference between the two is equal to:

$$\overline{\rho_k} - \overline{\rho_k}' = \sum_{t=1}^{N} \frac{P(T=t)}{t}(T_k(t) - T'_k(t)) = (P(R_l = 1) - P(R_m = 1)) \sum_{t=1}^{N} \frac{P(T=t)}{t} \tag{3.15}$$

Observe that the sum $\sum_{t=1}^{N} \frac{P(T=t)}{t}$ is equal to zero if, and only if, $P(T = 0) = 1$ (i.e. it is impossible that any retrieved element is relevant). In such case, $P(R_l = 1) = P(R_m = 1) = 0$. In any other case, the previous sum is strictly positive and it does not affect the sign of the difference in the expected value of the R@k. Thus:

$$\overline{\rho_k} - \overline{\rho_k} \begin{cases} > 0 & P(R_l = 1) > P(R_m = 1) \\ = 0 & P(R_l = 1) = P(R_m = 1) \\ < 0 & P(R_l = 1) < P(R_m = 1) \end{cases} \tag{3.16}$$

Equation (3.16) implies that, for any sequence of results, if we swap an arbitrary element among the top-k, with any other element not in the top-k, we will improve the expected value of the R@k if, and only if, the relevance probability of first one is strictly lower than that of the latter.

Thus, if we sort the sequence of N results by their decreasing relevance probability, we guarantee that the expected value of R@k is the maximum for any $1 \leq k \leq N$. □

3.2.3 AVERAGE PRECISION

In general, the Average Precision (AP) is defined as the area under the Recall–Precision curve.

$$\mathrm{AP} \stackrel{\mathrm{def}}{=} \int_0^1 \pi(\rho) d\rho \tag{3.17}$$

However, this definition cannot be applied in practice since the precision is not a function of the recall, and, most importantly, neither

the recall nor precision are continuous. In practice, researchers use a different definition of the AP.

Definition 3.3. The Average Precision (AP) of a ranked list of elements with corresponding relevance variables $R_1, \ldots, R_N$ is defined as:

$$\mathrm{AP}(R_1, \ldots, R_N) \stackrel{\text{def}}{=} \sum_{i=1}^{N} \pi_i \Delta\rho_i = \begin{cases} 0 & T = 0 \\ \frac{1}{T} \sum_{i=1}^{N} \frac{R_i}{i} \sum_{j=1}^{i} R_j & \text{otherwise} \end{cases} \tag{3.18}$$

Here, $\Delta\rho_i = \rho_i - \rho_{i-1}$ (see eq. (3.5)). The special case of $T = 0$ is due to the fact that $\frac{1}{T}$ is not defined.

Next, we calculate an expression for the expected value of the AP. Because T depends on the relevance variables, we need to use the conditional expectation in order to extract it from the expectation operator, as we did in section 3.2.2.

$$\begin{aligned} &\mathbb{E}\left[\mathrm{AP}(R_1, \ldots, R_N)\right] = \\ &\sum_{t=0}^{N} P(T = t)\, \mathbb{E}\left[\mathrm{AP}(R_1, \ldots, R_N) \mid T = t\right] = \\ &\sum_{t=1}^{N} \frac{P(T = t)}{t}\, \mathbb{E}\left[\sum_{i=1}^{N} \frac{R_i}{i} \sum_{j=1}^{i} R_j \mid T = t\right] = && (3.19) \\ &\sum_{t=1}^{N} \frac{P(T = t)}{t} \sum_{i=1}^{N} \frac{\mathbb{E}[R_i T_i \mid T = t]}{i} = && (3.20) \\ &\sum_{t=1}^{N} \frac{P(T = t)}{t} S(R_1, \ldots, R_N \mid T = t) && (3.21) \end{aligned}$$

with

$$S(R_1, \ldots, R_N \mid T = t) \stackrel{\text{def}}{=} \sum_{i=1}^{N} \frac{\mathbb{E}[R_i T_i \mid T = t]}{i} \tag{3.22}$$

Notice that the equality between eqs. (3.19) and (3.20) is due to the definition of T_i (see eq. (3.5)) and the linearity of the expectation. For the sake of clarity, when the set of relevant variables $R_1, \ldots, R_N$ can be inferred from the context, we will use $S(t)$ as an alternative to $S(R_1, \ldots, R_N \mid T = t)$.

In addition, we can express $S(t)$ with respect to an auxiliary value $S_{\not{k},\not{k+1}}(t)$, which is equal to the former sum excluding the indexes at positions k and $k+1$, by simply using basic properties of the addition operation:

$$\begin{aligned} S(t) = S(R_1, \ldots, R_N \mid T = t) = \\ \sum_{i=1}^{k-1} \frac{\mathbb{E}[R_i T_i \mid T = t]}{i} + \sum_{i=k+2}^{N} \frac{\mathbb{E}[R_i T_i \mid T = t]}{i} + \\ \frac{\mathbb{E}[R_k T_k \mid T = t]}{k} + \frac{\mathbb{E}[R_{k+1} T_{k+1} \mid T = t]}{k+1} = \\ S_{\not{k},\not{k+1}}(t) + \frac{\mathbb{E}[R_k T_k \mid T = t]}{k} + \frac{\mathbb{E}[R_{k+1} T_{k+1} \mid T = t]}{k+1} \end{aligned} \tag{3.23}$$

Definition 3.4. The random variable denoting the total number of relevant elements, excluding the elements at positions i and j, is represented by $T_{\not{i},\not{j}}$. It's value is given by the expression:

$$T_{\not{i},\not{j}} \stackrel{\text{def}}{=} \sum_{\substack{1 \leq k \leq N: \\ k \notin \{i,j\}}} R_k = T - R_i - R_j$$

For any pair $1 \leq i, j \leq N$ such that $i \neq j$.

Lemma 3.4. *Given a ranked list with relevance variables $R_1, \ldots, R_N$, if we swap any pair of elements at positions k and $k+1$, for any value of $1 \leq k < N$, the difference in the values of the sum $S(t)$ is equal to:*

$$\frac{P(R_k = 1) - P(R_{k+1} = 1)}{k(k+1)} \mathbb{E}[T_{k-1} + 1 \mid T_{\not{k},\not{k+1}} = t - 1]$$

For any value of $1 \leq t \leq N$.

Proof. Let's consider two sequences of relevance variables $R_1, \ldots, R_N$ and $R'_1, \ldots, R'_N$, such that $R_k = R'_{k+1}$, $R_{k+1} = R'_k$, and $R_i = R'_i$, $\forall i \notin \{k, k+1\}$ (that is, the two sequences are equal except that elements at positions k and $k+1$ have been swapped). We will use S and S' to denote the value of the sum $S(t)$ for the first and the second lists, respectively.

Thus, for the first sequence of relevant variables we have:

$$\begin{aligned} S = S(R_1, \ldots, R_N \mid T = t) = \\ S_{\cancel{k,k+1}}(t) + \frac{\mathbb{E}[R_k T_k \mid T = t]}{k} + \frac{\mathbb{E}[R_{k+1} T_{k+1} \mid T = t]}{k+1} = \\ S_{\cancel{k,k+1}}(t) + \frac{\mathbb{E}[R_k (T_{k-1} + R_k) \mid T = t]}{k} + \frac{\mathbb{E}[R_{k+1} T_{k+1} \mid T = t]}{k+1} \end{aligned} \tag{3.24}$$

And for the second sequence, we have:

$$\begin{aligned} S' = S(R'_1, \ldots, R'_N \mid T' = t) = \\ S'_{\cancel{k,k+1}}(t) + \frac{\mathbb{E}[R'_k (T'_{k-1} + R'_k) \mid T' = t]}{k} + \frac{\mathbb{E}[R'_{k+1} T_{k+1} \mid T' = t]}{k+1} = \\ S_{\cancel{k,k+1}}(t) + \frac{\mathbb{E}[R_{k+1} (T_{k-1} + R_{k+1}) \mid T = t]}{k} + \frac{\mathbb{E}[R_k T_{k+1} \mid T = t]}{k+1} \end{aligned} \tag{3.25}$$

The previous equations use the fact that $S_{\cancel{k,k+1}}(t) = S'_{\cancel{k,k+1}}(t)$ and $T_{k+1} = T'_{k+1}$, since these values do not change when we swap the elements at positions k and $k+1$ (review eqs. (3.5) and (3.23)).

Then, the difference $S - S'$ is equal to:

$$\begin{aligned} S - S' = \\ \frac{\mathbb{E}[R_k (T_{k-1} + R_k) \mid T = t]}{k} + \frac{\mathbb{E}[R_{k+1} T_{k+1} \mid T = t]}{k+1} - \\ \frac{\mathbb{E}[R_{k+1} (T_{k-1} + R_{k+1}) \mid T = t]}{k} + \frac{\mathbb{E}[R_k T_{k+1} \mid T = t]}{k+1} = \\ \frac{\mathbb{E}[R_k (T_{k-1} + R_k) - R_{k+1} (T_{k-1} + R_{k+1}) \mid T = t]}{k} + \\ \frac{\mathbb{E}[R_{k+1} T_{k+1} - R_k T_{k+1} \mid T = t]}{k+1} = \\ \frac{\mathbb{E}[(R_k - R_{k+1}) T_{k-1} + R_k R_k - R_{k+1} R_{k+1} \mid T = t]}{k} + \qquad (3.26) \\ \frac{\mathbb{E}[(R_{k+1} - R_k) T_{k+1} \mid T = t]}{k+1} \qquad (3.27) \end{aligned}$$

Now, let's focus on the value of each expectation in the previous equation. First, notice that if $R_k = R_{k+1}$ the value of all expectations

is canceled, and thus the difference $S - S'$ is equal to zero. Using the fact that the relevance of an element is independent of the others, definition 3.4 and basic properties of probabilities, the expectation in eq. (3.26) is equal to:

$$\begin{aligned}
&\mathbb{E}[(R_k - R_{k+1})T_{k-1} + R_k R_k - R_{k+1}R_{k+1} \mid T = t] = \\
&P(R_k = 0)P(R_{k+1} = 1)\,\mathbb{E}[-T_{k-1} - 1 \mid T = t, R_k = 0, R_{k+1} = 1] + \\
&P(R_k = 1)P(R_{k+1} = 0)\,\mathbb{E}[T_{k-1} + 1 \mid T = t, R_k = 1, R_{k+1} = 0] = \\
&-P(R_k = 0)P(R_{k+1} = 1)\,\mathbb{E}[T_{k-1} + 1 \mid T = t, R_k = 0, R_{k+1} = 1] + \\
&P(R_k = 1)P(R_{k+1} = 0)\,\mathbb{E}[T_{k-1} + 1 \mid T = t, R_k = 1, R_{k+1} = 0] = \\
&-P(R_k = 0)P(R_{k+1} = 1)\,\mathbb{E}[T_{k-1} + 1 \mid T_{\cancel{k,k+1}} = t - 1] + \\
&P(R_k = 1)P(R_{k+1} = 0)\,\mathbb{E}[T_{k-1} + 1 \mid T_{\cancel{k,k+1}} = t - 1] = \\
&(P(R_k = 1) - P(R_{k+1} = 1))\,\mathbb{E}[T_{k-1} + 1 \mid T_{\cancel{k,k+1}} = t - 1]
\end{aligned} \tag{3.28}$$

The last step is due to the fact that, for binary random variables, $P(A = 1)P(B = 0) - P(A = 0)P(B = 1) = P(A = 1) - P(B = 1)$, which can be easily derived given that $P(A = 0) = 1 - P(A = 1)$, and respectively for the random variable B.

Similarly, for the second expectation we obtain:

$$\begin{aligned}
&\mathbb{E}[(R_{k+1} - R_k)T_{k+1} \mid T = t] = \\
&P(R_k = 0)P(R_{k+1} = 1)\,\mathbb{E}[T_{k-1} + 1 \mid T = t, R_k = 0, R_{k+1} = 1] + \\
&P(R_k = 1)P(R_{k+1} = 0)\,\mathbb{E}[-T_{k-1} - 1 \mid T = t, R_k = 1, R_{k+1} = 1] = \\
&(P(R_{k+1} = 1) - P(R_k = 1))\,\mathbb{E}[T_{k-1} + 1 \mid T_{\cancel{k,k+1}} = t - 1]
\end{aligned} \tag{3.29}$$

Note that the value of $\mathbb{E}[(R_{k+1} - R_k)T_{k+1} \mid T = t]$ is the same as that of $\mathbb{E}[(R_k - R_{k+1})T_{k-1} + R_k R_k - R_{k+1}R_{k+1} \mid T = t]$ with the opposite sign.

Finally, replacing eqs. (3.28) and (3.29) into eqs. (3.26) and (3.27), respectively, and after a few basic algebra operations to simplify the expressions, the result is:

$$S - S' = \frac{P(R_k = 1) - P(R_{k+1} = 1)}{k(k+1)}\,\mathbb{E}[T_{k-1} + 1 \mid T_{\cancel{k,k+1}} = t - 1] \tag{3.30}$$

□

Theorem 3.5. *A sequence of retrieved elements ordered by decreasing relevance probability (i.e. $P(R_i = 1) \geq P(R_j = 1), 1 \leq i < j \leq N$) maximizes the expected value of the Average Precision.*

Proof. Let's consider two sequences of relevance variables $R_1, \ldots, R_N$ and $R'_1, \ldots, R'_N$, and an arbitrary position, $1 \leq k < N$, such that $R_k = R'_{k+1}$, $R_{k+1} = R'_k$, and $R_i = R'_i$, $\forall i \notin \{k, k+1\}$ (that is, the two sequences are equal except that elements at positions k and $k+1$ have been swapped).

Following eqs. (3.22) and (3.23), the expected value of the AP for the first and the second sequences are, respectively:

$$\overline{AP} = \sum_{t=1}^{N} \frac{P(T=t)}{t} S(t) \;\; ; \;\; \overline{AP}' = \sum_{t=1}^{N} \frac{P(T'=t)}{t} S'(t) \tag{3.31}$$

where $S(t) = S(R_1, \ldots, R_N \mid T = t)$ and $S'(t) = S(R'_1, \ldots, R'_N \mid T' = t)$.

Using the fact that $T = T'$ (since the two sequences contain the same elements, just with a different order the total number of relevant elements is the same in both cases), he difference of the two expected values results in:

$$\overline{AP} - \overline{AP}' = \sum_{t=1}^{T} \frac{P(T=t)}{t} (S(t) - S'(t)) \tag{3.32}$$

Taking into account lemma 3.4, the previous equation is equivalent to:

$$\overline{AP} - \overline{AP}' = \\ (P(R_k = 1) - P(R_{k+1} = 1)) \sum_{t=1}^{T} \frac{P(T=t)\, \mathbb{E}[T_{k-1} + 1 \mid T_{\cancel{k,k+1}} = t-1]}{tk(k+1)} \tag{3.33}$$

Observe that the sum is strictly greater than 0, since the expectation includes a $+1$ and the rest of variables have values ≥ 0. Thus, its

value does not alter the sign of the difference $\overline{AP} - \overline{AP}'$, which results in:

$$\overline{AP} - \overline{AP}' = \begin{cases} > 0 & P(R_k = 1) > P(R_{k+1} = 1) \\ = 0 & P(R_k = 1) = P(R_{k+1} = 1) \\ < 0 & P(R_k = 1) < P(R_{k+1} = 1) \end{cases} \tag{3.34}$$

Equation (3.34) implies that, for any sequence of results, if we swap any element at position k with its next neighbor (position $k+1$), the expected value of the AP will increase if, and only if, the relevance probability of k is lower than that of $k+1$. Thus, if we sort the sequence of N results by their decreasing relevance probability, we guarantee that the expected value of the AP is the maximum. □

3.2.4 DISCOUNTED CUMULATIVE GAIN

Definition 3.5. The Discounted Cumulative Gain (DCG) of a sequence of elements with relevance $R_i \in \{0, 1\}$ is defined as:

$$\mathrm{DCG}(R_1, \ldots, R_N) \stackrel{\text{def}}{=} \sum_{i=1}^{N} \frac{2^{R_i} - 1}{\log_2(i+1)} \tag{3.35}$$

Now, let's compute the expected value of the DCG with respect to $P(R_i = 1)$. Again, the following expressions assume that the relevance of the elements are independent.

$$\begin{aligned} \mathbb{E}\left[\mathrm{DCG}(R_1, \ldots, R_N)\right] = \\ \sum_{i=1}^{N} \mathbb{E}\left[\frac{2^{R_i} - 1}{\log_2(i+1)}\right] = \\ \sum_{i=1}^{N} \frac{\mathbb{E}[2^{R_i}] - 1}{\log_2(i+1)} = \\ \sum_{i=1}^{N} \frac{2^1 P(R_i = 1) + 2^0 P(R_i = 0) - 1}{\log_2(i+1)} = \\ \sum_{i=1}^{N} \frac{P(R_i = 1)}{\log_2(i+1)} \end{aligned} \tag{3.36}$$

Theorem 3.6. *A sequence of elements ordered by decreasing relevance probability (i.e. $P(R_i = 1) \geq P(R_j = 1), 1 \leq i < j \leq N$) maximizes the expected value of the Discounted Cumulative Gain.*

Proof. Let's consider two sequences of relevance variables $R_1, \ldots, R_N$ and $R'_1, \ldots, R'_N$, such that $R_k = R'_{k+1}$, $R_{k+1} = R'_k$, and $R_i = R'_i$, $\forall i \notin \{k, k+1\}$ (that is, the two sequences are equal except that elements at positions k and $k+1$ have been swapped). The expected value of the DCG for each sequence of variables is:

$$\overline{DCG} = \sum_{i=1}^{N} \frac{P(R_i = 1)}{\log_2(i+1)} \;\; ; \;\; \overline{DCG}' = \sum_{i=1}^{N} \frac{P(R'_i = 1)}{\log_2(i+1)} \tag{3.37}$$

Since the elements at any position $i \notin \{k, k+1\}$ are identical in both sequences, and $R_k = R'_{k+1}$ and $R_{k+1} = R'_k$, the difference of the two expected values is:

$$\begin{aligned} &\overline{DCG} - \overline{DCG}' = \\ &\frac{P(R_k = 1)}{\log_2(k+1)} + \frac{P(R_{k+1} = 1)}{\log_2(k+2)} - \frac{P(R_{k+1} = 1)}{\log_2(k+1)} + \frac{P(R_k = 1)}{\log_2(k+2)} = \\ &(P(R_k = 1) - P(R_{k+1} = 1)) \left(\frac{1}{\log_2(k+1)} - \frac{1}{\log_2(k+2)} \right) = \\ &(P(R_k = 1) - P(R_{k+1} = 1)) \frac{\log_2(k+2) - \log_2(k+1)}{\log_2(2k+3)} \end{aligned} \tag{3.38}$$

Observe that the second term in the multiplication is always greater than 0. Hence, this element does not affect the sign of the difference of the expected values, resulting in:

$$\overline{DCG} - \overline{DCG}' \begin{cases} > 0 & P(R_k = 1) > P(R_{k+1} = 1) \\ = 0 & P(R_k = 1) = P(R_{k+1} = 1) \\ < 0 & P(R_k = 1) < P(R_{k+1} = 1) \end{cases} \tag{3.39}$$

This implies that, if the retrieved objects are not ordered by decreasing relevance probability (i.e. $\exists k : P(R_k = 1) < P(R_{k+1} = 1)$), we can increase the expected value of the DCG by swapping these

two elements. We can repeat this operation as long as the sequence of probabilities is not ordered. Once all the elements are completely ordered by decreasing relevance, we cannot longer increase its expected value, proving the theorem. □

3.2.5 NORMALIZED DISCOUNTED CUMULATIVE GAIN

Notice that the DCG is not upper bounded, and thus sequences of different number of elements cannot be relatively compared. In order to solve this issue, the Normalized Discounted Cumulative Gain (NDCG) was defined.

Definition 3.6. The Normalized Discounted Cumulative Gain (NDCG) of a sequence of retrieved elements is equal to the value of its DCG divided by the maximum DCG achieved by any possible ordering of the sequence, $\text{MDCG}(R_1, \ldots, R_N)$.

$$\text{NDCG}(R_1, \ldots, R_N) = \frac{\text{DCG}(R_1, \ldots, R_N)}{\text{MDCG}(R_1, \ldots, R_N)} \tag{3.40}$$

Corollary 3.6.1. *The expected value of the Normalized Discounted Cumulative Gain for a sequence of elements ordered by decreasing relevance probability (i.e. $P(R_i = 1) \geq P(R_j = 1), 1 \leq i < j \leq N$) is equal to 1, which is the maximum.*

Proof.

$$\begin{aligned} \mathbb{E}[\text{NDCG}(R_1, \ldots, R_N)] = \\ \mathbb{E}\left[\frac{\text{DCG}(R_1, \ldots, R_N)}{\text{MDCG}(R_1, \ldots, R_N)}\right] = \\ \int_0^\infty \frac{P(\text{MDCG} = m)}{m} \mathbb{E}[\text{DCG} \mid \text{MDCG} = m]\, dm \end{aligned} \tag{3.41}$$

Notice that theorem 3.6 states that ordering the retrieved objects by decreasing relevance probability maximizes the expected value of the DCG. Thus, if the maximum DCG (i.e. MDCG) is equal to m, then for such a sequence of elements, $\mathbb{E}[\text{DCG} \mid \text{MDCG} = m] = m$.

$$\mathbb{E}[\text{NDCG}(R_1, \ldots, R_N)] = \int_0^\infty \frac{P(\text{MDCG} = m)}{m} m \, dm = \int_0^\infty P(\text{MDCG} = m) dm = 1 \tag{3.42}$$

The fact that this is the maximum value comes from the definition of NDCG itself. □

3.3 Global and mean measures

In the Keyword Spotting literature (and also in many other applications of Information Retrieval), performance measures of ranking systems can be divided into *global* and *mean* measures[1]. Global measures, consider the retrieved elements of all queries used to evaluate the system *at the same time*, while *mean* measures evaluate each query in isolation and later average the results.

As an example, consider table 3.1. A (global) ranking of retrieved elements for different queries (v_1 and v_2) is shown, with their relevance and the *score* (not necessarily a probability) used to order the ranking list[2]. As the example shows, depending on the value of the relevance variable of each element, the global and mean average precision can be quite different.

Finally, observe that in section 3.2 we did not make any assumption on the queries that originated each element in the ranked list. Thus, it is fairly easy to prove that the PRP is optimal for both the global and mean versions of the measures in the previous section (see theorem 3.7 below).

Theorem 3.7. *The Probability Ranking Principle is optimal with respect to the Global and the Mean versions of the measures in section 3.2, assuming that the queries are independent.*

[1] In the literature, these two types of measures have been sometimes referred to as *micro* and *macro* measures, respectively [Perronnin et al., 2009, Tsoumakas et al., 2010].

[2] The values of the scores are irrelevant for this particular example.

Table 3.1. Example illustrating the calculation of the Global and Mean Average Precision (AP). The elements of the ranking list are pairs of (keyword, region), sorted by their decreasing score. R_i and R'_i show two hypothetical values of the relevance variable of each element in the ranked list.
In the first case, the Global AP is equal to $\frac{1}{3} + \frac{2}{9} + \frac{3}{12} = \frac{29}{36}$, while the Mean AP is equal to 1 (since the AP of each individual keyword is 1).
In the second case, the Global AP is equal to $\frac{1}{3} + \frac{1}{3} + \frac{1}{6} = \frac{5}{6}$, while the Mean AP is equal to $\frac{1}{2}\left(\frac{1}{6} + 1\right) = \frac{7}{12}$ (since the AP of v_1 is equal to $\frac{1}{6}$ and that of v_2 is equal to 1).

(a) Ranking

Element	R_i	R'_i	Score
(v_2, x_1)	1	1	3.9
(v_2, x_3)	0	1	2.8
(v_1, x_1)	1	0	1.7
(v_1, x_2)	1	0	0.4
(v_2, x_2)	0	0	-0.2
(v_1, x_3)	0	1	-1.1

(b) Performance with R_i

Global AP	$\frac{29}{36} \approx 0.81$
Mean AP	$\frac{1}{2}(1+1) = 1$

(c) Performance with R'_i

Global AP	$\frac{5}{6} \approx 0.83$
Mean AP	$\frac{1}{2}\left(\frac{1}{6}+1\right) = \frac{7}{12} \approx 0.58$

Proof sketch. When we proved the optimality of the ranking principle for all the previous performance measures, we did not make any assumption on the query that originated each element in the ranked list.

Thus, if the ranking is *globally* ordered (considering all retrieved elements of all queries), it will be optimal with respect the global measure of interest (e.g. the Global Average Precision).

In addition, if the ranked list is *globally* ordered, it is also *locally* ordered for the elements corresponding to each individual query. Thus, the ranking will be optimal for each individual query.

Finally, since the mean measure is just the average of the measure of each individual query, and each of these are the optimal values, the mean measure of interest (e.g. the Mean Average Precision) is also optimal.

Thus, the probability ranking principle is optimal for both the *global* and *mean* versions of the measures. □

4 Probabilistic Models for Handwritten Text

In chapter 2, we saw that the relevance probability needed to tackle the problem of Keyword Spotting (in a principled way) involves, essentially, the conditional probability over the transcript of the document images.

Later, in chapter 3, we used these relevance probabilities to *optimally* rank a set of elements, with respect to different quality measures widely used in the Information Retrieval and Keyword Spotting literature. However, the reader should remember that the proofs of optimality only hold under a few assumptions.

One of these assumptions requires that the distributions used to obtain the relevance probabilities must be the *true* ones that describe the data. Of course, these distributions are unknown in any real scenario, and reasonable good models have to be *estimated* from data.

In this chapter, two families of models used to represent the text (and its alignment) written in images are reviewed: Hidden Markov Models and Recurrent Neural Networks. Both models have been widely and successfully used, for Keyword Spotting and Handwritten Text Recognition applications (as well as other applications).

4.1 Image preprocessing

Before we introduce these probabilistic models used, we must explain how the scanned page images are first processed in order to extract meaningful regions containing text and to reduce the variability of these text regions that is not useful to "read" their content. As we shall see later, some methods for modeling the transcription of im-

ages are more robust (i.e. Artificial Neural Networks) than others (i.e. Hidden Markov Models) and require less of these steps.

4.1.1 TEXT SEGMENTATION

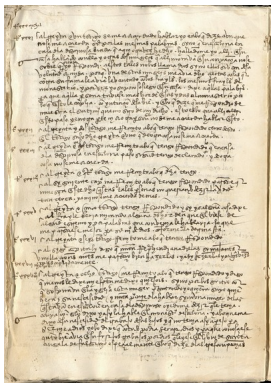
First the different text regions in the document are localized. Text regions are typically formed by several text lines with a meaningful (sequential) reading order. This problem is, by itself, quite challenging in real applications, mainly due to the ambiguity of the concept "text region". [Mao et al., 2003] offers a survey on traditional approaches tackling the structure analysis of text documents. Also, as in many other key aspects of text documents processing, methods based on Artificial Neural Networks have become the state-of-the-art in recent years [Bukhari et al., 2012, Chen et al., 2015, Chen et al., 2017, He et al., 2017].

Other preprocessing steps that are carried at page level are the removal of bleed-through and other ink noise, the correction of rotated pages or contrast normalization. Some of these steps play a less significant role under controlled environments, but they are very important if the page images were not obtained using good scanning equipment (e.g. they were obtained using a cell phone).

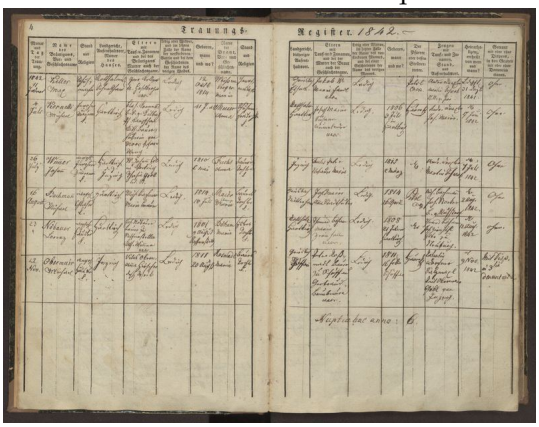
Next, individual text lines are isolated, which may be an even more difficult task and is considered an open research problem, as shown by recent competitions [Diem et al., 2017]. Common problems are the presence of diacritics, ascenders and descenders non-horizontal lines, overlapping lines, etc. [Likforman-Sulem et al., 2007, Louloudis et al., 2009] report a comparison on different approaches to tackle line segmentation, and recent relevant works include [Garz et al., 2012, Saabni et al., 2014, Cruz and Terrades, 2018]. Text line segmentation is considered a critical step in the processing of historical handwritten images, since virtually all approaches to model the text in these images assume that text lines have been (more or less) accurately segmented, including the models presented in this chapter.

Figure 4.1 shows three examples from different data sets used in text recognition and keyword spotting research. The easiest scenario, represented by the IAM data set depicted in fig. 4.1a, already senses some ambiguity: one could just separate the handwritten zones from

(a) IAM



(b) Inquisición



(c) Passau

Figure 4.1. Page images from different collections used for handwritten text recognition and keyword spotting experiments. (a) shows a page from the IAM data set, with a quite easy structure since all pages in the collection are formed by two main text regions: printed and handwritten text; (b) shows a page from the "Inquisición" data set, where annotations are interleaved with the main text of the page; and (c) shows a page from the Passau collection, which contains many tables and records with a particular structure.

the printed text areas, but one could also argue that the printed areas are separated in three regions: header, description and footer. Nevertheless, text line segmentation is quite straightforward in this collection, since text lines are well separated by blank regions.

On the contrary, fig. 4.1b and fig. 4.1c show two data sets where text region localization and line segmentation are considerably more challenging, since the main text body is filled with annotations (see fig. 4.1b), and there are dozens of close-by text regions (one for each cell in fig. 4.1c) with few text lines in each.

4.1.2 TEXT LINE NORMALIZATION

Once the page images have been segmented into individual text lines, several normalization steps are applied to reduce the variability in the handwriting, as discussed above.

Text lines do not follow a strict horizontal (or vertical) orientation, especially in handwritten documents written without constraints. Plus, quite often, handwritten characters tend to be inclined towards the left or the right, producing the *italic* effect. The first distortion is known as the *skew* of the line, whereas the latter is known as the *slant*.

To correct the skew of a text line, some methods estimate a single angle for the whole text line. For instance, a popular approach is based on the horizontal projection profile of the text line [Senior and Robinson, 1998, Vinciarelli and Luettin, 2001]. A survey can be found in [Hull, 1998]. Other authors detect several regions within the text line to apply different skewing corrections in each segment [Toselli et al., 2004, Vinciarelli et al., 2004, Pesch et al., 2012].

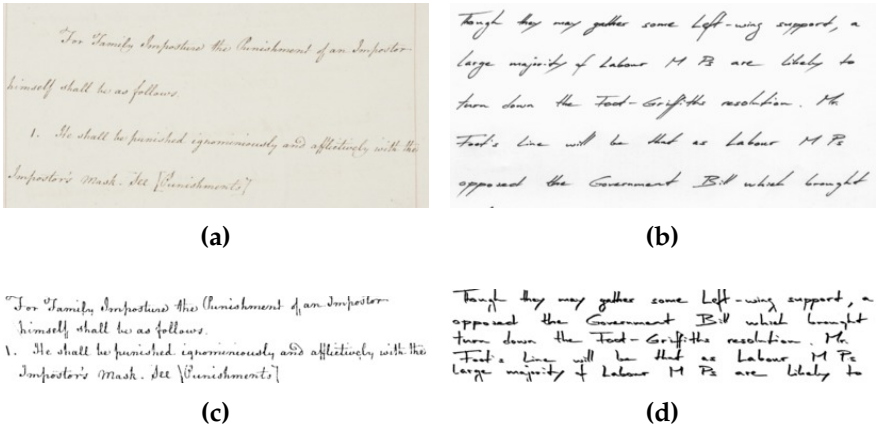
Regarding the slant correction, virtually all methods find the angle between the vertical axis of the text line and the strokes of text in the image. Then, an affine (shear) transformation is applied to correct the angle. For instance [Vinciarelli and Luettin, 2001, Pastor et al., 2004] try a range of slant angles and choose the one that maximizes the variance of the histogram of the pixel intensities.

Finally, the size of the writing is also normalized since it may vary significantly between different writers and/or documents. This normalization usually includes the height of the text, as well as its thick-

ness. A common solution (that we have used in most experiments) is to scale all text line images to a fixed height, preserving the aspect ratio. This works well when the height of the text line is proportional to the height of the characters, but when the lines have not been accurately segmented, this approach may be troublesome.

it result in a damaging loss of information. Another approach that has been proposed more recently is to use the first and second order moments over a sliding window, to re-position the center of gravity in each window at the center of the image and re-scale the size so that the second order moments are constant [Kozielski et al., 2012].

Figure 4.2 depicts two examples of skewed and slanted text lines and the normalized images after using the normalization software that most of our experiments use[1].



(a) **(b)**

(c) **(d)**

Figure 4.2. Figure (a) shows a portion of a text page which includes skewed (and also slanted) text lines. Figure (b) shows a heavily slanted text. Figures (c) and (d) show the text lines of each segment after normalization.

4.1.3 FEATURE EXTRACTION

Traditionally, statistical models could not model accurately the transcript (and alignment) distributions using directly raw images (not

[1] `https://github.com/mauvilsa/textfeats`

even images after normalization), and a sequence of handcrafted feature vectors needed to be extracted from the raw pixels. Although this is no longer the case with modern methods based on Artificial Neural Networks (this will be discussed in section 4.3), feature extraction is still needed to operate with traditional Hidden Markov Models.

An enormous set of alternatives exists to extract features from the raw pixels. Here we describe some of the popular techniques used in the past. It is extremely important to notice that, very often, the features employed are highly correlated with the image processing steps carried before, meaning that if one processes the images in a different way than the proposed by the author of the feature extraction technique, the overall performance may be poor (e.g. some features need the images to be binarized).

Many of the early techniques relied on counting the black and/or white pixels (after binarization) in the image. [Marti and Bunke, 2000, Marti and Bunke, 2001b, Bertolami and Bunke, 2008] use the number of black pixels in each column of the image; [Marti and Bunke, 2000, Bertolami and Bunke, 2008] also count the number of transitions between black and white pixels; and [Toselli et al., 2010, Espana-Boquera et al., 2011] use the average intensity of the pixels in rectangular cells of the image.

In addition to these simple features, higher-level features have also been used in the literature. For instance, [Toselli et al., 2010, Espana-Boquera et al., 2011] use the derivative of the pixel intensities in each dimension in rectangular cells of the image; and common features in the Computer Vision community such as Speeded Up Robust Features (SURF) [Bay et al., 2006] and Scale-Invariant Feature Transform (SIFT) [Lowe, 1999], have also been used in the context of handwritten text modeling [Wang et al., 2012, Rothacker et al., 2012].

4.2 Hidden Markov Models

4.2.1 Description

A Hidden Markov Model (HMM) describes a stochastic process involving two random variables: the random variable representing the sequence of *observed values*, denoted by X, and the random variable

representing the sequence of *states* that produced such values, denoted by S. In particular, Hidden Markov Models are probabilistic graphical models of the joint likelihood of the two variables $p(X, S)$.

Hidden Markov Models are typically defined using the following elements:

- A set of emitting states $\mathcal{S} = \{s_1, \ldots, s_N\}$, and a special (non-emitting) final state s_F.
- A probability distribution over initial states: $P(S_1 = s), \forall s \in \mathcal{S}^*$.
- A probability distribution describing the transition model between states: $P(S_{t+1} = s' \mid S_t = s), \forall s \in \mathcal{S}, \forall s' \in \mathcal{S} \cup \{s_F\}$.
- And a probability mass or density function describing the likelihood of the an observed value according to each non-final state: $p(X_t = \boldsymbol{x} \mid S_t = s), \forall s \in \mathcal{S}$.

Figure 4.3 shows an example of an *ergodic* HMM with three emitting states, depicting the previous elements. The name ergodic simply denotes that any possible transition between two states has a non-zero probability mass.

We will refer to a particular sequence of observed values with the notation $\boldsymbol{x}_{1:T} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$. Notice that a sequence of observed values of length T is *emitted* by a sequence of states of length $T + 1$, since all valid sequences of states must end at the non-emitting final state, s_F. Thus, a particular sequence of states of length $T + 1$ is represented by $s_{1:T+1} = s_1, \ldots, s_T, s_F$.

Since X and S are both variables over sequences of arbitrary length, instead we could interpret each of them as a sequence of random variables: $X = X_1, \ldots, X_T$ and $S = S_1, \ldots, S_T, S_{T+1}$, denoting the random variables involved in a sequence of length T. Yet, it is important to emphasize that, in general, the model does not restrict in any way the length of the sequences, i.e. $T \in [1, \infty)$.

*It is possible to include the final state s_F in the set of possible initial states, in order to allow sequences of length 0, but this is not the norm.

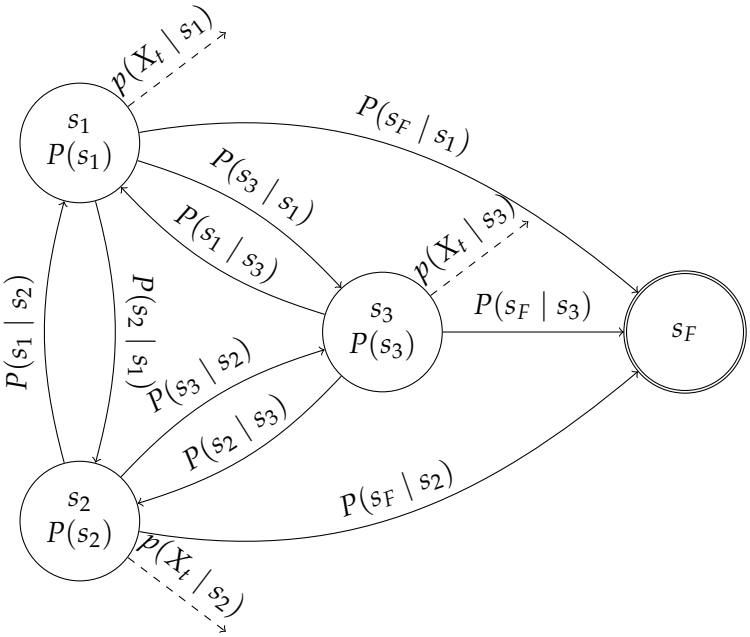


Figure 4.3. An example of a Hidden Markov Model with all the involved probability functions. Each state s_1, s_2, s_3, s_F, is represented by a circle, with the name and the initial state probability written in it (observe that the final state does not have an initial probability). Solid arcs represent the *values* of the transition probability $P(S_{t+1} = s_j \mid S_t = s_i)$ between pairs of states (s_i, s_j). Dashed arcs represent the density *distribution* $p(X_t \mid S_t = s_i)$ of each emitting state, s_i.

For sequences of observed values of length T, HMMs make two assumptions:

1. The sequence of states is described by a first-order Markov process (remember that $s_{T+1} = s_F$).

$$P(S = s_{1:T+1}) \stackrel{\star}{=} P(S_1 = s_1) \prod_{t=1}^{T} P(S_{t+1} = s_{t+1} \mid S_t = s_t) \quad (4.1)$$

2. The value of the observation at time t only depends on the value of the state at that time.

$$p(X = \boldsymbol{x}_{1:T} \mid S = s_{1:T+1}) \stackrel{\star}{=} \prod_{t=1}^{T} p(X_t = \boldsymbol{x}_t \mid S_t = s_t) \quad (4.2)$$

Realize that the probability and density distributions do not depend on any particular value of t, only on the state. That is:

$$\begin{aligned} \forall t, t', \boldsymbol{x} \quad s_t = s_{t'} \Rightarrow \\ P(X_t = \boldsymbol{x} \mid S_t = s_t) = P(X'_t = \boldsymbol{x} \mid S_{t'} = s_{t'}) \\ \forall t, t', s \quad s_t = s_{t'} \Rightarrow \\ P(S_{t+1} = s \mid S_t = s_t) = P(S_{t'+1} = s \mid S_{t'} = s_{t'}) \end{aligned} \tag{4.3}$$

Under these assumptions, $p(X = \boldsymbol{x}_{1:T}, S = s_{1:T+1})$ is equal to:

$$\begin{aligned} p(X = \boldsymbol{x}_{1:T}, S = s_{1:T+1}) = \\ p(X = \boldsymbol{x}_{1:T} \mid S = s_{1:T+1}) P(S = s_{1:T+1}) \overset{\star}{=} \\ \left(\prod_{t=1}^{T} p(\boldsymbol{x}_t \mid s_t) \right) \left(P(s_1) \prod_{t=1}^{T} P(s_{t+1} \mid s_t) \right) \end{aligned} \tag{4.4}$$

Sometimes, researchers refer to HMMs as *generative models*, since they model the likelihood of observed values, and examples of can be generated sampling from it.

4.2.2 TRAINING

4.2.2.1 Generative training

Traditionally, HMMs have been trained using the Maximum Likelihood Estimation (MLE) criterion on the observed values of the training data. Thus, we try to find the optimal set of parameters $\boldsymbol{\theta}^*$ that maximizes the probability density of the feature vectors of the training data, given the reference transcripts. The set of parameters to optimize includes both the emission and transition parameters. In general, this procedure can be described by the following optimization problem:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(X' = \{\boldsymbol{x}_i : 1 \leq i \leq m\} \mid W' = \{w_i : 1 \leq i \leq m\}; \boldsymbol{\theta}) \tag{4.5}$$

where $p(X' \mid W'; \boldsymbol{\theta})$ is the joint likelihood of *all* the training feature vectors given *all* the training reference transcripts, according to a particular model parameterized by $\boldsymbol{\theta}$.

When one assumes that each of the training examples is sampled independently, an equivalent expression can obtained:

$$\boldsymbol{\theta}^* \stackrel{\star}{=} \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p(X = \boldsymbol{x}_i \mid W = w_i; \boldsymbol{\theta}) =$$

$$\arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log \sum_{s_{1:T+1}} p(X = \boldsymbol{x}_i, S = s_{1:T+1} \mid W = w_i; \boldsymbol{\theta}) \stackrel{\star}{=} \tag{4.6}$$

$$\arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log \sum_{s_{1:T+1}} p(X = \boldsymbol{x}_i \mid S = s_{1:T+1}; \boldsymbol{\theta}) P(S = s_{1:T+1} \mid W = w_i; \boldsymbol{\theta}) \tag{4.7}$$

The previous equation maximizes the logarithm of the density of the feature vectors given the transcript, instead of the density itself. However, the two are equivalent since the logarithm is a monotonically increasing function.

The sum over $s_{1:T+1}$ in eq. (4.6) is due to the marginalization of the density among all sequences of states of the composite model of HMMs, which is one of the hidden variables of the model.

Because closed-form solutions to the previous optimization problem cannot be found for complex probabilistic models with hidden variables (such as HMMs and GMMs), the Expectation–Maximization algorithm [Dempster et al., 1977] (in particular, the Baum–Welch algorithm [Baum and Eagon, 1967, Baum et al., 1970] for HMM) is used to find a *local optimum* of the parameters. The algorithm starts with an arbitrary set of parameters and these are iteratively modified to increase log-density until convergence. All popular HMM-based toolkits for speech recognition and handwritten text recognition implement these algorithms, or variations of them (such as the Viterbi training for HMMs) [Young et al., 2002, Povey et al., 2011].

4.2.2.2 Discriminative training

Notice that the previous approach finds a set of parameters for our probabilistic models that (locally) maximizes the density of the feature vectors given the reference transcript of the image.

However, in reality we are usually in the opposite scenario, since the transcript of the images is unknown. Then, we typically want to find the transcript with the maximum probability given the sequence of feature vectors. Once the HMMs and GMMs are trained, we can solve the latter problem using the Bayes theorem, but when training our models in a generative way, we are solving a different problem than the original (find a probabilistic model that accurately represents the transcript of a given image).

There exists a wide variety of estimation criteria in the context of Hidden Markov Models in order to train these models to have a better *discriminative* performance. The general idea is to try to find a model that maximizes the probability of the reference transcript given the feature vectors, although slightly different variations of this idea have also been widely used.

For instance, the Maximum Mutual Information (MMI) criterion [Bahl et al., 1986] maximizes the mutual information between the sequences of feature vectors and the corresponding transcripts; the Minimum Classification Error Rate (MCE) criterion [Juang et al., 1997] tries to find a set of parameters that minimizes the expected number of wrongly transcribed samples; the Minimum Word and Phone Error Rate (respectively, MWE and MPE) [Povey and Woodland, 2002] try to optimize the parameters of the model so that the expected number of wrongly transcribed words (or phones in the case of speech signals) is minimized.

In this thesis, we have generally used the more traditional generative training (using the Baum–Welch algorithm) in order to estimate the parameters of our HMMs (and GMMs). However, in some cases we have also used the Minimum Character Error Rate (i.e. analogous to the MPE in speech) to have better probabilistic models (see section 8.12).

4.2.3 Hidden Markov Models for Handwritten Text

In order to reduce the number of free parameters of the model (and thus, reduce the chances of *overfitting*), some structural decisions are made in practice for modeling handwritten text.

Typically, HMMs adopt a left-to-right topology, as depicted in fig. 4.4. In a left-to-right topology, each emitting state has two transitions: a self-loop and a transition to the next state. Some authors add *skip* transitions so that samples shorter than the number of emitting states can be processed by the HMM (otherwise, these sequences would have a null likelihood) [El-Yacoubi et al., 1999, Tay et al., 2001].

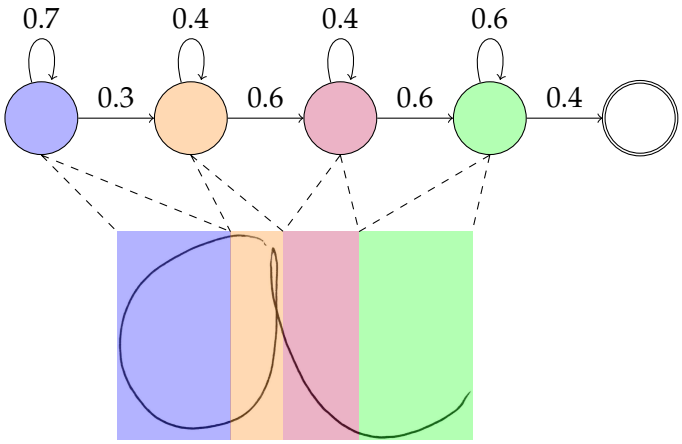


Figure 4.4. Example of the alignment produced by a character HMM modeling the letter "a". The HMM is composed of four states in a left-to-right topology. The probabilities in the arcs represent the transition probabilities of the HMM, i.e. $P(S_{t+1} \mid S_t)$, and the final state also includes the *final* probability.

As depicted in fig. 4.4, most works that need to model handwritten text choose to use an individual HMM to represent each character in the alphabet. There are some works that represent full words by a single HMM, however this becomes problematic when dealing with vocabularies of a large number of words, since the number of parameters required to estimate grows significantly. Notice that the number of characters of the Latin and other western alphabets is typically in the 10–100s, while the number of words that one typically encounters in large collections of texts is in the 10 000–100 000s. This is very similar to the works in speech recognition, where they typically use an individual HMM to represent a phoneme (or a tri-phoneme), since this is the fundamental unit of speech.

The number of states in each HMM can be fixed (i.e. all characters have the same number of states), or can be variable, since the length of each character is expected to be different from one class to the other. For instance characters like "i" or "l" are typically much shorter (horizontally) than characters like "m" or "n".

Finally, regarding the emitting states, we will use Gaussian Mixture Models (GMM) with diagonal covariance matrices as the probability density functions used to model $p(X_t = \boldsymbol{x}_t \mid S_t = s_t)$. Although this is the common choice in the handwritten text and speech community, the reader should be aware that it is not the only option, and models for sequences of discrete observed values have also been used in the past [Huang et al., 1993,Digalakis et al., 2000,Giménez and Juan, 2009]. Equation (4.8) shows the density of a diagonal GMM:

$$p(X_t = \boldsymbol{x}_t \mid S_t = s_t) \stackrel{\text{def}}{=} \sum_{k=1}^{K} \omega_{s_t,k} \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{\mu}_{s_t,k}, \text{diag}(\boldsymbol{\sigma}^2_{s_t,k})) \tag{4.8}$$

where $\omega_{s_t,k}$ is the component weight, and $\boldsymbol{\mu}_{s_t,k}$ and $\text{diag}(\boldsymbol{\sigma}^2_{s_t,k})$ are the mean and diagonal covariance matrix of the k-th multivariate Gaussian corresponding to the state s_t.

[Günter and Bunke, 2004] offers a very good analysis of different methods to optimize the topology of HMMs for modeling handwritten text, including the number of states and the number of components in the GMM.

4.3 Artificial Neural Networks

4.3.1 Description

During the last decade, Artificial Neural Networks (or NNs) have become predominant in many Pattern Recognition applications. In particular, Convolutional and Recurrent Neural Networks have become the *de facto* standard to model handwritten text.

The fundamental components of a NN are the (artificial) *neurons*, also known as *units*. An artificial neuron, in a similar way to real neurons present in living organisms, is connected to other neurons and produces an output given the inputs from the connected cells,

using an *activation function* σ. The first definition of an artificial NN appeared in the work of [McCulloch and Pitts, 1943], and the first algorithm to adjust the weights of the neurons was described in [Rosenblatt, 1958]. Figure 4.5a shows the mathematical model of the neuron introduced by [McCulloch and Pitts, 1943].

In order to solve challenging problems, the neurons are grouped together into several *layers* of neurons, so that the neurons from one layer are connected to the neurons of the next layer[2]. The layers of neurons which are neither the input nor the output are called *hidden layers*. Figure 4.5b shows an illustration of a multilayer artificial neural networks with two hidden layers. This type of network is also known as a fully connected multilayer network, since each neuron is connected to all the units from the previous layer, as shown in the figure.

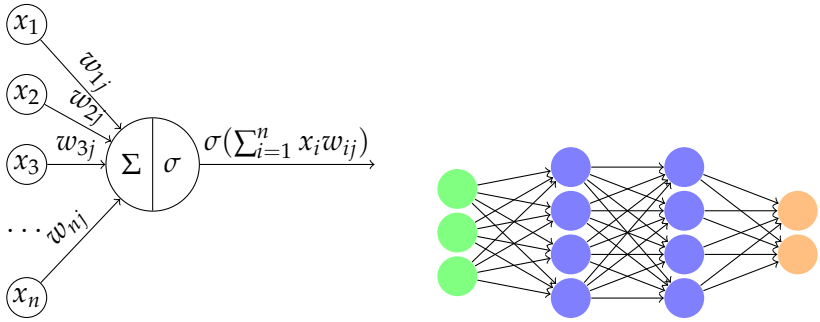


Figure 4.5. Diagram of the mathematical model of an artificial neuron (fig. 4.5a) and an illustration of a multilayer neural network with two hidden layers (fig. 4.5b), with three input and two output neurons.

It has been proven that, for several families of activation functions, a multilayer neural network is a *universal approximator*. This means that, although composed by rather simple units, NNs are powerful models able to approximate any function arbitrary well. This result was first proved for the sigmoid activation function [Cybenko, 1989]

[2]Typically, neurons also have a constant input called the bias, but we will omit it from the notation for simplicity.

defined as:

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{4.9}$$

Nevertheless, due to the characteristics of the algorithm used to adjust the parameters of the model, other activation functions have been proposed that perform better in practice: the hyperbolic tangent (denoted by tanh), the rectifier linear unit (also known as ReLU, and denoted by R) [Hahnloser et al., 2000], and the leaky ReLU (denoted by L_a, typically with $a = 0.01$) [Maas et al., 2013] are a few examples. The following equations describe each of these activation functions.

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{4.10}$$

$$R(x) = \max\{0, x\} \tag{4.11}$$

$$L_a(x) = \begin{cases} x & x > 0 \\ a \cdot x & \text{otherwise} \end{cases} \tag{4.12}$$

In classification problems, the output of the neural network typically represents the posterior probability of a label given the input data [Bourlard and Wellekens, 1990]. Thus, the output units must have their value in the range $[0, 1]$ and the sum of all of their outputs must be equal to 1. For that, the *softmax* function [Bridle, 1990] is typically used:

$$y_i = \phi(x_1, \ldots, x_n) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \tag{4.13}$$

where y_i is the i-th output neuron of the NN.

4.3.2 CONVOLUTIONAL LAYERS

Fully connected layers present two problems in practice. First, since all neurons from one layer are connected to all neurons of the next layer, and each connection has its own parameter, this means that the number of neurons in each layer has to be limited (since we need the number of parameters of the model to be finite, for training purposes). Thus, for instance, if we needed to process images with our NN, we

first would need to get a fixed-size representation of our images (e.g. by re-scaling them to a fixed size).

The second problem arises when the input data is high dimensional, but presents strong local patterns between *neighbor* input units. For instance, consider a rather small squared input image of 100 pixels height and width. Since each unit in a fully connected layer would be connected to *all* input pixels, we would need 10^4 parameters for each unit in the layer. However, the correlation between the pixel intensities of far away coordinates is low. This makes fully connected layers very slow when processing images with a medium or large size, and highly *overparameterized*, which may create or aggravate the problem of overfitting[3].

Convolutional layers [Fukushima and Miyake, 1982, LeCun et al., 1989] are one solution to the aforementioned problems. They are similar to fully connected layers, but each neuron of a Convolution layer is only connected to the *neighbor* units from the previous layer. In addition, all units share the same parameters. Thus, the number of parameters does not depend on the size of the input data, but only on the *receptive field* (i.e. the size of the neighborhood around each unit). For instance, fig. 4.6 shows a diagram of a convolution layer operating on a input image of width and height of 4 pixels, with a receptive field of 3 squared pixels. Each output pixel is the product of each neighboring input pixel weighted by the parameters.

For multichannel images (e.g. RGB), for each output channel the convolution operation typically weights all input channels from all neighboring pixels. Let $\boldsymbol{X}$ be the input image represented as a tensor[4] of size (W, H, C) where W is the width, H is the height and C is the number of input channels. Then, if $\boldsymbol{X}$ is convolved with a receptive field of size $(\vartheta_i, \vartheta_j)$ and K output channels, the result will be an image

[3]Overfitting happens when a model is highly accurate for training data but highly inaccurate for new data coming from the same distribution.

[4]Tensors are a generalization of matrices for an arbitrary number of dimensions. For instance, vectors are one-dimensional tensors, matrices are two-dimensional tensors and multichannel images can be represented as three-dimensional tensors.

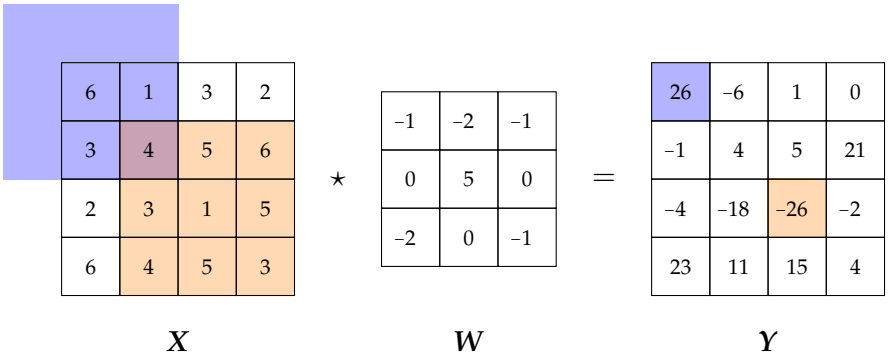


Figure 4.6. Diagram of a two dimensional convolution operation. The input image $\boldsymbol{X}$ is convolved with the weight matrix $\boldsymbol{W}$ to produce the output image $\boldsymbol{Y}$. The value of each output pixel is the weighted sum of the neighboring input pixels, weighted by the corresponding parameter in $\boldsymbol{W}$.

of size (W, H, K) described by the following equation[5]:

$$[\boldsymbol{Y}]_{i,j,k} = [\boldsymbol{X} \star \boldsymbol{W}]_{i,j,k} = \sum_{i'=0}^{\vartheta_i - 1} \sum_{j'=0}^{\vartheta_j - 1} \sum_{c=1}^{C} [\boldsymbol{X}]_{i+i'-\lfloor \frac{\vartheta_i}{2} \rfloor, j+j'-\lfloor \frac{\vartheta_j}{2} \rfloor, c} [\boldsymbol{W}]_{i',j',c,k} \tag{4.14}$$

Here, the operator $\lfloor \cdot \rfloor$ denotes the *floor* operator. Very often, a *padding* value is defined (usually 0) for pixels laying outside of the input images (e.g. the blue regions outside the input matrix in fig. 4.6).

Finally, the reader should be aware that, although convolutional layers were first popularized to process images [LeCun et al., 1989], they have also been used to process one-dimensional sequences [Hu et al., 2014], videos [Ji et al., 2013], or even sparse graphs [Defferrard et al., 2016].

4.3.3 RECURRENT LAYERS

Recurrent layers [Elman, 1990] (RNNs) are a type of neural network layer with an internal *state* for each unit. These were originally designed to process sequences of vectors instead of a single vectors, as fully connected neural networks. At each time-step, the output of

[5] Typically, as in the case of fully connected layers, each neuron has an input bias, but we omitted this from the notation for simplicity.

of layer depends on the current input and the previous state. In its simplest form, the *state* of each neuron in a recurrent layer is just its output. Thus, for a sequence of T elements of m-dimensional (row) vectors, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, the output of a simple RNN, parameterized by the matrices $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{R} \in \mathbb{R}^{n \times n}$, is a sequence of n-dimensional vectors, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$, given by the equation[6]:

$$y_{t,j} = \sigma\left(\sum_{i=1}^{n} x_{t,i}[\boldsymbol{W}]_{i,j} + \sum_{i'=1}^{m} y_{t-1,i'}[\boldsymbol{R}]_{i',j}\right) \tag{4.15}$$

where $y_{t,j}$ is the j-th component of the output vector at time t (i.e. $\boldsymbol{y}_t$) and $x_{t,i}$ is the i-th component of the input vector at time t (i.e. $\boldsymbol{x}_t$). Typically, at $t = 1$ the recurrent connection is ignored, which is equivalent to assume that the state at $t = 0$ is $\boldsymbol{y}_0 = \boldsymbol{0}$.

The previous equation can be simplified using vectorial notation to describe all units in the recurrent layer.

$$\boldsymbol{y}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W} + \boldsymbol{y}_{t-1} \boldsymbol{R}) \tag{4.16}$$

where the non-linear function σ is applied to all components of the vector.

RNNs can be visualized as regular fully connected layers if one *unrolls* the sequence of input and output vectors, as depicted in fig. 4.7.

Very often, the sequence is processed from left-to-right and right-to-left directions. This architecture is known as a bidirectional layer (BRNN) [Schuster and Paliwal, 1997]. After the sequence has been processed in the two directions, the two output vectors at each time-step are combined into a single one by summing, averaging or concatenating them (among other combination strategies). Then, the combined output is fed into the next layer.

4.3.3.1 Long Short-Term Memory layers

Simple RNN are conceptually powerful models. In fact, it has been argued that they are Turing-complete machines [Siegelmann, 1995],

[6]Recurrent units typically have a bias input too, but this has been omitted for simplicity.

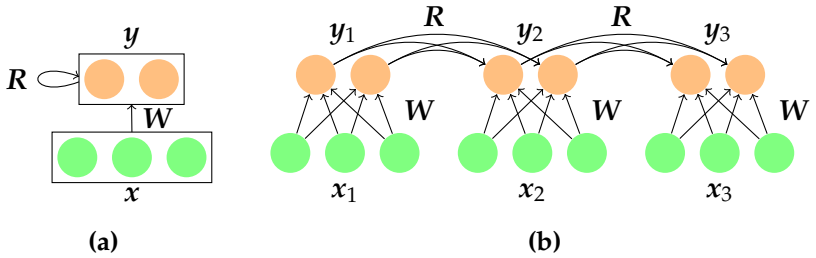
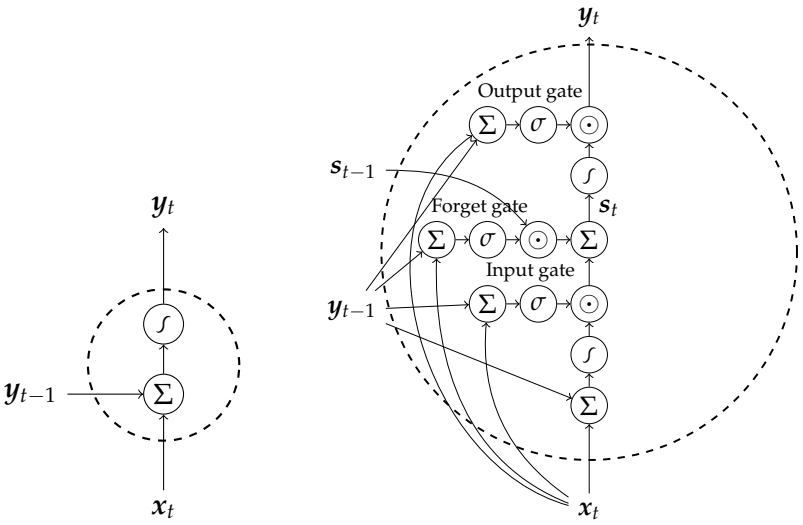


Figure 4.7. Compact and unrolled representation of a simple recurrent layer. The connections in the left figure are replicated at each time-step and the resulting neural network is equivalent to a (very deep) fully connected network with several layers, as shown in the right figure.

although in practice this theoretical capability is quite limited by several reasons. Among other reasons, the algorithms typically used to learn the parameters of our network have some issues with the simplest RNN architectures. In particular, they are prone to the vanishing and exploding gradient problem, which prevents the network to learn dependencies in a large time context, and makes the training very unstable. In order to address these issues, Long Short-Term Memory (LSTM) units were proposed [Hochreiter and Schmidhuber, 1997]. These specially-designed units, control the flow of information between the units at different time-steps using a set of *gates*. The original LSTMs were later improved with *forget gates* [Gers et al., 2000], which can be used to "reset" the internal state of the units. Figure 4.8 shows the details of the simple recurrent units and the LSTM units (with forget gates).

Given a input sequence of T elements of m-dimensional vectors, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$, the output sequence of n-dimensional vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$ provided by an LSTM layer is described by the following set of (vec-



(a) Simple recurrent unit **(b)** LSTM unit

Figure 4.8. Detailed diagram of the units of a simple recurrent layer (left) and a LSTM layer (right). The LSTM unit has two gates (input and forget gates) that control the input to the unit's state (s_t), and an output gate that controls the output of the unit, at each time-step.

torial) equations[7].

$$\boldsymbol{a}_t = \tanh(\boldsymbol{x}_t \boldsymbol{W}_a + \boldsymbol{y}_{t-1} \boldsymbol{R}_a) \tag{4.17}$$
$$\boldsymbol{i}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}_i + \boldsymbol{y}_{t-1} \boldsymbol{R}_i) \tag{4.18}$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}_o + \boldsymbol{y}_{t-1} \boldsymbol{R}_o) \tag{4.19}$$
$$\boldsymbol{f}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}_f + \boldsymbol{y}_{t-1} \boldsymbol{R}_f) \tag{4.20}$$
$$\boldsymbol{s}_t = \boldsymbol{i}_t \odot \boldsymbol{a}_t + \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1} \tag{4.21}$$
$$\boldsymbol{y}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{s}_t) \tag{4.22}$$

where all the $\boldsymbol{W}$ parameter matrices are in $\mathbb{R}^{m \times n}$ and the recurrent parameter matrices, $\boldsymbol{R}$, are in $\mathbb{R}^{n \times n}$. The operator $\odot$ denotes the Hadamard (or element-wise) product of vectors. As the simplest recurrent units do, most implementations of LSTMs assume that $\boldsymbol{y}_0 = \boldsymbol{s}_0 = \boldsymbol{0}$.

[7] Once more, we have omitted the bias inputs in the first four equations to simplify the notation.

Long Short-Term Memory layers are in the core of almost all state-of-the-art solutions to countless applications such as handwritten text recognition [Graves and Schmidhuber, 2009, Voigtlaender et al., 2016, Puigcerver et al., 2017], speech recognition [Graves et al., 2013b,Graves et al., 2013a], machine translation [Bahdanau et al., 2014, Cho et al., 2014], language modeling [Sundermeyer et al., ,Sutskever et al., 2014], image captioning [You et al., 2016, Vinyals et al., 2017], etc.

4.3.4 TRAINING

Artificial Neural Networks used to model the transcript of a handwritten image are typically *discriminative*, and directly model the posterior distribution of the transcript given the image. Typically, the (log-)posterior probability of the reference transcript is maximized during training. It is easy to prove that this is equivalent to minimizing the Kullback–Leibler divergence between a Dirac delta distribution representing the reference text, and the probability distribution modeled by the neural network (assuming that the samples are independent).

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\left[P(W \mid X) \parallel P(W \mid X; \boldsymbol{\theta})\right] \equiv \\ &\qquad \arg\max_{\boldsymbol{\theta}} \log P(W = \hat{w} \mid X = \boldsymbol{x}; \boldsymbol{\theta}) \end{aligned} \tag{4.23}$$

4.3.4.1 Connectionist Temporal Classification

In order to model $P(W \mid X)$ employing neural networks, the key idea is to transform the input image into a sequence of d-dimensional feature vectors, each of which represents the characteristics of the particular *label* in that position of the sequence.

If we assume that the label in each position is independent from the other position, given the corresponding feature vector, then we can easily write down the probability of a given sequence of *labels*, $a_1, \ldots, a_T$:

$$P(A = a_1, \ldots, a_T \mid X = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \overset{\star}{=} \prod_{t=1}^{T} P(A_t = a_t \mid X_t = \boldsymbol{x}_t) \tag{4.24}$$

This model, assigns (in a stochastic manner) a *label* to each feature vector of the sequence, in a similar way that HMMs assume that each feature vector is generated by some state.

Notice that we used the term *label* in the previous description, not *character* or *word*. Thus, the remaining question is, how can we define a distribution over transcripts (sequences of characters) from the distribution over labels?

In order to define a distribution over transcripts of different lengths, the Connectionist Temporal Classification (CTC) [Graves et al., 2006] method is used. This assumes that the set of labels is equal to the set of characters, plus an additional auxiliary label, typically called the "CTC-blank" or "no-symbol" label, which is typically denoted by the symbol $\varnothing$. Precisely, the "CTC-blank" is needed to allow transcripts shorter than the number of feature vectors and to differentiate two consecutive and equal characters in the transcript, from the sequence of labels.

Given this, the sequence of labels is mapped into a sequence of characters in a deterministic way, using a simple function, which we refer to as the *CTC function*, denoted by $\mathcal{F}$:

1. Remove all contiguous repetitions of labels from the sequence (e.g. $\{\varnothing, \varnothing, a, a, b, \varnothing, b, b, a, a\} \Rightarrow \{\varnothing, a, b, \varnothing, b, a\}$).

2. Remove all CTC-blank symbols from the sequence (e.g. $\{\varnothing, a, b, \varnothing, b, a\} \Rightarrow \{a, b, b, a\}$).

While we are train the neural network, we need to compute (the logarithm of) $P(W = \hat{w} \mid X = \boldsymbol{x}; \boldsymbol{\theta})$, given the given reference transcript, $\hat{w}$, and the sequence of T feature vectors, $\boldsymbol{x}$. In order to define this value as a function of the distribution over the *labels*, we use

marginalization:

$$\begin{aligned} P(W = \hat{w} \mid X = \boldsymbol{x}; \boldsymbol{\theta}) = \\ \sum_a P(W = \hat{w}, A = a \mid X = \boldsymbol{x}; \boldsymbol{\theta}) = \\ \sum_a P(A = a \mid X = \boldsymbol{x}; \boldsymbol{\theta}) P(W = \hat{w} \mid A = a, X = \boldsymbol{x}; \boldsymbol{\theta}) = \\ \sum_{a:\mathcal{F}(a)=\hat{w}} P(A = a \mid X = \boldsymbol{x}; \boldsymbol{\theta}) \end{aligned} \tag{4.25}$$

The last equality is due to the fact that, since $\mathcal{F}$ is a function (a deterministic process), $P(W = \hat{w} \mid A = a, X = \boldsymbol{x}; \boldsymbol{\theta})$ is simply a Dirac delta function, whose value is 1 for the sequences of labels that generate the reference transcript, and 0 for the rest.

Figure 4.9 shows an example of the probabilistic interpretation that the CTC makes of the output of a neural network. Notice that the posterior distribution can be represented as a weighted automaton with $T + 1$ states (recall, T is the number of feature vectors). The automaton has such a simple form because of the conditional independence assumption made by the CTC algorithm. Moreover, the sum of the probabilities of all label sequences that produce the given reference sequence of characters can be computed efficiently thanks to this independence assumption.

4.3.4.2 Learning through gradient descent

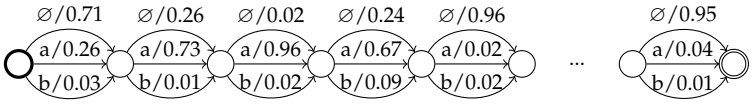
We now have a way of computing (in a efficient way), the probability $P(W = \hat{w} \mid X = \boldsymbol{x})$, which we would like to maximize, adjusting the parameters (or weights) of the artificial neural network.

In order to do so, gradient-based algorithms are typically used. Since all the components of the NN based on convolutional and recurrent layers are differentiable, one can compute the gradient of the *loss function* (i.e. $-\log P(W = \hat{w} \mid X = \boldsymbol{x})$) with respect to each of these parameters, and update them in the opposite direction of the gradient in order to (iteratively) *minimize* the objective loss function. The classical algorithm to perform these iterative updates is Stochastic Gradient Descent (SGD) [Robbins and Monro, 1951], but different alternatives are used very often, such as Adagrad [Duchi et al., 2011],

(a) Input image.

∅	2	1	−3	2	3	0	0	1	−5	2
a	1	2	1	3	−1	−1	1	−4	5	−1
b	−1	−3	−3	1	−1	2	1	2	−2	−3

(b) Output sequence of vectors produced by the neural network.



(c) Probability distribution of the output labels represented as a weighted automaton (softmax outputs).

Figure 4.9. Example of the probabilistic interpretation that the CTC makes of the output of a neural network, applied to a text image. The probability of a given sequence of *characters* is equal to the sum of all sequences of *labels* that produce such characters.

Adam [Kingma and Ba, 2014], RMSProp [Tieleman and Hinton, 2012], and many others.

The gradient of the loss function with respect to each parameter of the neural network can be computed efficiently thanks to the Backpropagation (BP) algorithm [Werbos, 1974, Rumelhart et al., 1986]. A version of this algorithm also exists for recurrent neural networks [Werbos, 1990], known as Backpropagation Through Time (BPTT).

In the early days of neural networks, BP was unable to effectively train the parameters of large neural networks with many stacked layers. Similarly, BPTT also had problems learning with long sequences for tasks that required long time dependencies between its outputs. However, better initialization of strategies [Glorot and Bengio, 2010], activation functions, types of layers (LSTMs vs. simple RNNs) [Hochre-

iter and Schmidhuber, 1997,Hahnloser et al., 2000], and gradient-based algorithms have alleviated this issue significantly. Nowadays, it is possible to train neural networks with dozens of stacked layers, over sequences of hundreds of elements.

4.3.5 NEURAL NETWORKS FOR HANDWRITTEN TEXT

During many years recurrent artificial neural networks based on Multidimensional Long-Short Term Memories dominated the state-of-the-art solutions to model handwritten text [Graves and Schmidhuber, 2009, B. et al., 2014, Voigtlaender et al., 2016]. Multidimensional recurrent layers are a variant of recurrent units designed to process signals of an arbitrary size and an arbitrary number of dimensions. Observe that, for both simple RNNs and LSTMs described before, the output at a given time-step depends on the current input and the state of the layer at the previous time-step. However, it is not clear what "the previous time-step" means with signals spanning across more than one axis. For instance, if we process an image (a two-dimensional input) from the left-top to the right-bottom corner, what would be the "previous" pixel? Multidimensional RNNs, and particularly Multidimensional LSTMs (MDLSTMs), are designed so that at each coordinate, the output depends on the input at the same coordinate and the outputs at the coordinates with a delay equal to 1 in each dimension, as illustrated in fig. 4.10. In addition, similarly to bidirectional RNNs described earlier, multidimensional RNNs can process the input signal in different directions (2^D different directions, for a signal with D coordinate axes).

Despite its wide adoption and success for handwritten text applications, in [Puigcerver, 2017] we argued that these powerful architectures were likely unnecessary to accurately model handwritten text. There are two main observations behind this hypothesis.

First, notice that human languages have an intrinsic sequential nature. Long before humans developed handwriting we communicated only using sounds, which are continuous vibrations of the air (or other mediums) through *time*. In order to allow for non-direct communication, humans later developed handwriting and represented the different phonemes of their language using combinations of a set

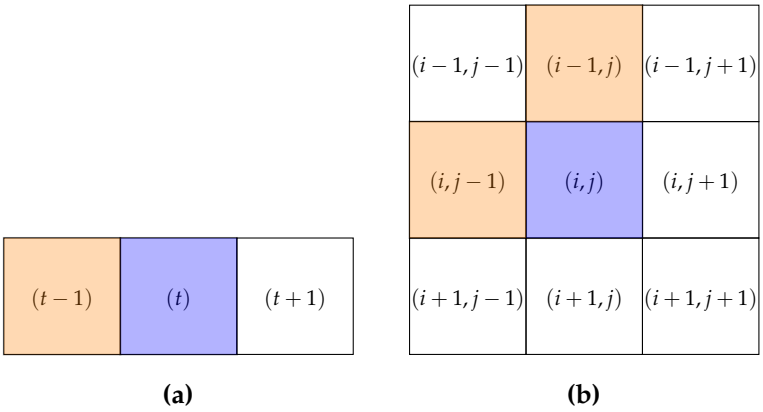
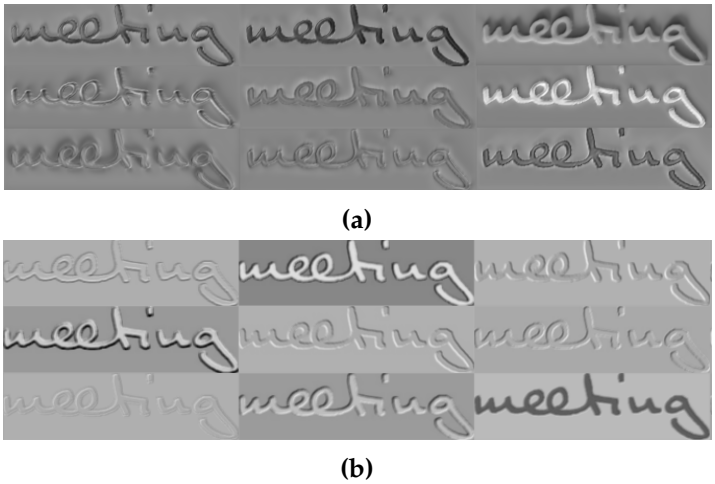


Figure 4.10. Coordinates in one-dimensional (left) and two-dimensional (right) input signals of a multidimensional recurrent layer. In the one-dimensional image (e.g. a sequence) there is a single "previous" element when processing the sequence, for instance, from left-to-right. However, in the two-dimensional signal (e.g. an image) there are two "previous" elements when processing the image from the top-left to the bottom-right corner.

of graphemes (such as characters) to be able to communicate using two-dimensional surfaces (such as paper sheets, stones, seeds, etc.). Thus, although two-dimensional information can be useful to model individual symbols (or a small group of symbols) in a given handwritten alphabet, it should not be necessary to model the language itself, since the latter has a sequential foundation.

Indeed, when one compares the output of a 2D-LSTM layer (a MDLSTM designed for processing images) trained for a handwritten text recognition task, with that of a comparable convolutional layer, the results are very similar, as shown in fig. 4.11. The figure suggests that the features learned by a 2D-LSTM layer use only a quite small 2D context, which can be mimicked with a simple convolutional layer with a small receptive field (3×3).

In fig. 4.11, the outputs of the 2D-LSTM show some "volumetric" effects that cannot be replicated with the small receptive field used by the convolutional layer. However, as shown in [Puigcerver, 2017], this limitation does not hinder the capability of the model to achieve the same accuracy as MDLSTMs.



(a)

(b)

Figure 4.11. Comparison of the outputs of a 2D-LSTM layer trained for a handwritten text recognition task (top), with that of a comparable convolutional layer (bottom).

Given these facts, we use an architecture composed of a stack of convolutional (and pooling) layers followed by a stack of one-dimensional bidirectional LSTM layers (BLSTMs), as shown in fig. 4.12. Pooling layers (in particular, Maximum pooling) are used very often to summarize the output of neighboring pixels and, thus, reducing the size of the images as they are processed by several layers of the neural network [Riesenhuber and Poggio, 1999]. The BLSTMs layers in our architecture process the images column-wise (i.e. all channels of all pixels of a column form a single vector). A similar architecture to this was first used in [Shi et al., 2017] and other variants have also been used more recently in different HTR applications [Bluche and Messina, 2017]. We call this *Convolutional and Recurrent Neural Network* architecture a CRNN.

4.4 Key differences between HMMs and NNs with CTC

As we described in section 4.3.4.1, when the CTC algorithm is used on top of a neural network, we can interpret the output of the neural network in a probabilistic fashion. In particular, recall that the output at each time-step is a posterior probability distribution over a set of

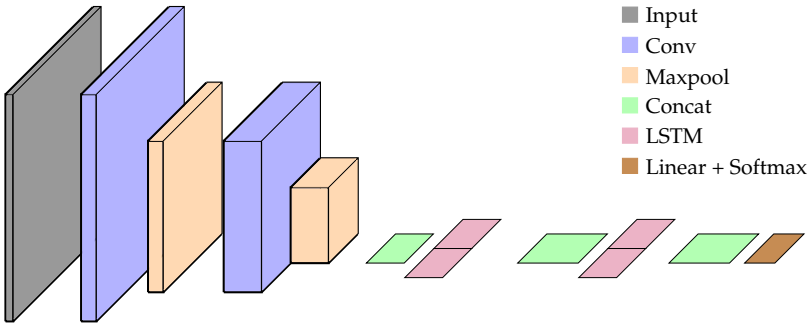


Figure 4.12. Diagram of the architecture of the artificial neural network used to model the transcripts of a handwritten text line. This model contains two convolutional blocks, followed by two BLSTM layers, and a final linear layer. Convolutional layers ("Conv") also include the ReLU activation function. If a large number of convolutional blocks is used, not all of them include the pooling layer ("Maxpool") to avoid discarding too much information. The concatenation layer ("Concat") after the last convolutional block concatenates all channels from all pixels of an individual column into a single vector. The other concatenation layers append the outputs of a BLSTM layer in each direction.

labels, given a feature vector. We argued before, that the output labels are similar to the states in HMMs, and can account for different alignment hypotheses of the transcript of a given image.

If one focuses only on the *probabilistic* models, observe that the CTC makes a very naive assumption with respect to a given sequence of labels: it assumes that the probability of the label at time t depends only on the corresponding feature vector, and not on the other vectors or labels. HMMs have a slightly weaker independence assumption: the density of a feature vector only depends on the state that emitted it, but the sequence of states follows a first-order Markov process (i.e. the probability of each state depends on the state before).

One might think that, since the probabilistic model of the CTC makes stronger assumptions than that of the HMMs, the latter could perform better when these assumptions do not hold. Yet, in reality, models trained with CTC typically perform better than HMM-based models for recognition tasks. How can this be explained?

Before, we mentioned the difference in the probabilistic models. However, when the CTC is used on top of deep and recurrent neural networks, the parameters of this neural network are adjusted to produce a sequence of feature vectors that helps the CTC probabilistic model to represent the transcript of the image accurately, even with its naive assumptions. On the other hand, the feature vectors in HMM-based solutions are fixed during training, and the free parameters of the probabilistic model are adjusted to maximize the likelihood of the fixed feature vectors (or the posterior of the reference transcript, if discriminative training is employed).

In his thesis, Théodore Bluche highlights the similarities between the CTC algorithm used during training and the Forward–Backward algorithm applied to hybrid models of HMMs and neural networks: essentially, CTC training is equivalent to Forward–Backward training on a special HMM model with a single state per character and an optional state between characters (the "CTC-blank"), and no transition probabilities betwen states (see chapter 7.3 in [Bluche, 2015]).

Although the independence assumption made by the CTC may not be critical for achieving a decent recognition accuracy, combining the output of the neural network with an external language model can improve the recognition results. For KWS applications, the independence assumption can be very hurtful since we are not only interested in obtaining the one-best transcription hypothesis, regardless of its particular posterior probability. But, we need to have an accurate model for the complete posterior probability distribution.

This is illustrated by fig. 4.13. The figure shows three probability distributions over five possible transcripts. The reference distribution assigns all the probability mass to a single event: the reference transcript. The two models assign different probabilities to different hypotheses, but in all cases the *order* of the hypotheses is the same as in the original distribution. Thus, the recognition accuracy achieved by the two models will be the same (100% accuracy), but the Kullback–Leibler divergence of the second model is much larger than the first one.

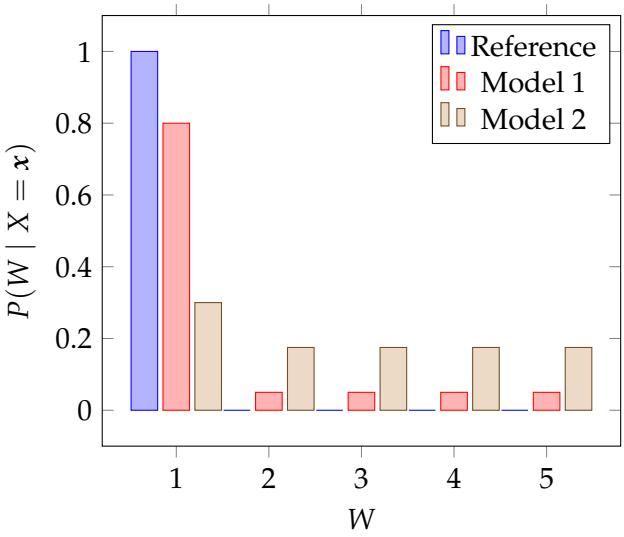


Figure 4.13. Example illustrating that achieving a good recognition accuracy does not necessary imply having a good representation of the target probability distribution. The two models will achieve a perfect recognition accuracy, but the first model is a better approximation to the reference distribution (has a lower Kullback–Leibler divergence).

4.5 N-gram Language Models

The effects of the independence assumption made by the CTC algorithm can be mitigated by combining the output distribution of the NN with an additional language model. A language model is just a prior distribution over all (meaningful) sequences of words (or characters) of a given language, or collection of documents.

In the case of generative models such as HMMs, using a prior distribution over the transcripts is mandatory, if one follows Bayesian Decision Theory. Given that HMMs (with GMMs) model the conditional density, one can use Bayes rule to obtain the transcript posterior:

$$P(W = w \mid X = \boldsymbol{x}) = \frac{p(W = w, X = \boldsymbol{x})}{p(X = \boldsymbol{x})} = \frac{p(X = \boldsymbol{x} \mid W = w)P(W = w)}{\sum_{w'} p(X = \boldsymbol{x} \mid W = w')P(W = w')} \tag{4.26}$$

where $P(W)$ is modeled by our language model.

One of the most popular choices to model this prior distribution is the n-gram, which was first used by Shannon to model the English language [Shannon, 1951]. An n-gram language model assumes that each symbol (word or character) in a given sequence only depends on the previous $n-1$ symbols. Thus, using the chain rule on the prior probability, one obtains:

$$\begin{aligned} P(W = w) = P(W = w_{1:L}) = \\ P(w_1) \prod_{i=2}^{L} P(w_i \mid w_1, \ldots, w_{i-1}) \stackrel{\star}{=} \\ P(w_1) \prod_{i=2}^{L} P(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) \end{aligned} \tag{4.27}$$

N-grams are typically estimated using the Maximum Likelihood Estimation criterion, on a given training set of strings. The MLE criterion yields to the following expression to estimate the conditional probability $P(w_i \mid w_{i-n+1}, \ldots, w_{i-1})$.

$$P(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_i, w_{i-1}, \ldots, w_{i-n+1})}{\sum_{w'} C(w', w_{i-1}, \ldots, w_{i-n+1})} \tag{4.28}$$

where $C(\cdot)$ represents the number of times that the given sequence of symbols appeared in the training set of strings.

Plain MLE presents a serious issue when the context (n) grows and the amount of training data is kept constant. Observe that if the number of symbols in the alphabet is $|\Sigma|$, there are $|\Sigma|^n = 2^{n \log_2 |\Sigma|}$ possible different contexts. For instance, the modern English language is made of more than 170 000 words. Using a modest 3-gram model gives roughly $5 \cdot 10^{15}$ different contexts, thus the likelihood of missing some n-grams increases exponentially with n.

The simplest technique to address this issue is additive smoothing, which consists in adding a small value to all counts, although more sophisticated (and preferred) methods exists, such as discounting, back-off and interpolation with lower-order n-grams. [Chen and Goodman, 1999] offer an excellent overview and empirical comparison of various smoothing techniques. In particular, in our experiments we have

used the modified Kneser–Ney [Kneser and Ney, 1995] or Witten–Bell [Witten and Bell, 1991] smoothing and interpolation.

N-grams have been widely used in speech recognition [Jelinek, 1976, Katz, 1987], HTR [Marti and Bunke, 2001b, Marti and Bunke, 2001a], machine translation [Brown et al., 1990, Brown and Frederking, 1995], and many other pattern recognition applications, including Keyword Spotting [Fischer et al., 2013, Toselli et al., 2015].

4.5.1 Combining the output of a neural network with a n-gram

We mentioned before that n-gram language models are very often combined with neural networks trained with CTC. In principle, this would not be necessary because networks trained with CTC already model the *posterior probability distribution* of the text given the input image. Thus, in theory, the underlying prior distribution (which is explicitly modeled by the language model), should be taken into account implicitly by the neural network.

However, it has been observed that this is not always the case and combining (in a probabilistic manner) the output of neural network with an additional n-gram language model improves the recognition accuracy [Puigcerver, 2017], as well as the keyword spotting performance (see experiments in section 8.3). There are two reasons that explain this seemingly counter-intuitive fact.

Firstly, notice that the underlying independence assumptions of the CTC algorithm are very different than those made by n-grams, especially when a large context is considered.

Secondly, and most importantly, very often we have access to additional text-only data, which can not be used to train the recurrent neural network. In such case, a language model trained with the additional text may represent a richer prior than the captured by the neural network.

In order to combine the two models in a Bayesian manner, we need to obtain a density distribution $p(X \mid A)$ from the output of the neural network $P(A \mid X)$. Recall that A is the random variable over the sequence of *labels* of the CTC algorithm. Applying Bayes rule to

the latter distribution we get:

$$\begin{gathered} p(X = \boldsymbol{x} \mid A = a_1, \ldots, a_n) = \\ \frac{p(A = a_1, \ldots, a_n, X = \boldsymbol{x})}{P(A = a_1, \ldots, a_n)} = \\ \frac{P(A = a_1, \ldots, a_n \mid X = \boldsymbol{x}) p(X = \boldsymbol{x})}{P(A = a_1, \ldots, a_n)} \propto \\ \frac{P(A = a_1, \ldots, a_n \mid X = \boldsymbol{x})}{P(A = a_1, \ldots, a_n)} \end{gathered} \tag{4.29}$$

In practice, we use the following approximations to the density $p(X = \boldsymbol{x} \mid A = a_1, \ldots, a_n)$ and the label prior $P(A = a_1, \ldots, a_n)$:

$$p(X = \boldsymbol{x} \mid A = a_1, \ldots, a_n) \approx \frac{P(A = a_1, \ldots, a_n \mid X = \boldsymbol{x})}{P(A = a_1, \ldots, a_n)^{\beta}} \tag{4.30}$$

$$P(A = a_1, \ldots, a_n) \approx \frac{1}{M} \sum_{m=1}^{M} P(A = a_1, \ldots, a_n \mid X = \boldsymbol{x}^{(m)}) \tag{4.31}$$

where $\{\boldsymbol{x}^{(m)} : 1 \leq m \leq M\}$ is the set of training examples, and the hyperparameter[8] β is tuned to maximize the performance on the corresponding task.

Then, we simply combine this (through WFST composition) with HMMs, (sometimes) a word lexicon and n-gram language models. This approach has been widely used in the past in many HTR works [Doetsch et al., 2014, Voigtlaender et al., 2016, Puigcerver, 2017, Bluche and Messina, 2017].

4.6 Weighted Finite State Transducers

Finite State Transducers (FST) represent a "mapping" between strings of two languages, and are an extension to Finite State Automatons (FSA) [Hopcroft et al., 2006]. Here, we use the formal definition of language: a set of sequences of symbols (strings) of a given alphabet.

[8] We use the term hyperparameter to refer to adjustable variables of a particular approach that are not the parameters of the statistical model itself.

Transducers were first introduced in [Shannon, 1948], although the current formal definition of FSTs is not completely compatible with that of Shannon's (which did not formalize them). Finite State Automatons are a representation of *regular (or rational) languages*, while Finite State Transducers are the formal representation of *rational relations* [Eilenberg, 1974].

Hidden Markov Models, n-grams, the CTC probabilistic model of a neural network, and many others can all be represented as Weighted Finite State Transducers (WFSTs), and most of the algorithms needed to estimate them, or make inferences once the models have been trained, can be formulated as algorithms operating on WFSTs [Mohri et al., 2008]. The term *weighted* simply means that each element of the relation represented by the FST has some weight value associated.

In fact, we have already used Weighted Finite State Transducers (or Automatons) through this dissertation, without properly defining them, since they are very intuitive tools to represent, for instance, the set of hypothetical transcripts of a given text image. WFSTs will play a crucial role in the next chapter, where the algorithms needed to build probabilistic indexes from text images are explained. Thus, it is very important that the reader understands the fundamentals of WFSTs.

4.6.1 DESCRIPTION

There are multiple ways of defining a WFST, in particular we will define a WFST, T, over a *semiring* $\mathbb{K}$, as a 6-tuple $T = (\Sigma, \Gamma, V, E, s_0, \rho)$ where:

- Σ is the finite input alphabet.
- Γ is the finite output alphabet.
- V is the finite set of states.
- $E \subseteq V \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times \mathbb{K} \times V$ is the finite set of transitions (also known as arcs or edges). ϵ is an special symbol indicating that no (input or output) symbol was consumed during the transition.
- $s_0 \in V$ is the initial state.

- $\rho : V \rightarrow \mathbb{K}$ is the final weight function.

Notice that a Weighted Finite State Automatons (WFSA), is just a WFST where the input and output labels of each transition are identical. Thus, we can define a WFSA, A, over a semiring $\mathbb{K}$, as a 5-tuple $A = (\Sigma, V, E, s_0, \rho)$.

A *semiring* $\mathbb{K}$ is an algebraic structure with operations $\oplus$ and $\otimes$ (called addition and multiplication), with the following properties:

- $(\mathbb{K}, \oplus, \bar{0})$ is a commutative monoid with identity element $\bar{0}$. That is: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$, $\bar{0} \oplus a = a \oplus \bar{0} = a$, and $a \oplus b = b \oplus a$, $\forall a, b, c \in \mathbb{K}$.
- $(\mathbb{K}, \otimes, \bar{1})$ is a monoid with identity element $\bar{1}$. $(a \otimes b) \otimes c = a \otimes (b \otimes c)$, and $\bar{1} \otimes a = a \otimes \bar{1} = a$, $\forall a, b, c \in \mathbb{K}$.
- The multiplication ($\otimes$) distributes over the addition ($\oplus$). That is: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ (left-distributivity) and $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ (right-distributivity), $\forall a, b, c \in \mathbb{K}$.
- The multiplication by $\bar{0}$ annihilates $\mathbb{K}$. That is: $\bar{0} \otimes a = a \otimes \bar{0} = \bar{0}$, $\forall a \in \mathbb{K}$

In addition, in some semirings a *division* operation can be defined. A semiring is called left divisible, right divisible or divisible when:

- Left divisible, *iff* $\forall a \in \mathbb{K} - \{\bar{0}\}$, $\exists b \in \mathbb{K}$ such that $b \otimes a = \bar{1}$, and b is unique (b is called the left-inverse of a).
- Right divisible, *iff* $\forall a \in \mathbb{K} - \{\bar{0}\}$, $\exists c \in \mathbb{K}$ such that $a \otimes c = \bar{1}$, and c is unique (c is called the right-inverse of a).
- Divisible, *iff* it is both left and right divisible and the left- and right-inverses are equal, $\forall a \in \mathbb{K} - \{\bar{0}\}$.

In a divisible semiring (left, right or both), we can define a corresponding inverse operation to the multiplication, the division, represented by the operator $\oslash$. Common semirings used in WFST are

shown in table 4.1. The *Tropical* and *Log* semirings are are widely used in the field of speech recognition and HTR (and are used in the following chapter), since they allow to define different operations and algorithms using a small set of operations on WFST. For instance, marginalization and the Viterbi decoding are equivalent to solving the shortest distance and the shortest-path problems on a WFST in the log and the tropical-semiring, respectively [Mohri, 2002].

Table 4.1. Common semirings used in Weighted Finite State Transducers.

Semiring	$\mathbb{K}$	$\bar{0}$	$\bar{1}$	$a \oplus b$	$a \otimes b$	$a \oslash b$
Real	$\mathbb{R}$	0	1	$a + b$	$a \cdot b$	$\frac{a}{b}$
Tropical	$\mathbb{R} \cup \{\infty\}$	∞	0	$\min\{a, b\}$	$a + b$	$a - b$
Log	$\mathbb{R} \cup \{\infty\}$	∞	0	$-\log(e^{-a} + e^{-b})$	$a + b$	$a - b$

As we already saw multiple times in the previous chapters, WFST have a very intuitive graphical representation. For example, fig. 4.14 represents a WFST with input alphabet $\Sigma = \{a, b, c\}$ and output alphabet $\Gamma = \{X, Y, Z\}$, five states $V = \{s_0, s_1, s_2, s_3, s_4\}$, with initial state s_0 and two final states (a state s is final *iff* $\rho(s) \neq \bar{0}$) with $\rho(s_3) = 1$ and $\rho(s_4) = 0$. The set of transitions is represented by the labeled arcs in the figure. For instance the transition $(s_0, a, X, 2, s_1) \in E$ is represented by the arc connecting states s_0 and s_1 labeled with $a : X/2$.

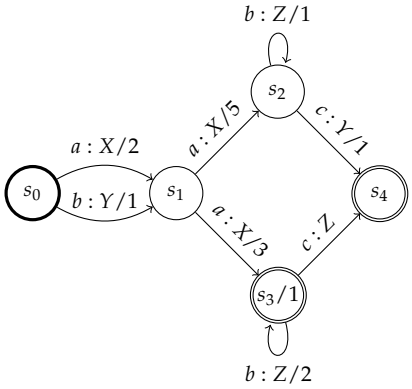


Figure 4.14. An example of a Weighted Finite State Transducer.

Usually, the transitions and final weights which are equal to $\bar{1}$ (i.e. 0 in the log or tropical semirings, as in the example) are omitted in the graphical representation. For instance, in fig. 4.14, notice that the final state s_4 and the transition between states s_3 and s_4 do not show any weight.

For convenience, given an edge $e \in E$, in our algorithms we typically use $p[e]$ to refer to the origin (or source, or previous) state, $n[e]$ to refer to the destination (or next) state, $\omega[e]$ to refer to its weight, and $l_i[e]$ and $l_o[e]$ to refer to its input and output symbols (or labels), respectively (in the case of a WFSA, we simply use $l[e]$, since $l_i[e] = l_o[e]$).

A *path* $\pi = e_1, \ldots, e_k$ is an element of E^* with *consecutive* transitions: $n[e_i] = p[e_{i+1}]$, $1 \leq i < k$.

The *weight of a path* $\omega[\pi]$ is the $\otimes$-product of the weights of the constituent transitions: $\omega[\pi] = \omega[e_1] \otimes \omega[e_2] \otimes \cdots \otimes \omega[e_k]$. Similarly, the *total weight of path*, $\hat{\omega}[\pi]$, is equal to the weight of the path and the final weight of the destination state of the last transition in the path: $\hat{\omega}[\pi] = \omega[e_1] \otimes \omega[e_2] \otimes \cdots \otimes \omega[e_k] \otimes \rho(n[e_k])$.

For instance, the weight of the path $\pi = (s_0, a, X, 2, s_1), (s_1, a, X, 3, s_3)$, in fig. 4.14, is equal to $\omega[\pi] = 2 \otimes 3 = 2 + 3 = 5$ and its total weight is $\hat{\omega}[\pi] = 2 \otimes 3 \otimes 1 = 2 + 3 + 1 = 6$.

Any path that starts at the initial (or start) state (i.e. $p[e_1] = s_0$) and has a total weight different than $\bar{0}$ is a *complete path*. Equivalently, a complete path is a path that starts in the initial state and ends in any final state. Sometimes, when we want to emphasize that a path *is not* complete, we will say that it is a *subpath*.

An important distinction between WFST is whether they are *acyclic* or not. A WFST is *acyclic* if there is no possible path in it that traverses the same state more than once. For instance, the WFST in fig. 4.14 *is not* acyclic, since the path $(s_1, a, X, 3, s_3), (s_3, b, Z, 2, s_3)$ goes through the state s_3 twice. It turns out that many problems related to WFST have efficient solutions for acyclic WFST, or some conditions are always met in such cases.

The rational relation defined by a (weighted) FST is given by its set of complete paths. For instance, fig. 4.14 transduces (relates) the string a, a, b into (to) X, X, Z with a weight equal to $2 \otimes 3 \otimes 2 \otimes 1 = 8$.

In the previous example, there was a single path representing the given relationship, but in general there may exist multiple complete paths. Then, we say that the FST is *ambiguous*. In such case, the weight of a given element of the relation (an input–output sequence of symbols) is the $\oplus$-sum of all complete paths representing that element. Thus, if T is a WFST and $(x, y) \in \Sigma^* \times \Gamma^*$ is an element of the relation represented by the WFST, its weight is defined as:

$$T(x, y) = \bigoplus_{\pi : l_i[\pi] = x \wedge l_o[\pi] = y} \hat{\omega}[\pi] \tag{4.32}$$

Observe that the WFST in fig. 4.14 is actually *unambiguous* since each complete path in the WFST represents a different pair of input–output strings.

Similarly to the previous definitions of ambiguity, we can define equivalent properties only looking at the input or output language of the WFST. For instance, we say that a WFST is *unambiguous on its input (output)* if, and only if, for each input (output) sequence of symbols there exists only a single path that accepts (produces) this string.

Another important distinction between WFSTs is whether they are *functional* or not. A WFST is *functional* if each input string relates only to a single output string (i.e. the relation that the WFST represents is a function). For instance, the WFST in fig. 4.14 *is not* functional, since the input sequence a, a, b, c is related to both output sequences X, X, Z, Y and X, X, Z, Z (with weights $2 + 5 + 1 + 1 + 0 = 9$ and $2 + 3 + 2 + 0 + 0 = 7$, respectively).

4.6.2 OPERATIONS

Next, we will briefly describe some of the most relevant operations on WFSTs which will be used in the following section to implement the algorithms developed in this thesis.

4.6.2.1 Composition

Composition is one of the fundamental operations to create complex WFST from simpler ones. This operation is used very often when different probabilistic models have to be combined.

Let $\mathbb{K}$ be a *commutative* semiring (i.e. a semiring with the property that $a \otimes b = b \otimes a, \forall a, b \in \mathbb{K}$). Let $T_1 = (\Sigma, \Omega, V_1, E_1, s_0, \rho_1)$ and $T_2 = (\Omega, \Gamma, V_2, E_2, s_0', \rho_2)$ be two WFSTs defined over $\mathbb{K}$, such that the output alphabet of T_1 is equal to the input alphabet of T_2, and assume that the sum $\bigoplus_{z \in \Omega^*} T_1(x, z) \otimes T_2(z, y)$ is well-defined and in $\mathbb{K}$ for all pairs of strings $(x, y) \in \Sigma^* \times \Gamma^*$. Then, the result of the composition of T_1 and T_2 is a WFST, denoted by $T_1 \circ T_2$, defined by:

$$[T_1 \circ T_2](x, y) = \bigoplus_{z \in \Omega^*} T_1(x, z) \otimes T_2(z, y) \tag{4.33}$$

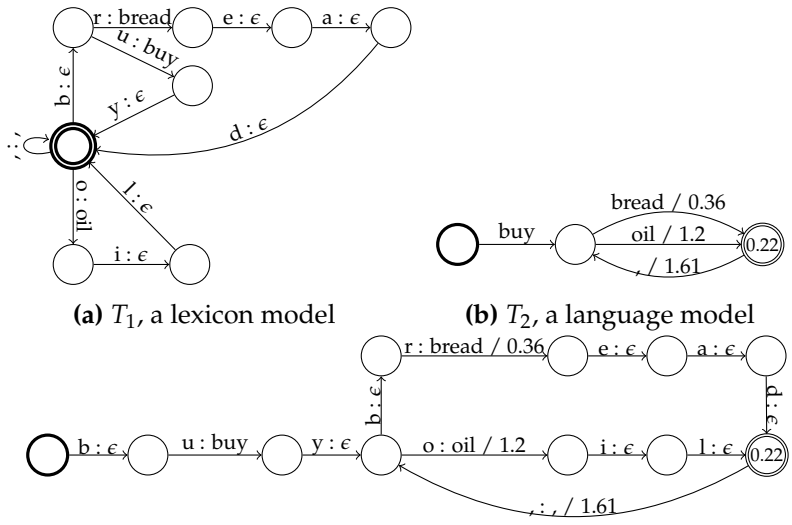
Each state in the WFST $T_3 = T_1 \circ T_2$ is represented by a pair of states from T_1 and T_2. Without taking into account transitions with input or output ϵ symbols (which require an special treatment), then:

$$\begin{aligned} (s_1, a, b, w, s_2) \in E_1 \wedge (s_1', b, c, w', s_2') \in E_2 \\ \Rightarrow \\ ((s_1, s_1'), a, c, w \otimes w', (s_2, s_2')) \in E_3 \end{aligned} \tag{4.34}$$

Thankfully, an efficient algorithm to perform the composition of arbitrary WFSTs exists [Mohri et al., 1996, Mohri et al., 2008], which is a generalization of the classical state–pair construction for the intersection of FSA [Hopcroft et al., 2006]. This algorithm has an asymptotic cost which is essentially[9] $\mathcal{O}(|V_1||V_2|D_1M_2)$, where $|V_i|$ is the number of states, D_i is the maximum output degree (maximum number of output edges from any state), and M_i is the maximum output multiplicity (maximum number of output edges with the same label) of the i-th WFST in the composition.

Figure 4.15 shows one of the many uses of the composition operation when handwritten text is modeled as the combination of different probabilistic models. In the figure, a *lexicon model*, which maps

[9] Depending on the implementation, the cost may be slightly different.



(a) T_1, a lexicon model

(b) T_2, a language model

(c) Composition of T_1 and T_2

Figure 4.15. Example of the composition of two Weighted Finite State Transducers.

characters into words, is composed with a *language model*, which is used to represent the set of possible sentences (and their respective probabilities). This is, of course, a very simple example of the composition operation. In practice, the resulting WFST would be further composed with others, representing other distributions of the composite model.

4.6.2.2 Shortest path and distance

The *shortest path* problem has been deeply studied in Computer Science. Generally speaking, given a graph with several nodes the goal is to obtain the path with the smallest total weight between a source node and the rest of the nodes in the graph. The problem was first tackled in [Dijkstra, 1959] for directed graphs with non-negative real weights. Generally speaking, we say that the total weight of the shortest path from node s to node s' is the *shortest distance* from s to s'.

The shortest path problem arises very often in speech and handwritten text recognition applications. Particularly, the Viterbi algorithm [Viterbi, 1967] finds the sequence of states in a probabilistic

model (e.g. a HMM or a combination of these) that maximizes the conditional density of the observed data. Thus, if we are able to express a probabilistic model as a WFST, we can then apply the generic shortest path algorithm to solve the same problem as the Viterbi algorithm, by minimizing the negated (logarithm of the) likelihoods [Mohri et al., 2008].

Take for instance the transducer represented in fig. 4.16 and assume that its weights are in the tropical semiring (i.e. $a \oplus b = \min\{a, b\}$, $a \otimes b = a + b$). Then, there are two paths with a total weight corresponding to the smallest: the path $\pi_1 = (s_0, a, X, 1, s_1)$, with a total weight of $1 \otimes 1 = 2$; and the path $\pi_2 = (s_0, a, X, 1, s_1), (s_1, c, Z, \bar{1}, s_1)$, with a total weight of $1 \otimes \bar{1} \otimes 1 = 1 + 0 + 1 = 2$.

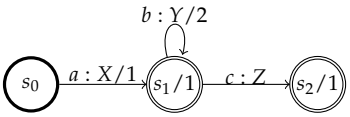


Figure 4.16. Example of a cyclic WFST which admits a shortest distance algorithm in the tropical and log semirings.

The single shortest path algorithm can be extended to obtain the n-shortest paths, and we actually use this algorithm in several of our probabilistic indexing algorithms. The asymptotic temporal cost of the n-shortest paths algorithm is $\mathcal{O}(|V| \log |V| + n|V| + n|E|)$.

The interesting particularity of WFSTs is that the shortest distance problem is not defined in terms of finding the smallest total weight of the path between two states, but in terms of computing the $\oplus$-sum of the weights of all paths between the two:

$$d[s, s'] = \bigoplus_{\pi : p[\pi] = s \wedge n[\pi] = s'} \omega[\pi] \tag{4.35}$$

On the one hand, if we assume that the weights are in the *tropical* semiring, the $\oplus$-sum defined above obtains the minimum weight (recall that $a \oplus b = \min\{a, b\}$). On the other hand, if we compute the same sum in the *log* semiring, we obtain the "log-sum-exp" of the weights (since $a \oplus b = -\log(\exp(-a) + \exp(-b))$). When transducers are acyclic the $\oplus$-sum from one state to the rest, can be computed very efficiently in $\mathcal{O}(|V| + |E|)$.

For instance, back to fig. 4.16 the shortest distance between state s_0 and s_2 in the tropical semiring is $1 \otimes \bar{1} = 1$, but the shortest distance in the log semiring is equal to:

$$(1) \oplus (1 \otimes 2) \oplus (1 \otimes 2 \otimes 2) \oplus \cdots = \bigoplus_{k=0}^{\infty} 1 \otimes \overbrace{2 \otimes \cdots \otimes 2}^{k \text{ times}} \approx 2.232 \tag{4.36}$$

Although we can loop indefinitely on the state s_1, the total cost in the log semiring finally converges.

This is very convenient in text and speech applications, since very common algorithms such as the Forward and Backward [Baum and Eagon, 1967, Baum et al., 1970], and the Viterbi algorithms have the same implementation, but using different semirings.

Although further details of these operations are not required to understand the algorithms presented in the next chapter, we recommend the reader to review the works of [Mohri, 2002, Mohri and Riley, 2002, Mohri, 2004, Mohri et al., 2008], if additional information is needed.

4.6.2.3 Determinization

We say that a FST is deterministic in its input (or its output) if there is no node with two output transitions with the same input (or output) label [Mohri, 2004]. This is analogous to the definition of deterministic automaton [Hopcroft et al., 2006].

The *determinization* operation consists on finding a deterministic and *equivalent* WFST to the given one. By equivalent, we mean that it represents the same weighted rational relation (i.e. the same probability distribution, in the case of probabilistic models).

Unlike in the unweighted case, some weighted transducers (and automata) are not determinizable (i.e. there is not any equivalent WFST which is deterministic). Luckily, we use the determinization operation typically on lattices, which are acyclic WFST, and these are always determinizable [Allauzen and Mohri, 2002].

A deterministic WFST has very interesting properties. In particular, a deterministic WFST (or automaton) is also unambiguous (in its input or output).

The determinization algorithm is an extension of the powerset construction [Rabin and Scott, 1959], used to determinize finite state automata. Thus, the worst case asymptotic cost is exponential with the number of states in the WFST, i.e. $\mathcal{O}(\exp(|V|))$. However, for our purposes this worst case scenario is rarely reached in practice, as we will see in the next chapter. In particular, we will use the determinization algorithm to create the probabilistic indexes from lexicon-free models (see section 5.3).

4.6.3 THE CTC ALGORITHM AS ELEMENTARY WFST OPERATIONS

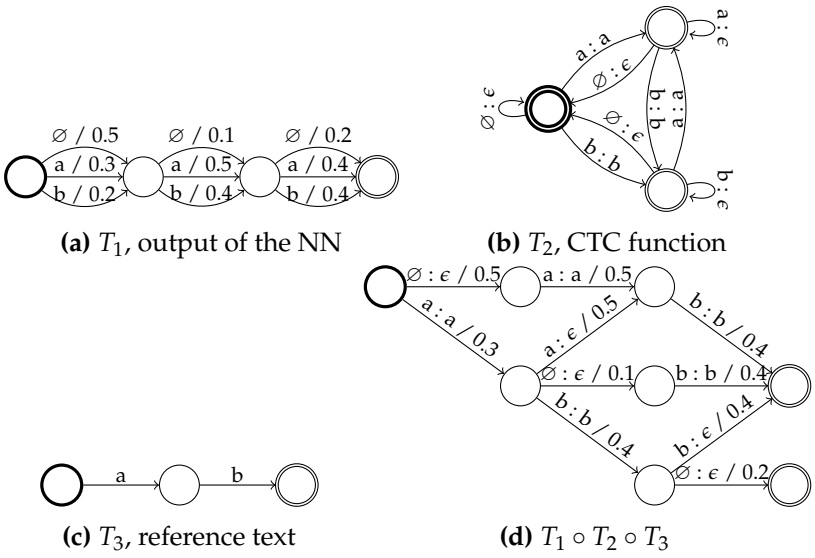
Earlier in this chapter, we described how HMM-based systems can be implemented using WFST. It turns out that the CTC algorithm can also be implemented as elementary operations on WFST.

In section 4.3.4.1 (particularly, fig. 4.9), we have already represented the output of the neural network using the CTC algorithm as a WFST, over the set of sequences of *labels*.

Nevertheless, recall that *labels* are not the same as *characters*, since a special CTC-blank symbol was added to account for the multiple alignments of a sequence of characters. To map from labelings to sequences of characters, we presented a simple function that the CTC uses.

In fact, this function can be represented as a WFST. Then, we can apply this function to a given NN output by simply composing the two WFST. If we want to obtain the set of labelings of a given reference transcript (in order to compute the sum of all their probabilities), we can then simply compose the resulting WFST with a transducer representing the reference transcript, as depicted in fig. 4.17.

Then, we can compute the (log of the) sum of all probabilities using the Forward or Backward algorithms. This, as we mentioned before, simply means to compute the shortest distance in the log semiring.



(a) T_1, output of the NN **(b)** T_2, CTC function

(c) T_3, reference text **(d)** $T_1 \circ T_2 \circ T_3$

Figure 4.17. Example of the CTC algorithm implemented as the composition of several Weighted Finite State Transducers.

In practice, the CTC algorithm is not implemented using these elementary WFST operations, because more efficient algorithms exist to treat this special case. In any case, treating the output of the neural network trained with the CTC as a WFST, allows us to obtain character lattices from the network without the need of any language model.

4.6.4 Lattices represented as WFST or WFSA

In the literature, there are multiple definitions of a lattice. Some authors [Ljolje et al., 1999] simply define a lattice as a labeled, weighted, directed acyclic graph containing the transcription hypotheses, and their (log-)probabilities or (log-)densities (depending on whether they represent $P(W \mid X)$ or $P(W, X)$). Most of the seminal works are applied to speech recognition, where it is sometimes useful to represent the "time" at which a certain word was uttered [Ortmanns et al., 1997]. In the traditional HTK toolkit, lattices can contain character and/or word "time" alignment information, as well as HMM or state-level alignment information [Young et al., 2002].

In our algorithms, we use the definition of lattices adopted in [Povey et al., 2012]. In this definition, lattices are defined as WFSTs where the output symbols are the words (or characters, in lexicon-free models), and the input symbols represent the most fine-grained hidden variable of our probabilistic model. For instance, if we use a HMM-based approach, each symbol in the input alphabet represents an individual state[10]. If we use a CTC-based approach, then the input alphabet represents *labels*, as we showed in fig. 4.17d.

Just as the aforementioned figure shows, ϵ-symbols are used at the output to cope with the fact the the number of output symbols (e.g. words) is typically smaller than the number of state transitions/labels. Epsilon symbols are not allowed at the input.

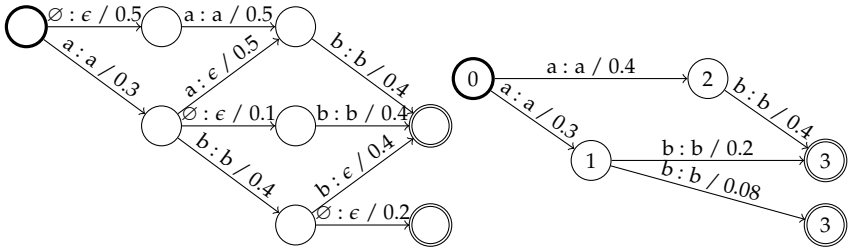
Given this definition, all paths leading to a given state have the same number of input symbols. Thus, we can easily associate a "time" (in speech) or "column" (in text) to each state. For instance, in fig. 4.17d, the WFST has a left-to-right order of the states. The "time" or "column" associated to the first (initial) state would be $t = 0$, the next two states would have a time $t = 1$, the next three states $t = 2$ and the last two (final) states would have a time $t = 3$, which is equal to the number of input feature vectors.

In most cases, the state/label symbols in the input of the transitions are only needed to determine the time-position of the states. Thus, we can represent the lattices in a more compact form (using less states and arcs). For the sake of brevity in our algorithms, we define a *compact lattice* as a WFSA representing only the possible sequences of words (or characters) of the transcription, and an associated function τ that contains the time of each state.

Figure 4.18 represents the lattice from fig. 4.17d and its equivalent compact lattice. Notice that the alignment information about the input symbols (CTC labels) is lost, but the time-alignment information is kept (τ is represented by the numbers inside each state).

Our definition of a compact lattice is not exactly as implemented, but we decided to avoid unnecessary details about the implementa-

[10] Actually, a transition between two states, for convenience.



(a) Lattice, T **(b)** Compact lattice, A

Figure 4.18. Example of a lattice, represented as a WFST (fig. 4.18a), and an equivalent compact lattice, represented as a WFSA (fig. 4.18b).

tion, for the sake of clarity in our algorithms. There is an efficient algorithm that converts an arbitrary lattice into its compact form.

5 Indexing for Fast Keyword Spotting

In chapter 2 we presented the framework of Probabilistic Keyword Spotting, which introduced theory-grounded principles to rank a set of results given a user's query. Essentially, the distinct flavors of relevance probabilities need a way of representing the posterior distribution over the transcripts, w, of a text image, $\boldsymbol{x}$. That is, a model of the distribution $P(W \mid X)$.

In chapter 4, and particularly in section 4.6, we saw that this distribution, for a given text image, can be efficiently represented using Weighted Finite State Transducers (WFST).

In principle, for each query, we could first build the representation of $P(W \mid X)$ as a WFST and then compute the $P(R = 1 \mid X = \boldsymbol{x}, V = v, \ldots)$ using standard operations on WFST (composition and shortest distance).

However, notice that $P(W \mid X)$ is independent of the particular query that a user might be interested in, and thus can be precomputed. In addition, if we want to build very fast applications that can be used in practical scenarios, we need to be able to compute, or approximate, $P(R = 1 \mid X = \boldsymbol{x}, V = v, \ldots)$ really fast.

Actually, we could even try to precompute this relevance probability in advance, for all keywords that are likely to be present in each of the documents of our collection. Then, we can simply store this information in a (probabilistic) search index, and let the users query this index directly.

In this chapter, we will introduce different algorithms to build such indexes, for each of the flavors of the relevance probability in-

troduced in chapter 2. These algorithms are all based on standard WFST algorithms, thus it is vital to understand the operations and algorithms described in section 4.6. If the reader has skimmed through the thesis, we highly suggest to go back and review the previous section.

5.1 Indexing lexicon-based lattices

5.1.1 POSITION-INDEPENDENT RELEVANCE

Here, we are interested in building an index that contains whole-region relevance probabilities, for each of the possible words written in the image. Essentially, we need an algorithm to compute, for all possible words v in a particular vocabulary Σ, the following probability (recall eq. (2.7)):

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) = \sum_{w \in L(v)} P(W = w \mid X = \boldsymbol{x})$$

Using basic WFST operations, this can be done quite easily and efficiently. First, we need to obtain the set of paths from the WFST that contain the label v in any of its arcs (we will assume that the transcript is represented by present the output language of the WFST). In order to do so, we can compute the composition between the input WFST and a deterministic automaton representing the language $L(v) = \Sigma^* \, v \, \Sigma^*$, which we will refer to as Q. Figure 5.1 represents the minimal deterministic[1] automaton for such language.

The result of this composition will be a new WFST containing all paths from the original WFST that contain at least one arc with the label v in its output. Then, we just need to compute the total sum of weights of all its paths (using the Backward algorithm, for instance).

In order to build an index for a given WFST, we can repeat this procedure for all the word labels in any of the edges of the input WFST. Algorithm 5.1 describes this procedure.

[1]It is crucial that Q is deterministic in order to avoid duplicated paths in the resulting WFST. However, it's not required that the automaton is minimal.

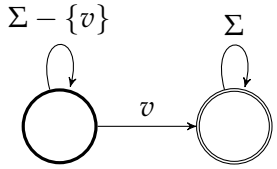


Figure 5.1. Minimal Deterministic Automaton accepting all sequences containing the symbol v, that is the language $\Sigma^* \; v \; \Sigma^*$. When a FST representing the set of hypotheses of a text image is composed with this automaton, the resulting FST includes all the original hypotheses (and their costs) that include at least one instance of the symbol v.

Algorithm 5.1 Compute a word index for whole text regions based on word compact lattices.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a *compact lattice*.

1: **procedure** LATTICEWORDINDEX(A)
2: $\beta \leftarrow$ BACKWARD(A) ▷Backward likelihood vector of each state
3: $S \leftarrow$ LABELSET(A) ▷Set of all labels in A
4: $I \leftarrow \emptyset$
5: **for all** $v \in S$ **do**
6: Let Q_v be the automaton represented in fig. 5.1.
7: $C \leftarrow A \circ Q_v$ ▷Composition
8: $\beta' \leftarrow$ BACKWARD(C)
9: Let s'_0 be the initial state of C.
10: SETINSERT($I, (v, \beta'_{s'_0} \oslash \beta_{s_0})$) ▷Normalize to get probability
return I

5.1.1.1 *Asymptotic cost of algorithm 5.1*

Observe that the cost of steps 2 and 3 is $\mathcal{O}(|V| + |E|)$, where $|V|$ is the number of states and $|E|$ the number of arcs in the WFSA. The cost of the composition operation in line 7 is also $\mathcal{O}(|V| + |E|)$, if special labels are used to represent the Σ-arcs in fig. 5.1, and a special matching algorithm is used during composition (i.e. ρ-composition).

In addition, if a WFST is trim (i.e. all states are accessible), then $|E| \geq |V| - 1$. Thus, the asymptotic cost of the entire algorithm is $\mathcal{O}(|S| \cdot |E|)$. Finally, since $|S| \leq |E|$, then we can simplify this expression as $\mathcal{O}(|E|^2)$.

5.1.2 LEXICON-BASED SEGMENT RELEVANCE

In section 2.2.2 we outlined the case that sometimes is required to highlight the location where a keyword's instance was found within a text region. In this scenario, we cannot use algorithm 5.1, since we need an algorithm that generates an index of the relevance probabilities given by eq. (2.16), for all possible keywords and keyword instances alignments.

In order to efficiently do this, we will use the lattices generated from a text line image, and build an index from them. Since we need information about the *alignment* of each word instance within the image, we will assume that, in addition to the WFST, we have a vector τ denoting the image column aligned to each state of the WFST (i.e. the lattice), as explained at the end of the previous chapter.

Recall from eq. (2.16), for a given word v and segment between the columns (c_0, c_1), we need to simply sum the posterior of all transcripts and alignments that contain an instance of that keyword in that precise location:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\sigma = (c_0, c_1)) = \sum_{\substack{w_{1:n} \in \Sigma^*, \exists k: \\ w_k = v}} \sum_{\substack{a_{1:n+1} \in \mathbb{N}^{n+1}: \\ a_k = c_0 \wedge a_{k+1} = c_1}} P(W = w_{1:n}, A = c_{1:n+1} \mid X = \boldsymbol{x})$$

The algorithm that computes this relevance probability, for all word present in the lattice, is depicted in algorithm 5.2. Since each arc is associated to a particular word and alignment (c_0 is the column associated the arc's origin state, and c_1 is the one associated the arc's destination), the algorithm simply traverses all the arcs in the lattice and accumulates the likelihood of all paths through the arc, and finally normalizes the joint likelihoods into posteriors.

5.1.2.1 Asymptotic cost of algorithm 5.2

The asymptotic cost of algorithm 5.2 is clearly linear in the number of states and arcs of the input weighted automaton (i.e. lattice). First, observe that both the forward and backward algorithms have an asymptotic cost of $\mathcal{O}(|V| + |E|)$, for acyclic weighted automata. Then, the

Algorithm 5.2 Compute a word index for text segments based on word compact lattices.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a *compact lattice*, τ is the associated function determining the frame aligned to each state in V.

1: **procedure** LATTICEWORDINDEXSEGMENT(A, τ)
2: $\alpha \leftarrow$ FORWARD(A) ▷Forward vector of each state
3: $\beta \leftarrow$ BACKWARD(A) ▷Backward vector of each state
4: $U \leftarrow \emptyset$ ▷Unnormalized index
5: **for all** $e \in E$ **do**
6: $v \leftarrow l[e]$ ▷Word associated to the arc
7: $i \leftarrow p[e]$ ▷Source state of the arc
8: $j \leftarrow n[e]$ ▷Destination state of the arc
9: $\omega \leftarrow w[e]$ ▷Weight (likelihood) of the arc
10: $t \leftarrow \alpha_i \cdot \omega \cdot \beta_j$ ▷Total likelihood through the arc
11: MAPINSERTORSUM($U, (v, \tau[i], \tau[j]), t$)
12: $I \leftarrow \emptyset$
13: **for all** $((v, \tau_s, \tau_e), \omega) \in U$ **do**
14: MAPINSERT($I, (v, \tau_s, \tau_e), \omega \oslash \beta_{s_0}$) ▷Segment relevance
return I

loop in step 5 of the algorithm traverses all the arcs in the WFSA again and performs different operations which are all done in constant time, and inserts (or updates) an element to a map structure (which can be done in $\mathcal{O}(1)$ if using hash tables, for instance). Finally, the loop in step 13 just normalizes the scores computed in the previous step to obtain posterior probabilities. Observe that the number of elements in the map U (the unnormalized index), will be at most $|E|$. Thus, the total cost of the algorithm is $\mathcal{O}(|V| + |E|)$.

Again, notice that the number of arcs is generally larger than the number of states in the WFSA. Thus, we can approximate the total asymptotic cost of algorithm 5.2 using a single variable as $\mathcal{O}(|E|)$.

5.1.2.2 Relevance conditioned on individual columns

In section 2.2.1, a relevance probability conditioned on individual columns was also presented. More precisely, eq. (2.13) was intro-

duced to compute it:

$$P(R = 1 \mid X = \boldsymbol{x}, V = v, L_\tau = c) = \sum_{\substack{w_{1:n} \in \Sigma^*, \exists k: \\ w_k = v}} \sum_{\substack{a_{1:n+1} \in \mathbb{N}^{n+1}: \\ a_k \leq c < a_{k+1}}} P(W = w_{1:n}, A = c_{1:n+1} \mid X = \boldsymbol{x})$$

Computing this relevance probability can be done using an algorithm very similar to algorithm 5.2, and has been deeply studied in the literature [Hazen et al., 2009, Toselli et al., 2013, Toselli et al., 2016b]. The difference is that once a given arc is traversed, we need an additional loop to increase the accumulator corresponding to each of the columns in that arc's segment.

5.1.3 LEXICON-BASED TRANSCRIPT POSITION RELEVANCE

A relevance probability of a transcript position was presented in section 2.2.3. Here, the relevance of a word position relative to the number of words in the transcript is presented. This relevance probability is very interesting since it allows to consider the relevance of word instances regardless of their multiple alignment hypotheses.

In addition, as we will see later, this relevance probabilities enable us to build probabilistic positional indexes, very similar to the ones used in traditional search engines, which enable the system to operate with phrase queries (i.e. searching for sequences of words).

5.1.3.1 Disambiguating word position associated to states

Lattices are WFSA for which all arcs entering a given state are aligned to the same column of the text line image. That is, all the words that enter a particular state *end* at the same physical position.

However, in general, the different arcs entering a state could be part of paths whose number of words is different, as in the example shown in fig. 5.2.

Algorithm 5.3 takes an input lattice, whose arcs represent words in the set of transcript hypotheses, and produces an equivalent WFSA such that all input paths to each state have the same number of words.

In short, the algorithm does that by decoupling the states where there are input paths with different lengths.

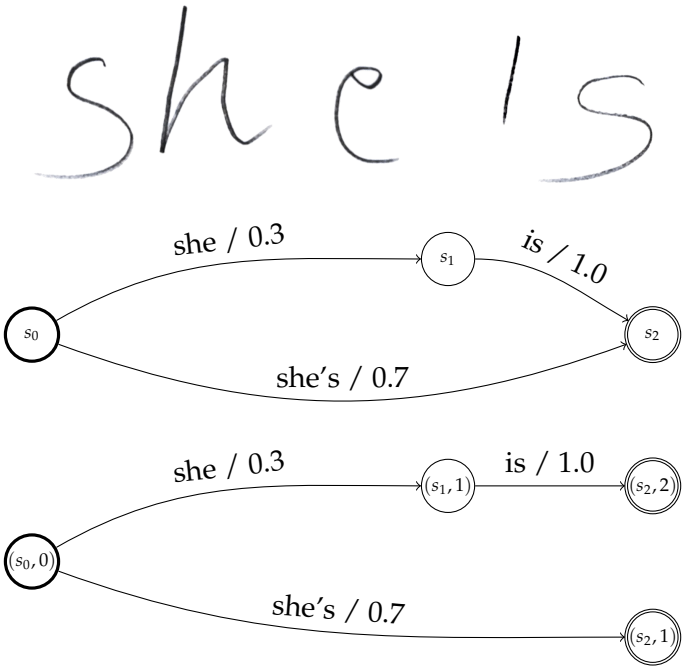


Figure 5.2. Example of the disambiguation of the word position associated to the states of a lattice. In he first lattice the only restriction is that each state is associated to the last column of the alignment of the input words to that state. In the second lattice, all paths entering each state have the same number of words. This way, we can associate a word position to each state (and arc) of the WFSA.

Notice that, because the arcs in the lattice represent full words, we just need to keep track of a counter which is incremented each time that a non-epsilon arc is traversed (see lines 12–15). When we reach a state with a different count than the previously observed, we add that pair of state and count to the queue of pending output states (lines 16–18).

Finally, observe that each state in the input WFSA will be added, at most, K_i times to the queue, where K_i is the *maximum word count input degree* (i.e. the maximum number of different path lengths arriving to any state). Thus the running time of the algorithm is $\mathcal{O}(K_i \cdot (|V| + |E|))$.

Algorithm 5.3 Disambiguate word lattice states to ensure that all paths entering a state have the same word count.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a *compact lattice*.

procedure LATTICEWORDDISAMBIGUATEWORDCOUNT(A)
  $V' \leftarrow \{(s_0, 0)\}$ ▷States of the output WFSA
  $E' \leftarrow \emptyset$ ▷Arcs of the output WFSA
  $Q \leftarrow \{(s_0, 0)\}$ ▷Pending output states
  while $Q \neq \emptyset$ **do**
    $(i, k) \leftarrow$ QUEUEPOP(Q)
    $\rho'[(i, k)] \leftarrow \rho[i]$ ▷Final weight for state (i, k)
    for all $e \in E : p[e] = i$ **do**
      $v \leftarrow l[e]$ ▷Word associated to the arc
      $j \leftarrow n[e]$ ▷Destination state of the arc
      $\omega \leftarrow w[e]$ ▷Weight (likelihood) of the arc
      if $v = \epsilon$ **then**
        $k' \leftarrow k$ ▷Do not increase the word position
      else
        $k' \leftarrow k + 1$ ▷Increase the word position
      if $(j, k') \notin V$ **then**
        QUEUEPUSH($Q, (j, k')$)
        $V' \leftarrow V' \cup \{(j, k')\}$ ▷Add the new state
      $E' \leftarrow E' \cup \{((i, k), (j, k'), v, \omega)\}$ ▷Add new arc
  return $A' = (\Sigma, V', E', (s_0, 0), \rho')$

5.1.3.2 Position index construction

Algorithm 5.4 describes the procedure used to obtain the word positional index from word lattices. The first step in the algorithm is to use algorithm 5.3 in order to disambiguate the input length of the paths arriving to each state. Once this step is complete, the remaining is quite trivial: we simply traverse all arcs of the resulting WFSA and add the word and positions associated to the arc to the unnormalized index. The last step in the algorithms traverses the index to normalize the likelihoods (or other unnormalized scores) in it. The total likelihood arriving to and leaving from each state is computed using the Forward and Backward algorithms, respectively.

Algorithm 5.4 Compute a positional index based on word compact lattices.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a compact lattice.

procedure LATTICEWORDINDEXPOSITION(A)
  $A' \leftarrow$ LATTICEWORDDISAMBIGUATEWORDCOUNT(A)
  Let A' be $A' = (\Sigma, V', E', (s_0, 0), \rho')$.
  $\beta \leftarrow$ BACKWARD(A')
  $U \leftarrow \emptyset$
  for all $e \in E'$ **do**
    $v \leftarrow l[e]$ ▷Word associated to the arc
    if $v \neq \epsilon$ **then**
      $(i, k) \leftarrow p[e]$ ▷Origin state of the arc
      $(j, k') \leftarrow n[e]$ ▷Destination state of the arc
      $\omega \leftarrow w[e]$ ▷Weight (likelihood of the arc
      $t \leftarrow \alpha_{(i,k)} \cdot \omega \cdot \beta_{(j,k')}$ ▷Total likelihood through the arc
      MAPINSERTORSUM($U, (v, k'), t$)
  $I \leftarrow \emptyset$
  for all $((v, k), \omega) \in U$ **do**
    MAPINSERT($I, (v, k), \omega \oslash \beta_{s_0}$) ▷Word position relevance
  return I

5.1.3.3 *Asymptotic cost of algorithm 5.4*

As we discussed earlier, the cost of the first step is $\mathcal{O}(K_i \cdot (|V| + |E|))$, which is the same as the Backward algorithm, and traversing all edges, *with respect to the original size*. Thus, this is the worst case asymptotic cost of algorithm 5.4.

We could simplify this expression with further assumptions. Observe that K_i can be upper bounded with $D_i \leq |E|$, where D_i is the maximum input degree (i.e. the maximum number of arcs arriving to a state). This is the case, for instance, when one state has inputs from all other states, each of which would have a different word position associated to it. Thus, the worst asymptotic cost can be (conservatively) upper bounded by $\mathcal{O}(|E|^2)$. In practice, the algorithm is much faster since $D_i \ll |E|$.

5.2 The out-of-vocabulary problem

In the previous section, we described a set of algorithms that can be employed to compute each one of the relevance probabilities that were introduced in chapter 2. However, as we already mentioned, all these algorithms assume that a *word lattice* representing the set of hypothetical transcripts of a text region was given. In order to obtain such lattices, researchers typically use a closed word lexicon and a word n-gram language model, as described in sections 4.5 and 4.6.

Nonetheless, this assumption carries an important restriction, which may be prohibitive in practical applications: the need to know which words can be written in our collection of documents. The only way to ensure that *all* words (including names, dates, numbers, etc) are present in the lexicon is to "read" the entire collection before indexing it. Of course, "reading" it is part of the problem we are trying to solve, thus we face a chicken and egg situation with this approach. In practice, many systems are restricted to work with a (typically very large) subset of words from the collection, usually extracted from the set of training samples, which have been manually transcribed to train the statistical models described in chapter 4.

Notice that when we restrict our statistical models to a small subset of words, by definition the rest of the words have a prior probability equal to zero, and thus will never be "recognized" or "spotted" by our text recognition or word spotting system.

On the contrary, if we use an excessively large lexicon, we may increase the chances that our system makes avoidable mistakes (since we are introducing noise to our statistical model) and, most importantly, the system can be excessively slow.

This problem has been widely studied in the fields of Speech Recognition and Text Recognition [Asadi et al., 1991, Young, 1994, Woodland et al., 2000, Bazzi, 2002], and it is a fundamental flaw of lexicon-based systems, which affects other domains such as Statistical Machine Translation, Spoken Dialog Systems, Image Captioning, etc. Some authors have proposed using a combination of word and character-based models, in order to avoid the issue [Yazgan and Saraclar, 2004, Szoke et al., 2008, Kozielski et al., 2013]. In the context of keyword

spotting, apart from using lexicon-free approaches, we have tried in the past to *smooth* the word indexes (produced by the methods described in the previous section) to account for potential out-of-vocabulary queries [Puigcerver et al., 2014b, Puigcerver et al., 2014a, Puigcerver et al., 2015a, Puigcerver et al., 2017].

In the following section we propose a different and more straightforward approach, based on the two following requirements:

1. We wish to build word-based indexes, since these allow for very fast searches on large collections of documents (i.e. constant with respect to the number of indexed documents).

2. We need to account for any potential word written in the collection, including proper names, foreign language words, etc.

Hence, the next solutions will employ character lattices obtained from lexicon-free statistical models (i.e. no closed lexicon is assumed), but will manipulate them to extract *pseudo-words* (and build an index for them). These algorithms assume that a word is any sequence of characters in-between some special delimiter characters such as whitespace, punctuation marks, etc.

This could be considered a serious limitation. For example, in early manuscripts it was very common to write several words, or even complete lines, without lifting up the quill from the paper (thereby resulting in text without any kind of optical word-separating clues).

However, it is important to emphasize that this does not mean that we expect word delimiters to actually appear in the images. Fortunately, lexical and/or language models help solving this problem very adequately. If training transcripts include these separators, (regardless whether they are actually rendered or not in the associated training images), the trained models generally provide significant probability to the required word separators on the right context, even if they do not appear at all in the test images.

In most languages, words can be decomposed as a sequence of characters. Generally speaking, the number of characters in a given alphabet, is much lower than the number of words that can be formed

in any of its languages, which one may argue that is not even bounded. Thus, when training statistical models of handwritten text, one may expect to have an instance of all interesting characters, but one may not assume that *all* possible words were observed during the training of the model. In this sense, lexicon-free models solve the problem of out-of-vocabulary words.

5.3 Indexing lexicon-free lattices

Unfortunately, lexicon-free (i.e. character) lattices prevent us from using the algorithms presented in the previous sections to build word indexes for fast user searches. This is because the arcs in the WFSA no longer represent words, but individual characters.

Recall that the goal of the algorithms in this section is to produce a "pseudo-word" index from character lattices. Ideally, pseudo-words should be sequences of characters that form "real" words, and the probabilistic indexes should assign a large probability to pseudo-words written in the image.

5.3.1 From character to word lattices

We first designed an algorithm that converts a character lattice (a lattice whose arcs represent individual characters of a transcript) into a word lattice (whose arcs represent full words). Typically, the delimiter character that we use to separate the words is the whitespace symbol (i.e. a word is anything written between two whitespaces).

More formally, given a function $\Lambda : \Sigma \to \mathcal{C}$ that assigns a *class* to each label (i.e. character) of the alphabet Σ, algorithm 5.5 takes a WFSA and produces an equivalent WFSA such that each arc in the output WFSA is a subpath of the input formed by arcs with the same label class. The function Λ is essentially used to determine whether or not a given character is a delimiter.

The two automata are equivalent, in the sense that all (and only) complete paths in the original WFSA are present in the output WFSA with exactly the same total weight. The equivalence is fundamental in order to preserve the distribution over transcript hypotheses.

Algorithm 5.5 Expansion of subpaths formed by labels of the same class in a WFSA. The algorithm obtains an equivalent WFSA such that the arcs represent subpaths in the input WFSA formed by labels of the same class.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is *compact lattice*, Λ is a function that assigns a class to each label (i.e. character).

procedure LATTICEEXPANDSUBPATHS(A, Λ)
  $c_\epsilon \leftarrow \Lambda[\epsilon]$ ▷Class of the epsilon label
  $\Sigma' \leftarrow \emptyset$ ▷Alphabet of the output WFSA
  $V' \leftarrow \{(s_0, c_\epsilon)\}$ ▷States of the output WFSA
  $E' \leftarrow \emptyset$ ▷Arcs of the output WFSA
  $X \leftarrow \emptyset$ ▷Origin arcs of subpaths, to avoid repetitions
  $S \leftarrow \{(s_0, c_\epsilon, s_0, c_\epsilon, \overline{1}, \epsilon)\}$ ▷Stack of pending subpaths
  while $S \neq \emptyset$ **do**
    $i, c_i, j, c_j, \omega, x \leftarrow$ STACKPOP(S)
    $z \leftarrow 0$ ▷Whether the subpath may end in state (j, c_j)
    for all $e \in E : p[e] = j$ **do** ▷For all output arcs from state j
      $a \leftarrow l[e]$ ▷Label (character) of the arc
      $k \leftarrow n[e]$ ▷Destination state of the arc
      $\nu \leftarrow w[e]$ ▷Weight (likelihood) of the arc
      if $a = \epsilon$ **then**
        $c_k \leftarrow c_j$ ▷Class of the arc is that of the predecessor
      else
        $c_k \leftarrow \Lambda[a]$ ▷Class of the arc is given by Λ
      if $c_j = c_\epsilon \vee c_k = c_j$ **then** ▷Keep expanding subpath
        STACKPUSH($S, (i, c_i, k, c_k, \omega \otimes \nu, xa)$)
      else
        $z \leftarrow 1$ ▷Subpath ends in state (j, c_j)
        if $(j, c_j, e) \notin X$ **then**
          ▷New subpath from (j, c_j) through arc e
          STACKPUSH($S, (j, c_j, k, c_k, \nu, a)$)
          $X \leftarrow X \cup \{(j, c_j, e)\}$
          $V' \leftarrow V' \cup \{(j, c_j)\}$
    if $i \neq j \wedge (\rho(j) \neq \overline{0} \vee z = 1)$ **then** ▷Subpath ends in (j, c_j)
      $V' \leftarrow V' \cup \{(j, c_j)\}$
      $E' \leftarrow E' \cup \{((i, c_i), (j, c_j), x, \omega)\}$
      $\Sigma' \leftarrow \Sigma' \cup \{x\}$
  for all $(s, c) \in V'$ **do**
    $\rho'((s, c)) \leftarrow \rho(s)$ ▷Set final weight of the output states
  return $F' = (V', E', (s_0, c_\epsilon), \rho')$

Each state in the output WFSA will be identified by the state of the input WFSA and the *class* of the subpaths that arrive to that state. Hence, the initial state of the output FST is the pair formed by the initial state of the input automaton and the class of the epsilon label (see lines 1–2), which is the class of the paths that do not contain any symbol (in the case that the automaton is not epsilon-free). An initial empty subpath is added to a stack that will process all the pending subpaths that are formed in the WFSA (see line 5).

While there are subpaths in the stack, the top one is extracted and the arcs leaving from the last state in the path are considered. If the considered arc has a label which is of the same class as the current subpath, it is extended with such arc and it is added to the stack (see lines 16–17). On the contrary, if the arc is from a different class, it means that a new subpath will start from the current state through that arc. The new subpath is added to the stack, and the pair formed by the output state and the traversed arc are added to the set X to avoid *expanding* subpaths through the same arc in the future (lines 20–25).

At the end of each iteration, if the last state of the current subpath is final or had an outgoing arc of a different class, it means that the subpath ends in the current state and, hence, an arc is added in the output WFSA to represent the current subpath (lines 26–28).

Finally, once the stack has been emptied, all the states from the output automaton are traversed and the final weight of each state is set to that of the corresponding state in the input WFSA.

5.3.1.1 Example of algorithm 5.5

In order to better illustrate how algorithm 5.5 operates, fig. 5.3 shows an example of the output produced by this algorithm on a small character lattice.

Notice that the third and last states in the original automaton (depicted in fig. 5.3a) are duplicated in the output (see fig. 5.3b) because they have input arcs with the two classes of characters (regular characters and the delimiter).

As we explained in the description of the algorithm, the two WFSA are equivalent, in the sense that they represent the same sequences of characters. In the original WFSA there are 12 accepted sequences (each of the complete paths in the automaton). In the output, WFSA the same sequences of characters are accepted with the same weights.

For instance, the sequence "b a b a" in the original WFSA is also represented in the output with total weight equal to 0.072, but there is a single arc in the output automaton to represent the whole sequence, since it is formed exclusively by "regular" symbols.

In contrast, the sequence of characters "b @ b a", represented by four arcs in the input (with total weight equal to 0.072), is represented by three arcs in the output WFSA: the arc "b" (with weight 0.6), the arc "@" (with weight 0.5), and the arc "b a" (with weight 0.24).

5.3.1.2 Asymptotic cost of algorithm 5.5

Since algorithm 5.5 expands subpaths in the original WFSA, it is clear that the worst case could have an exponential cost with the size of the graph. However, in practice, it behaves very well because the complete paths in the input automaton are not fully expanded, thanks to the *delimiter* symbols.

A single state in the input WFSA may be "replicated" in the output WFSA, depending on the class of the input arcs arriving to that state. For instance, if C_i distinct classes arrive to the state i, there will be C_i replicas of that in the output WFSA. Thus, the number of states in the output automaton is, at most, $C_i \cdot |V|$.

Now, suppose that the *maximum output degree* of the input automaton is D_o, and the maximum length (in arcs) of a subpath is L. Then, since each state in the output WFSA is the origin of a subpath, there will be, at most, D_o^L arcs leaving each of the output states. Then, the output WFSA will have at most $C_i \cdot |V| \cdot D_o^L$ arcs.

Consequently, the worst asymptotic cost is essentially $\mathcal{O}(C_i \cdot |V| \cdot D_o^L)$. Nevertheless, in practice we typically consider only two classes: delimiter characters (i.e. white spaces, punctuation symbols, ...) and regular characters (i.e. letters, numbers, ...). Therefore, the cost can be (approximately) expressed as $\mathcal{O}(|V| \cdot D_o^L) = \mathcal{O}(\exp(L \log D_o + \log |V|))$.

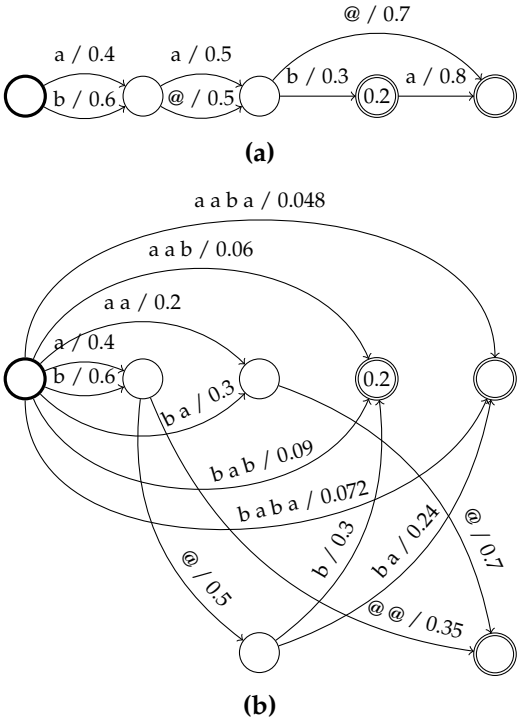


Figure 5.3. Example of algorithm 5.5. Figure 5.3a shows the original character WFSA, where the symbol "@" represents the character that separates words and symbols "a" and "b" represent regular characters. The output of the algorithm is the automaton depicted in fig. 5.3b. Observe that the 12 complete paths present in the original WFSA are exactly the same complete paths that the output WFSA accepts, with the same weights. However, the arcs in the output WFSA are formed by subpaths of the input instead of individual characters, formed by symbols of the same class.

The resulting WFSA can be used with any of the lexicon-based algorithms presented in section 5.1. However, explicitly expanding the subpaths of the lattice can generate some very big lattices in some cases. Thus, it would be preferable if could use some algorithm that avoids this expansion.

5.3.2 LEXICON-FREE SEGMENT RELEVANCE

The algorithm essentially consists of four steps, two of which are well studied algorithms available in any software packaged designed to manipulate WFST: First, it is disentangled so that all arcs arriving to a given state are part of the same class, in a similar way to algorithm 5.3 with some ingredients from algorithm 5.5. Next, the lattice is transformed so that the initial state is connected to every node which is the start of a path corresponding to the non-delimiter class, and each state which is the final of such path is connected to the final state. Then, an equivalent deterministic lattice is obtained, which sums multiple equivalent alignments of the same word, and finally the n-best word segments are obtained from the resulting lattice.

5.3.2.1 Encode character alignment

We need an initial step to encode the alignment of each character as part of the label of the arc. In particular, we choose to represent the alignment information using a transducer, but other options would also be equivalent. Algorithm 5.6 takes a lattice represented by an acyclic WFSA and the function mapping each state to its aligned column, and outputs an equivalent *WFST* which encodes the alignment information at the output labels of its arcs.

5.3.2.2 Disambiguating the input class associated to states

The first step consists of disambiguating the input class associated to each state in the original lattice, so that all arcs arriving to a particular state are of the same label class (given by Λ, as seen in algorithm 5.5). Algorithm 5.7 describes such disambiguation algorithm.

We start traversing the input WFST from the initial state (whose class is the class of the epsilon label) and add a new output state each

Algorithm 5.6 Encode the alignment of each arc in a lattice as part of the output labels of a WFST. The original label is given by the input label of the arc, and the alignment is given by a tuple in the output label.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a *compact lattice*, τ associates a frame to each state in V.

procedure LATTICEENCODEALIGNMENT(A, τ)
  $E' \leftarrow \emptyset$
  $\Gamma \leftarrow \emptyset$
  for all $e \in E$ **do**
    $s_i \leftarrow p[e]$ ▷Source of the arc
    $s_j \leftarrow n[e]$ ▷Destination of the arc
    $a \leftarrow l[e]$ ▷Label (character) of the arc
    $\omega \leftarrow w[e]$ ▷Weight (likelihood) of the arc
    $E' \leftarrow E' \cup \{(s_i, s_j, a, (\tau[i], \tau[j]), \omega)\}$
    $\Gamma \leftarrow \Gamma \cup \{(\tau[i], \tau[j])\}$ ▷Output arc
  return $T = (\Sigma, \Gamma, V, E', s_0, \rho)$ ▷The output is a WFST

time we arrive to an input state with a different class (according to the label of the output alphabet), continuing the expansion from that state. The asymptotic cost is $\mathcal{O}(C_i \cdot (|V| + |E|))$, where $|V|$, $|E|$ and C_i are the number of states, arcs and the maximum number of different input classes to an state, respectively. The C_i factor is due to the fact that each input state is added this number of times to the queue, and we add copies of all of its arcs each time it is extracted from the queue. Figure 5.4 shows the result of algorithm 5.7 applied on a small weighted automaton.

5.3.2.3 From subpaths to complete paths

In this step, we transform the lattice so that the initial state is connected to every node which is the start of a path corresponding to the non-delimiter class, and each state which is the final of such path is connected to the final state. More generally, we transform the lattice so that each subpath of a particular class in the lattice becomes a complete path in the modified lattice. When we do so, the complete paths in the lattice no longer represent transcript hypotheses of the

Algorithm 5.7 Disambiguate the input symbols of the states in a WFST. The algorithm obtains an equivalent WFST such that all labels in the input arcs of each state are of the same class. The algorithm disambiguates with respect to the input alphabet of the WFST.

Require: $T = (\Sigma, \Gamma, V, E, s_0, \rho)$ is an *acyclic* WFST, Λ is a function that assigns a class to each label (i.e. character).

procedure LATTICEDISAMBIGUATEINPUTCLASS(T, Λ)
  $c_\epsilon \leftarrow \Lambda(\epsilon)$ ▷Class of the epsilon label
  $V' \leftarrow \{(s_0, c_\epsilon)\}$ ▷States of the output WFST
  $E' \leftarrow \emptyset$ ▷Arcs of the output WFST
  $Q \leftarrow \{(s_0, c_\epsilon)\}$
  while $Q \neq \emptyset$ **do**
    $(s_i, c_i) \leftarrow$ QUEUEPOP(Q) ▷Process output state (s_i, c_i)
    $\rho'((s_i, c_i)) \leftarrow \rho(s_i)$ ▷Final weight for output state (s_i, c_i)
    for all $e \in E : p[e] = s_i$ **do** ▷Arcs leaving state s_i
      $s_j \leftarrow n[e]$ ▷Destination state of the arc
      $a \leftarrow l_i[e]$ ▷Input symbol (character)
      $b \leftarrow l_o[e]$ ▷Output symbol
      $\omega \leftarrow w[e]$ ▷Weight (likelihood) of the arc
      if $a = \epsilon$ **then**
        $c_j \leftarrow c_i$ ▷Class of the arc is that of s_i
      else
        $c_j \leftarrow \Lambda(a)$ ▷Class of the arc is given by Λ
      if $(s_j, c_j) \notin V'$ **then** ▷Add output state (s_j, c_j)
        $V' \leftarrow V' \cup \{(s_j, c_j)\}$
        QUEUEPUSH($Q, (s_j, c_j)$)
      $E' \leftarrow E' \cup \{((s_i, c_i), (s_j, c_j), a, b, \omega)\}$ ▷Add output arc
  return $T' = (\Sigma, \Gamma, V', E', (s_0, c_\epsilon), \rho')$

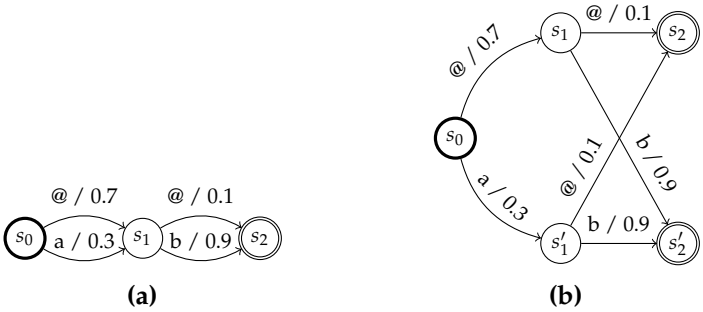


Figure 5.4. Example showing the result of algorithm 5.7 on a small WFST. On the left, the input WFST (actually, a weighted automaton). On the right, the output of the algorithm. Observe that the two WFST are equivalent, but the states in the output WFST ensure that all input arcs have labels of the same class (in the example, "@" denotes the delimiter, and "a" and "b" are non-delimiter characters.

full image, but hypotheses of individual segments (corresponding to a specific class). In addition, the weight (likelihood) of the complete path is equal to the sum of all complete paths which shared that subpath. Algorithm 5.8 describes such algorithm.

The worst asymptotic cost of the algorithm is $\mathcal{O}(|V| + |E|)$, where $|V|$ and $|E|$ are number of states and arcs, respectively, of the input lattice. The forward and backward calculation and the while loop share this cost.

Figure 5.5 shows an example of the output produced by algorithm 5.8 on a small WFSA. Observe that all complete paths in the output automaton correspond to some subpath in the input, and the weight of a complete path in the output is equal to the sum of all paths containing that subpath in the input. Hence, we can obtain the probability of a segmentation hypothesis as the total weight of a path.

5.3.2.4 From character to word alignments

Recall from algorithm 5.6 that we encoded the alignment information of each character (i.e. each label in the input lattice) as the output label of a WFST.

Algorithm 5.8 Convert subpaths of the same class in a character lattice to complete paths.

Require: $T = (\Sigma, \Gamma, V, E, (s_0, c_\epsilon), \rho)$ is an *acyclic* WFST, output of algorithm 5.7.

procedure LATTICECONVERTSUBTOCOMPLETEPATH(T)
  $\alpha \leftarrow$ FORWARD(T)
  $\beta \leftarrow$ BACKWARD(T)
  $E' \leftarrow \emptyset$ ▷Arcs of the output WFSA
  $X \leftarrow \{(s_0, c_\epsilon)\}$
  $Q \leftarrow \{(s_0, c_\epsilon)\}$ ▷Pending states
  while $Q \neq \emptyset$ **do**
    $(i, c_i) \leftarrow$ QUEUEPOP(Q) ▷Initialize final weight
    $\rho'[(i, c_i)] \leftarrow \rho[(i, c_i)]$
    for all $e \in E : p[e] = (i, c_i)$ **do** ▷Traverse arcs from (i, c_i)
      $a \leftarrow l_i[e]$ ▷Input label (character)
      $b \leftarrow l_o[e]$ ▷Output label (alignment)
      $(j, c_j) \leftarrow n[e]$ ▷Destination state
      $\omega \leftarrow w[e]$ ▷Weight (likelihood) of the arc
      if $c_i \neq c_\epsilon \wedge c_i \neq c_j$ **then** ▷States with different classes
        ▷Add arc from initial to (j, c_j)
        $E' \leftarrow E' \cup \{((s_0, c_\epsilon), (j, c_j), a, b, \alpha_{(i,c_i)} \otimes \omega\}$
        ▷Update (i, c_i) final weight
        $\rho'[(i, c_i)] \leftarrow \rho'[(i, c_i)] \oplus (\omega \otimes \beta_{(j,c_j)})$
      else
        $E' \leftarrow E' \cup \{e\}$ ▷States with same class, copy arc
      if $(j, c_j) \notin X$ **then**
        $X \leftarrow X \cup \{(j, c_j)\}$
        QUEUEPUSH($Q, (j, c_j)$)
  return $T' = (\Sigma, \Gamma, V, E', (s_0, \epsilon), \rho')$

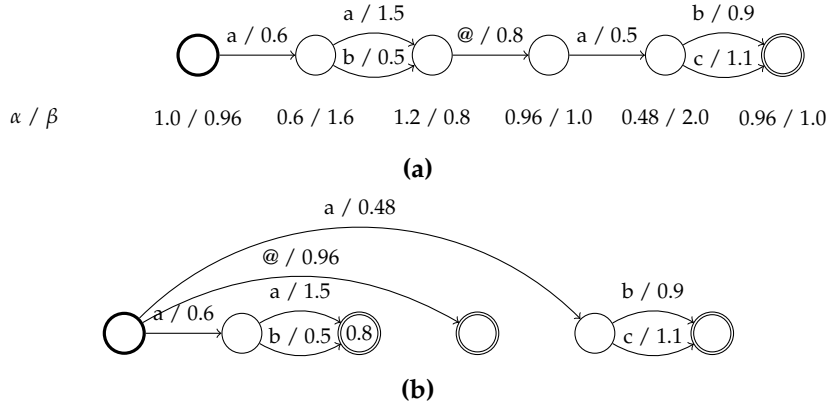


Figure 5.5. Example of algorithm 5.8 on a small WFSA. Figure 5.5a shows the original WFSA, where the arcs labeled with "@" are considered word delimiters. The forward and backward likelihoods of each state are shown below. Figure 5.5b shows the output of the algorithm, observe that all complete paths in the output automaton correspond to some subpath in the input, and the weight of a complete path in the output is equal to the sum of all paths containing that subpath in the input.

Now, in order to make the algorithm comparable to algorithm 5.2, we need to sum the likelihoods of all segments corresponding to the same *word* alignment, regardless of the particular alignment of each individual character in this word.

Before we can compute this sum, we need to extract the full word alignment, instead of the individual character-level alignments. For that, we use the fact that after algorithm 5.8, words start with arcs leaving from the initial state and finish with arcs entering a final state. Algorithm 5.9 uses this fact to remove the alignment information from intermediate characters of a word, and keep only the initial frame of the first character and final frame of the last character of the word. This way, all paths with the same word alignment will have the same sequence of input and output labels in the resulting WFST.

The running time of the algorithm is in $\mathcal{O}(|V| + |E|)$, with respect to the size of its input transducer.

Figure 5.6 shows an example of the result of algorithm 5.9, on a small WFST that would have been produced by algorithm 5.8. Notice that algorithm 5.8 transforms the input lattice so that, all complete

Algorithm 5.9 Keep only word alignments from a character lattice, processed by algorithm 5.8. The algorithm simply uses the fact that complete paths represent full words in the input lattice, so that we can remove the alignment associated to any arc not leaving the initial state or not entering a final state.

Require: $T = (\Sigma, \Gamma, V, E, s_0, \rho)$ is an *acyclic* WFST, output of algorithm 5.8.

procedure KEEPONLYWORDALIGNMENT(T)
  $V' \leftarrow V$
  $E' \leftarrow \emptyset$
  for all $e \in E$ **do**
    $s_i \leftarrow p[e]$ ▷Source state of the arc
    $s_j \leftarrow n[e]$ ▷Destination state of the arc
    $\omega \leftarrow w[e]$ ▷Weight (likelihood) of the arc
    $a \leftarrow l_i[e]$ ▷Character represented by the arc
    $(\tau_1, \tau_2) \leftarrow l_o[e]$ ▷Alignment of the character
    if $s_i = s_0 \wedge \rho[s_j] \neq \overline{0}$ **then** ▷Word with a single character
      $V' \leftarrow V' \cup \{(s_i, s_j)\}$ ▷New auxiliary state
      $E' \leftarrow E' \cup \{(s_i, (s_i, s_j), a, \tau_1, \omega)\}$
      $E' \leftarrow E' \cup \{((s_i, s_j), s_j, \epsilon, \tau_2, \overline{1})\}$
    else if $s_i = s_0$ **then** ▷First character of the word
      $E' \leftarrow E' \cup \{(s_i, s_j, a, \tau_1, \omega)\}$
    else if $\rho[s_j] \neq \overline{0}$ **then** ▷Last character of the word
      $E' \leftarrow E' \cup \{(s_i, s_j, a, \tau_2, \omega)\}$
    else ▷Intermediate character
      $E' \leftarrow E' \cup \{(s_i, s_j, a, \epsilon, \omega)\}$
  return $T' = (\Sigma, \mathbb{N}, V', E', s_0, \rho)$

paths representing the same word with the same word-level alignment (i.e. the word starts and ends at the same columns) will be represented by the same sequence of input/output symbols in the WFST. In particular, the sequence of input symbols represent the sequence of characters and the sequence of output symbols denotes the first and last columns of the image where the word is written.

5.3.2.5 *Indexing words with alignment from character lattices*

Finally, we are in the position to put all pieces together in order to describe the algorithm that extracts an index for word segments directly from character lattices. Algorithm 5.10 describes such algorithm.

Algorithm 5.10 Compute a word index for text segments based on character lattices.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a compact lattice, the function τ gives the frame associated to each state, and Λ assigns a class to each label (i.e. character). The maximum number of words to index is given by n.

1: **procedure** LATTICECHARACTERINDEXSEGMENT(T, τ, Λ, n)
2: $\beta \leftarrow$ BACKWARD(A)
3: $T_1 \leftarrow$ LATTICEENCODEALIGNMENT(A, τ)
4: $T_2 \leftarrow$ LATTICEDISAMBIGUATEINPUTCLASS(T_1, Λ)
5: $T_3 \leftarrow$ LATTICECONVERTSUBTOCOMPLETEPATH(T_2)
6: $T_4 \leftarrow$ KEEPONLYWORDALIGNMENT(T_3)
7: $T_5 \leftarrow$ DETERMINIZEASWFSA(T_4)
8: $I \leftarrow \emptyset$
9: **for all** $(x, y, \omega) \in$ NBESTPATH(T_5, n) **do**
10: Let $x = x_1, x_2, \ldots, x_m$ be the input symbols in the path.
11: Let $y = y_1, y_2, \ldots, y_m$ be the output symbols in the path.
12: Let ω be the total weight of the path.
13: MAPINSERT($I, (x, y_1, y_m), \omega \oslash \beta_{s_0}$)
14: **return** I

The backward algorithm, at line 2, has a cost of $\mathcal{O}(|V| + |E|)$, which is the same as algorithm 5.6 (line 3). Recall that algorithm 5.7 (line 4) has an asymptotic cost of $\mathcal{O}(C_i \cdot (|V| + |E|))$. However because the number of classes is usually two (delimiter and non-delimiter char-

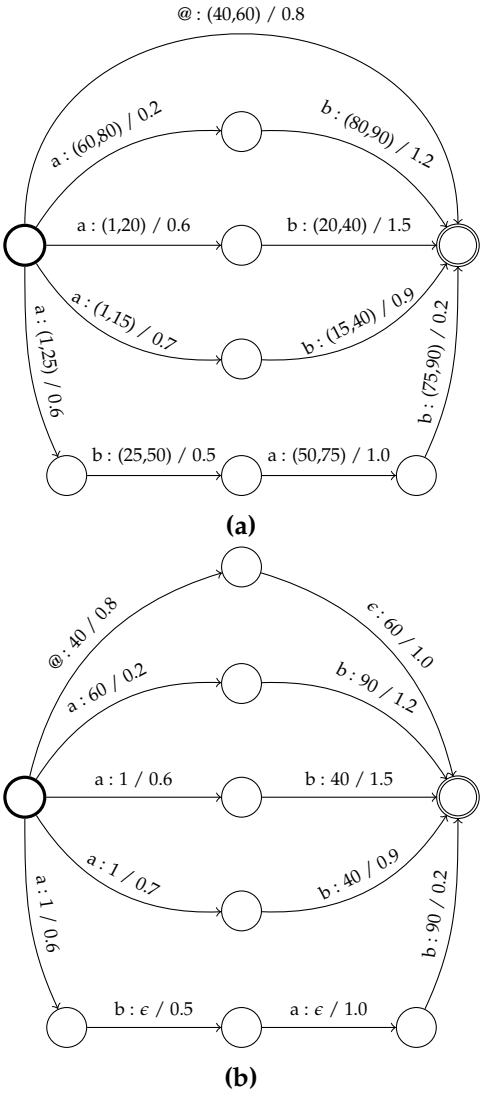


Figure 5.6. Example showing the result of algorithm 5.9. Figure 5.6a shows the input WFST. Symbols of the input alphabet represent characters and symbols of the output alphabet represent the alignment of the character with the corresponding image (i.e. first and last column where the character is written). Figure 5.6b shows the output WFST. Observe that the input symbols in a path represent the characters of the word, and the first and last output symbols in a path denote the first and last columns of the word. Two sequences with alternative character-level alignment, but the same word-level alignment will have the same sequence of input and output symbols.

acters), the cost can be expressed as $\mathcal{O}(|V| + |E|)$. This is, again, the cost of algorithm 5.8 (line 5), and algorithm 5.9 (line 6).

The "determinization as WFSA" (line 7) step simply means that we treat the pair of input–output symbols in each arc as an individual symbol (i.e. the alphabet of the equivalent WFSA is $\Sigma \times \Gamma$). Since lattices are acyclic, the determinization is always possible. It is important to highlight that the determinization has to be done in the log or real semiring, in order to obtain the sum of the likelihoods (or log-sum-exp of the log-likelihoods).

Notice that any finite set of strings with associated weights can be represented, in the worst case, with a weighted prefix tree. If L is the length of the longest string, $|\Sigma \times \Gamma|$ is the number of symbols in the alphabet of the WFSA, and D_o is the maximum output degree, then the number of states (and arcs) in the prefix tree will be, approximately, $\mathcal{O}(\exp(L \log D_o))$, which is very similar to that of algorithm 5.5.

Finally, extracting the n-best paths of a WFST has a time complexity of $\mathcal{O}(|V| \log |V| + n|V| + n|E|)$. Since the number of states and arcs as the result of the determinization is upper bounded in $\mathcal{O}(\exp(L \log D_o))$, the worst case asymptotic cost of algorithm 5.10 can be expressed as $\mathcal{O}((L \log D_o + n) \exp(L \log D_o))$. In practice, the cost is much smaller than this, and the exponential worst case is rarely seen. As the results of the experiments in section 8.2 show, we are able to generate a word–segment index for a whole data set (229 pages) in less than 6 seconds.

5.3.3 Lexicon-free transcript position relevance

In order to build a so-called positional index, similar to the ones used by traditional search engines, from character lattices, we will make use of algorithms very similar to the described before.

First, we need a procedure to determine where the words are within the transcript, given a sequence of characters. For that, we employ the notion of delimiter characters (e.g. the whitespace). In addition, we needed to sum the likelihoods of all paths corresponding to the same word alignment, in order to build the index of word positions.

5.3.3.1 *Disambiguating the word position associated to states*

First, we need to disambiguate the word position associated to a state. However, the problem is more difficult now, since the arcs represent individual characters instead of words, and we cannot simply count the number of arcs in the paths arriving to a given state.

In contrast, as we explained earlier, words are defined by a sequence of characters between two delimiter characters. Thus, we need to keep track of the number of times that we switch from a delimiter to a non-delimiter symbol in the paths that enter a given state, and we need to decouple the states of the input WFSA so that we produce an equivalent output WFSA with a constant number of transitions from non-delimiter to delimiter characters in all entering paths.

Algorithm 5.11 performs this operation, assuming that the input WFSA is the output of algorithm 5.7. In addition, this algorithm takes a function $\Delta : \mathcal{C} \rightarrow \{0, 1\}$, which maps the class of a label/state to a binary value that denotes whether or not the transition to the given class should increase the word count. For instance, take a path with symbols "m y @ c a t @ i s @ b l a c k", where the symbol "@" denotes the delimiter character. When we transition from any character to "@", we must increase the word count, but when we transition from "@" to any other character we must not. Thus, given the previous path the word count at each position of the sequence would be "1 1 1 2 2 2 2 3 3 3 4 4 4 4 4".

The asymptotic cost of the algorithm is $\mathcal{O}(K_i \cdot (|V| + |E|))$, where K_i is the maximum position input degree, and $|V|$ and $|E|$ are the number of states and arcs of the input WFST. Notice that, because the input WFST to this algorithm has to be pre-processed with algorithm 5.7, the total asymptotic cost of the two is $\mathcal{O}(C_i \cdot K_i \cdot (|V| + |E|))$ where C_i is the maximum class input degree. The maximum position and class input degrees are the maximum number of distinct classes and word positions entering a state.

Algorithm 5.11 Disambiguate character lattice states to ensure that all paths entering any state have the same word count. The algorithm is designed to operate after algorithm 5.7.

Require: $T = (\Sigma, \Gamma, V, E, (s_0, c_\epsilon), \rho)$ is a an *acyclic* WFST, output of algorithm 5.7, such that that all input arcs to any state are of the same class. Δ is a binary function that determines whether or not the transition into a given label class increases the word count.

procedure LATTICECHARDISAMBIGUATEWORDCOUNT(T, Δ)
  $V' \leftarrow \{(s_0, c_\epsilon, 0)\}$ ▷Output states
  $E' \leftarrow \emptyset$ ▷Output arcs
  $Q \leftarrow \{(s_0, c_\epsilon, 0)\}$
  while $Q \neq \emptyset$ **do**
    $(s_i, c_i, k_i) \leftarrow$ QUEUEPOP(Q)
    for all $e \in E : p[e] = (s_i, c_i)$ **do** ▷Arcs leaving state (s_i, c_i)
      $(s_j, c_j) \leftarrow n[e]$ ▷Destination state
      $a \leftarrow l_i[e]$ ▷Input label
      $b \leftarrow l_o[e]$ ▷Output label
      $\omega \leftarrow w[e]$ ▷Weight of the arc
      if $c_i \neq c_j \wedge \Delta[c_j] = 1$ **then**
        $k_j \leftarrow k_i + 1$ ▷Increase word count
      else
        $k_j \leftarrow k_i$ ▷Keep word count
      $E' \leftarrow E' \cup \{((s_i, c_i, k_i), (s_j, c_j, k_j), a, b, \omega)\}$ ▷Add arc
      if $(s_j, c_j, k_j) \notin V'$ **then**
        $V' \leftarrow V' \cup \{(s_j, c_j, k_j)\}$ ▷Add state
        QUEUEPUSH($Q, (s_j, c_j, k_j)$)
  for all $(s_i, c_i, k_i) \in V'$ **do**
    $\rho'[(s_i, c_i, k_i)] \leftarrow \rho[(s_i, c_i)]$ ▷Set final weight
  return $T' = (\Sigma, \Gamma, V', E', (s_0, c_\epsilon, 0), \rho')$

5.3.3.2 *Encode word counts*

Similarly to what we did in algorithm 5.6, where we encoded the alignment of the characters as labels of the WFST, here we will encode the word count of each character (i.e. the position of the word that the given character belongs to). The cost of this algorithm is linear with the size of the input transducer.

Algorithm 5.12 Encode the word count of each arc in a lattice as the output labels of a WFST. The input label of the WFST is preserved. This algorithm operates on the output of algorithm 5.11.

Require: $T = (\Sigma, \Gamma, V, E, (s_0, c_\epsilon, 0), \rho)$ is an *acyclic* WFST, output of algorithm 5.11, such that all paths entering a state have the same word count.

1: **procedure** LATTICEENCODEWORDCOUNT(T)
2: $E' \leftarrow \emptyset$
3: **for all** $e \in E$ **do**
4: $(s_i, c_i, k_i) \leftarrow p[e]$ ▷Source state
5: $(s_j, c_j, k_j) \leftarrow n[e]$ ▷Destination state
6: $a \leftarrow l_i[e]$ ▷Label (character)
7: $\omega \leftarrow w[e]$ ▷Weight (likelihood)
8: $E' \leftarrow E' \cup \{((s_i, c_i, k_i), (s_j, c_j, k_j), a, k_j, \omega)\}$ ▷Output arc
9: **return** $T' = (\Sigma, \mathbb{N}, V, E', (s_0, c_\epsilon, 0), \rho)$

5.3.3.3 *Indexing words with positions from character lattices*

Finally, we can put all the pieces together to build the positional index of words. Algorithm 5.13 describes such procedure, which shares many steps in common with algorithm 5.10.

First, the lattice states are disambiguated so that all labels entering a given state are of the same class. Then, the states are also disentangled so that all paths entering any state have the same number of words (i.e. groups of sequences of characters between delimiters). Afterwards, the word count (i.e. word position) is encoded as part of the labels of a WFST. Later, the subpaths corresponding to (pseudo)words are isolated, so that a complete path in the resulting WFST represents a word, in a particular transcript position. Then, the

Algorithm 5.13 Compute a word index for transcript positions based on character lattices.

Require: $A = (\Sigma, V, E, s_0, \rho)$ is a *compact lattice*. Λ assigns a class to each label (i.e. character) in the alphabet. Δ is a function telling for each label class whether or not it should increase the word position. The maximum number of words to index is given by n.

procedure LATTICECHARACTERINDEXPOSITION(A, Λ, Δ, n)
  $\beta \leftarrow$ BACKWARD(A)
  $T_1 \leftarrow$ LATTICEDISAMBIGUATEINPUTCLASS(A, Λ)
  $T_2 \leftarrow$ LATTICEDISAMBIGUATEWORDCOUNT(T_1, Δ)
  $T_3 \leftarrow$ LATTICEENCODEWORDCOUNT(T_2)
  $T_4 \leftarrow$ LATTICECONVERTSUBTOCOMPLETEPATH(T_3)
  $T_5 \leftarrow$ DETERMINIZEASWFSA(T_4)
  $I \leftarrow \emptyset$
  for all $(x, y, \omega) \in$ SHORTESTPATHS(T_5, n) **do**
    Let $x = x_1, x_2, \ldots, x_m$ be the input symbols in the path.
    Let $y = y_1, y_2, \ldots, y_m$ be the output symbols in the path.
    Let ω be the total weight of the path.
    MAPINSERT($I, (x, y_1), \omega \oslash \beta_{s_0}$)
  return I

resulting WFST is determinized, so that the likelihoods of the equivalent pairs of word–position are summed. And finally, the n-shortest paths (i.e. pairs of word–position) are indexed.

The total cost is essentially the same as the one of algorithm 5.10, that is $\mathcal{O}((L \log D_o + n) \exp(L \log D_o))$. As in the case of algorithm 5.10, the cost of this algorithm does not have an exponential behavior in the typical cases. Our implementation of the algorithm, also extracts the best alignment for each pair word–position, thus the cost is usually larger than that of algorithm 5.10. As the results of the experiments in section 8.2 show, we are able to generate a word–position index for a whole data set (229 pages) in less than 12 seconds.

6 Probabilistic Interpretation of Traditional Approaches

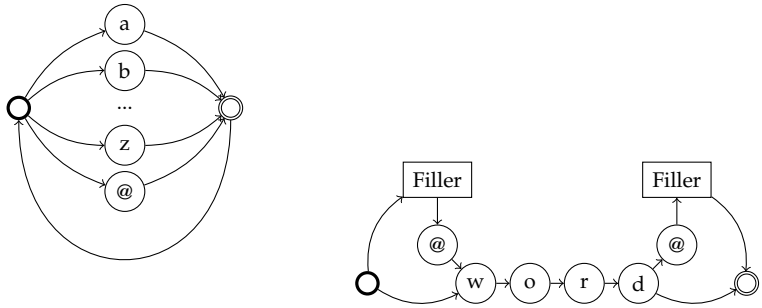
6.1 HMM-filler method

One of the traditional methods with great success for query-by-string KWS was the so-called *HMM-filler* approach [Fischer et al., 2012]. This approach models the handwritten text using HMMs with diagonal GMM as the emitting probability density function in the states (see section 4.2). Earlier variants of this method were applied on speech utterances [Rohlicek et al., 1989, Rose and Paul, 1990, Rose, 1995].

Each character is represented by an individual HMM and composite models are used to build: a) a generic (or garbage) model, known as "filler HMM" or "garbage HMM", and b) a specific "keyword HMM" for the query keyword that needs to be spotted. Figure 6.1 depicts these two composite models. Each circular node represents the HMM model (with several states and transitions) of an individual character in the alphabet (including the whitespace symbol). The rectangular nodes represent copies of the garbage model. Observe that the generative model represented by the filler can produce any character sequence, while the keyword-specific model can only accept sequences that contain at least one instance of "word".

[Fischer et al., 2012] suggested to use the following score to express how likely is that the query keyword, v, is written in a segmented text line image x.

$$S_{\text{filler}}(v, x) \stackrel{\text{def}}{=} \frac{\max_{w \in \Sigma^*} \log p(w, \boldsymbol{x}\,;\, \mathcal{M}_{k_v}) - \max_{w' \in \Sigma^*} \log p(w', \boldsymbol{x}\,;\, \mathcal{M}_f)}{|v|^\gamma} \tag{6.1}$$



(a) Filler HMM : $\mathcal{M}_f$ **(b)** Keyword HMM : $\mathcal{M}_{k_v}$

Figure 6.1. Representation of the models used in the HMM-filler approach. The two automata represent: (a) the "filler HMM", $\mathcal{M}_f$, and (b) "keyword HMM", $\mathcal{M}_{k_v}$, built for the keyword $v =$ "word". Circular nodes represent full HMM models (including several states and transitions) of each character, and the rectangular nodes represent copies of the garbage model. Observe that the filler can accept any character sequence, while the keyword-specific model can only accept sequences that contain at least one instance of "word".

First, observe that, on the one hand, $\log p(w', \boldsymbol{x}\,;\,\mathcal{M}_f)$ depends only on the image $\boldsymbol{x}$, and does not depend on the specific query keyword. On the other hand, $\log p(w, \boldsymbol{x}\,;\,\mathcal{M}_{k_v})$ will (hopefully) have high values when the text line contains the keyword v written at some location, while it will have very low values when it does not. Thus, by comparing how large is the likelihood provided by the keyword-specific model with respect to the keyword-agnostic (i.e. garbage) model, we can get a sense of how likely is that the word is written somewhere in the image.

The denominator in the score served to normalize the scores with respect to the length of the query keyword. In general, models based on HMMs tend to give larger likelihoods to shorter sequences of observations. So, without the denominator in eq. (6.1), long keywords used to have much lower scores than short keywords, which damaged the Global AP (see section 3.3). Sometimes $|v|$ refers to the length (number of characters) of the keyword, while others use the number o frames aligned with the keyword. In most cases, the value of the hyperparameter γ is set to 1, like in the original HMM-filler

publication, but other authors decided to tune it on a validation set to improve the performance of the system [Toselli et al., 2016a].

In [Toselli and Vidal, 2013], they showed that the HMM-filler score can be efficiently approximated by means of a character lattice. They first obtain a lattice from the text line using the HMM-filler model, and then compare the score of the best path (highest likelihood) in this lattice containing the query keyword against the score of the best path in this lattice (which would be equivalent to the right-hand part of the subtraction in eq. (6.1). Both approaches produce virtually the same results, but the lattice approach was much faster (2 orders of magnitude) serving queries.

Later, in [Toselli et al., 2015, Toselli et al., 2016a], instead of using the simple HMM-filler model, regular n-gram language models were used to obtain the character lattices. It was shown that, as one increases the order of the n-gram language model, the results improve. Actually, one can also observe that if one adjusts the parameter γ in eq. (6.1) using validation data, its value gets closer to 0 as one increases the order of the language model (see the results in section 8.9). This suggests, that the denominator was only needed in the first place because extremely naive language models were used (actually, the original HMM-filler assumes that all characters in a sequence are independent).

In addition, in [Puigcerver et al., 2015c] it was later shown that the HMM-filler score can actually be interpreted as an approximation to the probability defined in eq. (2.7), which directly relates this classical method to the probabilistic framework described in this work.

As stated before, the denominator in the original formulation is unnecessary when a large context is used to model the prior information of the language. Thus, ignoring this term and applying the exponent function to both sides of eq. (6.1) results in:

$$\exp S_{\text{filler}} = \frac{\max_{w \in \Sigma^*} p(w, \boldsymbol{x}\,;\, \mathcal{M}_{k_v})}{\max_{w' \in \Sigma^*} p(w', \boldsymbol{x}\,;\, \mathcal{M}_f)} \tag{6.2}$$

Observe that all the character strings with a likelihood greater than 0 in the keyword HMM model of v (i.e. $\mathcal{M}_{k_v}$), contain at least

one instance of this keyword. Thus, in the numerator, the maximum over all character strings in Σ^* is equivalent to the maximum over all character strings in $L(v)$, which is the set of all character strings that contain the keyword v, as explained in section 2.1.

$$\exp S_{\text{filler}} = \frac{\max_{w \in L(v)} p(w, \boldsymbol{x}\,;\, \mathcal{M}_{k_v})}{\max_{w' \in \Sigma^*} p(w', \boldsymbol{x}\,;\, \mathcal{M}_f)} \tag{6.3}$$

Now, let's rewrite eq. (2.7) in terms of the joint likelihood of the image and the transcript, instead of the posterior. We will also drop the name of the random variables, since it is clear to which random variable each value corresponds to:

$$\begin{aligned} P(R = 1 \mid X = \boldsymbol{x}, V = v) = \\ \sum_{w \in L(v)} P(W = w \mid X = \boldsymbol{x}) = \\ \frac{\sum_{w \in L(v)} p(w, \boldsymbol{x})}{p(\boldsymbol{x})} = \\ \frac{\sum_{w \in L(v)} p(w, \boldsymbol{x})}{\sum_{w' \in \Sigma^*} p(w', \boldsymbol{x})} \end{aligned} \tag{6.4}$$

Notice that there are two main differences between eq. (6.3) and eq. (6.4):

1. The HMM-filler score replaces the summation in with a maximum operation. This approximation is sometimes called the "Viterbi approximation to the sum".

2. The HMM-filler approach uses two different models for the joint likelihood: $\mathcal{M}_{k_v}$ and $\mathcal{M}_f$. However, our formulation is not based on any particular model. In addition, [Toselli and Vidal, 2013] showed that one can simply use the same model in both cases, as we mentioned earlier.

Hence, the original HMM-filler score and subsequent improvements are approximations to the relevance probability, presented in this thesis. Moreover, it is no surprise that when the probabilistic

models improve, the traditional Keyword Spotting performance measures (e.g. Global and Mean AP) also improve, since ranking according to the relevance probability results in the *optimal* ranking, as proven in chapter 3.

6.2 BLSTM-CTC method

One of the methods that showed an excellent performance for line-level Keyword Spotting, was the presented in [Frinken et al., 2012]. This method, also lexicon-free, was one of the first ones to use Bidirectional Long Short-Term Memories, a form of Recurrent Neural Networks (see section 4.3) to tackle Keyword Spotting.

Their method is in fact a modification of the classical CTC algorithm [Graves et al., 2006], typically used to train models for speech and handwritten text using RNNs.

The traditional CTC algorithm is used to compute the posterior probability of a sequence of characters, given the output of the neural network (whose parameters will be tuned to improve the posterior probability of the reference text, during training). The total posterior probability of a sequence of characters is computed by summing all alignments of that particular sequence of characters, using dynamic programming.

Instead of summing all alignments, the algorithm described in [Frinken et al., 2012] keeps track of the alignment with the maximum posterior. In addition, because we are not interested in the best alignment of the full text line image, but only to know whether or not that image contains the given keyword, they introduce an special symbol (i.e. $\star$), that can be optionally matched against any character at the start and at the end of the desired keyword. Thus, if $v = v_1, v_2, \ldots, v_n$ is the keyword (represented by its sequence of characters), the following sequence is aligned using their algorithm:

$$v' = \star, \text{␣}, v_1, v_2, \ldots, v_n, \text{␣}, \star$$

where "␣" represents the whitespace symbol, which is added here to ensure that only isolated words are spotted, and not subwords. A fixed "probability" equal to 1 for any position is assigned to this spe-

cial symbol, so that anything before or after the keyword does not affect its score. Then, using the CTC algorithm substituting the sum with max operations, and adding the artificial $\star$ at all positions with "probability" equal to 1, actually computes the probability of the subpath with the highest probability containing the given query (i.e. the sum of all paths going through that maximal subpath).

This equivalence is due to the fact that the CTC algorithm assumes that the labels at each position are independent. As we explained in section 4.3.4, the output of the neural network can be represented using an automaton similar to the one depicted in fig. 6.2 (without the $\star$ transitions). Thus, the sum of all paths entering an state is always 1.0, which is the equivalent to transitioning through the $\star$ transitions. Recall that the transitions in the CTC automaton do not represent characters, but *labels*, and a sequence of labels is transformed into a sequence of characters according to the CTC decoding rules.

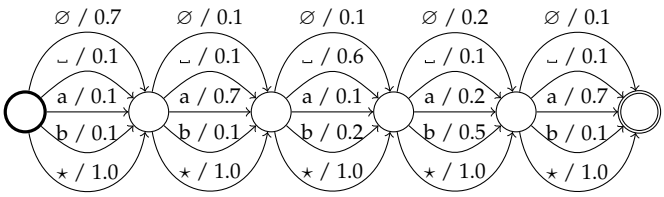


Figure 6.2. Example of the automaton implicitly used by the BLSTM-CTC method for Keyword Spotting. The method uses the same neural network output as used by the regular CTC algorithm, but adds an special label $\star$ with probability 1.0 in any position. This allows to align any frame of the image with this label without decreasing the probability, and then the probability of the best alignment for the sequence of characters $\star, \text{␣}, v_1, \ldots, v_n, \text{␣}, \star$ is found, where $v_1, \ldots, v_n$ is the character sequence corresponding to the keyword v.

We can express the probability computed by the algorithm proposed in [Frinken et al., 2012] as:

$$\Psi(v) \stackrel{\text{def}}{=} \max_{\substack{l_{i:j}: \\ \mathcal{L}(l_{i:j})=v}} \sum_{l_{1:i-1}} \sum_{l_{j+1:M}} P(l_1, \ldots, l_{i-1}, l_i, \ldots, l_j, l_{j+1}, \ldots, l_M \mid \boldsymbol{x}) \quad (6.5)$$

As we just mentioned, the previous equation computes the maximum probability of a subpath that is an alignment of the query key-

word. Now, observe that if we substitute the max operation with a sum, the previous equation can be rewritten as:

$$\sum_{\substack{l_{i:j}: \\ \mathcal{L}(l_{i:j})=v}} \sum_{l_{1:i-1}} \sum_{l_{j+1:M}} P(l_1, \ldots, l_{i-1}, l_i, \ldots, l_j, l_{j+1}, \ldots, l_M \mid \boldsymbol{x}) =$$

$$\sum_{\substack{l_{1:M}: \\ \mathcal{L}(l_{1:M}) \in L(v)}} P(l_1, \ldots, l_M \mid \boldsymbol{x}) =$$

$$\sum_{w \in L(v)} \sum_{l_{1:M}} P(W = w, L = l_{1:M} \mid X = \boldsymbol{x}) =$$

$$\sum_{w \in L(v)} P(W = w \mid X = \boldsymbol{x}) =$$

$$P(R = 1 \mid X = \boldsymbol{x}, V = v) \tag{6.6}$$

Thus, the only difference between the score devised [Frinken et al., 2012] and the relevance probability defined in eq. (2.7) is that the former does not sum the probability of all alignments containing the query keyword, but only considers the maximum subpath containing it.

Finally, due to (essentially) the same reasons as in the HMM-filler, [Frinken et al., 2012] introduce the following heuristic to mitigate the fact that long keywords tend to produce much lower scores than short ones:

$$S_{\text{CTC}} \stackrel{\text{def}}{=} \frac{\log \Psi(v)}{|v|} \tag{6.7}$$

Realize that the CTC model assumes that the label at position t is independent of the other labels (given the feature vector at position t). Now, suppose that a given keyword has 10 characters. Thus, it needs to be aligned at least to 10 frames. If the probability of the correct label at each frame is 0.9, then the probability of the subpath corresponding to the keyword drops to $0.9^{10} \approx 0.35$.

Again, this is only due to the independence assumptions of the CTC model, and the use of n-gram language models in addition to RNNs (or HMMs) removes the need for such heuristic.

6.3 Distance-based methods

Many traditional approaches, predominantly in the Query-by-Example paradigm, interpret Keyword Spotting as finding *similar* images to the given query. In most cases, a fixed-size feature vector is extracted from word images (which have been previously segmented manually, by automatic means or using a sliding window approach), and then compare (using a distance or other dissimilarity measure) this feature vector with the vector obtained from the query image (e.g. [Almazán et al., 2012, Retsinas et al., 2016, Sfikas et al., 2016]). Sometimes, the word images are represented as a sequence of feature vectors, and then distances over sequences are used to "compare" the two sequences (e.g. [Rath et al., 2004, Rodriguez-Serrano et al., 2009, Mondal et al., 2016, Mondal et al., 2018]).

This point-of-view of KWS is predominant among the researchers with a Computer Vision background, perhaps because this approach is also predominant in other related applications such as Content-based Image Retrieval [Toshikazu et al., 1991, Bird et al., 1996, Smeulders et al., 2000]. Many of these researchers consider that their methods are "recognition-free", and some of them have the advantage that are based on purely unsupervised methods and well-designed heuristics. This is clearly evidenced in the following excerpt from a recently published comprehensive survey on KWS [Giotis et al., 2017]:

...] recognition-free retrieval, which is also known in the literature as word spotting or keyword spotting, is the main subject of this study. The goal here is to retrieve all instances of user queries in a set of document images which may be segmented at text lines or words. Actually, the user formulates a query and the system evaluates its similarity with the stored documents and returns as output a ranked list of results which are most similar to the query.

This formulation of KWS presents some problems: First, one needs to establish how to measure this *similarity* (i.e. which vector space and which measure should be used). Secondly, and most importantly, the fact that two images are "similar" or "dissimilar" does not necessary imply that one is *relevant* or irrelevant given the other as query. In addition, it is true that these methods do not perform explicit "word

recognition", but they are still indirectly solving a classification problem, as we show below. Yet, they are not solving the *right* classification problem.

6.3.1 DISTANCE-BASED DENSITY ESTIMATION

The link between distance measures (or, more generally, dissimilarity measures) and density functions is well know and studied from the origins of Patter Recognition. Any properly defined dissimilarity (not strictly necessary a metric), $d(\cdot,\cdot)$, can be used to estimate a density function $p(Y = \boldsymbol{y})$ over a vector space [Silverman, 1986, Duda et al., 2000, Biau and Devroye, 2015].

Here we adopt perhaps the most basic and simple approach, namely kernel-based Parzen windows:

$$p(Y = \boldsymbol{y}) = \frac{1}{nb^D} \sum_{\boldsymbol{x} \in X} \mathcal{K}\left(\frac{d(\boldsymbol{x}, \boldsymbol{y})}{b}\right) \tag{6.8}$$

where X is a set of prototype samples, $n = |X|$, $\mathcal{K}$ is a kernel function (a non-negative real-valued integrable function which integrates to 1), D is the number of dimensions of our data, and b is a smoothing parameter called the window bandwidth.

Many traditional parametric density families have non-parametric kernel-based counterparts. For instance, if we use the (squared of) the euclidean distance and a Gaussian kernel, we obtain a mixture of multivariate normal distributions with the identity as the covariance matrix.

Likewise, we can define conditional densities on some class.

$$p(Y = \boldsymbol{y} \mid C = c) = \frac{1}{n_c b^D} \sum_{\boldsymbol{x}_c \in X_c} \mathcal{K}\left(\frac{d(\boldsymbol{x}_c, \boldsymbol{y})}{b}\right) \tag{6.9}$$

where X_c is the subset of prototypes of class c and $n_c = |X_c|$.

Now, following the classical formulation of Keyword Spotting, suppose that we have some query keyword, represented by its feature vector $\boldsymbol{y}$, and we have a collection of candidate images $\{\boldsymbol{x}_i\}$, for $1 \leq i \leq n$ that we want to rank by "similarity".

We can take the density estimation using Parzen windows to the limit and suppose that there are as many classes as candidate images (i.e. $C \equiv [1, \ldots, N]$), and we have a single prototype for each class (i.e. the candidate image). This results in the conditional density:

$$p(Y = \boldsymbol{y} \mid C = i) \propto \frac{1}{b^D} \mathcal{K}\left(\frac{d(\boldsymbol{x}_i, \boldsymbol{y})}{b}\right) \tag{6.10}$$

The previous equation uses the proportionality symbol ($\propto$) instead of the equality symbol because the density is ill-defined in this extreme case. Nevertheless, we can still rank our candidate images using this pseudo-density (and we can define proper class posteriors, as we shall see soon).

We can devise a generative model of the queries that would include this density function: we assume that the query image has been *generated* from the distribution defined by one of the candidates (i.e. it is a copy of one of the candidates distorted by some stochastic process). Figure 6.3 shows a representation of this generative model.

Observe that, if we use a function $\mathcal{K}$ which monotonically decreases as the distance (or divergence) increases, either sorting the candidate images in descending order of the pseudo-likelihood, or sorting them according to increasing distances will produce exactly the same ranking, with respect to a given query.

This approach, (implicitly) widely adopted among the Keyword Spotting community, presents two fundamental problems, that we coined the "multi-variance" and "multi-mode" problems.

6.3.1.1 The multi-variance problem

The first problem is caused due to simply using the raw distances (or some monotonic equivalent function, like the pseudo-likelihood described above) in order to rank the candidate images. Observe that some candidate images (for instance, corresponding to some particular word) may have much larger variances than others. This becomes very problematic when, instead of considering a single query, one considers a set of such queries, as global measures of the retrieval performance do (such as the Global Average Precision). Notice that some

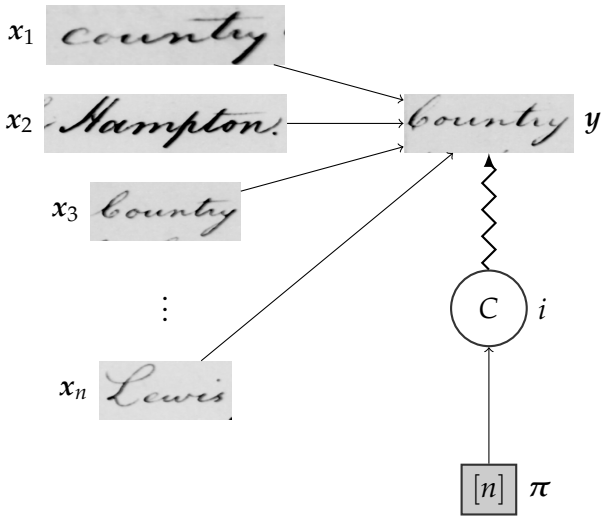


Figure 6.3. Probabilistic graphical model assumed by traditional distance-based approaches. The value of the random variable Y, representing a particular query $\boldsymbol{y}$, is generated from a mixture of n distributions, where each of them parameterized by one the n candidate images $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_n$. The value of the random variable C is used to select the i-th component of the mixture (i.e. the candidate) from which $\boldsymbol{y}$ is generated, based on the prior probability π_i.

"dissimilar" pairs will be ranked on top of other "similar" pairs just because the class of the similar pairs happens to have a large variance. Thus, we refer to this issue as the "multi-variance problem".

Figure 6.4 shows an illustrative instance of the issue. The example includes samples from two classes (which are unknown by the KWS system). Candidate images are represented by dots labeled as $\boldsymbol{x}$, and queries as $\boldsymbol{y}$. Notice that the class "red-square" has a much larger variance than the class "blue-circle", hence the Euclidean distances tend to be larger for the query $\boldsymbol{y}_2$ than for $\boldsymbol{y}_1$. Therefore, for the raw negative distances, the pair $(\boldsymbol{x}_2, \boldsymbol{y}_1)$ is better ranked than $(\boldsymbol{x}_2, \boldsymbol{y}_2)$, resulting in a noticeable drop in the Global AP (gAP).

Instead of directly using the negative distance (or equivalently, the density represented in eq. (6.10)), we can compute the posterior of the

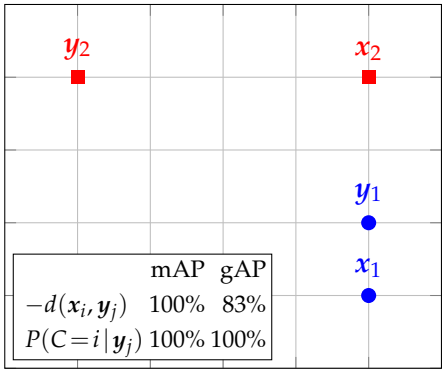


Figure 6.4. An instance of the multi-variance problem of traditional distance-based KWS. The color and shape of each point represents its true class, x represent candidate images and y represent image queries.
The the negative Euclidean distance scores are:
$-d(x_1, y_1) = -1$, $-d(x_2, y_1) = -2$, $-d(x_2, y_2) = -4$, $-d(x_1, y_2) = -5$.
On the other hand, eq. (6.12) (with $s = 1$) yields to:
$P(C = 2 \mid y_2) = 0.9991$, $P(C = 1 \mid y_1) = 0.9526$, $P(C = 2 \mid y_1) = 0.9526$, $P(C=1 \mid y_2) = 0.0009$.
The gAP for raw distances is 83%, while the candidate posteriors achieve 100% gAP. The mAP is identical in both cases because the two are monotonically related.

candidate given the query image:

$$P(C = i \mid Y = \boldsymbol{y}) = \frac{P(\mid Y = \boldsymbol{y} \mid C = i)P(C = i)}{\sum_{i'} P(\mid Y = \boldsymbol{y} \mid C = i')P(C = i')} \tag{6.11}$$

Now, let's assume that the prior among candidates is uniform, and define $\mathcal{K}(u) = \exp(-su^2)$, where s is simply a smoothing constant tuned on validation to maximize performance. Then, the previous equation becomes:

$$P(C = i \mid Y = \boldsymbol{y}) = \frac{\exp(-sd^2(\boldsymbol{x}_i, \boldsymbol{y}))}{\sum_{i'} \exp(-sd^2(\boldsymbol{x}_{i'}, \boldsymbol{y}))} \tag{6.12}$$

Since the denominator is independent of the i-th candidate, for a fixed query $\boldsymbol{y}$, eq. (6.12) is proportional to $\exp(-sd^2(\boldsymbol{x}_i, \boldsymbol{y}))$, and therefore monotonous with $-d(\boldsymbol{x}_i, \boldsymbol{y})$. Consequently, for a set of queries, the Mean AP (mAP) obtained using traditional distance-based ap-

proaches and that of eq. (6.12) will be identical. However, the normalization of eq. (6.12) provides a probabilistically sound KWS score, which is bounded in $[0, 1]$ and therefore uniformly comparable across different queries. As illustrated in fig. 6.4 and the experiments in section 8.10, this normalization entails significant gAP improvements with respect to using the raw negative distance KWS score.

As we already mentioned, there are countless types of kernels. The reader should know that, as our experiments suggest, the choice of the kernel is not critical (as long as it has the required properties) in order to achieve decent results (see experiments in section 8.10).

Finally, the reader may believe that normalizing the feature vectors would mitigate the multi-variance problem. For instance, many authors use unit vectors as features (e.g. [Aldavert et al., 2015, Sfikas et al., 2016]). This limits the maximum distance between two vectors on the unit hypersphere (i.e. for any norm, the distance between two unit vectors is always ≤ 2), but the multi-variance effect can also be present, as shown in fig. 6.5. One can also see from the figure that other popular measures for unit-normed spaces, such as the cosine similarity, will also present exactly the same problem. Thus, the multi-variance problem is not due to a particular feature space or dissimilarity measure.

We could try to come up with some feature representation and dissimilarity measure that does not present this problem. However, it is not clear that such requirement can be easily guaranteed across different data sets (languages, scripts, ages, etc.), especially using unsupervised methods for feature extraction, which is the usual case in traditional query-by-example KWS publications.

On the other hand, using the generative model presented earlier and the posterior of the candidate image given the query, solves the problem of multi-variance.

6.3.1.2 The multi-mode problem

In any case, we still may face an additional problem, caused by the naive generative model presented earlier (and implicitly assumed by most traditional KWS methods). Observe that the posterior in eq. (6.12)

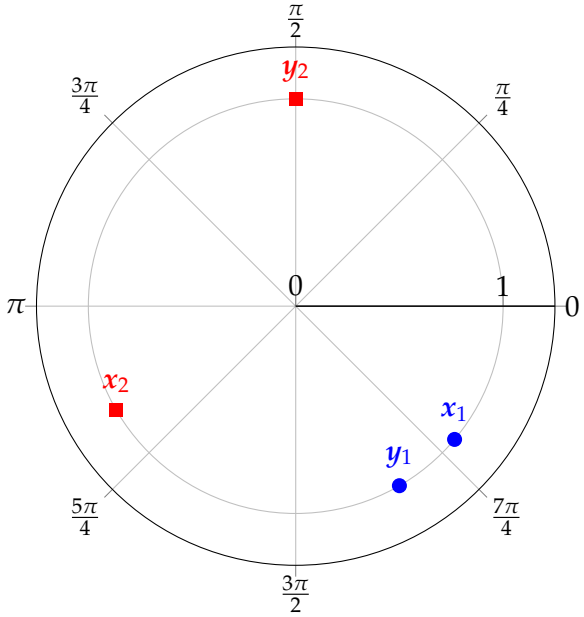


Figure 6.5. An instance of the multi-variance problem in a unit norm space. Similarly to the example depicted in fig. 6.4, the class "red-square" has a much larger variance than the class "blue-circle" and the Euclidean distances tend to be larger for the former class than the latter. Therefore, the pair of candidate-query samples $(\boldsymbol{x}_2, \boldsymbol{y}_1)$ is ranked higher than $(\boldsymbol{x}_2, \boldsymbol{y}_2)$, resulting in a drop in the Global AP (gAP).

is not equivalent at all to the (optimal) relevance probability derived in eq. (2.23) (review section 2.4).

Using the former candidate posterior probability (or the raw distances) to rank the candidate images may present the so-called "multi-mode problem". Observe that in reality we (usually) don't have as many classes as candidate images, but several candidate images are part of the same class (i.e. we have multiple instances of the same word). When one (or several) of the density functions conditioned on a class are multi-modal, a given query will be "close" to the candidates of the same mode, but may be arbitrarily "far away" from samples of other modes of the same class. This may happen, for instance, if one is considering a collection of documents with multiple

writers: it is reasonable to think that the density $p(\text{image} \mid \text{word})$ will have several modes (one for each writer).

In the example depicted in fig. 6.6, there are two classes (words), labeled as "blue-circle" and "red-square". The collection of candidate images are $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ and we consider only two queries, $\boldsymbol{y}_1, \boldsymbol{y}_2$, both from the "blue-circle" class. In addition, we have a set of labeled training samples z_1, z_2, z_3 in order to estimate the class (i.e. word) posteriors using a distance-based approach.

Observe that the class-conditional density $p(\boldsymbol{x} \mid \text{"blue-circle"})$ is bi-modal. Therefore, as we just mentioned, a query such as $\boldsymbol{y}_1$ is only close to images from the same mode (e.g. $\boldsymbol{x}_1$), while collection images of the same class but a different mode (e.g. $\boldsymbol{x}_2$) may be farther away than those from other classes, like $\boldsymbol{x}_3$. In this situation, the traditional distance-based KWS approach only achieves 83% mAP and the same gAP. Using eq. (6.12) does not alleviate the trouble either, yielding the same poor gAP and mAP.

However, if we follow the definition of the relevance probability, given candidate image $\boldsymbol{x}$ and query $\boldsymbol{y}$ (as we explained in section 2.3), we get:

$$P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}) = \sum_{v \in \Sigma} P(W = v \mid X = \boldsymbol{x}) P(V = v \mid Y = \boldsymbol{y}) \tag{6.13}$$

Recall, this simply means that the relevance probability, given a query $\boldsymbol{y}$ and a candidate $\boldsymbol{x}$, is equal to the sum over all possible keywords of the product of the word-posterior given $\boldsymbol{x}$ and the word-posterior given $\boldsymbol{y}$.

Given that we have a few training samples, we are able to estimate $P(W = v \mid X = \boldsymbol{x})$ (respectively $P(V = v \mid Y = \boldsymbol{y})$) using the traditional k-nearest neighbor classifier. Let $\mathcal{N}(k, \boldsymbol{x})$ and $\mathcal{N}(k, \boldsymbol{y})$ be the sets of k training images which are nearest neighbors of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, according to the given distance $d(\cdot, \cdot)$. Let $\mathcal{N}(k, \boldsymbol{x}, v)$ and $\mathcal{N}(k, \boldsymbol{y}, v)$ be, respectively, the subsets of images in $\mathcal{N}(k, \boldsymbol{x})$ and $\mathcal{N}(k, \boldsymbol{y})$ labeled with the word v and let $k_{\boldsymbol{x}v}$, $k_{\boldsymbol{y}v}$ be the sizes of these

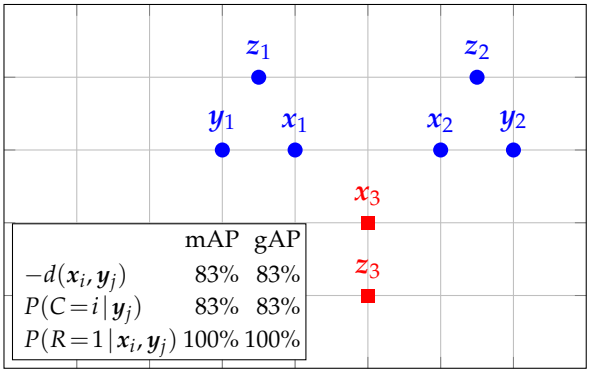


Figure 6.6. An instance of the multi-mode problem of traditional distance-based KWS (sample colors and shapes as in fig. 6.4).
Negative Euclidean distance scores:
$-d(x_1, y_1) = -d(x_2, y_2) = -1$, $-d(x_3, y_1) = -d(x_3, y_2) = -\sqrt{5}$, $-d(x_1, y_2) = -d(x_2, y_1) = -3$.
Posterior probabilities given by eq. (6.12) (with $s = 1$):
$P(C = 1 \mid y_1) = P(C = 2 \mid y_2) = 0.9817$, $P(C = 3 \mid y_1) = P(C = 3 \mid y_2) = 0.0180$, $P(C=1 \mid y_2) = P(C=2 \mid y_1) = 0.0003$.
Relevance probabilities given by eqs. (6.13) and (6.15) (with $k = 3, s = 1$):
$P(R \mid x_1, y_1) = P(R \mid x_2, y_2) = P(R \mid x_1, y_2) = P(R \mid x_2, y_1) = 0.9721$, $P(R \mid x_3, y_1) = P(R \mid x_3, y_2) = 0.0155$.
Ranking according to $-d(x, y)$ or $P(C \mid y)$ yields the same poor mAP and gAP, but using $P(R \mid x, y)$ does provide perfect results.

sets. Then:

$$P(W = v \mid X = x) = \frac{k_{xv}}{k}, \quad P(W = v \mid Y = y) = \frac{k_{yv}}{k} \tag{6.14}$$

However, to achieve best performance in real, finite cases, a smoothed version of eq. (6.14) is generally adopted to take into account not only the number of neighbors, but also the actual distances involved. To this end, we use the same Gaussian kernel than before, leading to:

$$\begin{aligned} P(W=v \mid \mathbf{x}) &= \frac{\sum_{\mathbf{z} \in \mathcal{N}(k,\mathbf{x},v)} \exp(-s\, d^2(\mathbf{z}, \mathbf{x}))}{\sum_{\mathbf{z}' \in \mathcal{N}(k,\mathbf{x})} \exp(-s\, d^2(\mathbf{z}', \mathbf{x}))} \\ P(V=v \mid \mathbf{y}) &= \frac{\sum_{\mathbf{z} \in \mathcal{N}(k,\mathbf{y},v)} \exp(-s\, d^2(\mathbf{z}, \mathbf{y}))}{\sum_{\mathbf{z}' \in \mathcal{N}(k,\mathbf{y})} \exp(-s\, d^2(\mathbf{z}', \mathbf{y}))} \end{aligned} \tag{6.15}$$

This way, we are able to give a well-principled probabilistic approach to the problem of query-by-example KWS, even with the original distances that were employed originally in an heuristic manner.

As a result of using the well-principled probabilistic framework for Keyword Spotting, proposed in this thesis, we are able to overcome both the multi-variance and multi-mode problems presented here, that traditional distance-based approaches to Keyword Spotting may suffer. As we shall see in section 8.10, these problems are not mutually exclusive: a method can suffer from both problems simultaneously.

The candidate posteriors that we proposed in eq. (6.12) can mitigate the multi-variance problem, but not the multi-mode. However, assuming that the class posteriors are properly estimated, the framework presented in this thesis solves both issues.

6.4 PHOC-based methods

A Pyramid of Histograms of Characters (PHOC) [Almazán et al., 2014] is a hierarchical set of character string "binary histograms", where a histogram bin is "1" if the character is present in the (sub)string and "0" if not. Here we prefer not to abuse the language and refer to it as a bit vector representation of the set of characters which appear in the string, or CS for short. So the CS of the string "`acbbab`" is `{a,b,c}` and that of the string "`a`" is `{a}`.

Let v be a word and v_c its character spelling. The first PHOC level is the (single) CS of v_c. At the l-th level, v_c is split in l disjoint substrings, and a CS is produced for each substring. Typically $l \leq 5$. Thus, if v_c = "`KOOKY`", for example, its PHOC encompasses the CSs shown in table 6.1.

For any word v shorter than 6 characters, its highest PHOC level is a sequence of CSs containing at most one character each. So, the PHOC of v is just v_c, accompanied by superfluous spelling information in the lower levels. For longer words, the intrinsic redundancy of spelling in natural languages also results in last-level CSs which typically represent unique words. In any case, the lower-level CSs contain superfluous information. Figure 6.7 shows that such an (almost) strict

Table 6.1. PHOC representation of the word "kooky" using a 5-level pyramid of binary histograms (i.e. character sets). Notice that at the last level, because the length of the word is equal than the number of character sets, each set contains at most one character, thus the sequence of sets is equivalent to the spelling of the word.

Level (splits)	Character sets (CSs)
1	{K,O,Y}
2	{K,O} {K,O,Y}
3	{K,O} {O} {K,Y}
4	{K} {O} {K,O} {Y}
5	{K} {O} {O} {K} {Y}

uniqueness of PHOC word representations actually happens in two of the most commonly used data sets in KWS research (i.e. IAM and George Washington databases).

6.4.1 PHOCNET

Originally, Support Vector Machines (SVM) were adopted to predict $(0/1)$ PHOC components from Fisher feature vectors extracted from word-cropped images, in [Almazán et al., 2014]. SVM outputs were then calibrated by means of elaborate ad-hoc methods and interpreted as posterior probabilities. The results of [Almazán et al., 2014] were later improved by the work presented in [Sudholt and Fink, 2016,Sudholt and Fink, 2017], using convolutional neural networks (CNN) to both extract "deep" image features and predict the PHOC components.

Let $\boldsymbol{x}$ be a a given word-cropped image, rendering a single unknown word v. Let v_c be the spelling of v and v_h the PHOC of v_c. A PHOCNet [Sudholt and Fink, 2016] tries to predict v_h from $\boldsymbol{x}$. Clearly, because of the one-to-one PHOC-word correspondence that we just showed, PHOCNet is essentially only one more (among the many) proposed character-based recognizers of isolated word images.

However, we hypothesize that since it needs to predict also the superfluous information in (the lower levels of) v_h, PHOCNet may be in disadvantage in terms of training demands with respect to other

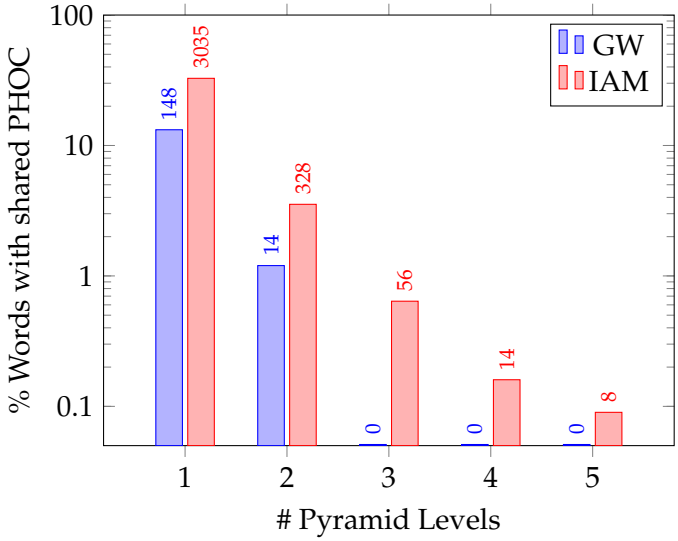


Figure 6.7. Percentage of words that share the PHOC representation with other words in IAM and George Washington data sets. For GW, a perfect one-to-one mapping is produced with only 3-level PHOCs. For IAM, a perfect mapping would require 7 levels, but with 4-level PHOCs only 0.16% of the words share their PHOC representation. The exact number of words sharing their PHOCs is shown above each bar.

less intricate, conventional text recognizers, which predict a sequential representation of v, namely v_c.

PHOCNet is trained by minimizing the Binary Cross Entropy loss function, assuming each PHOC component is independent of the others [Sudholt and Fink, 2016]. This assumption is obviously false; in particular the low-level histograms are obviously correlated with the upper levels. Yet, the method still works very well in practice.

For a given input image $\boldsymbol{z}$ a trained PHOCNet produces estimates of the posterior probabilities $P(H_i = 1 \mid \boldsymbol{z})$, $1 \leq i \leq D$, where H_i is a binary random variable associated to the i-th component of a PHOC. Let $\hat{h}_i(\boldsymbol{z})$ be the i-th estimate and $\hat{\boldsymbol{h}}(\boldsymbol{z})$ a vector composed of all the D estimates. For QbE KWS, $\hat{\boldsymbol{h}}(\boldsymbol{x})$ is obtained for each collection image $\boldsymbol{x}$. Then, for a given query image, $\boldsymbol{y}$, $\hat{\boldsymbol{h}}(\boldsymbol{y})$ is similarly produced and the collection images are ranked according to the Bray-Curtis [Sud-

holt and Fink, 2016], or the cosine dissimilarities between $\hat{\mathbf{h}}(\boldsymbol{x})$ and $\hat{\boldsymbol{h}}(\boldsymbol{y})$ [Sudholt and Fink, 2017].

Even if the PHOCNet approach could quite naturally be considered "recognition-based", it still overlooks standard optimal-decision principles of Pattern Recognition and sticks to the most traditional KWS view of just heuristically relying on image representation dissimilarities. Below we will explain how the PHOCNet posterior estimates can be properly used in the optimal QbE KWS framework introduced in section 2.3.

6.4.2 PROBABILISTIC PHOCNET

We can apply a similar marginalization approach that lead to eqs. (2.23) and (6.13), in order to use PHOCNet posterior estimates. Observe that here we do not have an estimate of $P(W \mid X = \boldsymbol{x})$ and $P(V \mid Y = \boldsymbol{y})$, but $P(H \mid X = \boldsymbol{x})$ and $P(H' \mid Y = \boldsymbol{y})$.

$$
\begin{aligned}
P(R = 1 \mid X = \boldsymbol{x}, Y = \boldsymbol{y}) = \\
\sum_{v \in \Sigma^*} P(W = v \mid X = \boldsymbol{x}) P(V = v \mid Y = \boldsymbol{y}) = \\
\sum_{v \in \Sigma^*} \left(\sum_{\boldsymbol{h}} P(W = v, H = \boldsymbol{h} \mid X = \boldsymbol{x}) \right) \left(\sum_{\boldsymbol{h}'} P(V = v, H' = \boldsymbol{h}' \mid Y = \boldsymbol{y}) \right) = \\
\sum_{v \in \Sigma^*} \left(\sum_{\boldsymbol{h}} P(H = \boldsymbol{h} \mid X = \boldsymbol{x}) P(W = v \mid X = \boldsymbol{x}, H = \boldsymbol{h}) \right) \\
\left(\sum_{\boldsymbol{h}'} P(H' = \boldsymbol{h}' \mid X = \boldsymbol{y}) P(V = v \mid Y = \boldsymbol{y}, H' = \boldsymbol{h}') \right)
\end{aligned}
\tag{6.16}
$$

Observe that, in general, W and V can be assumed to be conditionally independent of the image if the corresponding PHOC representations (i.e. H and H') are given. This assumption implies that $P(W = v \mid X = \boldsymbol{x}, H = \boldsymbol{h}) = P(W = v \mid H = \boldsymbol{h})$ (and equivalently for the probabilities involving the query image). In addition, if we assume that there is a one-to-one correspondence between words and PHOC representations, as we described earlier, the previous equation simpli-

fies to:

$$P(R=1 \mid X=\boldsymbol{x}, Y=\boldsymbol{y}) \overset{\star}{=} \sum_{\boldsymbol{h}} P(H=\boldsymbol{h} \mid X=\boldsymbol{x}) P(H'=\boldsymbol{h} \mid X=\boldsymbol{y}) \quad (6.17)$$

This equation implies that we can compute the relevance probability using the PHOCNet output of the query and the word images.

However, recall that $\boldsymbol{h}$ is a binary vector of D components (i.e. the vector of elements h_i, $1 \leq i \leq D$), and D must be very large to have a one-to-one mapping between PHOC representations and words. Thus, generally computing eq. (6.17) may be infeasible in practice.

However, as the PHOCNet method does during training, we can assume that each of the D random variables is independent from the others when the image is given, resulting in:

$$P(H=\boldsymbol{h} \mid X=\boldsymbol{x}) \overset{\star}{=} \prod_{i=1}^{D} P(H_i=h_i \mid X=\boldsymbol{x}) = \prod_{i=1}^{D} \hat{h}_i(\boldsymbol{x})^{h_i} (1-\hat{h}_i(\boldsymbol{x}))^{1-h_i} \quad (6.18)$$

$$P(H'=\boldsymbol{h} \mid X=\boldsymbol{y}) \overset{\star}{=} \prod_{i=1}^{D} P(H'_i=h_i \mid Y=\boldsymbol{y}) = \prod_{i=1}^{D} \hat{h}_i(\boldsymbol{y})^{h_i} (1-\hat{h}_i(\boldsymbol{y}))^{1-h_i} \quad (6.19)$$

where $\hat{h}_i(\boldsymbol{x})$ is the i-th output of the PHOCNet when $\boldsymbol{x}$ is given as its input (i.e. $\hat{h}_i(\boldsymbol{x}) = P(H_i = 1 \mid X = \boldsymbol{x})$), and equivalently for the probabilities involving the query image.

Finally, given this independence assumption and using the PHOCNet to model the probabilities, we can approximate the relevance probability as:

$$P(R=1 \mid X=\boldsymbol{x}, Y=\boldsymbol{y}) \overset{\star}{=} \sum_{\boldsymbol{h}} \prod_{i=1}^{D} (\hat{h}_i(\boldsymbol{x}) \hat{h}_i(\boldsymbol{y}))^{h_i} ((1-\hat{h}_i(\boldsymbol{x}))(1-\hat{h}_i(\boldsymbol{y})))^{1-h_i} \quad (6.20)$$

This sum can be efficiently computed using a dynamic programming algorithm in $\mathcal{O}(D)$, which is the same asymptotic cost as the Bray-Curtis or the cosine dissimilarities used in papers using the PHOCNet approach [Sudholt and Fink, 2016, Sudholt and Fink, 2017].

To summarize this section, we have derived an efficient way (same complexity as the Bray–Curtis dissimilarity and cosine similarity) of computing a probabilistically sound measure (i.e. the relevance probability presented in section 2.3.1.1), which can be used to rank pairs of images, for word-segmented query-by-example scenarios, based on the PHOC representation of words.

7 Beyond Traditional Keyword Spotting

7.1 Multi-word spotting in handwritten documents

During this thesis we framed Keyword Spotting as a particular instance of an Information Retrieval application (see section 1.3 and chapter 2). In section 2.3 we explained how the probabilistic framework employed can cope with both query-by-string and query-by-example scenarios, seamlessly. However, we always assumed the query to be an individual keyword, which is the traditional scenario in KWS publications.

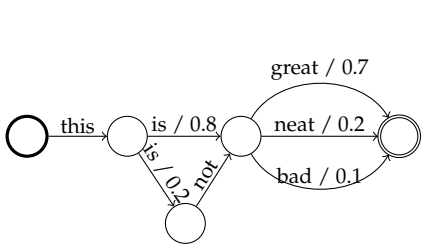
Nevertheless, the same probabilistic framework can be used as well to tackle more complex types of queries. Since we represent the probability distribution of transcripts of a text image as a Weighted Finite State Transducer, we can easily compute the relevance probabilities described in chapter 2 using simple WFST operations, if we manage to represent the query as a WFST as well.

The complexity of the WFST equivalent to our query will vary depending on whether we work with a lexicon-free or lexicon-based model, whether the query is presented as a string or an image, etc.

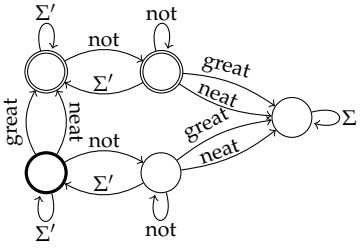
In any case, the important remark is that in our approach the query is simply another FST (possibly weighted) that is *composed* with the WFST of a particular text image (or any other *stochastic* source of information).

Given that FSTs are a generalization of Finite State Automatons, and the latter are equivalent to regular languages, we can use any type of query that can be represented as a (weighted) regular language or, equivalently, a regular expression. This is illustrated in

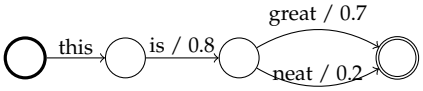
fig. 7.1, where the relevance probability of a stochastic document is computed for a query expressed as the regular expression `(great ∨ neat) ∧ ¬("not great" ∨ "not neat")`. Here, we compute the relevance probability defined in section 2.1. That is, a given document is relevant for the query if the content of the document is in the regular language represented by the query.



(a) T_1: WFST of the document $\boldsymbol{x}$



(b) T_2: DFA of the query $\mathcal{Q}$



(c) $T_1 \circ T_2$: Possible transcripts of the document that are relevant to the query

Figure 7.1. Example of the computation of the relevance probability for a given regular expression. The content of the stochastic document $\boldsymbol{x}$ (e.g. a text line image) is represented as a WFST. The regular expression query, $\mathcal{Q}$, `(great ∨ neat) ∧ ¬("not great" ∨ "not neat")` is represented as a DFA. Σ'-transitions denote all labels in Σ not present in the other state's transitions. The relevance probability, is the total probability mass in the WFST result of the composition of the two previous transducers. $P(R = 1 \mid X = \boldsymbol{x}, Q = \mathcal{Q}) = 0.72$.

Despite the fact that our approach can handle such a rich set of queries, it does not mean that it is worthy to implement this full support in practice. After all, there is a reason why most popular web search engines (e.g. Bing, Google, Yahoo!, etc.) *do not* support generic regular expressions as queries.

In order to support fast searches that scale well with the number of indexed documents, word-based search indexes have to be built. When we do so, we are essentially able to serve the results of the users requests in a time which *does not* depend on the number of documents indexed in our database. Yet, in order to support all types

of regular expressions we would need to store the lattice for each of the documents in our collection, and then compute this composition at query time, which would become impractical once the number of documents (and users) reaches the thousands.

Although we cannot compute the *exact* relevance probability without the lattices, we can compute good approximations to it using probabilistic word indexes obtained from the algorithms (described in chapter 5) and the use of Fréchet bounds (see section 2.5.1). In particular, in order to allow for the so-called *phrase queries*, we build a positional probabilistic word index using the algorithms described in section 5.1.3 (if we use lexicon-based lattices) or section 5.3.3 (if we use lexicon-free lattice).

Instead of providing bounds of the relevance probability, we typically give a single value to the user by using the min operation in the conjunction, and the max operation in the disjunction (the upper and lower Fréchet bounds of each operation, respectively). This is a very natural way of combining single probabilities and we used it effectively in [Toselli et al., 2018b] (see experiments in section 8.12). Also, notice that the phrase queries (such as `"not neat"` or `"not great"`) can be interpreted as the conjunction of the multiple words with the additional restriction that their text positions are contiguous.

In particular, our large scale demonstrations allow for multi-word queries with support of Boolean expressions and phrase queries at word level. At query time, we use the parsing tree of the query to perform intersection (conjunction operation) and union (disjunction) of partial results and use the Fréchet approximations to the relevance probability. Although it would be possible in theory, our current implementation does not support prefix or suffix searches, and other more advanced regular expression operations. Yet, we believe that Boolean expressions and phrase queries cover the vast majority of the queries that typical users need.

Table 7.1 shows an example where we use a positional probabilistic index of the example depicted in fig. 7.1, in order to compute an approximation to the relevance probability (and its bounds). In section 2.2.3 we explained how this positional index is computed from a word lattice. For instance, the probability of the pair of word–position

(this, 1) is equal to 1.0, because all paths in the word lattice contain this word in that position. On the contrary, the probability of the pair (not, 3) is 0.2, since only one path (whose posterior probability is 0.2) includes that word in that position.

Table 7.1. Example computing the relevance probability of a multi-word query. The ultimate goal is to compute an approximation to the relevance probability of the query $\mathcal{Q}$ depicted in fig. 7.1. For that, we use the Fréchet bounds and the probabilistic positional index in table 7.1a.

(a) Probabilistic positional index

Term	Pos.	Prob.
this	1	1.0
is	2	1.0
not	3	0.2
great	3	0.56
neat	3	0.16
bad	3	0.08
great	4	0.14
neat	4	0.04
bad	4	0.02

(b) Relevance probabilities of different queries

	Relevance probability		
Query	Exact	Approx.	Bounds
`not`	0.2	0.2	$[0.2, 0.2]$
`great`	0.7	0.56	$[0.56, 0.7]$
`neat`	0.2	0.16	$[0.16, 0.2]$
`great` $\vee$ `neat`	0.9	0.56	$[0.56, 0.9]$
`"not great"`	0.14	0.14	$[0, 0.14]$
`"not neat"`	0.04	0.04	$[0, 0.04]$
`"not great"` $\vee$ `"not neat"`	0.18	0.14	$[0, 0.2]$
$\neg$`("not great"` $\vee$ `"not neat")`	0.82	0.86	$[0.8, 1]$
$\mathcal{Q}$	0.72	0.56	$[0.36, 0.9]$

Following the Fréchet bounds we can compute an approximated value of each query using the probabilistic index, as table 7.1a shows. For example, there are two entries associated to the word `"great"`

(since there is one path with that word at position 3, with posterior 0.56, and another at position 4, with posterior 0.14). The exact relevance probability, computed directly from the lattice, is equal to 0.7, which in this case is the sum of the two. The exact probability will be in the range $[\max\{0.56, 0.14\}, \min\{1, 0.56 + 0.14\}] = [0.56, 0.7]$, according to the Fréchet bounds. Notice that, in general, this probability is not equal to the sum of all entries. For example, if the same word is repeated multiple times within the same transcript hypothesis, the relevance probability *is not* the sum of all the index entries. In our implementation, we usually keep only the *lower* bound of this interval, for OR (and single word) queries, that is 0.56, which is shown in the "Approx." column of the table 7.1b.

Similarly, we can also use the Fréchet bounds to approximate the value of AND (and phrase) queries. Take for instance the query `"not great"`. Its exact relevance probability is 0.14, since there is only one path in the lattice containing this sequence of words, with this posterior probability. There is one entry for the word "not" in the index, at position 3, with probability 0.2, thus we need to take into account the entry of the word "great" at position 4, with probability 0.14. Thus, the Fréchet bounds for the relevance probability of the phrase query are $[\max\{0, 0.2 + 0.14 - 1\}, \min\{0.2, 0.14\}] = [0, 0.14]$. In our implementation, we typically keep the upper bound for AND (and phrase) queries, as shown in the "Approx." column of the table 7.1b.

7.2 The future of Keyword Spotting with perfect transcripts

One of the main reasons that motivated the development of Keyword Spotting, back in the 1990s, was to mitigate the problems of indexing inaccurate transcripts provided by the handwritten text recognition systems.

However, in the last ten years the recognition accuracy of different HTR approaches have improved considerably and the character error rate in most (academic) databases is below 5%.

Thus, the reader might wonder what the future of Keyword Spotting (and this thesis) will be once the accuracy reaches a 0% error rate. After all, once we achieve a reasonable low error rate, why don't we

simply automatically transcribe all the documents and use a typical search engine to index them?

This apparently simple (but important) question hides two key assumptions.

First, it assumes that the performance on academic benchmarks is a good proxy of the performance of a technology in real scenarios. Yet, it is well known that real collections of historical documents are much "messier" than typical academic benchmarks. The documents are sometimes badly preserved, layout analysis is trickier, they cannot be manually segmented into lines (much less into words), the reading order is unclear (especially when dealing with multi-column documents), etc. In addition, results shown in academic publications are prone to underestimate the error rates since we (researchers) keep using a few (rather small) data sets and improve the results on them iteratively by looking at what worked best in previous publications.

Secondly, it assumes that the transcription of a text image is a *deterministic* (non-stochastic) process. That is, we assume that there is only one *right* transcript of the given image. However, in reality, professional transcribers do not always agree on their transcripts, especially when ancient (and not well-preserved) documents, or documents with many abbreviations or difficult writing styles are considered.

Even if we assume that the transcription of a text image is actually a deterministic process, and that we can characterize it (i.e. given a text line image we can obtain unequivocally its only possible transcript), still many other applications exist where this assumption is obviously false. In all such cases, we could essentially use all the methods described in this thesis.

For instance, in order to illustrate such scenario, we will consider the so-called *Image Captioning* problem, where the goal is to obtain a textual description of a given image. Under this perspective, we can interpret the process of transcribing a text image as an instance of Image Captioning, where the set of images is restricted to handwritten images containing text, and the "description" (or caption) of the image is just the transcript of this text.

In practice, though, Image Captioning refers to describing "natural" images, such as landscapes or portraits captured with a digital camera.

Clearly, for a given natural image, there are multiple possible descriptions, and if we asked two persons to write down their description of such picture, these would hardly be identical. Even if the two persons did interpret the same "meaning" of the image, the wording (description) of such "meaning" would likely not be the same, given the large amount of ambiguity present in natural languages.

In order to illustrate this phenomenon, let's consider the Flickr-30k data set, which is typically used in academic publications tackling Image Captioning. This data set is composed of 31 783 natural images (pictures from the Flickr[1] website) and five annotations for each of them, provided by different persons. For instance, fig. 7.2 shows an image from the data set and its five reference descriptions.

The example shows that although the general meaning of the picture is common for all five annotators, their description is indeed very different, and there are some nuances that are only described by one annotator (e.g. "mature hikers", "playing golf", etc).

Given this, if a random person had to describe the content of the image, we could assume that she would produce one of the five descriptions with equal probability. This is equivalent to say that $P(W = w_i \mid X = \boldsymbol{x}) = \frac{1}{5}, 1 \leq i \leq 5$.

Typically, in Image Captioning publications, they measure the BLEU score between the system's annotation and the reference one(s). However, since we want to tackle this problem as an Information Retrieval application, we can imagine that we want to retrieve relevant images given the user queries in our web page (i.e. like in a traditional Content-based Image Retrieval application). Thus, given a set of queries, we can compute the Global and Mean Average Precision performance measures.

In particular, we will compare two systems that use the *true* probability distribution $P(W \mid X)$ for all images in the Flickr-30k collection.

[1] `http://www.flickr.com`

Figure 7.2. Example of an image from the Flickr-30k data set used in Image Captioning publications. The reference transcripts for this image are:
w_1 = "Two men wearing hats and holding canes are standing silhouetted against a large body of water with sunlight reflecting off the water and a tree to the side".
w_2 = "Two men wearing hats and using walking sticks are walking near a body of water during sundown".
w_3 = "Two men stand beneath a tree as they watch the sunset over the ocean".
w_4 = "Two mature hikers take a break and admire the sunset over the lake".
w_5 = "Two men playing golf near water while the sun is setting".

The first system uses a deterministic index on the output of a traditional Image Captioning solution, i.e. it obtains $\arg\max_w P(W \mid X)$. Then, it retrieves all images that contain the given query word. This system, given that $P(W \mid X)$ is the *true* distribution, gives the Bayes optimal decision for the *expected value* of the 0–1 loss.

In the second case, an index is built using our probabilistic framework. More precisely, $P(R = 1 \mid X = \boldsymbol{x}, V = v)$, for each image $\boldsymbol{x}$ and each word in the data set, as we described in section 5.1.1. Then, given a query keyword, it retrieves all images in decreasing order of the relevance probability. As we proved in chapter 3 (particularly in sections 3.2.3 and 3.3), given that we are using the *true* distribution, this

system gives the Bayes optimal ranking that maximizes the expected value of the Average Precision.

We use all the words in the data set as queries, excluding the stopwords. This gives a total of 23 346 keyword queries. Given this set of queries and the *true* distribution of the descriptions in the images, we compute the *expected value* of the Global and Mean Average Precision. Notice that we need to compute an expected value because the reference description of each image is an stochastic process.

Because the cost of computing the exact value of the expected Average Precision is very large, we compute an approximate value by making repeated trials. In each trial, we pick the reference annotation of the image randomly, and then we average the results over the multiple trials. In addition, we compute 95% bootstrapped confidence intervals. Table 7.2 shows the results obtained in this illustrative experiment.

Table 7.2. Global and Mean Average Precision on the Flickr-30k data set with a deterministic and probabilistic indexes, when the ground-truth (i.e. true) distribution is known. BCa bootstrapped confidence intervals at 95% are shown.

Index	gAP (%)	mAP (%)
Deterministic (1-best)	18.1 [18.0–18.1]	18.3 [18.2–18.3]
Probabilistic	62.5 [62.5–62.5]	56.6 [56.6–56.6]

As the table shows, using the probabilistic index which optimizes the value that we are measuring, clearly improves the results that a traditional solution would give. Not only that, given the proofs in chapter 3 we can assert that no other system will perform better than the one based in our probabilistic index (at least, if the expected Average Precision or other measures described in chapter 3 are used).

Thus, the reader can rest assured that the methods developed in this thesis are worthy not only for traditional KWS applications, but also for other related applications, with a large interest in current research and industrial settings.

8 Experiments

In this chapter we will experimentally validate the probabilistic framework, algorithms and claims, described through the previous chapters. In particular, some of the questions that we aim to answer in this chapter are:

1. How can the different relevance probabilities defined in chapter 2 be used under a line-level KWS paradigm, given that some are more efficient to index (see algorithms in chapter 5)? This will be developed in section 8.2.

2. What effect has the language model in a KWS task? Are lexicon-based or lexicon-free approaches preferable? These questions are tackled in section 8.3.

3. How does the number of training samples affect the performance of our probabilistic indexes? This is studied in section 8.4.

4. Given that both KWS and HTR use the same underlying probability distribution, what is the correlation between the performance on HTR and KWS tasks? This topic is examined in section 8.5.

5. How does our approach compares with state-of-the-art line-level KWS works? This is studied in section 8.6.

6. How does the use of more traditional statistical models for handwritten text affects the KWS assessment measures of our approach? We analyze this question in section 8.7.

7. How can we use our line-oriented probabilities and indexes to tackle KWS under a segmentation-free approach? In section 8.8 we empirically evaluate it.

8. How do other seminal works in KWS relate to our approach, from an experimental point of view? We already studied this in chapter 6, and we provide some experimental justification in sections 8.9 to 8.11.

9. What is the performance of our probabilistic approach to tackle, in a efficient but approximate way, Boolean queries containing multiple words? The theoretical justification of our approach was given in section 7.1, and we evaluate it in section 8.12.

8.1 Overview of the experimental setup

8.1.1 Databases

Most of the experiments reported here are evaluated at line level. That is, for a given query, our system has to determine the set of relevant text lines in a particular data set, regardless of the position of the query keyword in the text line, or the number of occurrences within the line. Nevertheless, some experiments under the fully segmentation-free paradigm are also reported, as well as a couple of experiments under the word-segmentation assumption.

Different databases are used as well to evaluate these experiments, but most of them (at line level) are conducted only using the IAM database (see appendix A.3). This is considered to be one of the toughest seminal databases under the line-level KWS paradigm, for which typically the KWS performance is much lower than other line-level data sets (such as George Washington, or Parzival). Yet, we will also report line-level results on other databases for completeness in the comparison with previous state-of-the-art works on KWS.

8.1.2 Statistical models for handwritten text

Most of the experiments described here have been conducted using neural networks based on convolutional and one-dimensional LSTMs layers (i.e. CRNNs, for short), as described in section 4.3.5. However,

some experiments using traditional Hidden Markov Models (with Gaussian Mixture densities) are also performed, in order to highlight that our approach can be used regardless of the underlying probabilistic model.

In the case of the IAM experiments, we use the simplest neural network architecture presented in [Puigcerver, 2017]. For the rest of the data sets, we will describe the employed architecture in each case. We have performed most of the experiments using PyLaia[1], a toolkit developed keeping in mind the needs of researchers working with handwritten documents.

In most of the experiments we make use of the RMSProp algorithm to perform the gradient descent updates of the neural network's parameters. In particular, for we used the setting described in [Puigcerver, 2017]. In the cases that we used a different setting, we will indicate so.

We used the Kaldi toolkit[2] to combine the output distribution of the neural network with the n-gram language models. The n-grams were trained typically using the OpenGrm libraries, with Kneser–Ney smoothing and interpolation.

8.1.3 EVALUATION PROTOCOL

We typically report both the Mean and Global Average Precision. In order to compute these, we used the so-called *interpolated precision* [Manning et al., 2008]:

$$\hat{\pi}_m = \max_{m' > m} \pi_{m'} \tag{8.1}$$

where π_m is the recall at the m-th position of the ranked list.

The interpolated precision is justified from a practical point of view: one typically would look at a few more results if this increases the percentage of relevant retrieved documents (that is, the precision of the larger set is higher). In addition, the interpolated precision at a recall of 0 is well-defined, whereas plain precision is not.

[1] `https://github.com/jpuigcerver/PyLaia`

[2] `http://kaldi-asr.org/`

Recall that the Average Precision is the area below the Recall–Precision curve. In order to compute this area, we use the *trapezoid integral* in most of the experiments. For a list of M ranked results simply means, this (together with the fact that we use the interpolated precision) simply means that the Average Precision is computed as:

$$\mathrm{AP} = \hat{\pi}_1 \rho_1 + \sum_{m=2}^{M} \frac{\hat{\pi}_{m-1} + \hat{\pi}_m}{2} (\rho_m - \rho_{m-1}) \tag{8.2}$$

where $\hat{\pi}_m$ and ρ_m are the interpolated precision and recall at the m-th position of the ranked list.

Nevertheless, in some experiments, for a fair comparison with previously published results, we have adopted a different evaluation criterion, that we will state in the corresponding section.

Notice that the Mean Average Precision can only be computed for *pertinent* queries. That is, queries for which at least one relevant element exists. This is because for non-pertinent queries, the Average Precision is undefined. In some data sets that we use, the query sets contain several non-pertinent keywords. In such cases, the Global AP is computed taking into account the full query set, but the Mean AP only takes into account the results of the pertinent keywords.

In addition, in some experiments we also plot the Recall–Precision curves to give a deeper insight on the performance of the corresponding KWS system.

8.2 Comparison of different relevance probabilities

In section 2.5 we saw that all relevance probabilities presented through chapter 2 are related. In particular, given an image $\boldsymbol{x}$ and query v, the maximum position-dependent relevance probability is a lower bound of the position-independent relevance probability, i.e. $P(R=1 \mid X=\boldsymbol{x}, V=v)$.

If we are tackling a line-oriented KWS scenario, where the relevance of a full text line for a given query keyword has to be determined, the natural approach would be to use $P(R=1 \mid X=\boldsymbol{x}, V=v)$ to answer this question. However, as we saw in chapter 5, creating a probabilistic word index from this distribution has a higher

asymptotic cost than creating a probability index from other position-dependent distributions.

Thus, in this section we will study the line-level KWS performance of the indexes created for each of the relevance distributions introduced in chapter 2, taking also into consideration the time required to construct them. In order to evaluate the position-dependent indexes in a line-level setting, where we only need to determine whether or not the full text line is relevant, we will simply keep, for each word in each line, the location (i.e. column, segment, or transcript position) with the maximum relevance probability, as the Fréchet bounds from section 2.5 suggest.

In order to do so, we evaluate each approach on the IAM database, measuring the Mean and Global AP. We consider both lexicon-based and lexicon-free approaches. In the first case, we use a word 3-gram language model with a vocabulary of 50 000, while in the second case we use a character 8-gram. In the lexicon-free case, some algorithms need to set the maximum number of (pseudo-)words to index per line, and we fixed this number to 100 (later on, in section 8.3.2.2, we will study how this tunable parameter affects both the mAP and gAP).

The relevance probability used in each case, and the algorithms used to build the respective word index are:

- Position independent (line-level) relevance probability, indicated by $P(R=1 \mid X=\boldsymbol{x}, V=v)$, and described in section 2.1. The lexicon-based algorithm to build such index is explained in section 5.1.1. In the lexicon-free case, we simply convert the character lattice into a word lattice, as explained in section 5.3.1, and finally we use the lexicon-based algorithm.

- (Maximum) Column relevance probability, given by the expression $\max_c P(R=1 \mid X=\boldsymbol{x}, V=v, L_\tau=c)$, and introduced in section 2.2.1. Although we have not explicitly introduced the algorithm in this thesis, we have briefly described it in section 5.1.2.2. In the lexicon-free case, we first convert the lattice into a word lattice, as in the previous approach, and the use the lexicon-based algorithm.

- (Maximum) Segment relevance probability, presented in section 2.2.2, and denoted by $\max_{c_0,c_1} P(R=1 \mid X=\boldsymbol{x},V=v,L_\sigma=(c_0,c_1))$. The lexicon-based algorithm is described in section 5.1.2, and the lexicon-free algorithm in section 5.3.2.

- (Maximum) Transcript position relevance probability, denoted by $\max_k P(R=1 \mid X=\boldsymbol{x},V=v,L_\kappa=k)$, described in section 2.2.3. The lexicon-based algorithm is described in section 5.1.3, and the lexicon-free algorithm in section 5.3.3.

All lexicon-free approaches need a set of delimiter characters to automatically extract the (pseudo-)words from the sequence of characters in the text line transcript(s), as we explained in section 5.3. In the case of the IAM database, the characters used as delimiters are: "#", "&", "(", ")", "*", ":", ";", "?" and the whitespace symbol.

Table 8.1. Average Precision (both Mean and Global) on the IAM database for different relevance probabilities, evaluated in a line KWS setting. Results using lexicon-based and lexicon-free are shown in tables 8.1a and 8.1b, respectively. In both cases, a CRNN and a n-gram language model were combined to generate the lattices from which the indexes were built.

(a) Word 3-gram, 50 000 words

Method	mAP (%)	gAP (%)
$P(R=1 \mid X=\boldsymbol{x},V=v)$	93.8	91.1
$\max_c P(R=1 \mid X=\boldsymbol{x},V=v,L_\tau=c)$	94.0	91.4
$\max_{c_0,c_1} P(R=1 \mid X=\boldsymbol{x},V=v,L_\sigma=(c_0,c_1))$	94.0	91.6
$\max_k P(R=1 \mid X=\boldsymbol{x},V=v,L_\kappa=k)$	94.0	91.1

(b) Character 8-gram

Method	mAP (%)	gAP (%)
$P(R=1 \mid X=\boldsymbol{x},V=v)$	95.7	93.0
$\max_c P(R=1 \mid X=\boldsymbol{x},V=v,L_\tau=c)$	95.4	92.8
$\max_{c_0,c_1} P(R=1 \mid X=\boldsymbol{x},V=v,L_\sigma=(c_0,c_1))$	95.5	92.9
$\max_k P(R=1 \mid X=\boldsymbol{x},V=v,L_\kappa=k)$	95.4	92.7

Tables 8.1a and 8.1b summarize the mAP and gAP on the IAM database (test set), for each of the lexicon-based and lexicon-free approaches, respectively. Observe that all lexicon-based approaches behave very similarly. The same trend can be observed among the lexicon-free methods, although both the mAP and gAP of the latter are higher than those of the former (in section 8.3.1.2, we will see that better results can be obtained using a larger vocabulary size, with the lexicon-based approaches).

It may surprise the reader that, in the case of the lexicon-based models, the performance of the position-independent relevance probability (i.e. $P(R=1 \mid X=\boldsymbol{x},V=v)$) is lower than that of the other lower bounds to this probability. After all, as we proved in chapter 3, ranking according to this relevance probability should give the maximum mAP and gAP values. Nevertheless, recall that these proofs require that the relevance probability distribution used to rank the results is actually the *real* distribution of the data. However, here we are not using the real distribution, but a (set of) statistical model(s) to estimate it from training data.

Anyhow, the differences between all methods are very small. Moreover, when better statistical models are used (i.e. using lexicon-free models), the performance is consistent with the theoretical results from chapter 3.

Table 8.2 contains the total indexing times needed by each approach, to build the probabilistic indexes. These total times include all the steps, from reading the lattices to writing the corresponding index to disk, for both the validation and test sets of the database.

Observe that the lexicon-based approaches are faster than the equivalent lexicon-free counterparts. This is because character lattices, are typically larger (have more arcs and states) than word lattices. In addition, remember that all the lexicon-free approaches have a worst case scenario cost which grows exponentially with the size of the lattice.

The exponential cost is especially damaging for the first two lexicon-free approaches, which actually need to expand the character lattices to create equivalent (pseudo-)word lattices. For most text lines, these

Table 8.2. Time needed by different algorithms to build the probabilistic index, on the IAM database, for both the validation and test sets (1 849 lines, 229 pages). Results using lexicon-based and lexicon-free approaches are shown in tables 8.2a and 8.2b, respectively. The total time needed to generate the respective lattices is also represented. Experiments performed on a single core of an Intel Core i7-3820 CPU at 3.60GHz

(a) Word 3-gram, 50 000 words

Method	Time (sec.)
$P(R=1 \mid X=\boldsymbol{x}, V=v)$	23.8
$\max_c P(R=1 \mid X=\boldsymbol{x}, V=v, L_\tau=c)$	1.0
$\max_{c_0,c_1} P(R=1 \mid X=\boldsymbol{x}, V=v, L_\sigma=(c_0,c_1))$	0.6
$\max_k P(R=1 \mid X=\boldsymbol{x}, V=v, L_\kappa=k)$	0.8
Lattice creation Avg. # states = 37.4, # arcs = 108.5, # paths = $2.8 \cdot 10^7$	105.8

(b) Character 8-gram

Method	Time (sec.)
$P(R=1 \mid X=\boldsymbol{x}, V=v)$	1754.9
$\max_c P(R=1 \mid X=\boldsymbol{x}, V=v, L_\tau=c)$	101.2
$\max_{c_0,c_1} P(R=1 \mid X=\boldsymbol{x}, V=v, L_\sigma=(c_0,c_1))$	5.3
$\max_k P(R=1 \mid X=\boldsymbol{x}, V=v, L_\kappa=k)$	11.4
Lattice creation Avg. # states = 506.2, # arcs = 713.5, # paths = $2.7 \cdot 10^{12}$	264.4

methods are very fast (less than 0.1 seconds/line), but for some extreme cases with an enormous amount of arcs, the exponential time (and memory) is excessive and we need to prune the expanded lattices. Another alternative would be to first extract a set of candidate words to index (using one of the faster lexicon-free approaches), and then compute the actual position-independent or column-relevance probability only for these (pseudo-)words.

Nevertheless, given that the segment and position relevance probabilities obtain virtually the same mAP and gAP than the position-independent or column relevances, in real application we can simply

use one of these. As a matter of fact, these faster approaches are also more convenient for real applications, since we can obtain a word bounding box for each indexed spot, and perform more advanced types of queries in a very fast manner, as we saw in section 7.1.

For the rest of the line-oriented KWS experiments in this chapter, unless stated otherwise, we will make use of the segment relevance probability (i.e. $\max_{c_0,c_1} P(R=1 \mid X=\boldsymbol{x},V=v,L_\sigma=(c_0,c_1))$), since it is the fastest method for both lexicon-based and lexicon-free approaches.

Taking into consideration both the time needed to generate the lattices and the time to create the index from these, we can process 229 pages in just 106.4 seconds (2.15 pages/second), using a lexicon-based approach; or in 269.7 seconds (0.85 pages/second), if a lexicon-free approach. Notice that these processing times were measured using a single core of a relatively modest CPU, while the whole indexing process can be widely parallelized, since each text line can be processed independently.

Although lexicon-free approaches are slower than lexicon-based ones, they do not suffer from the out-of-vocabulary problem, which may be a real issue in databases with very few training samples (recall that we used a large amount of external data in this experiment to create the language models). Thus, in real scenarios (see large-scale demonstrators in chapter 9), we typically use lexicon-free approaches.

8.3 Effect of the language model

In this section, we will study the impact of the language model in the Keyword Spotting performance (measured using the Mean and Global AP). These experiments are all performed on the IAM database.

In order to build a line-level index from each text line image, we build a *segment index* for each text line (using section 5.1.2 and section 5.3.2, for lexicon-based and lexicon-free models, respectively. Since multiple segments of the same word can be present in the index, we approximate the line relevance probability with the maximum of all segments, for each word, just as we did in the previous section.

Recall that the maximum segment-level relevance probability is a lower bound of the exact line-level relevance probability (see section 2.5). Nevertheless, as we saw in the previous section, this approach produces results very similar to the exact line-level relevance probability, and it is much faster to obtain.

Regarding the hyperparameters needed in the combination of the output of the neural networks and the n-gram language models, in each of the following experiments, we have tuned both the optical and the prior scales, using Bayesian optimization, just as we did before. Later, we will show that tuning both parameters is not critical to achieving very competitive results, but we have tuned both for completeness in our experiments.

8.3.1 LEXICON-BASED MODELS

First, we evaluate the performance using word language models. That is, each token in the n-gram represents a full word. The whitespace characters are modeled as part of the word in the lexicon. In particular, we modeled each word to start with a whitespace character.

It is important to highlight that we have not performed any "tokenization" of the data before training the language model. This step is usually done in HTR and many other applications, in order to reduce the effective lexicon size (e.g. words like "don't", and "aren't" are split into "do" + "n't" and "are" + "n't", respectively). This typically improves the recognition accuracy, but it introduces some problems in the case of KWS. Notice that we could split some word that was a full keyword, and all of our lexicon-based algorithms, introduced in section 5.1, assume that the arcs of the lattices are labeled with *full words*. Thus, to keep the experiments simple we decided to train a language model with the original text data.

There are two hyperparameters to adjust in this regard. One is the size of the context in the language model (n, in the n-gram), and the other is the size of the lexicon.

8.3.1.1 *N-gram order*

Here, we study the effect of the n-gram context size in the KWS performance. We kept the lexicon size fixed to 50 000 words. Below, fig. 8.1 shows the Mean and Global AP for each of the considered n-gram sizes (i.e. 1, 2, 3, 4, and 5), for both the validation and the test sets of the IAM database.

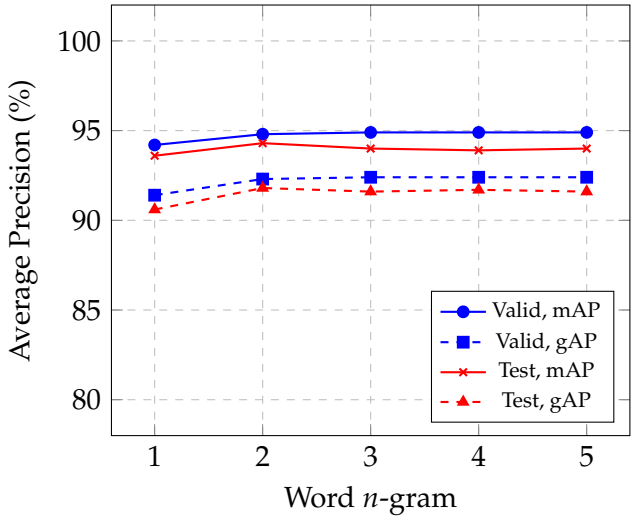


Figure 8.1. Evolution of the Mean and Global Average Precision (mAP and gAP, respectively) with respect to the order of a word n-gram language model, on the IAM database. The vocabulary of all the language models was restricted to 50 000 words. The figure shows the results for both the Validation (Valid) and Test sets.

In the validation set, all n-gram sizes achieve a competitive Average Precision, and for $n \geq 2$ the results are almost identical (i.e. mAP: 94.8%, 94.9%, 94.9%, 94.9%; and gAP: 92.3%, 92.4%, 92.4%, 92.4%). Considering these results, the optimal choice is $n = 3$.

The performance on the test set, although similar, does not yield the same conclusions. Here, the AP of all $n \geq 2$ are indeed very similar, but there is more fluctuation and the optimal choice for n is $n = 2$ (i.e. mAP: 94.3%, 94.0%, 93.9%, 94.0%; and gAP: 91.8%, 91.6%, 91.7%, 91.6%).

Although we have not conducted formal experiments to determine whether or not these differences are statistically significant, they are very likely not significant[3].

8.3.1.2 *Lexicon size*

As we mentioned above, the size of the lexicon must also be studied for lexicon-based approaches, since it has (as we are about to show) a significant impact on the quality of our system.

Here, we kept the context size of the n-gram constant ($n = 3$) and tried with different lexicon sizes (10k, 20k, . . ., and 100k). Notice that we are only able to conduct this experiment with such large lexicons because we are using the Brown, LOB and Wellington databases as additional text data to train the language models. The lexicon size of the training partition of the IAM database itself is only of 7 772 words.

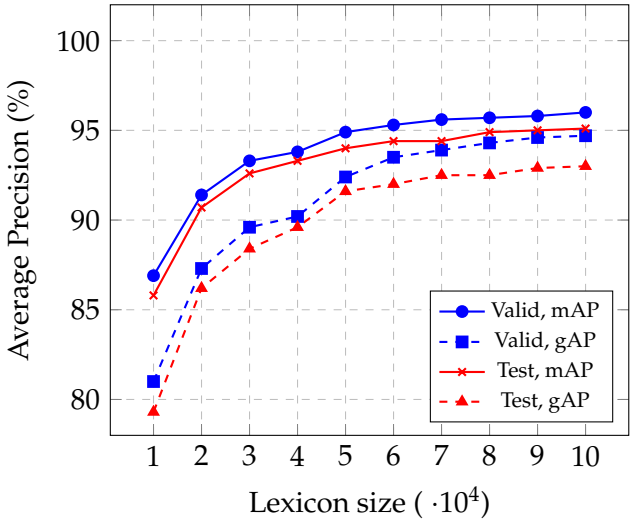


Figure 8.2. Evolution of the Mean and Global Average Precision (mAP and gAP, respectively) with respect to the lexicon size, on the IAM database. The order of all language models was fixed to $n = 3$. The figure shows the results for both the Validation (Valid) and Test sets.

[3]This hypothesis is based on the typical width of the confidence intervals from other KWS publications, which is about 1–2% in AP [Sudholt and Fink, 2018].

Figure 8.2 shows the evolution of the Mean and Global AP with respect to the lexicon size, for both the validation and test sets of the IAM database.

As expected, the quality of the KWS system improves consistently with the number of words in the lexicon. The best results, for both validation and test sets, are achieved with a vocabulary size of 100 000 words. For this lexicon size, the mAP is equal to 95.1% and the gAP is 93.0%, in the test set.

Of course, the relevance probability of any OOV query will always be zero, for any text line image. However, OOVs affect the performance even if not queried. While the lattice is being built, the image segment corresponding to the OOV will be aligned with many other (wrong) words, all with a very low likelihood. This may cause issues with the beam pruning used during decoding (if a narrow beam is used, all terminal states may be pruned; if the beam is increased, the lattice generation can be much slower), and may introduce errors in the subsequent words in the path, since the probability of n subsequent words is tied due to the context size of the n-grams.

8.3.2 LEXICON-FREE MODELS

Given the issues that lexicon-based approaches may present, we now investigate the performance of a lexicon-free approach.

In order to train the character n-gram language model, we have split the original words in the training partition of the IAM database (as well as the Brown, LOB, and Wellington) into characters. The "whitespace" symbol is included in the character-level transcript. Notice that this is the same reference text used to train the CRNN with the CTC algorithm.

8.3.2.1 N-gram order

First, we evaluate the performance with respect to the order of the n-gram (character) language model. Here, we extract only the 100-best word segments from each text line. Later, we will see that this provides a good trade-off between quality and speed.

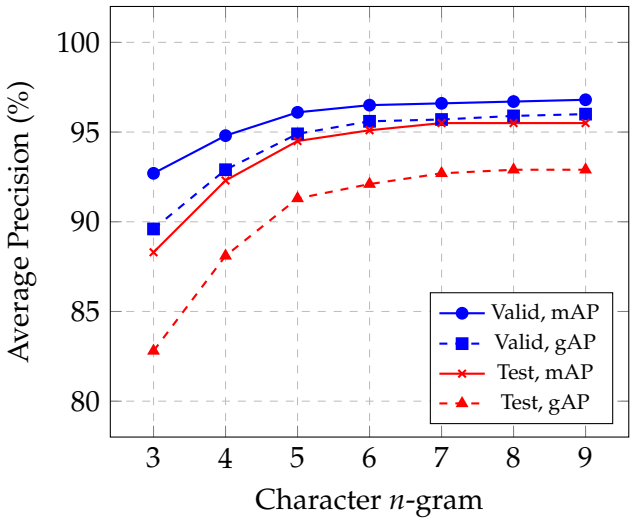


Figure 8.3. Evolution of the Mean and Global Average Precision (mAP and gAP, respectively) with respect to the order of a character n-gram language model, on the IAM database. The figure shows the results for both the Validation (Valid) and Test sets.

Figure 8.3 shows the results from this experiment, where the Mean and Global AP are plotted with respect to the order of the n-gram. As expected, the results also improve with the context size of the n-gram language model. The best validation results are achieved with the 9-gram, although they are almost identical to the 8-gram (mAP: 96.8 vs. 96.7; gAP: 96.0 vs. 95.9, respectively). However, the results on the test set are exactly the same.

Observe that in order to have a character n-gram that covers the same (or similar) context as a word m-gram, we typically need bigger context sizes. Since we include the whitespace character as part of the word in the lexicon-based system, we need a character context of, approximately, $n = m \cdot \overline{w} + (m - 1)$ to cover the same number of characters as a m-gram word model, where $\overline{w}$ is the average number of characters per word. For example, in the IAM database, since the average number of characters per word is $\overline{w} = 4.1$, we need a context size of about $2 \cdot 4.1 + 1 = 9.2$ characters to cover the same context as

a 2-gram word-based language model. This is a good rule of thumb to estimate the required character context size, from a word n-gram.

Anyhow, given that all these experiments are actually very fast to perform, one can simply sweep across multiple context sizes, just as we did here.

Observe that the results that we achieve with 8-gram characters are virtually the same as the achieved with *a very large lexicon* and a 3-gram word language model. In real applications, where no additional training data is available, we often prefer character language models. Thus, for the rest of the experiments, unless otherwise stated, we will use a lexicon-free approach.

8.3.2.2 Number of indexed spots per line

So far, we only indexed the 100-best segments from each text line, using the algorithm described in section 5.3.2. Nevertheless, observe that this is the *maximum* number of indexed spots per line. For a particular text line, the actual number may be slower, if the lattice contains fewer hypotheses (due to beam pruning). In any case, the (maximum) number of indexed spots per line can be adjusted in our algorithm. Thus, we can study how this number affects the performance of our system.

Generally speaking, depending on the entropy of the true distribution of the transcript posterior, i.e. $P(w \mid \boldsymbol{x})$, and the expected length of the transcript, the required number of indexed spots will need to be increased, in order to achieve a good Average Precision.

In the IAM database, if we assume that the reference text represents the true transcript posterior, then the entropy of the true distribution is 0 (only one transcript per text line is given). Regarding the expected length of the transcripts, among both validation and test sets, each text line contains an average of 9.1 words, with a standard deviation of 2.2 words/line. The text line with fewer words has 3 of them, and the text line with most words has 18. This gives a lower bound on the number of spots to index by our algorithm[4].

[4]In real scenarios, we can compute the expected number of words from the lattices.

Figure 8.4 depicts the evolution of the Average Precision with respect to the (maximum) number of indexed spots per text line. Observe that once we set this maximum to 30, we start achieving very good results. The performance achieved by the system with a maximum of 100 (which we used in the rest of the experiments in this thesis) is virtually the same as for larger indexes.

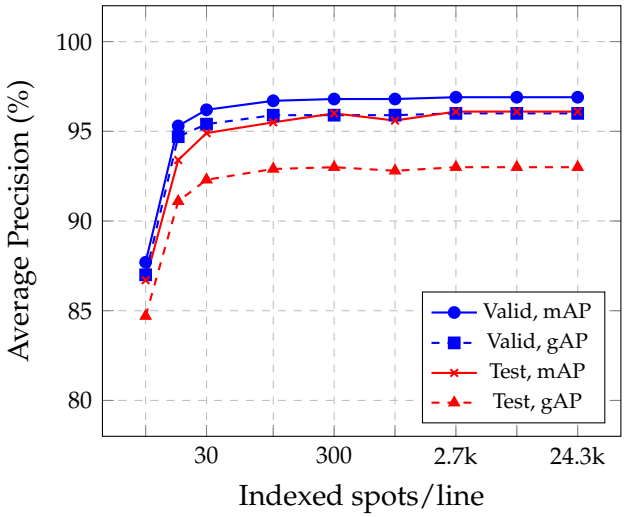


Figure 8.4. Evolution of the Mean and Global Average Precision (mAP and gAP, respectively) with respect to the number of indexed segments per line, on the IAM database. The figure shows the results for both the Validation (Valid) and Test sets.

Similarly, fig. 8.5 shows the evolution of the total size of the index (including both the validation and test lines) as well as the total time required to produce such index[5], with respect to the maximum number of indexed spots per line.

Observe that, in the worst case scenario (extracting at most 24 300 spots per text line), we need just about 15 seconds to generate the index from the lattices, for the 1 849 text lines (226 pages) that comprise both the validation and test sets. Yet, virtually the same mAP and gAP results can be obtained in about 5 seconds, by indexing (at most) 100 spots per line.

[5]Using a single core of an Intel Core i7-3820 CPU at 3.60GHz

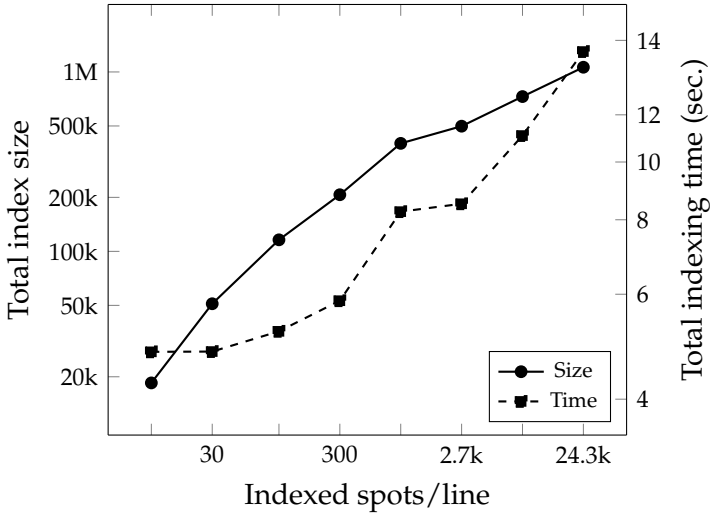


Figure 8.5. Evolution of the total index size and time with respect to the number of indexed segments per line, on the IAM database. The total numbers include both the validation and test sets of the database.

We must emphasize that naively indexing the text resulting of an automatic transcription achieves considerably worse results, even though the recognition error rates are quite low (4.4% CER, and 11.73% WER, on the test set). The mAP and gAP obtained by this naive approach are 85.8% and 80.7%, using exactly the same statistical models.

8.3.3 EFFECT OF THE OPTICAL AND PRIOR SCALES

In most works, the optical scale[6], α, and the prior scale, β, are tuned independently, typically using grid search (e.g. [Doetsch et al., 2014, Voigtlaender et al., 2016]). Also, due to limitations in the software that we (and many others) use to generate the lattices, the effective prior scale is $\alpha \cdot \beta$, and not just β.

In any case, it is not clear that the effects of both hyperparameters on the desired metric (e.g. mAP or gAP) are independent. Thus, in all

[6]The acoustic/optical scale plays a similar role to the grammar scale factor. The former scales the likelihoods of the optical model (the CRNN in our case), while the latter scales the language model probabilities.

our previous (and following) experiments, we tuned both parameters at the same time.

In particular, we first generate lattices using $\alpha = 1$ for different values of the prior scale, $\beta \in [0.0, 0.1, \ldots, 1.0]$. Because the optical cost is stored separately in the lattice arcs, we can adjust α once the lattice is generated, avoiding to re-generate lattices for each value of α that we need to evaluate.

Then, we use Bayesian optimization using a Tree-structured Parzen Estimator (TPE) [Bergstra et al., 2011] to find the pair (α^*, β^*) that maximizes the average of the mAP and the gAP[7], for a given experiment. We use a Bayesian optimization approach, instead of simple grid search, because it makes the search much faster (in our case, about 2–3 times).

Figures 8.6 and 8.7 show the evolution of the mAP and the gAP with respect to the optical and the prior scales, in the validation set of the IAM database. The color of the heat map represents the value of the mAP (resp. gAP), the x-coordinate represents the value of the optical scale ($\alpha \in [0.2, 0.4, \ldots, 4]$), and the y-coordinate represents the value of the prior scale ($\beta \in [0.0, 0.1, \ldots, 1.0]$). The sampled results were interpolated to produce a smoother plot. Figure 8.6 was produced using a 8-gram character language model, and fig. 8.7 using a a 3-gram word language model with a vocabulary size of 50 000 words.

Observe that, in both figures, the optical scale has a more significant impact on the performance, compared to the prior scale. In the case of the lexicon-free model, if one fixes the optical scale to its optimum value, then all values of the prior scale have a very similar performance. The mAP in the lexicon-based approach (see the left plot in fig. 8.7) is slightly more susceptible to the combined effect of the two hyperparameters.

This behavior is not particular of the IAM database. We conducted additional experiments in other data sets and all show the similar

[7] We could optimize only the value of the mAP or the gAP, or some other combination of the two (e.g. geometric mean). However, for simplicity we decided to use the average.

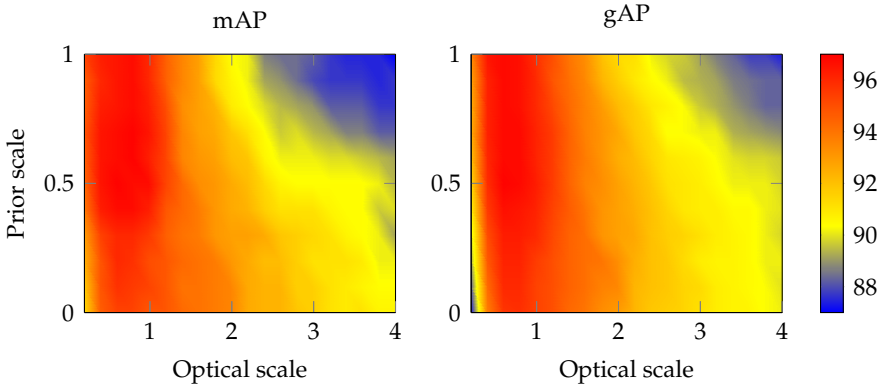


Figure 8.6. Evolution of the Mean and Global Average Precision (mAP and gAP) with respect to the optical and prior scales, in the validation set of the IAM database, using a 8-gram character language model.

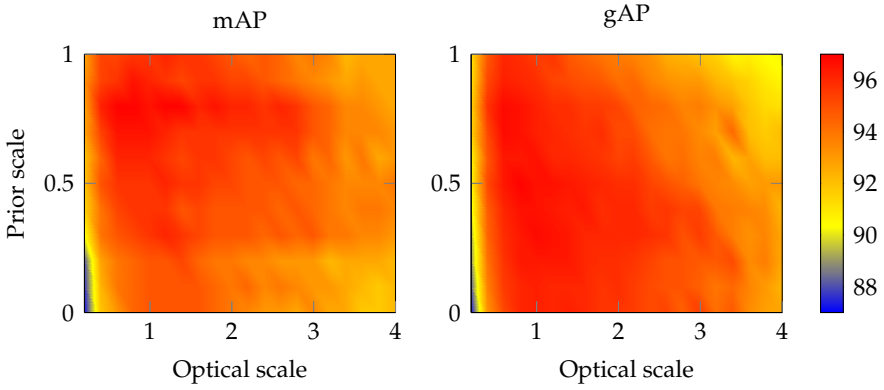


Figure 8.7. Evolution of the Mean and Global Average Precision (mAP and gAP) with respect to the optical and prior scales, in the validation set of the IAM database, using a 3-gram word language model, with a vocabulary of 50 000 words.

trends. For instance, fig. 8.8 shows the results of the same experiment in the George Washington data set.

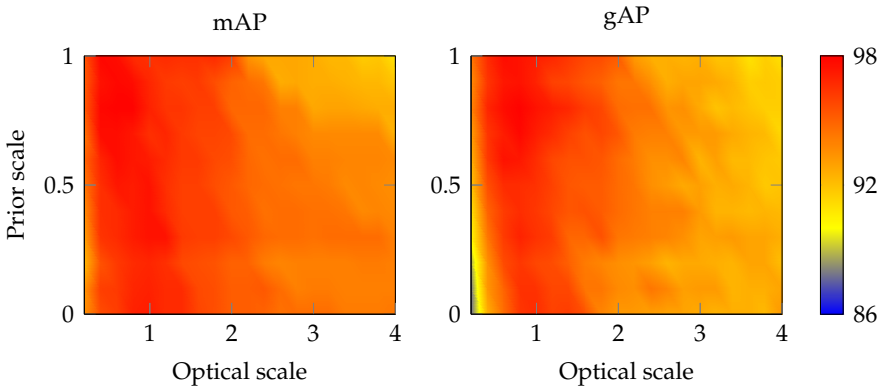


Figure 8.8. Evolution of the Mean and Global Average Precision (mAP and gAP) with respect to the optical and prior scales, in the validation set of the George Washington database, using a 6-gram character language model. We show the average across the validation sets of the four cross-validation folds.

These results suggest that the value of the prior scale is not critical to achieve very good KWS results. In fact, one could even use a null prior scale, which is equivalent to using the raw output distribution to perform the lattice generation, and still get very good results. This is good news, because estimating the prior distribution requires processing all the training data with the neural network and accumulate the posteriors across all frames, which can take a few minutes.

In any case, optimizing the prior scale together with the optical scale using a Bayesian optimization approach is very fast as well (in our case, it took 5 minutes for the IAM database and 8 minutes for the George Washington database).

8.4 Effect of the training data size and augmentation

Here, we study how the number of training samples (i.e. text lines) available for training influences the results of our KWS approach. Recall that in order to train our probabilistic models, we need segmented text lines with its transcription. Thus, if our solution is able to per-

form well enough even with less training data, then we can potentially save some of the costs of ground-truth production.

The partition of the IAM database that we use consists of 6 161 training, 920 validation, and 929 test lines (747, 116, and 110 pages, respectively). However, the total number of transcribed lines in the database amounts to 13 353 (1 539 pages). Thus, we can exclude or add several text lines to the original training set and study the evolution of the Average Precision (both Mean and Global) with respect to the number of available training lines.

For the experiments in this section, we have used the same character 8-gram language model used before. Recall, that this model was trained using three external text-only databases. We have not studied the influence of the external text data because this type of data is considered "free", in comparison to the training data needed to train the neural networks or GMM-HMMs, since human supervision is barely needed to gather it.

In addition, we also study the effect of using artificial data augmentation. This has been widely used for many applications in the field of Pattern Recognition. In the handwritten text domain, we showed in [Puigcerver, 2017] that simple (but adequate) random affine distortions, applied to the training images, can significantly boost the performance of a HTR system.

During training, in order to generate a random affine matrix for a given image, we translate the upper-left, upper-right, and bottom-left coordinates of the image in a random direction, such that the maximum translation is 30% of the height of the image. Then, we compute the matrix of the affine transformation that maps the three original coordinates to the new ones. This allows us to efficiently generate arbitrary affine transformations which do not alter significantly the content of the image. Our training software performs this operation on-the-fly, for each training image independently.

Figure 8.9 shows the evolution of the Mean and Global Average Precision, for both validation and test sets, with respect to the number of training lines used to train the CRNN. The plot on the left of the figure shows the evolution when no artificial data augmentation is

used, while the plot on the right shows this evolution when random affine distortions are used on the training samples.

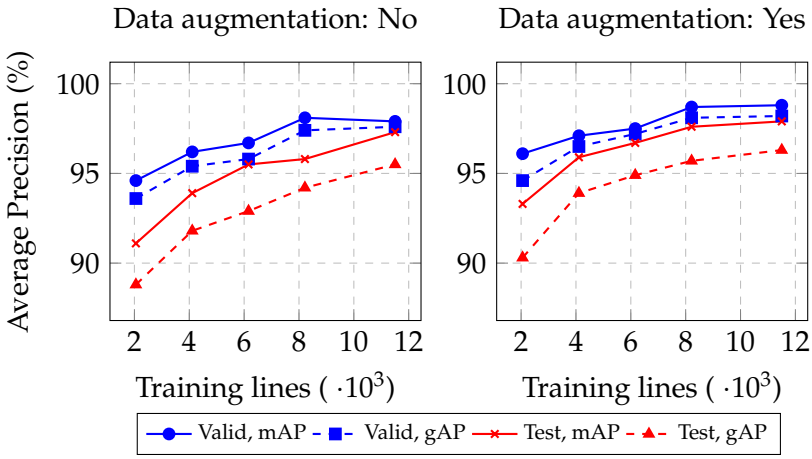


Figure 8.9. Evolution of the Mean and Global Average Precision (mAP and gAP, respectively) with respect to the number of lines used to train the CRNN, on the IAM database. The left plot shows the evolution when no artificial data augmentation is used, while random affine distortions were used in the right plot. The results for both the Validation (Valid) and Test sets are shown.

Observe that all measures typically improve with the number of training lines used. The only exception is the validation mAP when using about 11 500 images with no data augmentation, which is slightly inferior than that of using only about 8 200 (97.9% and 98.1%, respectively). However, the differences are likely not statistically significant. Even using a very small subset of training lines (about 2000), our approach provides a decent Mean and Global AP (91.1% and 88.8%, respectively), which are higher than any other previously reported results for line-level KWS on the IAM database (see section 8.6.3).

In addition, using the random distortions that we just proposed to artificially augment the training data also improves the results in all cases. For instance, using the original training set (6 161 lines), the results on the test set improve from 95.5% mAP and 92.9% gAP, to 96.7% mAP 94.9% gAP.

8.5 Correlation between Average Precision and Recognition Error

Given that our probabilistic approach uses the same probability distribution typically used in Handwritten Text Recognition tasks, i.e. $P(W \mid X)$, one might wonder what is the correlation of the performance in a KWS and a HTR task, when the same probabilistic model is used. In particular, we study the correlation between the mAP and gAP, and the Character and Word Error Rates (CER and WER, typically used in HTR tasks).

In order to study this correlation, we use the same lexicon-free model used in previous sections, considering different orders of the n-gram language model, and different values of the optical and prior scales. For each combination of hyperparameters, we compute the CER, WER, mAP and gAP (on the test set of the IAM database). In fig. 8.10, the x-coordinate in each plot represents the value of one of the HTR measures (i.e. CER or WER), and the y-coordinate the corresponding value achieved in the KWS task (i.e. mAP or gAP).

Certainly, extreme cases can be artificially constructed such that the error rates are arbitrarily large, and the Average Precision still be close to 100% (or vice versa). For instance, suppose that our recognition system recognizes perfectly all the query keywords (which are very infrequent, compared to the total amount of words), but not any other word. Then, the recognition error rates will be arbitrarily high, but both mAP and gAP will be equal to 100%.

Nevertheless, as fig. 8.10 shows, a fairly linear relationship can be established between the mAP/gAP and the CER/WER. Observe that the correlation between the WER and both APs is larger that that of the CER. This is obvious, since the relevance definition is done at word-level (i.e. a pair of words is relevant if both words are exactly the same), and the number of character errors in a given word, if greater than zero, does not matter.

This linear relationship can be very useful to estimate the results expected in a KWS task, for a given model, using results from an HTR experiment (which is typically easier to conduct). First, one just needs to estimate the slope and bias of the linear function, and then simply

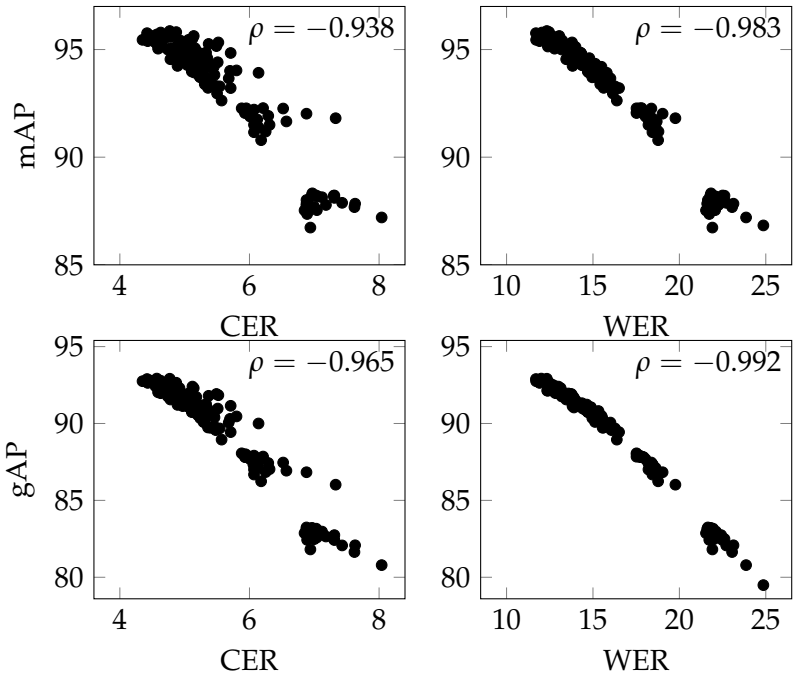


Figure 8.10. Correlation between Average Precision measures and Recognition Error Rates, for the test set of the IAM database. All points in the plots use the same CRNN, and a lexicon-free language model. However, different values for the order of the n-gram, and the optical and prior scales were evaluated to generate the plots.

evaluate the probabilistic models on the HTR tasks, and predict its KWS performance.

8.6 Results on other academic databases

In this section, we evaluate one of our lexicon-free approaches in other line-level academic data sets: the George Washington and the Parzival databases.

In each database, we tuned the order of the character n-gram language model, as well as the optical scale and prior scale, as described above. For each text line, we extracted only the 100-best pseudo-word segments using the lexicon-free approach described in section 5.3.2.

Finally, the results in each corpus are then compared with other scientific publications.

8.6.1 GEORGE WASHINGTON

We trained a CRNN neural network similar to the one used in the IAM experiments. The details are depicted in table 8.3. The CTC loss was minimized using the RMSProp algorithm, a learning rate equal to $3 \cdot 10^{-4}$ and a batch size of 16 images, for about 230k updates of the parameters[8] In this particular database, we observed that using batch normalization [Ioffe and Szegedy, 2015] in the convolutional layers improved the recognition results, so we also used this technique in the KWS experiments. In addition, since the data set is so small, we also performed random affine transformations of the images during training.

Table 8.3. Architecture of the CRNN used in the George Washington experiments.

Configuration	Values
Convolutional block	
Num. layers	4
Activation	LeakyReLU
Conv. filters	{16, 32, 64, 64}
Conv. size	{3, 3, 3, 3}
Max. pooling	{2, 2, 2, 0}
Batch normalization	yes
Recurrent block	
Num. layers	4
Type	BLSTM
Units	{128, 128, 128, 128}
Dropout	{0.5, 0.5, 0.5, 0.5}
Output layer	
Units	72
Dropout	0.5

[8]This is about 700 epochs through the training data of each cross-validation fold, which took about 2h12m using a NVIDIA Titan X.

Figure 8.11 shows the evolution of the Average Precision (both Mean and Global) with respect to the order of the character n-gram. The delimiters used to generate the word indexes were the parentheses (i.e. "(" and ")") and the whitespace symbol. Since this database uses four-fold cross-validation, we tuned a unique optical and prior scale by averaging the results across the four partitions.

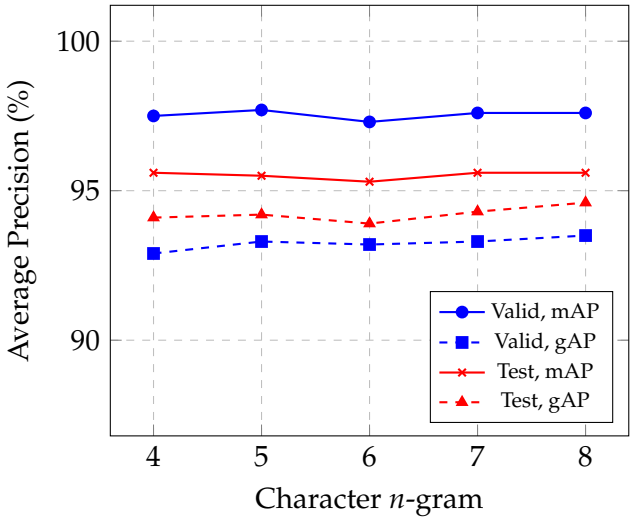


Figure 8.11. Evolution of the Mean and Global Average Precision (mAP and gAP) with respect to the order of a character n-gram language model, on the George Washington database. The figure shows the average across the four-folds for both the Validation (Valid) and Test sets.

The results are very similar across all values of n. In order to chose a single value for n, we picked the one that maximized the average of the mAP and gAP in the validation set. There, the 8-gram model achieves a mAP of 97.6% and a gAP of 93.5%, while in the test set it achieves 95.6% (mAP) and 94.6% (gAP). Note that these numbers are the averages across the four cross-validation folds. These results were achieved with an optical and prior scales equal to 0.85 and 0.80, respectively.

8.6.2 PARZIVAL

We also performed experiments in the Parzival data set. The details of the architecture of the CRNN are represented in table 8.4. The CTC loss was minimized using the RMSProp algorithm, a learning rate equal to $5 \cdot 10^{-4}$ and a batch size of 16 images, for about 28k parameter updates[9] Contrary to the previous section, we did not use batch normalization, nor training data augmentation for this experiment.

Table 8.4. Architecture of the CRNN used in the Parzival experiments.

Configuration	Values
Convolutional block	
Num. layers	4
Activation	ReLU
Conv. filters	{16, 16, 32, 32}
Conv. size	{3, 3, 3, 3}
Max. pooling	{2, 2, 2, 0}
Batch normalization	no
Recurrent block	
Num. layers	3
Type	BLSTM
Units	{256, 256, 256}
Dropout	{0.5, 0.5, 0.5}
Output layer	
Units	97
Dropout	0.5

Once more, we tried with different context sizes of the n-gram character language model, adjusting for each of them both the optical and the prior scale. The delimiters used to generate the word indexes were the dot character, the hyphen (resp. "pt" and "eq" in the reference text), and the whitespace symbol. Figure 8.12 shows the results of this experiment, for different values of n.

[9]This is about 200 epochs through, which took about 2h08m using a NVIDIA Titan X.

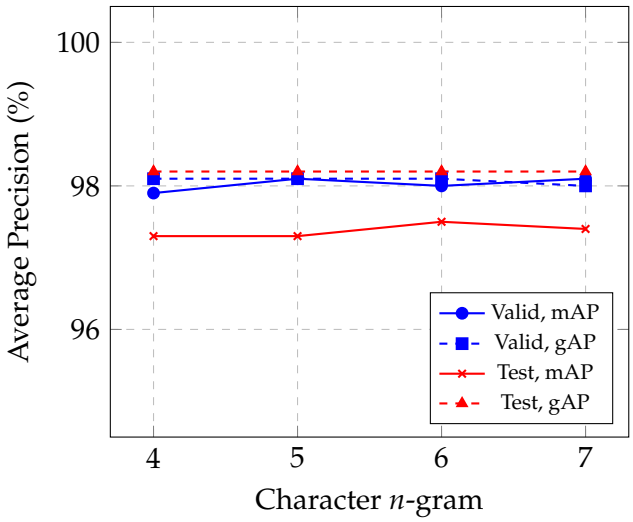


Figure 8.12. Evolution of the Mean and Global Average Precision (mAP and gAP) with respect to the order of a character n-gram language model, on the Parzival database. The figure shows the results for both the Validation (Valid) and Test sets.

The best results are achieved with the 6-gram. In the validation set, this language model obtains 98.1% for both mAP and gAP (with an optical and prior scales equal to 0.55 and 0.20, respectively). In the test set, it achieves 97.3% and 98.2% points of mAP and gAP, respectively. Nevertheless, the differences among all orders are small and very likely not statistically significant.

8.6.3 Comparison with other published works

To put our results in comparison with previously published work, table 8.5 presents results in the query-by-string, line-level Keyword Spotting scenario, obtained by other authors on the IAM, George Washington and Parzival databases.

The following approaches have been considered: The method presented in [Terasawa and Tanaka, 2009] uses histogram of gradients (HOG) and dynamic time warping (DTW); [Fischer et al., 2012, Fischer et al., 2013] use approaches based on the classical HMM-Filler, including the use of character language models to improve its results;

in [Kumar and Govindaraju, 2014] they use a Bayesian logistic regression classifier; [Wshah et al., 2014] present a method based on the HMM-Filler but with additional background modeling; [Wicht et al., 2016b,Wicht et al., 2016a] use deep belief neural networks in combination of DTW and HMMs, respectively; [Toselli et al., 2016b] essentially use the column-wise relevance probability (see section 2.2.1), making use of GMMs, HMMs and a word language model to build the necessary statistical models. Finally, [Frinken et al., 2012] use the BLSTM-CTC approach briefly discussed in section 6.2.

In the case of the IAM database, we report in this table the results achieved by the segment lexicon-free approach, using 8-gram character language model, with artificial data augmentation (i.e. using random affine distortions on the original training images). For the rest of the databases, the reported results are the ones described earlier in this section. Observe that the methods developed in this thesis significantly improve previous state-of-the-art results, for all the considered databases.

8.7 Using traditional GMM-HMM models

Although in the previous experiments we have only used models based on CRNNs (combined with n-gram language models), other types of statistical models for the text line transcripts, whose output can be represented as a WFST can also be used.

For instance, as we explained in section 4.2, the traditional Hidden Markov Models (with Gaussian Mixture densities) can also be used to build the probabilistic indexes. In fact, at the early stages of the development of this thesis, we mainly used HMMs to perform our experiments.

In this section, we will use HMMs to tackle line-oriented KWS tasks, using the approach described in this thesis to create probabilistic word indexes. As we will see, these models can also achieve very high performance for KWS tasks, when used under our formulation, although they have been almost entirely replaced by modern CRNNs.

HMM training was carried out with the embedded Baum–Welch algorithm, using the HTK toolkit [Young et al., 2002]. A left-to-right

Table 8.5. Average Precision (%) results achieved by different query-by-string, line-level KWS approaches on the IAM, George Washington and Parzival databases. Publications that did not report the first decimal point are marked with "?".

	Measure	IAM	GW	PAR
HOG-DTW [Terasawa and Tanaka, 2009]	mAP	—	79.1†	—
Classic Filler-HMM [Fischer et al., 2012]	mAP	68.9†	79.3	88.2
	gAP	47.8†	62.1	85.5
BLSTM-CTC [Frinken et al., 2012]	gAP	78.?	85.?	94.?
2-gram Filler-HMM [Fischer et al., 2013]	gAP	55.1†	73.9	—
BLRC [Kumar and Govindaraju, 2014]	mAP	49.0†	—	—
Filler-BGR [Wshah et al., 2014]	mAP	57.7†	—	—
CDBN-DTW [Wicht et al., 2016b]	mAP	—	67.4	62.4
	gAP	—	55.7	58.8
CDBN-HMM [Wicht et al., 2016a]	mAP	72.4†	85.1	94.6
	gAP	64.7†	71.2	92.3
HMM + 2-gram LM [Toselli et al., 2016b]	gAP	72.?	77.?	90.?
This thesis	mAP	96.7‡	95.6	97.3
	gAP	94.9‡	94.6	98.2

† Smaller query set and/or number of evaluation lines.

‡ Artificial data augmentation.

HMM was used for each character. The number of states and Gaussian densities per state were roughly set up taking into account the average number of frames aligned to each character in the alphabet, and other data set features, and finally tuned using validation data. Since the HTK toolkit does not offer good support for generating lattices with large language models, in this section we will only used 2-gram word language models. More details about the HMMs setting and the language model are given in [Toselli and Vidal, 2015].

We use two large data sets, from the Bentham and Plantas collections. The Bentham partition that we use is the one used in the *ICFHR2014 Competition on Handwritten Text Recognition on Transcriptorium Datasets* [Sánchez et al., 2014], while the Plantas partition was introduced in [Toselli et al., 2018a]. More details about these databases can be found in appendices A.1 and A.5, respectively.

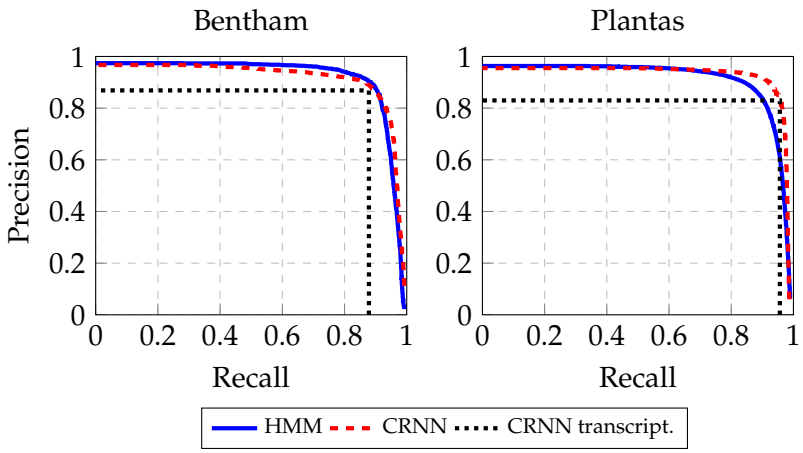


Figure 8.13. Recall–Precision curves of HMMs, CRNNs, and fully automatic transcription, on the Bentham and Plantas databases. In the Bentham database (left), the gAP of HMMs is 90.7%, while that of the CRNN is 91.4%. In the Plantas database (right), the gAP of the HMMs is 90.9%, and that of the CRNN is 92.9%. The plots also show the performance of a (non-probabilistic) index built after automatically transcribing all text lines with the CRNN, which achieves a gAP of 76.3% and 79.4%, in each database, respectively.

In fig. 8.13, the recall–precision curves of the test set of both databases are plotted (the Global Average Precision is the area below this curve).

According to these results, using CRNNs in the place of HMMs only gives modest improvements with respect to the gAP. Observe that in both cases, the gAP is very high. Thus, little gains can be expected from using a better statistical model.

Nevertheless, as we will see in the next section, CRNNs offer a more robust performance when automatically segmented text lines are used to create a probabilistic index for a segmentation-free KWS scenario, which is closer setting to reality.

8.8 Segmentation-free evaluation

So far, we have performed experiments in documents with manually segmented lines, and evaluated the spotting results at line level (without the need of predicting accurate bounding boxes for the spots). However, as we suggested in section 2.4, the position-dependent probability relevances described in this thesis can be directly used in segmentation-free evaluations, by using automatic text line segmentation techniques.

In this section we will show the results of our approach evaluated under a segmentation-free scenario. In particular, we have conducted experiments using the same databases as two international competitions: the *ICFHR2014 Handwritten Keyword Spotting Competition*[10], and the *ICDAR2015 Competition on Keyword Spotting for Handwritten Documents*[11].

8.8.1 ICFHR2014 Handwritten Keyword Spotting Competition

Here, we use the Bentham collection employed in this competition to evaluate the performance of a lexicon-based system, using GMMs, HMMs, and a 2-gram word language model as the statistical models; as well as a lexicon-free system, using a CRNN with a similar architecture to all the other networks used before, and a 7-gram character language model. The results of the HMM-based system were originally published in [Vidal et al., 2015].

[10] `http://vc.ee.duth.gr/H-KWS2014/`

[11] `http://transcriptorium.eu/~icdar15kws/`

Because in the original competition they did not provide any training data, we used the training (350 pages) and validation (50 pages) from the *ICFHR2014 Competition on Handwritten Text Recognition on tranScriptorium Datasets* [Sánchez et al., 2014], from which we excluded the 50 pages used in the evaluation of the KWS competition.

In the case of the HMM-based system, an additional corpus of 10M running words (about 78 000 distinct words) was used to estimate the 2-gram language model. However, in the case of the CRNN-based system, no additional training data, nor data augmentation, was performed to estimate any of the statistical models.

We use the evaluation software provided by the organizers of the competition. Contrary to most of the experiments in this thesis, they do not use interpolated precision or the trapezoid integration method to compute the Average Precision. Also, their software did not report the gAP, only the mAP.

Since we are in a segmentation-free scenario, the systems must provide a bounding box with the localization of each spotted keyword in each page. Thus, an additional measure is needed in order to decide whether the given bounding box is sufficiently correct. The organizers of the competition used the overlapping area between the reference bounding boxes and the detected ones, defined as $\frac{A \cap B}{A}$, where A and B are the reference and the detected bounding boxes, respectively. All detected keywords with an overlapping area greater than 0.7 are considered correct[12].

The details about the training of the GMM-HMM, and the automatic text line segmentation can be read in [Vidal et al., 2015]. The architecture of the CRNN that we used in this thesis is described in table 8.6. We trained the neural network using RMSProp, with a learning rate equal to $5 \cdot 10^{-4}$, with a batch size of 16, for about 28k parameter updates.

[12]We are aware that this overlapping measure can be easily fooled by producing spots with sizes equal to the whole page. However, for the sake of fair comparison, we did not take advantage of this shortcoming of the evaluation protocol.

Table 8.6. Architecture of the CRNN used in the ICFHR2014 Handwritten Keyword Spotting Competition.

Configuration	Values
Convolutional block	
Num. layers	4
Activation	ReLU
Conv. filters	{12, 24, 48, 48}
Conv. size	{7, 5, 3, 3}
Max. pooling	{2, 2, 2, 0}
Batch normalization	no
Recurrent block	
Num. layers	3
Type	BLSTM
Units	{256, 256, 256}
Dropout	{0.5, 0.5, 0.5}
Output layer	
Units	62
Dropout	0.5

In the case of the lexicon-free (CRNN-based) approach, we used the algorithm described in section 5.3.2 to build a (pseudo-)word index, storing the relevance probability of each segment.

Because the competition was evaluated under the query-by-example paradigm, we modeled the transcript posterior of the query image using the n-best transcription hypotheses, either using the HMM-GMM model or the CRNNs. In both cases, for the query image, instead of using a line-level language model, a character n-gram language model for individual words was used.

With the probabilistic word index, and the query transcript posterior, we follow the approach described in section 2.3 to obtain the relevant bounding boxes for each query image.

Table 8.7 shows the results of the HMM and CRNN-based systems, using both manual and automatic line segmentation of the evaluation pages.

Table 8.7. Results for manual and automatic line segmentation in the ICFHR2014 KWS competition. Results are reported using the Mean Average Precision (%) as computed by the official evaluation software.

	Manual	Automatic
HMM-GMM + word 2-gram	86.5	71.5
CRNN + char 7-gram	88.9	87.3

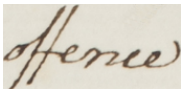
Observe that the results achieved by the HMM and the CRNN models are similar, but the later one is much more robust when automatic line segmentation is used during evaluation (both use manually segmented lines during training). In addition, recall that the CRNN-based system used fewer training data to estimate the character language model.

Regardless of the particular statistical approach used to model the required probability distributions, our probabilistic approach is much superior than the other solutions submitted to the competition, as table 8.8 shows.

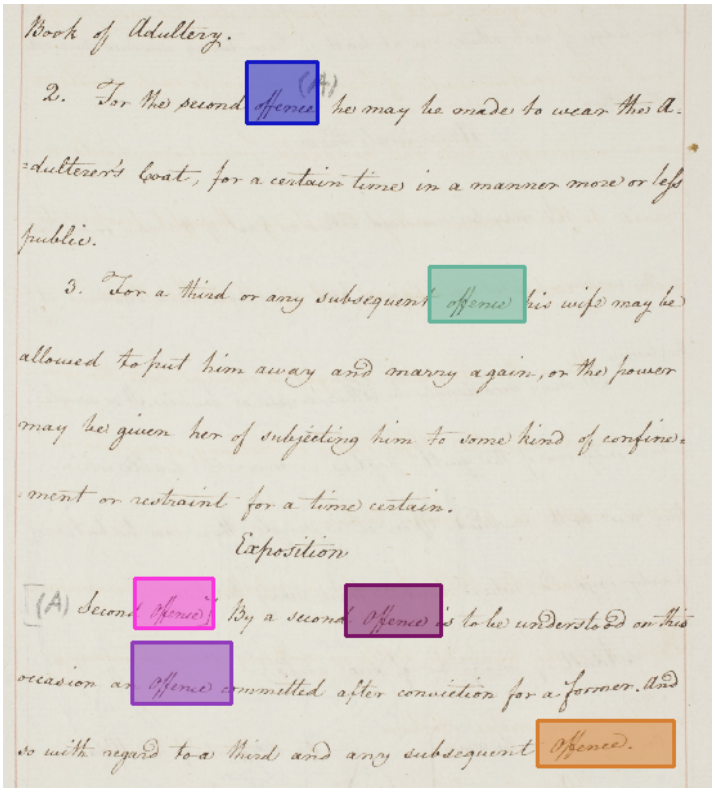
In this table, the four top rows represent the official scoreboard for the segmentation-free task of the competition, and the two rows at the bottom represent the solutions based in our probabilistic perspective. In addition to the mAP, other performance metrics are also shown: the precision at 5 (P@5), and two normalized discounted cumulative gain measures, one assuming a binary relevance judgment (NDCGbin) and the other assuming non-binary judgment (NDCG). Further details about these evaluation measures are explained in the competition report [Pratikakis et al., 2014].

Finally, fig. 8.14 shows some qualitative results of the CRNN-based approach, for a given query image (with the word "offence" written in it) and a test page of the competition (with several instances of this keyword). It is worth mentioning that we did not make use of any image processing technique to improve the bounding boxes of the segments indexed by our system.

Finally, we must highlight that this particular competition violates the definition of binary relevance that is used in this thesis. In partic-



(a)



(b)

Figure 8.14. Figure (a) shows a query of the ICFHR2014 H-KWS Competition, and (b) a fragment of a page where multiple instances of this query were spotted by the CRNN system using automatic text line segmentation.

Table 8.8. Comparison of multiple systems submitted to the original competition ("Team 1", "Team 3", ...), and two systems employing the probabilistic KWS approach described in this thesis and automatic text line segmentation: one using GMM-HMMs and a word language model ("GMM-HMM + word 2-gram"); and another one using CRNNs and a character language model ("CRNN + char 7-gram").

	P@5	NDCG(bin)	NDCG	mAP
Team 1	61.1	64.0	65.7	41.9
Team 3	56.8	51.8	53.6	37.2
Team 4	34.1	36.3	37.6	20.9
Team 5	55.0	51.3	53.1	34.7
GMM-HMM + word 2-gram	87.9	82.2	82.3	71.5
CRNN + char 7-gram	96.0	92.2	91.9	87.3

ular, they consider as relevant some word regions of the evaluation pages that contain different words than that of the given query image. For example, for a query image with the text "possess" written in it, some text regions in the test pages are considered relevant, while the text actually written in these are words like "possesst", "possessed" or "possession".

Although one of the assumptions in our approach is clearly violated, our probabilistic perspective still outperforms all the other (word-segmented distance-based) heuristic solutions, by a significant margin, even using different statistical models, like traditional GMM-HMM and more recent neural network-based solutions.

8.8.2 ICDAR2015 COMPETITION ON KEYWORD SPOTTING FOR HANDWRITTEN DOCUMENTS

The *ICDAR2015 Competition on Keyword Spotting for Handwritten Documents* [Puigcerver et al., 2015b] was organized in order to provide a benchmark to fairly compare different KWS approaches. Two tracks were created: a training-free (for systems that do not need labeled training data), and a training-based (for other systems that do need labeled data, like the presented in this thesis). In the training-based track, two sub-tracks were created: one following the query-by-string paradigm and the other following the query-by-example.

Since the data used in this competition is based on the Bentham collection, we used the same CRNN architecture and the same training procedure for the neural network and the character language model.

The submissions to the training-based track were only evaluated under a segmentation-free paradigm. Thus, we first automatically segmented the evaluation pages into lines, using the Transkribus tool, which can be downloaded and used freely[13]. Then, a lattice was generated for each text line, from which a segment-level probabilistic index was created, as described in section 5.3.2 (and as we did for the previous competition). Using the text line coordinates obtained from the automatic segmentation, and the indexed segments, we can obtain word-level bounding boxes relative to the whole page.

These bounding boxes need to be refined in a last step, since the evaluation measure used in the competition is very sensitive to the overlap between the detected and the reference bounding boxes. Contrary to the previous competition, the intersection over the union measure was used in this case. This measure is defined as $\frac{A \cap B}{A \cup B}$, where A and B are the areas of the reference and spotted bounding boxes, respectively. Only correct matches with an overlapping ratio greater or equal than 0.7 were considered correct. We used the evaluation software provided by the competition to measure the Global and Mean Average Performance.

Table 8.9. Comparison of multiple systems and measures in the ICDAR2015 Competition on KWS.

	Query-by-String	
	mAP (%)	gAP (%)
Team 1	87.1	85.3
Team 2	38.2	18.2
Ours	91.5	86.4

[13] `https://transkribus.eu/`

Table 8.9 shows the Mean and Global[14] Average Precision of our approach, compared to the best results of the two teams that participated in the training-based track. Team 1 uses 2D-LSTM and a fairly similar approach to the BLSTM-CTC, described in section 6.2. Team 2 use a combination of multiple BLSTM networks, with handcrafted features, and a KWS approach similar to the HMM-Filler, described in section 6.1.

The lower performance achieved by Team 2 is very likely due to problems recovering accurate word bounding boxes, and due to the fact that they used handcrafted features to feed their neural networks. Our performance is very similar to that of Team 1 (the winner), but slightly superior. Indeed, according to their description, they follow an approach very similar to the BLSTM-CTC, but multiple segments are retrieved per text line (up to 4). As we explained in section 6.2, the BLSTM-CTC approach can also be interpreted under our probabilistic umbrella, thus it is reasonable that both approaches perform similarly.

In a real application, our approach has the benefit that we do not need to know the potential queries in advance, to build the word index, and thus, we can respond to queries instantly.

In summary, we have seen that our probabilistic indexes can be easily used in segmentation-free scenarios, by automatically segmenting the pages. This is of vital importance for real-life applications, since accurately segmented text lines or words are never available. Later, in chapter 9 we will follow the same procedure as in this section, to apply our method to large-scale collections (with several tens of thousands of pages).

8.9 Probabilistic interpretation of the HMM-Filler

In this section, we describe the experiments that we carried out to highlight the probabilistic interpretation of the HMM-Filler, which we described in section 6.1, as well as the improvements achieved by using better approximations to the relevance probability.

[14]The gAP was not originally reported in the competition, but report it here, since we had access to the submissions.

The experiments are performed on the IAM database at line level. That is, we assume that the text lines have been segmented and we aim to localize the relevant text lines for a set of query keywords. We use the IAM partition used in the original publication describing the HMM-Filler approach [Fischer et al., 2012].

8.9.1 DESCRIPTION

For each character in the training lexicon, a left-to-right HMM-GMM was used in order to model the likelihood of the images. The number of states and the number of mixtures was tuned using the annotated validation partition (in order to minimize the CER). The HTK software [Young et al., 2002] was used to train the HMM-GMM models using the Baum–Welch algorithm. In order to perform a fair comparison, we used exactly the same image processing, feature extraction and even the same models used in [Fischer et al., 2012], which were kindly provided by the authors of the paper.

Once the models were trained, character lattices were obtained for each of the text lines using HTK or iATROS [Luján-Mares et al., 2008]. In order to generate the lattices, character n-gram language models of different orders were trained on the well-known Lancaster-Oslo/Bergen text corpus (LOB) [Johansson et al., 1978]. Since the IAM corpus was build from the LOB, we excluded from the original LOB corpora all sentences used in the test set of the IAM database.

In order to limit the size of the generated CLs, a maximum node input degree of 30 edges and beam search were used during decoding. Additionally, the grammar scale factor and character insertion penalty parameters were adjusted using the validation data to optimize the CER. We followed the approach described in [Toselli and Vidal, 2013] in order to compute the HMM-Filler scores from character lattices, since this greatly reduces the computational burden. We used the same lattices to compute the relevance probability, as described by eq. (6.4).

We use the length of the keyword (number of character) as the heuristic to perform length normalization of the scores. That is, the

retrieved lines were ranked in increasing order of:

$$\log P'(R = 1 \mid X = \boldsymbol{x}, V = v) \approx \frac{\log P_m(R = 1 \mid X = \boldsymbol{x}, V = v)}{|v|^\gamma} \quad (8.3)$$

where $|v|$ is the length of the keyword and $\log P_m(R = 1 \mid X = \boldsymbol{x}, V = v)$ is the relevance log-probability computed using the HMM-Filler approximation or the exact probability[15].

8.9.2 RESULTS

Table 8.10 shows the results of the comparison. We denote the HMM-Filler results as "Viterbi" (since they can be interpreted as a Viterbi approximation of the exact probability), and the exact probability as "Forward" (since it is computed using a Forward-like algorithm on the lattices). The optimal value of γ was tuned on the validation data in order to maximize the Global Average Precision.

Table 8.10. Line-level Global Average Precision (gAP) results in IAM, using HMM-GMM, and different character n-grams and approximations to $P(R = 1 \mid \boldsymbol{x}, v)$. "Viterbi" refers to the HMM-Filler approximation, and "Forward" refers to the exact computation of the probability using lattices. The length normalization hyperparameter (γ) was adjusted on the validation set, its optimal value (γ^*) is also shown in each case.

	n-gram	0	1	2	3	4	5	6
Viterbi	$\text{gAP}_{\gamma=1}$	34.5	37.5	41.2	45.4	48.8	49.9	50.5
	$\text{gAP}_{\gamma=\gamma^*}$	40.2	42.9	45.2	47.7	49.4	50.0	50.5
	γ^*	1.9	2.0	1.9	1.8	1.6	1.3	0.8
Forward	$\text{gAP}_{\gamma=1}$	34.6	38.1	41.8	47.1	52.8	54.4	55.8
	$\text{gAP}_{\gamma=\gamma^*}$	40.3	43.4	45.7	49.4	53.3	54.5	55.8
	γ^*	2.0	1.9	1.9	1.8	1.6	1.3	0.8

First, we can observe that using better character language models (higher order n-grams) greatly improves the Average Precision of

[15] Actually, the relevance probabilities are *exact* modulo the pruning used to obtain the lattices. However, since we were using large lattices and a large decoding beam, we can assume that these probabilities are very close to the exact ones.

the two methods. This is consistent with the probabilistic reasoning followed through this thesis: the better the probabilistic models of the handwritten text, the better Keyword Spotting performance one should expect.

Secondly, observe that as the order of the n-gram language model augments, the optimal value of γ decreases. This suggests that, as we hypothesized in section 6.1, the length normalization heuristic is only necessary when poor probabilistic models are used.

Thirdly, the exact computation of the relevance probabilities proves to be better than the Viterbi (i.e. HMM-Filler) in all cases, as expected. In the case of low-order n-grams, the two methods perform quite similar, but when the order of the LM increases, the exact computation clearly shows its superiority. This is due to the fact that, when using poor statistical models, there is not much difference between computing the *exact* probability or a not-so-good approximation.

We must highlight the fact that the absolute and relative improvements, with respect to the HMM-Filler increase with the size of the n-gram. This suggests that not only using good probabilistic models is important, but also using the the *right* quantities to rank the text lines.

Finally, fig. 8.15 shows the Recall–Precision curves of the Viterbi approximation and the exact computation of $P(R=1 \mid X=\boldsymbol{x}, V=v)$, using a 6-gram LM with $\gamma = 1$. The figure gives additional insights about the behavior of two algorithms. The flat regions in both curves tell that there are many events with the same (high) score. The forward algorithm seems to discriminate better between the true relevant events and the false positive cases, which explains the increase in the maximum precision with only a slight drop in the minimum recall. Since the flat region is present in both cases, we hypothesize that it is due to the underlying statistical models, and not the algorithms. Finally, once the confidence of the underlying probabilistic model decays, both algorithms behave very similar as shown by the curve.

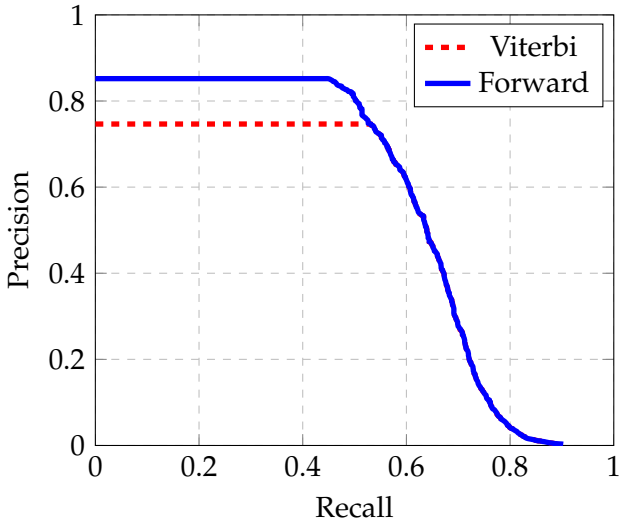


Figure 8.15. Recall–Precision curves of the Viterbi approximation and the exact computation (Forward) of $P(R=1 \mid X=\boldsymbol{x}, V=v)$. The gAP is the area below these curves.

8.10 Probabilistic interpretation of traditional distance-based systems

8.10.1 Description

In this section, we present some results using the probabilistic interpretation of traditional distance-based methods that was presented in section 6.3. The aim of this section is to highlight the issues that these methods could generally present and were described in the aforementioned section, namely: the "multi-variance" and "multi-mode" problem.

There are countless distance-based works in the KWS literature. Unfortunately, most of them do not share the source code of their methods. Hence, re-implementing and exploring how the "multi-variance" and "multi-mode" problems affect each of these works would be a very time consuming task, and out of the scope of this dissertation. Thus, we have chosen the work presented in [Sfikas et al., 2016] to illustrate the effect of these problems. We did so because it is a

fairly recent work and, most importantly, the source code of the paper is publicly available on the Internet[16].

This work presents a query-by-example (QbE) system that operates on segmented words and makes use of the so-called Zoning Aggregated Hypercolumn features, which is a fixed-size vector representation of a word image.

Each word image is partitioned into 6 vertical (overlapping) zones. Each zone has the same height as the image and a variable width (which depends on the configuration of the method). For each of these zones, pixel-level descriptors are extracted using a ConvNet which was previously trained to perform character recognition on "street-view" images (in particular, they use the network from [Jaderberg et al., 2014]). These descriptors are known as "hypercolumns", which are then aggregated into a single vector per zone, and the vectors from all zones are concatenated to produce a fixed-size, word-level feature vector. As a result, a vector of 1536 features is obtained from each image. These vectors are finally normalized using the euclidean norm. Figure 8.16 describes the feature extraction work-flow.

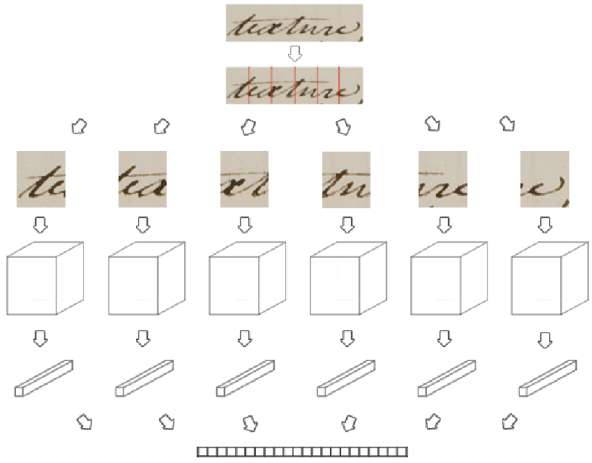


Figure 8.16. Work-flow of the feature extraction process used to obtain Zoning Aggregated Hypercolumn features. Figure kindly provided by Giorgos Sfikas.

[16] `https://github.com/sfikas/zah`

Finally, in order to rank the set of candidate images, the distance between the feature vector corresponding to the query image and each of the candidate vectors is computed, and candidates are ranked according to the increasing euclidean distance.

The experiments are carried on the George Washington database. More particularly, we use the same partitions as in [Almazán et al., 2014, Sudholt and Fink, 2016] and many others. We chose to perform experiments in the George Washington database because both [Sfikas et al., 2016] and [Sudholt and Fink, 2016] used it (and tuned their systems to work well on it). This is the same partition as we will use in the next section, where we will study the probabilistic interpretation of the PHOCNet model. However, notice that this is not the partition used in the original [Sfikas et al., 2016] work (our evaluations sets contain more queries), thus our results do not match those published in the original work. In order to know more details about the George Washington database and this partition in particular, refer to appendix A.2.

We will use the Global and Mean Average Precision measures (gAP and mAP, respectively) to compare three ranking alternatives. In particular, we rank a set of candidate images, represented by their feature vectors $\{\boldsymbol{x}_i : 1 \leq i \leq m\}$ (i.e. all word-segmented images from the test pages), for each of the query images, $\{\boldsymbol{y}_j : 1 \leq j \leq n\}$, using these approaches:

1. *Increasing* order of the euclidean distance (i.e. $\|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2$), as proposed in the original work.
2. *Decreasing* order of the candidate posterior, $P(C = i \,|\, Y = \boldsymbol{y}_j)$, as explained in section 6.3, and particularly using eq. (6.12).
3. *Decreasing* order of the relevance probability, $P(R = 1 \,|\, X = \boldsymbol{x}_i, Y = \boldsymbol{y}_j)$. In particular, using eqs. (6.13) and (6.15).

8.10.2 RESULTS

The candidate posterior in eq. (6.12) and the word posterior in eq. (6.15) (used to compute the relevance probability) have a "sharpness" hyperparameter, s, that needs to be tuned.

Figure 8.17 shows the Mean and Global AP obtained with different values of the sharpness parameter, using each of the probabilistic approaches described before.

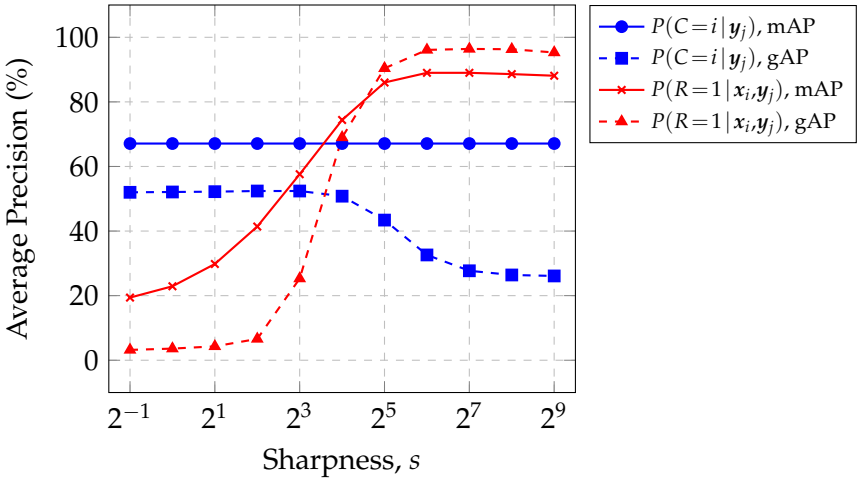


Figure 8.17. Global and Mean Average Precision (gAP and mAP, respectively) for different values of the "sharpness" hyperparameter, s, used in eqs. (6.12) and (6.15) to estimate $P(C=i\,|\,\boldsymbol{y}_j)$ and $P(R=1\,|\,\boldsymbol{x}_i,\boldsymbol{y}_j)$, respectively.

In the case of the candidate posterior, $P(C=i\,|\,\boldsymbol{y}_j)$, the mAP is always 67.1% because this parameter does not affect the relative order of the candidate images when compared to the same query image. In fact, this is the same mAP as the one obtained by the original method (ranking in increasing euclidean distance). On the other hand, this parameters affects both the mAP and the gAP when using the relevance posterior, $P(R=1\,|\,\boldsymbol{x}_i,\boldsymbol{y}_j)$. In order to compute the word posteriors from eq. (6.15), involved in the computation of the relevance probability, we used *all* available training examples.

Given the results depicted in fig. 8.17 we decided to use $s=8$ to compute $P(C=i\,|\,\boldsymbol{y}_j)$, and $s=128$ to compute (the word posteriors involved in) $P(R=1\,|\,\boldsymbol{x}_i,\boldsymbol{y}_j)$. These results are compared with the first (original) approach of simply ranking using the euclidean distance. Table 8.11 shows the mAP and gAP in each of the cross-validation folds of the George Washington database, as well as their average.

Table 8.11. Mean and Global Average Precision (mAP and gAP, respectively) on the George Washington database, obtained by different ranking strategies based on the features extracted by [Sfikas et al., 2016]. The results are shown for each of the individual cross-validation folds, as well as the average across the four sets.

Metric	Order by	CV1	CV2	CV3	CV4	Avg.
mAP (%)	$-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2$	66.6	69.3	66.3	66.3	67.1
	$P(C=i \mid \boldsymbol{y}_j)$	66.6	69.3	66.3	66.3	67.1
	$P(R=1 \mid \boldsymbol{x}_i, \boldsymbol{y}_j)$	89.0	90.5	89.0	87.4	89.0
gAP (%)	$-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2$	32.3	29.0	28.0	28.7	29.2
	$P(C=i \mid \boldsymbol{y}_j)$	52.0	52.7	52.8	52.1	52.4
	$P(R=1 \mid \boldsymbol{x}_i, \boldsymbol{y}_j)$	96.5	96.2	97.0	95.9	96.4

8.10.3 DISCUSSION

As we explained in section 6.3, and as shown in the experimental results from table 8.11, ranking the retrieved results according to some distance may not be a good idea, since this is prone to both the "multi-variance" and "multi-mode" problems, introduced earlier in this thesis.

We can normalize the distances among all candidate images (as $P(C=i \mid \boldsymbol{y}_j)$ does) in order to solve the "multi-variance" issue, which has a large impact on the Global AP (but not the Mean AP, which is unaffected by this). However, these results can be further improved by actually computing a relevance probability (i.e. $P(R=1 \mid \boldsymbol{x}_i, \boldsymbol{y}_j)$) if we have some labeled training examples. As we have seen during this thesis, ranking according to the later probability (if well modeled) is the *optimal strategy* for the usually used Mean and Global Average Precision measures.

Even though we are using a very simple model of the word posterior probability (based only on nearest-neighbors), we will see in the next section (see table 8.13), that our probabilistic framework reduces the gap between the approach described here and more advanced methods (such as PHOC and CTC-based neural networks).

It is worth emphasizing that using other distance metrics would probably not solve the problem. In particular, as we discussed in sec-

tion 6.3, and given that our feature vectors are unit normed, using the popular cosine similarity would have produced exactly the same mAP *and* gAP results[17]. We decided to use the euclidean distance because that was the metric used in the original paper.

Of course, the effectiveness of the relevance probability depends on how good the model of the word posterior probability is. Thus, in the next section, instead of using a simple model based on nearest-neighbors, we will use a neural network to model the word posterior of an image (in a lexicon-free manner).

8.11 Probabilistic interpretation of the PHOCNet

8.11.1 Description

In section 6.4 we discussed that, in practice, the PHOC representation of a word, is exactly equivalent to the word itself. This is because there is virtually a one-to-one mapping between words (strings of characters) and PHOC binary vectors, for the restricted set of words of a given data set (see fig. 6.7).

Then, we saw that the neural networks (or support vector machines, in the original work of [Almazán et al., 2014]) used to predict the PHOC from an image, can be interpreted in a probabilistic way (they are estimating the posterior probability of PHOC binary vectors, given the image, assuming that each of the components of the vector is independent from the others).

Hence, given the (practical) equivalence between words and PHOC binary vectors, the neural networks built are equivalent to a word recognizer. Thus, given this and our probabilistic interpretation of the model, we derived a probabilistically-sound equation to compute the relevance probability from the network's output (see section 6.4.2).

One of the goals of this experiment is to show that this probabilistic approach can achieve results as good as the traditional (dissimilarity-

[17] $\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 = \|\boldsymbol{x}\|_2^2 + \|\boldsymbol{y}\|_2^2 - 2\boldsymbol{x}^\intercal\boldsymbol{y}$. $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1 \Rightarrow \|\boldsymbol{x}-\boldsymbol{y}\|_2^2 = 2 - 2\boldsymbol{x}^\intercal\boldsymbol{y}$. Thus, $\|\boldsymbol{x}-\boldsymbol{y}\|_2$ and $\boldsymbol{x}^\intercal\boldsymbol{y}$ produce an equivalent (inverse) ranking, since one is a monotonic function of the other.

based) PHOCNet approach, which are more interpretable and with some theoretical guarantees (see chapter 3).

Given that a neural network trained to predict the PHOC vectors is essentially as powerful as a word recognizer, the second goal, is to use a more traditional model of the transcript's posterior of a word image.

Regarding the second goal, we tried to use as many parts as possible from the original TPP-PHOCNet architecture. In [Sudholt and Fink, 2016] they tried to replace the last layer from the network with a word classifier (Softmax CNN). However, the results that they obtained were not as good as with the PHOCNet approach, for a KWS task. Although they did not compute the relevance probability using the Softmax output as word posteriors, based on our experience with lexicon-based approaches (see section 8.3.1), we believe that building a word classifier is not the best option due to the out-of-vocabulary problem. Thus, we have replaced the fully connected layers from the PHOCNet architecture with a single recurrent layer, followed by a Softmax layer to predict *character labels*, and have used the CTC loss to train this network. With the network's posteriors, we generate lattices as described in section 4.6.3, we prune this lattices for speed purposes with a beam equal to 15, and then we simply use the method presented in section 2.3.1.1 to compute the relevance probability. This latter step is performed simply by a composition of two WFSTs and the Backward algorithm in the result to compute the sum of all paths (review section 4.6.2.2).

In summary, we will compare the results obtained by three different approaches:

1. Ranking the elements by increasing Bray–Curtis measure, of the pairs of vectors produced by the PHOCNet architecture (actually, the TPP-PHOCNet from [Sudholt and Fink, 2017]). We will refer to this approach as PHOCNet.

2. Ranking the elements by decreasing relevance probability, computed from our probabilistically-sound interpretation of the PHOC representation. We will refer to this as Probabilistic PHOCNet.

3. Ranking the elements by decreasing relevance probability, computed from a similar neural network to the TPP-PHOCNet, but trained using the CTC loss. We will refer to this approach as CTCNet.

For the PHOC-based methods (1 and 2 from the list above) we use the TPP-PHOCNet architecture described in [Sudholt and Fink, 2017]. We have re-implemented their architecture in PyTorch, since we later needed to slightly adapt it to use the CTC loss. Since the authors published their source code, we made sure to be using the same training loss (binary cross entropy); learning algorithm (Stochastic Gradient Descent, with batch size 10, learning rate equal to 10^{-4}, momentum equal to 0.9, and weight decay equal to $5 \cdot 10^{-5}$); and the same data augmentation procedure.

We use the same evaluation measures as in the paper. That is, we compute the Mean Average Precision (mAP), *without* precision interpolation. In addition, we also compute the Global Average Precision (gAP), which is not typically reported in PHOC-based works.

In order to compare these results with those of the previous section, we have replicated the experiments from [Sudholt and Fink, 2017] on the George Washington database, using exactly the same partitions, ground-truth transcripts and query sets, to make sure that we made a fair comparison. Since the results of all methods are very close in this data set, we have trained 8 neural networks with different random seeds for each of the cross-validation (CV) fold of the database. We report the average of the 8 runs in each CV partition, for each of the assessment measures (mAP and gAP).

8.11.2 RESULTS

Table 8.12 shows the mAP and gAP achieved by each approach, in each of the George Washington CV test sets, as well as the average across the four cross-validation folds.

First, observe that our own implementation of the TPP-PHOCNet achieves almost the same mAP as the original work (97.7% and 97.8%, respectively). The differences are likely due to the random seed used in each case, and due to the fact that some operations executed in

GPUs are non-deterministic. Our implementation achieves 98.4% gAP, while in [Sudholt and Fink, 2017] they did not report this figure.

The Probabilistic PHOCNet formulation (Prob. PHOCNet) achieves slightly better results for both mAP and gAP: 98.1% and 98.7%. The CTCNet approach gives an additional (small) improvement on the mAP, achieving a value of 98.3%, while the gAP is the same as the Prob. PHOCNet.

Table 8.12. Comparison of the Mean and Global Average Precision achieved by PHOC-based works and our distance-based methods, in the George Washington database.

Metric	Method	CV1	CV2	CV3	CV4	Avg.
mAP (%)	PHOCNet (orig.)	—	—	—	—	97.8
	PHOCNet (ours)	97.7	98.2	97.9	97.1	97.7
	Prob. PHOCNet	98.0	98.5	98.3	97.7	98.1
	CTCNet	98.8	98.4	98.3	97.5	98.3
gAP (%)	PHOCNet (orig.)	—	—	—	—	—
	PHOCNet (ours)	98.3	98.4	99.0	98.0	98.4
	Prob. PHOCNet	98.3	98.9	99.4	98.3	98.7
	CTCNet	98.3	98.8	99.5	98.1	98.7

8.11.3 DISCUSSION

It is important to emphasize that we did not performed any fine-tuning for any of the approaches, we simply used the same architecture and hyperparameters used in the original TPP-PHOCNet paper. This means that the Probabilistic PHOCNet and the CTCNet may be in a slight disadvantage in front of the original TPP-PHOCNet, since in the later they presumably did tune these hyperparameters, as well as the architecture of the neural network. Nevertheless our (lower bounded, in this sense) mAP and gAP are still better than those of the PHOCNet, although the differences are not significant, since all methods achieve very high values for both metrics.

One advantage of the CTCNet, which we have not exploited here, is that we could have combined its output with an external language

model, as we did in previous sections. However, we decided not to do so to keep this experiment simple and comparable with traditional word segmentation-based publications.

Moreover, the CTCNet has far less parameters than the PHOCNet (17 935 589 and 59 859 420, respectively), since all the fully connected layers in the latter have been replaced with a single bidirectional LSTM, and a linear layer for the final mapping. This suggests that indeed the PHOCNet is over-parameterized for the *task*. This is consistent with our hypothesis that a given PHOC binary vector is a redundant representation of a particular word. Thus, by eliminating this redundancy we can reduce the number of parameters needed by the model, as the CTCNet results shows.

Table 8.13. Summary of the KWS results on the word-segmented George Washington database, for different approaches described in this thesis. The first three rows are results obtained following a traditional distance-based approach. The three middle rows were obtained using PHOC-based neural networks. And the final row shows the result of a CRNN trained using the CTC algorithm.

Method	mAP (%)	gAP (%)
(Features from [Sfikas et al., 2016])		
Raw distance $-\|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2$	67.1	29.2
$P(C = i \mid Y = \boldsymbol{y}_j)$	67.1	52.4
$P(R = 1 \mid X = \boldsymbol{x}_i, Y = \boldsymbol{y}_j)$	89.0	96.4
PHOCNet [Sudholt and Fink, 2017]	97.8	—
PHOCNet (ours)	97.7	98.4
Prob. PHOCNet	98.1	98.7
CTCNet	98.3	98.7

Table 8.13 summarizes the results of the KWS experiments performed using segmented word images. That is, the results from the probabilistic interpretation of the the traditional distance-based methods (described in the previous section), the PHOC-based neural networks (including its probabilistic interpretation described in this section), and the CTC-based neural network (also described here).

Observe that applying the probabilistic formulation described in this thesis over the traditional distance-based and PHOC-based ap-

proaches improves both the mAP and gAP. In addition, using a neural network trained with the CTC loss also improves these results. The most remarkable result is the huge improvement achieved using a simple distance-based method, which significantly reduces the gap between traditional approaches based on nearest-neighbors and modern convolutional neural networks.

8.12 Multi-word queries

In the last, but not least, experiment we evaluate the performance of a system based on our probabilistic perspective that can respond to multi-word Boolean queries.

8.12.1 Description

This system, essentially follows the method described in section 7.1. Since multi-word queries rarely occur on the same line, the evaluation in this experiment is done at page level. That is, our goal is to determine whether or not a complete page is relevant for the given (Boolean) query. A given page is relevant for a multi-word AND query if all words in the query are written in the page. Similarly, the page is relevant with respect to an OR query if any of the words is written in the page.

We use the Bentham collection, previously used in section 8.7. The multi-word queries that we use are all pairs of distinct words from the original query set used in this database. The original query set consists of 3 293 query keywords, extracted from the training set, with a frequency of occurrence in the training set ranging from 2 to 10. This avoids including most stop words (generally with word frequencies greater than 10) and also many (singleton) words that are unlikely to appear in the test partition. This gives a total number of 5 420 278 multi-word queries, that have to be spotted in 33 test pages.

Despite the large number of single-word and multi-word queries, only a small portion of queries have some relevant page in the database. In the case of the single-word queries, only 674 words appear in the test images. Similarly, for the multi-word AND and OR queries, only 1 992 007 and 11 784 have some relevant page, respectively. We call

these queries "pertinent", and the queries without any relevant page in the test set, "non-pertinent".

In our original publication, we study the performance for all types of queries (singletons, AND, and OR) when the number of "non-pertinent" queries increases. However, for the sake of brevity, we will only report the results for "pertinent" queries (i.e. queries for which at least one relevant page exists), and for a set of queries with the same amount of "non-pertinent" and "pertinent" queries. We refer to these two groups as $r = 0$ and $r = 1$, respectively (i.e. the ratio between "non-pertinent" and "pertinent" queries).

In order to build the probabilistic index, we used Hidden Markov Models and word 2-grams. That is, our approach is lexicon-based. The models that we used were essentially the same as in section 8.7, but here we also used discriminative training to improve the training of the HMMs. The results from this section were published in [Toselli et al., 2018b], which contains additional details about the statistical models and the training procedure.

Since we operate at page level (without caring about the actual location of the words within the page), we build a probabilistic index that contains the (approximate) relevance probability for each word in the lexicon and each page from the test set. The page-level relevance probability is approximated by the probability of the best segment within the page, just as we did in most line-level experiments in the rest of this chapter. Then, the relevance probability for multi-word queries is simply approximated by min and max operations, as explained in section 7.1.

8.12.2 RESULTS

Table 8.14 shows the results of the multi-word experiments performed, for the two types of query sets: $r = 0$ is the set of queries that are all "pertinent" and $r = 1$ is the set of queries with the same ratio of "pertinent" and "non-pertinent" queries. Only the Global AP is reported, since the Mean AP cannot be computed on "non-pertinent" queries. As shown in the table, the system has an excellent performance for all types of queries (single-word and multi-word), in the two considered query sets.

Table 8.14. Global Average Precision (gAP) obtained in the multi-word KWS experiments, on the Bentham database. $r = 0$ denotes the set of all "pertinent" queries, and $r = 1$ the set of the same number of "non-pertinent" and "pertinent" queries.

		Single	AND	OR
gAP (%)	r = 0	94.6	93.1	93.5
	r = 1	92.7	91.9	90.9

It is important to understand that, from a practical point of view, a ratio of "non-pertinent" to "pertinent" queries equal to 1 is quite unlikely: it would correspond to the use of an information retrieval system where half of the queries try to find information which cannot actually be found in the indexed collection.

Obviously, a perfect system should not produce any result for a non-pertinent query. But a real system may spot, with non-negligible confidence, pages containing words similar to those stated in the query. This typically tends to result in degradations of the precision.

8.12.3 DISCUSSION

First, observe that the total number of pairs of multi-word queries and pages in this experiment is extremely large. Without the use of a probabilistic index, computing the relevance probability on-the-fly, for such a large number of pairs would be simply unfeasible.

Secondly, we have seen that our simple combination of the single-word probability relevances yields to an excellent performance for both types of multi-word queries. This experiment validates the approach presented in section 7.1. Recall that the max and min combination of probabilities, for the OR and AND Boolean queries, gives a lower and upper bound to the real probability, respectively (assuming that the real probability is well represented by the used statistical model).

As a result, we can use this simple approach to support multi-word queries in real (and large-scale) databases, as we will see in the demonstrators described in the next chapter.

8.13 Summary

Here, we summarize the main results of each of the experiments conducted in this chapter, that give the answer to each of the questions posed at the start of the chapter.

1. In section 8.2 we showed that the line relevance probability can be efficiently and effectively approximated by the segment or transcript position probabilities. We have algorithms that can construct such indexes in a very efficient way, for both lexicon-based and lexicon-free models (2 pages/sec. and 1 page/sec., respectively, without using any parallelization).

2. In section 8.3, we showed that lexicon-based approaches, although usually faster, can produce worse results if they are not trained with a large lexicon. Thus, in real applications, we recommend using lexicon-free approaches with a sufficiently large order n-gram language model (e.g. $n \in [6, 8]$).

3. We saw that the number of labeled samples that are needed to train the used neural networks is not excessively high (see section 8.4). In fact, for the considered data set, and using synthetic data augmentation, we can get the same performance using only about one half of the available training data. Of course, using additional training data does not hurt the performance.

4. There is a clear linear correlation between the performance in HTR and KWS tasks, as shown in section 8.5. In particular, we can alleviate the experimentation costs since HTR experiments are much faster to perform, and we know that improvements in CER and WER will translate to gAP and mAP.

5. In section 8.6, we showed that our approach produces state-of-the-art results for several line-level KWS benchmarks, widely used in prior academic works.

6. In section 8.7, we also showed that more traditional models for the handwritten text, namely HMMs, can also be effectively

used in our framework, since our indexing algorithms take WFSTs as the input. Also, HMMs provide virtually the same performance as RNNs, for well segmented text lines.

7. We tackle segmentation-free KWS by simply performing automatic text line segmentation. We showed in section 8.8 that this provides excellent results, on the data sets of two different competitions, using different methods for text line segmentation. In addition, we also show that RNNs are more robust to errors in the automatic line segmentation than HMMs. Thus, we prefer the former in real applications like the large-scale demonstrators, described in the next chapter.

8. In sections 8.9 to 8.11, we perform experiments interpreting traditional KWS approaches from a probabilistic perspective (namely the HMM-Filler, distance-based, and PHOCNet-based methods). We show that a probabilistic interpretation with the correct models can greatly improve the mAP and gAP measures.

9. In section 8.12, we have showed that our methods can be effectively (and efficiently) used to support multi-word queries. This approach is used in all the large-scale demonstrators used in the next chapter.

9 Large-scale demonstrators

In order to demonstrate the effectiveness and efficiency of the probabilistic framework and the algorithms developed in this thesis, we built several large-scale demonstrators, within the scope of different research projects. All demonstrators allow the users to search for arbitrary Boolean expressions and phrase queries on different historical collections, composed of several thousands of pages, written in different languages and from different periods of time.

9.1 Architecture design

The demonstrators follow a client–server architecture made of 4 main components: the web client, the web server, the data server and the KWS server, as shown in fig. 9.1.

9.1.1 Description of the servers

9.1.1.1 Web and Data servers

The main task of the web server is to serve the HTML pages of the user interface and prepare the responses to the user's requests. For that, a HTTP web server (Apache) is used with PHP support. The web server connects to the other servers when the requested information is not available.

For instance, this services does not directly stores the document images and metadata, which are stored in a separate database, handled by the Data server (although in practice, the data server and the web server reside in the same machine).

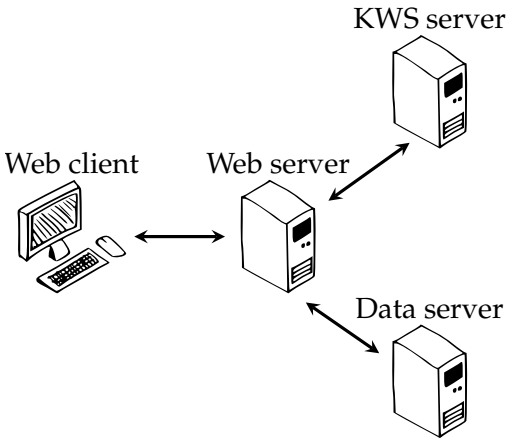


Figure 9.1. Client–server architecture used by the large-scale demonstrators. The web client is a regular web browser, the web server receives the user's request and prepares the response as HTML files. The KWS server contains the probabilistic index, similar to an "inverted index" used by a regular search engine. The data server stores all images and metadata of the documents in a database.

Likewise, when the user sends a KWS query to the web server, this redirects the query to the KWS server, and prepares the HTML pages that will present the relevant results to the user.

9.1.1.2 KWS server

The KWS server is the main component of the system and implements a probabilistic word index, similar to the ones used by regular search engines. This server implements a RESTful API using HTTP which is used by the web server to redirect the user's queries.

The index has a conceptual hierarchical structure, as represented in fig. 9.2. In our current implementation, we actually use a hierarchy of hash tables to represent this conceptual hierarchy.

At the top of the hierarchy, we use a hash table that maps from words to the set of collections that contain (with an estimated probability greater than 0) at least one instance of such word. Then, an additional set of hash tables are used to map to the collection's books that contain each term. An additional level of the hierarchy maps to relevant pages within the book. Finally, for each page, a (ranked) list

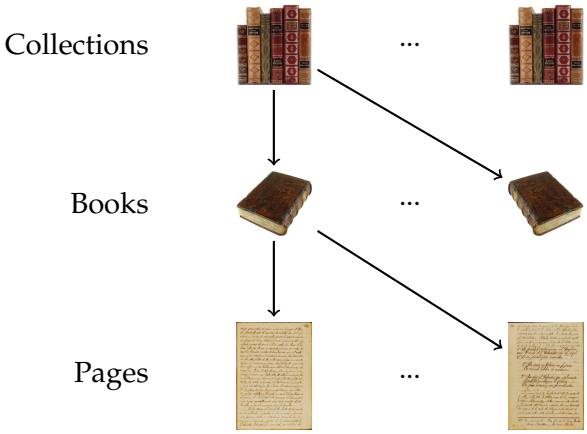


Figure 9.2. Representation of the hierarchical index used in the large-scale demonstrators.

of all relevant word instances within that page is stored with the actual relevance probability, and the bounding box coordinates of the word instance.

Notice that using hash tables allows for extremely fast searches at any particular level of the hierarchy, since accessing a particular level can be done in constant time (on average). However, this representation is not the most memory efficient one, since the explicit implementation of the hierarchy has a significant memory overhead.

This memory footprint could be reduced significantly using a "linear" layout of the hierarchy, for instance keeping the tuples (word, collection, book, page) sorted. Then, any level could be accessed in $O(\log n)$ (where n is the total number of indexed pages), which would result in virtually the same time as the nested hash table architecture in real use cases, but much lower spatial cost.

An additional advantage of this linear layout is that, since words would be lexicographically ordered, we could support prefix queries very efficiently. Nevertheless, our current implementation does not support this feature.

9.1.2 DESCRIPTION OF THE WEB CLIENT

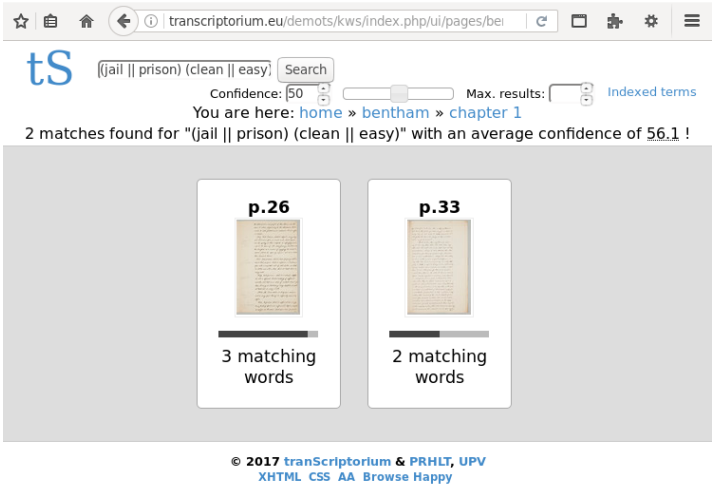
From the user perspective, the system is simply a web page, similar to a regular search engine, which she can access using her favorite web client (Firefox, Chrome, Safari, etc.). This way, we can support users using multiple devices and operating systems. The web client only communicates through the web server with the rest of the components.

The web interface consists of a search box where the user can type the query. The user can also specify the maximum number of results that the system should retrieve and/or a confidence threshold to filter out the results with a relevance probability below that threshold.

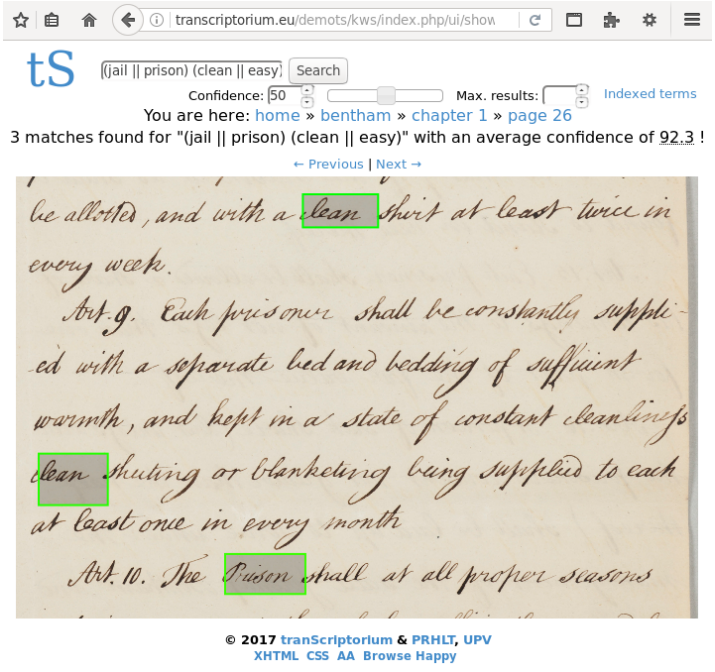
Queries can be formed by individual terms, or Boolean combinations of these terms using the logical AND, OR and NOT operations, which are represented, respectively by the characters "`&&`" (i.e. double ampersand), "`||`" (double vertical bar) and "`-`" (dash). If the user types two consecutive words without any operators, the system interprets it as a logical AND. The user can form nested Boolean expressions using the parentheses (e.g. `(jail || prison) && (clean || easy)`). In addition, phrase-queries are also supported by our engine using the square brackets (e.g. `[the white house]`).

The web client offers different views, depending on the level of the hierarchical index that the user is currently visiting (see the previous subsection). If the user is at the root and sends a request to the system, this will report all relevant collections. Then she can look into a specific collection and the set of relevant books of that collection will be shown. Next, the user can move to a specific book and the set of relevant pages will be shown. Finally, she can visualize the location of the spotted words in a particular page.

In the inner stages of the hierarchy, the visual interface is as depicted in fig. 9.3a. This figure is showing all relevant pages for a particular query within a book. At the leafs of the hierarchy (pages), the interface is as seen in fig. 9.3b, where all the instances of the spotted terms within a particular page are shown. The color of the box surrounding the spotted word indicates the confidence (relevance probability) of the match (green: high confidence, red: low confidence).



(a) Book level



(b) Page level

Figure 9.3. Web client graphical interface at different levels of the index hierarchy displaying the search results for the Boolean query `(jail` $\vee$ `prison)` $\wedge$ `(clean` $\vee$ `easy)`.

9.2 Trésor des Chartes

The large collection of the "*Trésor des Chartes*", produced by the French royal chancery and encompassing about 68 000 images, dating from 1302 to 1483, was used to evaluate the outcome of the HIMANIS project[1]. The main goal of HIMANIS was to build effective and efficient solutions serving queries over large sets of historical handwritten document images.

The "*Trésor des Chartes*" collection is a challenging and particularly interesting case study. This large and emblematic collection is a key source of information to understand the origin of centralized nation states in Europe, in the medieval age.

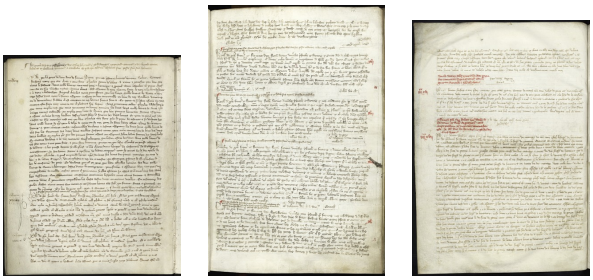
In order to train the statistical models required to create the probabilistic index, 436 images were used. The transcripts corresponding to these images were obtained from the text edition provided by Paul Guérin [Guérin, 1896]. The training data covers the registers dating from 1302 to 1361, which represents merely 0.67% of the complete corpus, but is largely representative of its diversity and challenges. Figure 9.4 shows some pages from this collection as well as a trimmed act image with its *modernized* transcript.

Medieval documents like the one considered in the HIMANIS project, entail two important linguistic challenges. Namely, they are written in more than one language and they are heavily abbreviated (in the Chancery data set, especially the text parts written in Latin). The latter problem is particularly insidious because the (only) image transcripts generally available are *modernized* versions where all the abbreviations are completely expanded.

As discussed in [Villegas et al., 2016c], this constitutes a serious drawback to train adequate optical character models. Moreover, according to the requirements of the search functionality aimed at in HIMANIS, textual queries must allow (only) modernized word forms.

Nevertheless, thanks to the combination of powerful statistical models for handwritten text (in particular CRNN) and the lexicon-free indexing approaches presented in this thesis (in particular sec-

[1] `https://himanis.org/`



Philippes, etc. Savoir faisons à touz presenz et avenir que, comme ou traitié du mariage de nostre amé et feal le seigneur de Harecourt
et de Ysabeaul, sa fame a present, file de nostre amé et feaul le seigneur de Partenay, certainnes convenances eussaint esté traite-
-es et acordées entre les dites parties par leur consoil de leurs amis, et sus ycelles convenances eust esté et fust entre eus aucun
descort, c'est assavoir que li diz sires de Harecourt dysoit et maintenoit que deux miles livres à tournois de rente annuelle et perpetu-
-elle que li diz sires de Partenay li avoit promises et données pour cause dou dit mariage, à avoir du dit seigneur de Harecourt après le
deceps du dit seigneur de Partenay, ou cas que il auroit hoir ou hoirs mâles, et le cas se estoit jà offert, li devoient estre assises à Chas-

Figure 9.4. Examples of page images from the "*Trésor des Chartes*" data set.Extract from Paris, Archives Nationales, JJ 67, fol. 34v, n. 97 (1 May 1329). The ground truth, modernized transcript of the text in the bottom image is also shown.

tion 5.3.2), the project was successful against these challenges, as shown in the results reported by [Bluche and Messina, 2017].

The quantitative results from the project are summarized in table 9.1. These were obtained from the available ground-truth data (341 images used for training, 95 used for evaluation). Words from the evaluation set, excluding numbers and punctuation marks, were used as query words, making a total of 6 506 query keywords. A large proportion of these keywords are expanded forms of words which in the images appear abbreviated in several ways. Thus, a query set exclusively composed of expanded forms of abbreviated words was also used (244 total keywords).

Lexicon-based and lexicon-free approaches were compared, as well as deterministic indexes (obtained from the 1-best automatic transcript given by the system) and probabilistic indexes. The results achieved for abbreviations-only are superior than those of the full query set, because most abbreviated words have, on average, more training instances than the expanded ones.

Table 9.1. Summary of Average Precision results in the "Chancery" data set from the HIMANIS project. Lexicon-free models (i.e. "Lex-free") use a combination of CRNN and character 5-grams. Lexicon-based models (i.e. "Lex-based") use the same CRNN, but a word 2-gram language model. Deterministic indexes ("deter.") use the automatic transcript produced by the system, while the probabilistic ones ("prob.") use the framework described in this thesis (particularly, algorithms described in sections 5.1.2 and 5.3.2).

Query set	System	mAP (%)	gAP (%)
Full	Lex-free deter.	44.5	58.1
	Lex-based prob.	46.2	63.7
	Lex-free prob.	68.0	75.0
Abbreviations-only	Lex-free deter.	52.2	68.6
	Lex-free prob.	73.6	83.8

The resulting probabilistic index contains about 366 million entries (indexed spots), as shown in table 9.2. The reader can access the demonstrator following the next hyperlink.

`http://prhlt-carabela.prhlt.upv.es/himanis`

Table 9.2. Statistics of the probabilistic index used in the *Trésor des Chartes* demonstrator.

# Pages (P)	83 290
# Spots	365 888 252
# of running words (W, expected value)	47 260 386
# Spots / W	7.7
# W / P (expected value)	567.4
# Spots / P	4392.9

9.3 Teatro del Siglo de Oro

The "*Teatro del Siglo de Oro*" collection was indexed within the READ project[2], as one of its multiple large-scale demonstrators. The goal of the Recognition and Enrichment of Archival Documents (READ) project was to make archival material more accessible by developing and implementing cutting-edge technology in Handwritten Text Recognition, Key Word Spotting, Layout Analysis, and related fields.

In particular, the "*Teatro del Siglo de Oro*" collection encompasses hundreds of play[3] manuscripts written by the Spanish playwright, poet and novelist, Lope de Vega, between the late XVI and early XVII centuries. Lope de Vega was one of the key figures in the Spanish Golden Age[4] of Literature, and is one of the most prolific authors in the history of literature. The complete collection of manuscripts is made of about 100 000 images, although at the moment of writing this dissertation only 23 000 images (230 manuscripts) have been indexed so far. The statistics of the probabilistic index are shown in table 9.3.

Table 9.3. Statistics of the probabilistic index used in the *Teatro del Siglo de Oro* demonstrator.

# Pages (P)	22 498
# Spots	31 227 062
# of running words (W, expected value)	3 464 620
# Spots / W	9.0
# W / P (expected value)	154.0
# Spots / P	1388.0

Although these documents are written by a single hand in a single language, they contain a large portion of abbreviated words (e.g. most names of characters and locations in the scripts are abbreviated), which makes the indexing process very challenging, in similar ways to the faced in the "*Trésor des Chartes*". Figure 9.5 shows some examples of the images included in the collection, and a detailed view of a

[2] `https://read.transkribus.eu/`

[3] *Teatro*, in Spanish.

[4] *Siglo de Oro*, in Spanish.

fragment of one page. The figure also shows a spotted result for the phrase query `"la India oriental"`.

In order to build the probabilistic indexes for the large scale demonstrator, 286 transcribed page images were used. The statistical models used to build the index were a neural networks (a CRNN like the one described in section 4.3.5 and used in many other experiments in this thesis) and a 8-gram character language model (trained only on the available transcripts). In order to perform a quantitative assessment of the quality of the probabilistic index, cross-validation was performed using a query set made of 5 409 keywords. The (lexicon-free) probabilistic index was built using the algorithm described in section 5.3.2. Figure 9.6 shows the Recall–Precision curve to summarize the results of this experiment.

These preliminary results show, once again, that the proposed probabilistic framework and algorithms introduced in this thesis can be effectively (and efficiently) used to tackle the task of indexing a large collection of document images. The demonstrator can be accessed following the link below.

`http://prhlt-carabela.prhlt.upv.es/tso/`

9.4 The Bentham Collection

Last, but not least, the Bentham collection is used as the third large-scale demonstrator to highlight the effectiveness of the technology introduced in this thesis. This collection includes several hundreds of manuscripts written by Jeremy Bentham, an influential English philosopher, jurist, and social reformer from the XVIII and XIX centuries; as well as many of his secretarial staff. A portion of this collection was first indexed in the Transcriptorium project[5]. The rest of the collection was processed during the READ project, described before. The entire collection is composed of around 100 000 images, from which about 76 000 are available through this large scale demonstrator. The statistics of the index used in the demonstrator are shown in table 9.4.

[5] `http://transcriptorium.eu/`

(a) Examples of page images from the "*Teatro del Siglo de Oro*" collection

(b) Spotted result for the phrase query `"la India oriental"`.

Figure 9.5. Examples of page images from the "*Teatro del Siglo de Oro*" collection, and a spotted result for a phrase query.

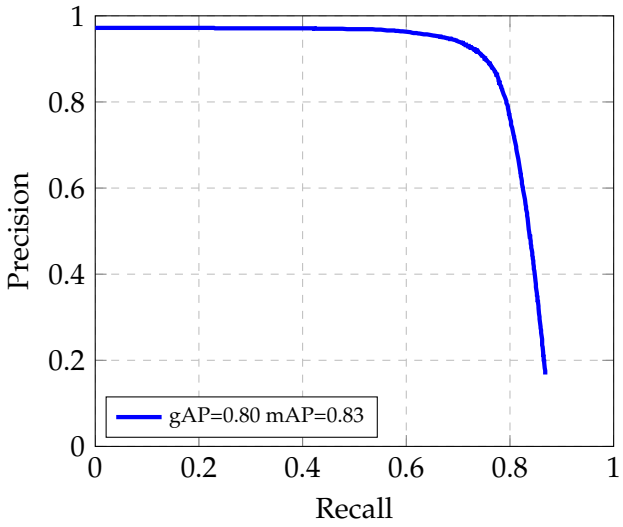


Figure 9.6. Recall–Precision curve summarizing the results of the experiments carried out in the "*Teatro del Siglo de Oro*" collection. The Global AP (area under this curve) is of 80%, while the Mean AP is of 83%.

Table 9.4. Statistics of the probabilistic index used in the Bentham demonstrator.

# Pages (P)	76 161
# Spots	171 991 957
# of running words (W, expected value)	22 770 041
# Spots / W	7.6
# W / P (expected value)	299.0
# Spots / P	2258.3

A subset of this collection has been used in many HTR and KWS competitions, as well as in some of the experiments described in the previous chapter. Some quantitative results can be reviewed in sections 8.8.1, 8.8.2 and 8.12. Details about the data used for these experiments can be reviewed in appendix A.1.

The demonstrator can be accessed through the link below.

`http://prhlt-carabela.prhlt.upv.es/bentham`

10 Conclusions

10.1 Contributions

In this section the main contributions of this dissertation are summarized.

10.1.1 Keyword Spotting probabilistic framework

After describing the many flavors of Keyword Spotting (see chapter 1), we have introduced a new probabilistic formulation of Keyword Spotting in chapter 2. Here, we present Keyword Spotting as an instance of Information Retrieval, where the relevant documents or pieces of documents (i.e. manuscript pages, individual lines or image segments) have to be presented to the user for a given query (represented by a string or an exemplar image).

The key idea of the formulation is to compute a (binary) relevance probability that takes into account the definition of relevance given the content of the document and the query, and the stochastic nature of the content, given the images (of the document, and also the query in the query-by-example paradigm).

Besides, this formulation can be applied (directly, or with minor modifications) to tackle Keyword Spotting for all of its typical segmentation assumption (word and line-based segmentation, as well as segmentation-free), as empirically shown in the experiments chapter (i.e. chapter 8).

Not only that, in chapter 3 we have mathematically proven that, under the necessary assumptions (i.e. the *correct* probability distributions are used), our framework is the best possible strategy (i.e. *opti-*

mal) for many of the typically used evaluation measures used across academia and industry, like the Average Precision, Normalized Discounted Cumulative Gain and Precision-at-k measures (both Mean and Global).

In short, we have built a principled and robust probabilistic framework for Keyword Spotting and have shown its validity both theoretically and in practice.

10.1.2 Probabilistic models of handwritten text

Because our framework works on top of probability distributions, we need good statistical models to approximate these and that can be automatically estimated from training data, (since the *real* distributions are never know in practice).

In chapter 4 we have reviewed some of the traditional models used to represent the content (transcript) of handwritten text documents, and we have proposed a new neural network-based architecture that only uses convolutional and one-dimensional recurrent layers. This architecture achieves a similar or better accuracy then previous state-of-the-art models in a fraction of time (6 to 9 times faster).

Finally, we have explained how the result (i.e. the stochastic transcription of a text image) of these models, and others yet to come, can be be represented using Weighted Finite State Transducers, which unifies its representation and allows to use our algorithms with arbitrary types of probabilistic models.

10.1.3 Indexing algorithms based on the framework

Using Weighted Finite State Transducers to represent the (stochastic) content of the document and queries, in chapter 5 we have developed several algorithms that allow us to build probabilistic word indexes, based on the framework developed earlier.

These indexes, similar to the ones constructed by traditional search engines on text data (like Web pages, or PDF documents) enable us to scale our framework to very large collections of documents, with a

search time which is virtually constant with respect to the the index size (the number of terms and documents indexed).

10.1.4 Probabilistic interpretation of other methods

In chapter 6, we have presented a probabilistic interpretation of several methods used in the Keyword Spotting literature in the past (and some still used).

This interpretation enabled us to understand the assumptions and limitations of these approaches, and we proposed different solutions for "fixing" these limitations and improving such methods. For instance, we studied (and improved) the traditional HMM-Filler, the BLSTM-CTC approach, a distance-based approach typically used for segmentation-based query-by-example KWS, and the popular PHOC-Net method.

10.1.5 Beyond traditional and academic Keyword Spotting

Finally, we have used our probabilistic framework and algorithms to tackle Keyword Spotting scenarios which are less frequent in the academic literature, but are of vital importance for real usage scenarios.

For example, in chapter 7 we have presented how to tackle, in a principled way, Keyword Spotting with Boolean and phrase queries, using our framework. Also, we have suggested how to use our probabilistic formulation in other applications related to Keyword Spotting (from an Information Retrieval perspective), such as Content-based Image Retrieval.

Moreover, in chapter 9 we have shown that our solutions can be applied, in an effective and efficient manner, to large collections of documents, which are rarely targeted in academic domains.

10.2 Scientific publications

Although a large portion of the concepts, algorithms, proofs and experiments presented in this thesis are brand new and have not been published before, some others were first published in several conference and journal papers. In summary, during the development of

this thesis, the author has collaborated in the publication of 2 refereed journal articles and 13 refereed conference papers (6 CORE A, 5 CORE B, 2 CORE C), which are directly related to the content of this thesis.

This section briefly describes these publications and its relationship to this dissertation.

10.2.1 Probabilistic models of handwritten text

The following paper described the architecture of the neural network widely used in this thesis to (probabilistically) model the transcript of a handwritten text line.

- **Joan Puigcerver**. Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Kyoto, Japan, 2017. 🏆

  The need for multidimensional recurrent layers (particularly, 2D-LSTMs) for handwritten text recognition was studied in this work. As explained in section 4.3.5, we found a much faster architecture based only on 1D-LSTMs and convolutional layers that matched the performance of the previous state-of-the-art. This work was awarded with the Best Student Paper Award.

10.2.2 Keyword Spotting probabilistic framework

The papers in this subsection present some of the concepts of the probabilistic framework described in this thesis or describe some of the algorithms.

- **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. Probabilistic interpretation and improvements to the HMM-filler for handwritten keyword spotting. *Proceedings of the 13th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Nancy, France, 2015.

  The core of the probabilistic framework presented in section 2.1

was first described here, as well as the probabilistic interpretation of the HMM-Filler, described in section 6.1. Experiments from this paper, and more additional results are described in section 8.9.

- Enrique Vidal, Alejandro H. Toselli, and **Joan Puigcerver**. High Performance Query-by-Example Keyword Spotting Using Query-by-String techniques. *Proceedings of the 13th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Nancy, France, 2015.

  The probabilistic framework developed in this thesis was used to tackle the query-by-example paradigm, as described in section 2.3. The results from this paper, and additional experiments using CRNNs are described in section 8.8.

- Alejandro H. Toselli, **Joan Puigcerver**, and Enrique Vidal. Context-Aware Lattice based Filler approach for Keyword Spotting in Handwritten Documents. *Proceedings of the 13th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Nancy, France, 2015.

  In this publication, we modified the traditional HMM-Filler and studied its performance using high-order n-gram language models. In addition to the results reported in this paper, we have conducted more related experiments, which are described in section 8.9.

- Alejandro H. Toselli, **Joan Puigcerver**, and Enrique Vidal. Two Methods to Improve Confidence Scores for Lexicon-Free Word Spotting in Handwritten Text. *Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016.

  Here we presented an algorithm which computes the relevance probability of a column (see section 2.2.1) from a character lattice. We compared its performance with the relevance probability of the whole text line (i.e. section 2.1).

- Alejandro H. Toselli, Enrique Vidal, **Joan Puigcerver**, and Ernesto Noya-García. Probabilistic multi-word spotting in handwritten

text images. *Pattern Analysis and Applications*. 2018. Here, we use the algorithm described in section 5.3.2 to build an index of words. Then, the approach described in section 7.1 is used to perform Boolean queries. The models to generate the lattices involved in the experiments are CRNNs, as described in section 4.3.5.

10.2.3 KEYWORD SPOTTING APPLICATIONS

The following papers describe the application and results of some of the probabilistic indexing algorithms described in this thesis, especially on large-scale collections of handwritten documents.

- Theodore Bluché, Sebastien Hamel, Christopher Kermorvant, **Joan Puigcerver**, Dominique Stutzmann, Alejandro H. Toselli, and Enrique Vidal. Preparatory KWS Experiments for Large-Scale Indexing of a Vast Medieval Manuscript Collection in the HIMANIS Project. *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Kyoto, Japan, 2017.

  In this publication the results of the HIMANIS project, briefly described in section 9.2, were presented. The experiments carried out in the paper were done using the lexicon-free indexing approach presented in section 5.3.2, which implements the probabilistic framework developed in this thesis, in particular the one described in section 2.2.2.

- Eva Lang, **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. Probabilistic Indexing and Search for Information Extraction on Handwritten German Parish Records. *Proceedings of the 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Niagara Falls, United States, 2018.

  The indexing methods presented in section 5.3.2 were applied to one of the large-scale collections of the READ project. This collection is composed of pages with tabular content, which we exploit to improve the information extraction accuracy.

10.2.4 KEYWORD SPOTTING COMPETITIONS

Several competitions were organized to promote the interest in the field of Keyword Spotting, and that provided useful data sets and results for this dissertation (e.g. sections 8.8.1 and 8.8.2).

- **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. ICDAR2015 Competition on Keyword Spotting for Handwritten Documents. *Proceedings of the 13th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Nancy, France, 2015.

  The principal goal of this competition was to provide a unified framework to evaluate distinct approaches to Keyword Spotting. In this thesis, experiments that outperform the best results in the segmentation-free scenario have been presented in section 8.8.2.

- Mauricio Villegas, **Joan Puigcerver**, Alejandro H. Toselli, Joan A. Sánchez, and Enrique Vidal. Overview of the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task. *Working Notes of the Conference and Labs of the Evaluation Forum (CLEF)*. Évora, Portugal, 2016.

  This competition aimed to bring research groups with interests in Information Retrieval to the Keyword Spotting community. It also introduced some new aspects, not typically found in KWS competitions, like multi word queries, possibly broken into multiple text lines. With the results from this competition, we could study the correlation between the performance of KWS systems at word level and line level, as mentioned in section 1.2.2.2.

- Ioannis Pratikakis, Konstantinos Zagoris, Basilis Gatos, **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016). *Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016.

  We organized this competition as a continuation of the ICDAR2015 Competition on Keyword Spotting. In addition to the previous

goals, we aimed to analyze the effects of the available training data in the performance of the submitted KWS solutions.

10.2.5 Other Keyword Spotting works

Before we developed the indexing algorithms for lexicon-free lattices, described in section 5.3, we tried to estimate the relevance probability of out-of-vocabulary keywords from lexicon-based lattices (or word graphs). The following works, focused on those aspects. Although this issue has been only briefly described in section 5.2, these works were an important step towards the conception of the lexicon-free indexing algorithms presented in this thesis.

- **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. Word-Graph-Based Handwriting Keyword Spotting of Out-of-Vocabulary Queries. *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR)*. Stockholm, Sweden, 2014.

  In this work several strategies to estimate relevance probabilities of out-of-vocabulary keywords from word graphs were presented.

- **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. Word-Graph and Character-Lattice Combination for KWS in Handwritten Documents *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Stockholm, Sweden, 2014.

  Here, we suggested to combine lexicon-based and lexicon-free approaches in order to improve the overall performance of the KWS system.

- **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. A New Smoothing Method for Lexicon-Based Handwritten Text Keyword Spotting. *Proceedings of the 7th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*. Santiago de Compostela, Spain, 2015.

  A better approach to estimate the relevance probability of out-of-vocabulary keywords was presented in this work, which operated on the line posteriorgrams.

- **Joan Puigcerver**, Alejandro H. Toselli, and Enrique Vidal. Querying out-of-vocabulary words in lexicon-based keyword spotting. *Neural Computing and Applications*. 2017.

  The problem of out-of-vocabulary keywords in the field of KWS is reviewed here, together with a summary of the *smoothing* methods described in the previous works and additional results and insights.

10.3 Open source software

Besides from scientific publications, an important fraction of time during the span of the PhD was dedicated to write software that enabled us to be more productive and make our experiments reproducible. It is very common that many details necessary to replicate the experiments in a given article are not always described in it. Plus, even with all the details, rewriting the software necessary to repeat the experiments is a very time consuming (and not rewarded) task. We did not want this to happen with the results from this thesis and our papers.

In addition, we aimed to produce not only a well principled formulation of Keyword Spotting, but software that could be easily used, improved and adapted by (private or public) organizations that needed to incorporate Keyword Spotting technologies in their own products and services.

Hence, we decided to release as open source most of the software (at least the critical parts) developed in during this doctorate.

- **Laia**. We built this deep learning toolkit for handwritten text recognition in order to showcase the architecture that we designed for handwritten text modeling (i.e. section 4.3.5). Since then, this software has been widely used in our experiments, for both handwritten text recognition and keyword spotting.

  `https://github.com/jpuigcerver/Laia`

- **PyLaia**. This project was the successor of Laia and has been built to be more modular, easier to maintain, and to eliminate

some of the limits of the previous software. Many experiments done in this thesis can be directly replicated from the examples available in the software's repository.

`https://github.com/jpuigcerver/PyLaia`

- **Probabilistic indexing software**. The implementation of all the algorithms described in chapter 5, which are the practical core of this thesis can be found in this project, as well as many other utilities to perform HTR and KWS-related tasks.

`https://github.com/jpuigcerver/kaldi-lattice-utils`

10.4 Future work

Finally, we would like to identify future lines of research and development that we have already considered, or we think may be important.

10.4.1 Stochastic definitions of relevance

During all the thesis, we have assumed that the notion of relevance is of deterministic nature, given the content of the document and the query, and only depends on these variables. Under the Information Retrieval perspective, the challenging part was not in the notion of relevance, but in the uncertainty regarding the content of the documents.

This is quite the opposite to the applications that the Information Retrieval community typically tackles, where the content of their documents is perfectly known (e.g. the content of a Web page or a PDF document is given), but the concept of relevance has an stochastic nature (e.g. we would not benefit from a search engine that would simply retrieve all web pages that contain the text in the given query).

In order to improve the usefulness of our solutions, especially when dealing with large collections of documents and queries regarding high-frequency words, we need to investigate how to extend our algorithms and models to consider this new source of uncertainty, in a principled, robust, and efficient manner.

10.4.2 Better statistical models and training

Every improvement in the statistical models of the text content of the images will have an impact on the final KWS performance. Our architecture based on convolutional and recurrent neural networks has done an excellent job to model the text on *text lines* in several languages. However, we need to continue exploring for better architectures, or even radically different models. Especially if we aim to tackle more challenging documents with difficult layout analysis, line segmentation.

For instance, these later steps, which are currently performed as a preliminary step, can also be modeled as stochastic processes, from which probabilistic models can be constructed and integrated with our current solutions.

In addition, the data efficiency of our models could be further improved. Although the human effort needed to produce labeled training data is definitely worth it when indexing large collections of documents, we need to explore new ways of training our statistical models, perhaps even using unlabeled data, to make our solutions more data efficient, and to allow their use by low-budgeted institutions, or in collections with very few available data.

10.4.3 Probabilistic framework applied to other domains

Finally, we need to explore other domains in which our framework can be applied with minor (or no) modifications. For instance, we have already suggested one of this applications, Content-based Image Retrieval, in section 7.2. But we need to continue exploring this direction and many others, like the retrieval of music scores, cross-lingual information retrieval, multi-modal information retrieval, etc.

In addition, we would also like to use our probabilistic Keyword Spotting approach in order to aid the transcription process of documents, in an iterative way. For instance, a small portion of documents could be manually transcribed. Then, the rest of documents would be probabilistically indexed and the most likely word instances would be presented to the user for a verification in a single step. Finally, we

could re-train our models using the additional data and repeat the process.

A Corpora

A.1 Bentham

The Bentham collection includes several hundreds of manuscripts written by Jeremy Bentham, an influential English philosopher, jurist, and social reformer in the XVIII and XIX centuries; as well as many of his secretarial staff. A portion of this collection was first indexed in the Transcriptorium project[1]. The rest of the collection was processed during the READ project[2]. The entire collection is composed of around 100 000 images, but only a small portion of those have been manually transcribed.

Many HTR and KWS publications and competitions have used this collection, very often with different partitions of the database. In the following subsections we will detail the statistics of each of the partitions used in this thesis. Figure A.1 shows two pages and a few segmented text lines from this collection.

A.1.1 ICFHR-2014 Competition on HTR

One of the partitions of the Bentham collection that we use, for line-oriented KWS experiments, is the *ICFHR-2014 Handwritten Text Recognition on a tranScriptorium Dataset* competition (also shortened as ICFHR-2014 HTRtS). This encompasses a total of 433 pages (11 473 lines with nearly 110 000 running words and a vocabulary of more than 9 700 different words. Table A.1 shows the main statistics of this data partition, for the train, validation and test sets that we used.

[1] `http://transcriptorium.eu/`

[2] `https://read.transkribus.eu/`

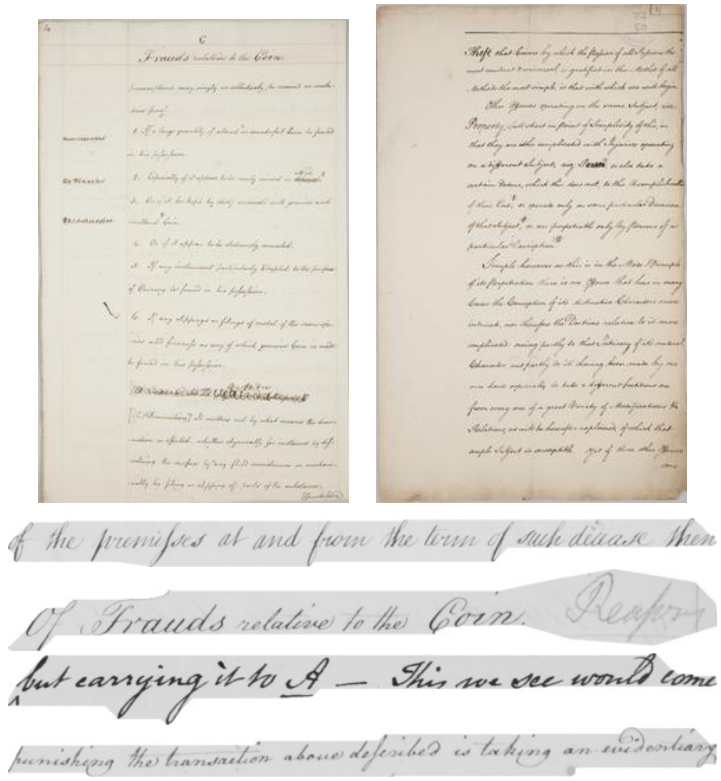


Figure A.1. Examples of a two pages and a few segmented lines from Bentham collection.

Table A.1. Statistics of the Bentham partition from the ICFHR-2014 Competition on HTR as used in our experiments.

	Train	Valid.	Test	Total
# Pages	350	50	33	433
# Lines	9 198	1 415	860	11 473
# Words	86 075	12 962	7 868	106 905
# Characters†	442,336	67,400	40,938	550,674
Alphabet size†	91	91	91	91
Lexicon size	8 658	2 709	1 946	9 716

† Not including the whitespace or other auxiliary symbols.

A.1.1.1 Line-level experiments

Several criteria can be assumed to select the keywords to be used in the assessment of a KWS system. Clearly, any given KWS system may perform better or worse depending on the query keywords it is tested with, and how these words are distributed in the test set. Of course, the larger the set of keywords, the more reliable the empirical results. Moreover, since our approach is aimed at indexing large collections, testing with a large set of keywords is mandatory.

In the experiments using this partition (i.e. section 8.7), we adopted all the words in the training set as queries. In this case, this gives a total number of 8 658 query keywords (the training lexicon size), and some of the statistics of these queries in the test set are shown in table A.2.

Table A.2. Statistics of the queries in the test set of the Bentham partition used in the ICFHR-2015 Competition on HTR.

# Queries	8 658
# Pairs	7 445 880
# Pertinent queries	1 487
# Pertinent pairs	6 727

A.1.1.2 Page-level experiments

For the multi-word experiments described in section 8.12, the KWS evaluation was performed at page level.

In this case, the query set was composed of 3 293 words whose frequency of occurrence in the training partition ranges from 2 to 10. This avoids including most stop words (generally with word frequencies greater than 10) and also many words that are unlikely to appear in the test partition. Despite the selection criteria adopted, only a relatively small subset of 674 words from the query set actually appear in any of the test lines. The corresponding query keywords are called "pertinent", while all the other queries are called "non-pertinent".

In addition, we also defined a pool of multi-word queries consisting of all the 5 420 278 pairs of different words . Similarly to the

previous case, not all the word-pairs in the multi-word query set are pertinent. The total (maximum) number of pertinent queries which can be composed for each query type are reported in table A.3, along with the other figures mentioned above. The table also reports the equivalent statistics for the number of pairs of (query, page) that can be formed.

Table A.3. Statistics of the query pools generated in the page-level multi-word experiments on the Bentham database.

	Query type	Total	Pertinent	$r_{\max}$
Queries	Single-word	3 293	674	3.89
	AND	5 420 278	11 784	458.97
	OR	5 420 278	1 992 007	1.72
Pairs	Single-word	108 669	836	128.99
	AND	178 869 174	12 438	14 379.86
	OR	178 869 174	2 739 674	64.29

We report results with two sets of queries generated from the previous pool. First, $r = 0$ is the set of (single or multi-word) queries that are all pertinent (i.e. queries for which at least one relevant page exists), and $r = 1$ is the set of queries with the same ratio of pertinent and non-pertinent queries. Thus, r denotes the ratio between non-pertinent and pertinent queries.

A.1.2 ICFHR-2014 COMPETITION ON KWS

One of the databases used in the *ICFHR-2014 Handwritten Keyword Spotting Competition*[3] [Pratikakis et al., 2014] was based on the Bentham collection.

Since the competition was targeted to researchers using training-free (and query-by-example) KWS approaches, no training data was officially released. However, the test set of 50 page was extracted from the 400 pages available for training and validation, released in the previous HTR competition. Thus, we excluded the 50 test pages from this 400 pages and used the manually segmented text lines, provided

[3] `http://vc.ee.duth.gr/H-KWS2014/`

by the HTR competition for training and validation purposes. The set of pages used for training and validation is not disjoint. We simply randomly sampled 10% of the text lines available for training to create the validation set.

Table A.4 shows the main statistics from the Bentham partition used in the ICFHR-2014 Competition on KWS. Notice that the alphabet size is much smaller than that of the HTR competition. This is because we converted all transcripts into lowercase, since the competition on KWS was scoped in a query-by-example setting.

Table A.4. Statistics of the Bentham partition from the ICFHR-2014 Competition on KWS as used in our experiments.

	Train	Valid.	Test	Total
# Pages	350	321	50	400
# Lines	8 337	973	1 303	10 613
# Words	70 223	8 187	10 786	89 196
# Characters†	308 507	35 946	46 644	391 097
Alphabet size†	60	60	60	60
Lexicon size	7 301	2 155	2 452	8 389

† Not including the whitespace or other auxiliary symbols.

In our experiments we only targeted the segmentation-free approach, by using both the provided manual segmentation (available from the ICFHR-2014 Competition on HTR), and fully automatic segmentation based on the Horizontal Projection Profile of the test pages [Likforman-Sulem et al., 2007].

This is essentially the same setting as in [Vidal et al., 2015], although there we used an additional text corpora to improve the word 2-gram language model, while in this thesis we only use the transcripts of the training images to train a character language model.

The query set is composed of 290 word images. The keywords depicted by these image correspond to words written more than 5 times in the test partition, and had a length greater than 6 characters. The query images come from several writers, have different writing styles, font sizes, and noise. Query images have to be spotted in images of

pages written by the same or different writers and/or exhibiting different writing variations. Some examples of the query images are shown in fig. A.2.

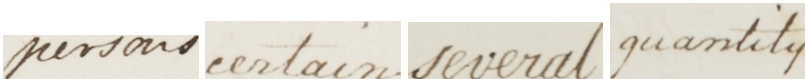


Figure A.2. Examples of the query images used in the ICFHR-2014 Competition on KWS.

Since we are in a segmentation-free scenario, there is an humongous amount of possible pairs of (query, bounding box) at page level. However, the test set only contains 2 850 reference bounding boxes for the 290 queries.

It is important to highlight that this database violates one of the assumptions made in our probabilistic formulation. In particular, some of the "relevant" bounding box instances in the test pages for a particular query, actually contain a different word written in them. For example, instances of words like "altogether" or "characteristic" were considered relevant for query images with the word "together" or "character" written on them, respectively.

A.1.3 ICDAR-2015 COMPETITION ON KWS

The *ICDAR-2015 Competition on Keyword Spotting for Handwritten Documents*[4] was was organized in order to provide a benchmark to fairly compare different KWS approaches.

The data used in this competition is also part of the Bentham collection, and is a subset of a similar competition, targeting HTR, that was launched within the ICDAR-2015 conference[5]. Some of the pages used in the ICFHR-2014 competitions on HTR and KWS were also used in here, as part of the train and validation sets.

The test set consists of 70 document images (never used in any previous competition), containing several difficult problems to be addressed, including writing from different authors, styles, font-sizes,

[4] `http://transcriptorium.eu/~icdar15kws/`

[5] `http://transcriptorium.eu/~htrcontest/contestICDAR2015/public_html/`

crossed-out words, etc. In addition, 10 document pages were provided for validation purposes. The main block of text was extracted from the original pages, so participants did not have to deal with marginalia.

An additional set of 423 document images, with manually segmented and transcribed text lines was also provided to the participants competing in the training-base track. The total number of text lines in this additional set is of 11 144. Since the 10 validation pages provided by the competition organizers (named *Valid.-I*) were not transcribed or segmented into lines, we used 10% of the training lines to tune some of the hyperparameters of our model (we call this set *Valid.-II*). Some statistics of the data, as used in our experiments, are reported in table A.5.

Table A.5. Statistics of the Bentham partition from the ICDAR-2015 Competition on KWS as used in our experiments.

	Valid.-I	Test	Train	Valid.-II	Total
# Pages	10	70	423	393	583
# Lines	—	—	10 013	1 131	11 144
# Words	~3 392	~16 840	84 071	9 359	~113 662
# Characters†	—	—	366 596	40 588	407 184
Alphabet size†	—	—	60	60	60
Lexicon size	—	—	8 012	2 321	8 449

† Not including the whitespace or other auxiliary symbols.

Notice that the full transcripts of the Valid.-I and test sets were not provided, thus the number of words is only approximated, based on the number of word images provided in the segmentation-based task. In addition, the provided transcripts were normalized by converting to lowercase and transliterating all the text.

Within the training-free track, two sub-tracks were defined: one following the query-by-string and the other the query-by-example paradigms (QbS and QbE, respectively). A total of 243 different keywords compose the the query set, with lenghts ranging from 6 to 15 characters. Each keyword is is represented by up to 6 different example images, making a total of 1 421 query images. All query keywords

were written at least 4 times in the test set. In total, there are 1 867 relevant bounding boxes to be retrieved for the 243 keywords (QbS scenario), and 11 006 for the 1 421 query images (QbE scenario).

Similarly to the ICFHR-2014 Competition on KWS, all casing instances of a word were considered equivalent (hence our lowercase normalization of the transcripts), but not plurals or derived words. For instance, "therefore" and "Therefore" are considered the same keyword, but not "according" and "accordingly", nor "instance" and "instances".

A.2 George Washington

The George Washington database (sometimes referred to as Washington, or simply GW), was created from the George Washington Papers at the Library of Congress[6]. These papers were written in the 18th century in English language, mainly by George Washington himself.

This collection has been used in seminal KWS publications, such as [Rath et al., 2004, Rath and Manmatha, 2007, Fischer et al., 2012, Frinken et al., 2012, Almazán et al., 2014], and it is one of the most used among the KWS researchers.

Note that different versions of this data set are used in the literature. We will perform experiments using two of these versions. Figure A.3 contains some examples of the page images extracted from this data set.

A.2.1 LINE-LEVEL EXPERIMENTS

The line-level partition is supported by the Research Group on Computer Vision and Artificial Intelligence (FKI) at the Institute of Computer Science an Applied Mathematics (IAM), in Bern (Switzerland). It was originally used in [Fischer et al., 2012], and it is based on the same 20 pages originally used by [Lavrenko et al., 2004], although the word and line segmentations, and the transcripts are of their own.

[6] https://www.loc.gov/collections/george-washington-papers/about-this-collection/

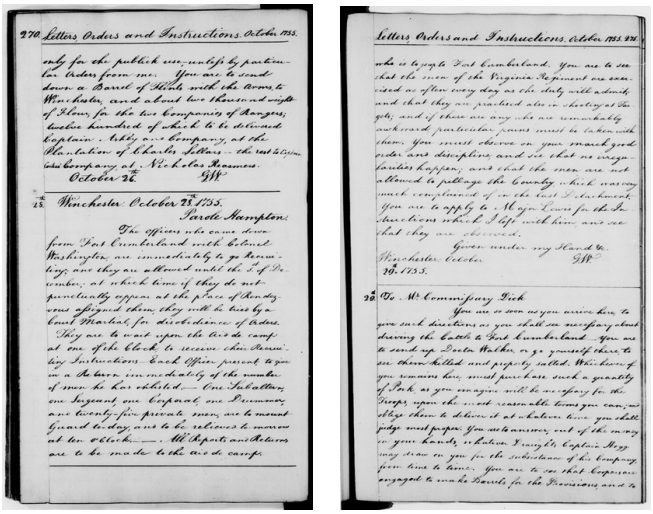


Figure A.3. Examples of page images extracted from the George Washington database.

A detrimental characteristic of this partition is that they only provide the binarized and normalized line and word images, which is a sub-optimal scenario for modern models, like CRNNs. Nevertheless, we use this partition because it has been widely used in previous line-level KWS works. A few of these lines are shown in fig. A.4.

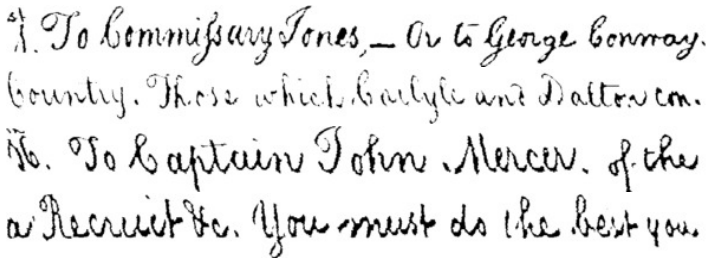


Figure A.4. Examples of a few normalized and binarized line images used from the line-level partition of the George Washington database.

The 20 pages in this partition encompass a total number of 656 text lines. Since this is a quite small data set, four-fold cross-validation (CV) partitions were defined. In each CV partition, 10 pages are used for training, 5 for validation and 5 for testing purposes. The average

statistics across the four folds of this partition are shown in table A.6. Notice that the average statistics in the validation and test sets are identical, due to the cross-validation procedure.

Table A.6. Statistics of the George Washington partition used in the line-level experiments.

	Train (avg.)	Valid./Test (avg.)	Total
# Lines	328.0	164.0	656
# Words	2 451.0	1 225.5	4 902
# Characters†	11 165.5	5 582.8	22 331
Alphabet size†	70.0	66.5	70
Lexicon size	898.0	538.8	1 466

† Not including the whitespace or other auxiliary symbols.

The alphabet size is different from that of the originally published in [Fischer et al., 2012], since they considered some frequent words (e.g. the signature "G.W.") as individual symbols of their alphabet.

To perform KWS experiments, the set of words in the training lexicon of each cross-validation fold with at least one occurrence in in the test set was used as the query set. Thus, all queries are pertinent in the test set, although not necessarily in the validation set of each fold. Table A.7 summarizes this statistics.

Table A.7. Statistics of the queries used in the IAM database.

	Valid. (avg.)	Test (avg.)
# Queries	104.5	104.5
# Pairs	17 139.0	17 144.5
# Pertinent queries	46.5	104.5
# Pertinent pairs	108.0	194.3

A.2.2 WORD-LEVEL EXPERIMENTS

Another popular partition of this data set is the one used in [Almazán et al., 2014], which is also based on [Lavrenko et al., 2004]. This partition is typically used by KWS works with word-segmentation needs,

such as the distance and PHOC-based approaches compared in sections 8.10 and 8.11.

Here, only a training and test sets are defined for each of the (four) cross-validation folds, which are not the same as in the previous partition. The number of words in each training and test partition is the same across the four folds.

Under this setting, all word images with more than one occurrence in the test set are used as queries, and the rest of instances in the test set have to be retrieved by the given KWS system. The transcriptions of the word images have been converted to lowercase, and punctuation marks have been removed, so that a pair of images containing the same word but with different cases is considered pertinent. For evaluation purposes, only pairs of distinct images are considered. Table A.8 summarizes the statistics of this partition of the George Washington database.

Table A.8. Statistics of the George Washington partition used in the word-level experiments.

	Train	Test	Total
# Words	3 645	1 215	4 860
# Pairs	—	1 471 365	—
# Pertinent pairs	—	16 842	—

A.3 IAM

The IAM database (or simply IAM) was compiled by the Research Group on Computer Vision and Artificial Intelligence (FKI) at the Institute of Computer Science an Applied Mathematics (IAM), in Bern (Switzerland). The database as of October 2002 is described in [Marti and Bunke, 2002]. It is publicly accessible[7] and freely available upon request for non-commercial research purposes.

The handwritten text images in this data set were produced by 500 different persons that transcribed fragments of text from the Lancaster-

[7] `http://www.fki.inf.unibe.ch/databases/iam-handwriting-database`

Oslo/Bergen (LOB) corpus [Johansson et al., 1978]. Text from the LOB corpus was split into fragments of 3–6 sentences, with at least 50 words each.

At the moment of transcribing the texts, no restrictions were imposed on the writing style or the type of pen and thus, very different styles and sizes are present. Images were scanned at 300dpi. Some pages and text lines are depicted in fig. A.5.

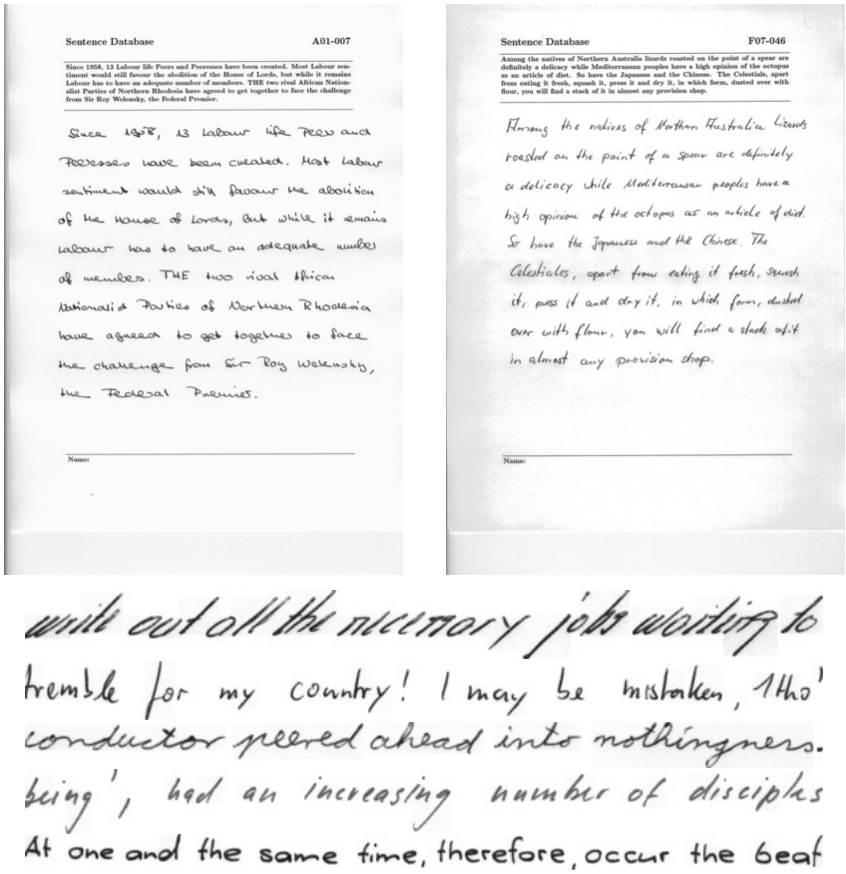


Figure A.5. Examples of a few pages and segmented lines extracted from the IAM database.

The official partition of the latest version of the database (version 3.0) consists of 747 training pages (6 161 text lines, from 283 writers),

two validation sets of 105 pages (900 lines, from 46 writers) and 115 pages (940 text lines, from 43 writers), respectively, and a test set of 232 pages (1 861 lines, from 128 writers). However, in this thesis, as well as in many other line-oriented KWS works, we use a different partition of the database, summarized in table A.9. Notice that this partition is different from the typically used in many recent HTR works (e.g. [Doetsch et al., 2014, Voigtlaender et al., 2016, Puigcerver, 2017]), although the training set is the same in all cases.

Table A.9. Statistics of the IAM partition used in the experiments.

	Train	Valid.	Test	Total
# Writers	283	25	25	333
# Pages	747	116	110	973
# Lines	6 161	920	929	8 010
# Words	53 767	8 599	8 315	70 681
# Characters†	275 466	41 031	40 445	364 952
Alphabet size†	78	74	71	78
Lexicon size	7 772	2 450	2 492	9792

† Not including the whitespace or other auxiliary symbols.

The query set employed in this thesis was also used in previous KWS publications (e.g. [Toselli et al., 2013, Toselli and Vidal, 2013, Puigcerver et al., 2015c] and many others). It consists of 3 421 words[8], most of which were extracted from the training set, excluding English stop words. The same set of queries is used in both the validation and test sets. Table A.10 shows some statistics of the query set, on both the validation and test sets.

Following [Bertolami and Bunke, 2008], we use three external text corpora to improve the n-gram language models. In particular, we use the the full LOB corpus (except the sentences that appear in the original test set of the IAM) [Johansson et al., 1978], the Brown corpus [Nelson and Kuçera, 1964] and the Wellington corpus [Holmes et al., 1998]. These three external corpora are typically used for experiments with the IAM database in HTR and KWS works. Table A.11 gives additional information of the three text corpora.

[8] `https://gist.github.com/jpuigcerver/a2c94a2196211aea6b2c08774f50f541`

Table A.10. Statistics of the queries used in the IAM database.

	Valid.	Test
# Queries	3 421	3 421
# Pairs	3 147 320	3 178 109
# Pertinent queries	1 053	1 101
# Pertinent pairs	1 849	1 919

Table A.11. Statistics of the external corpora used in the IAM database experiments.

	LOB	Brown	Wellington
# Words	1 119 904	1 045 213	1 144 401
# Characters†	5 803 916	5 582 023	6 055 820
Lexicon size	52 724	53 115	58 919

† Not including the whitespace or other auxiliary symbols.

The three of them contain full sentences much larger than the text lines of the IAM database (actually, equivalent to a full page of the IAM database). We split the sentences in these data sets so that the distribution of the number of words in each split "line" matches that of the lines in the IAM database, in order to improve the n-gram models.

A.4 Parzival

This database contains images of medieval German manuscripts, from the 13th century, written in Gothic script. The full Parzival collection consists of 16 books, although only a small subset has been manually annotated by German transcribers. Transcriptions are available for 47 pages, out of the 318 available in the full data set.

We use the partition released[9] by the Research Group on Computer Vision and Artificial Intelligence (FKI). Their release provides the page images in color, scanned at 300dpi, as well as binarized and normalized text line images.

[9] `http://www.fki.inf.unibe.ch/databases/iam-historical-document-database/parzival-database`

Figure A.6 shows some examples of the pages and (processed) text lines available in the database. Observe that the binarization and normalization carried out is quite severe and degrades the images significantly. Still, we have used these in our experiments.

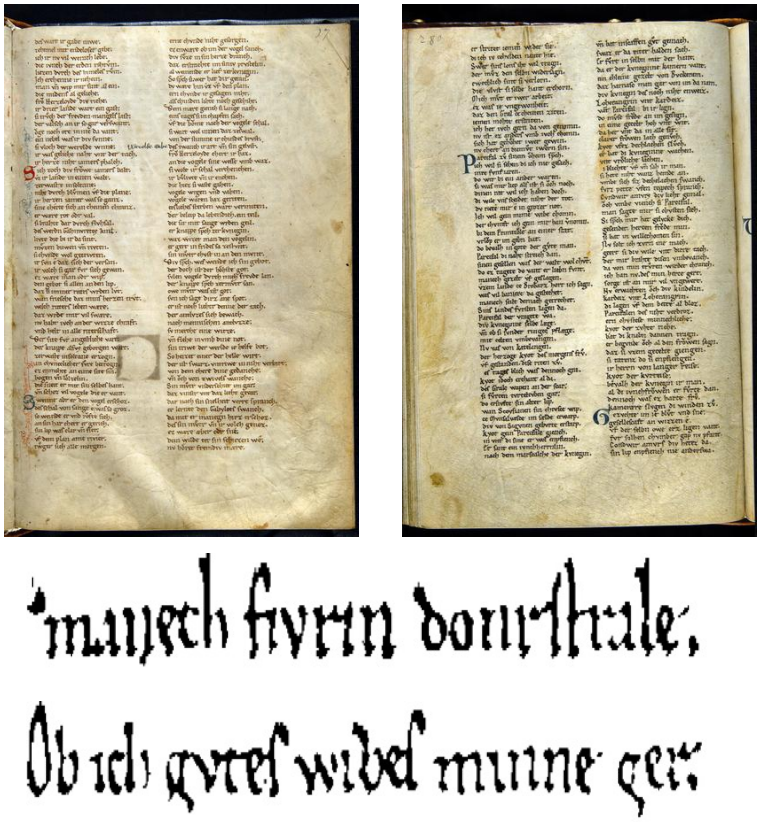


Figure A.6. Examples of a couple of pages and segmented (binarized and normalized) text lines extracted from the Parzival database.

The statistics of the partition can be found in table A.12. Notice that, on the one hand, the 47 pages are not distributed disjointly among the train, validation, and test sets, which makes this data set somehow easier than others (e.g. IAM). On the other hand, some of the characters present in the validation and test sets are not included in the training set, meaning that these characters are badly modeled (if considered at all) by the statistical models. This is the

same partition used in many seminal KWS papers such as [Fischer et al., 2012, Frinken et al., 2012, Toselli et al., 2016a].

Table A.12. Statistics of the Parzival partition used in the experiments.

	Train	Valid.	Test	Total
# Pages	47	47	47	47
# Lines	2 237	912	1 328	4 477
# Words	14 042	5 671	8 407	28 120
# Characters†	64 436	26 211	38 339	105 343
Alphabet size†	89	79	81	95
Lexicon size	3 221	1 753	2 305	4 936

† Not including the whitespace or other auxiliary symbols.

We use the official query set released by the FKI group. This consists of 1 217 different keywords, which are used in both validation and test. The statistics of the query set are summarized in table A.13.

Table A.13. Statistics of the query set used in the Parzival database.

	Valid.	Test
# Queries	1 217	1 217
# Pairs	1 109 904	1 616 176
# Pertinent queries	718	1 217
# Pertinent pairs	3 533	5 764

Notice that all the queries are pertinent in the test set, meaning that at least one test line is relevant for each query keyword.

A.5 Plantas

The *"Historia de las plantas"* manuscript (shorted as Plantas), produced in the XVII century by the Spanish botanist Bernardo de Cienfuego, is a source of valuable information for researchers interested in the botanical knowledge of the modern era.

In this work, only the first volume of Plantas was used for experimentation. Table A.14 shows basic statistics of the Plantas data set.

We excluded empty page images or those containing only drawings. Full details about this database are found in [Toselli et al., 2018a].

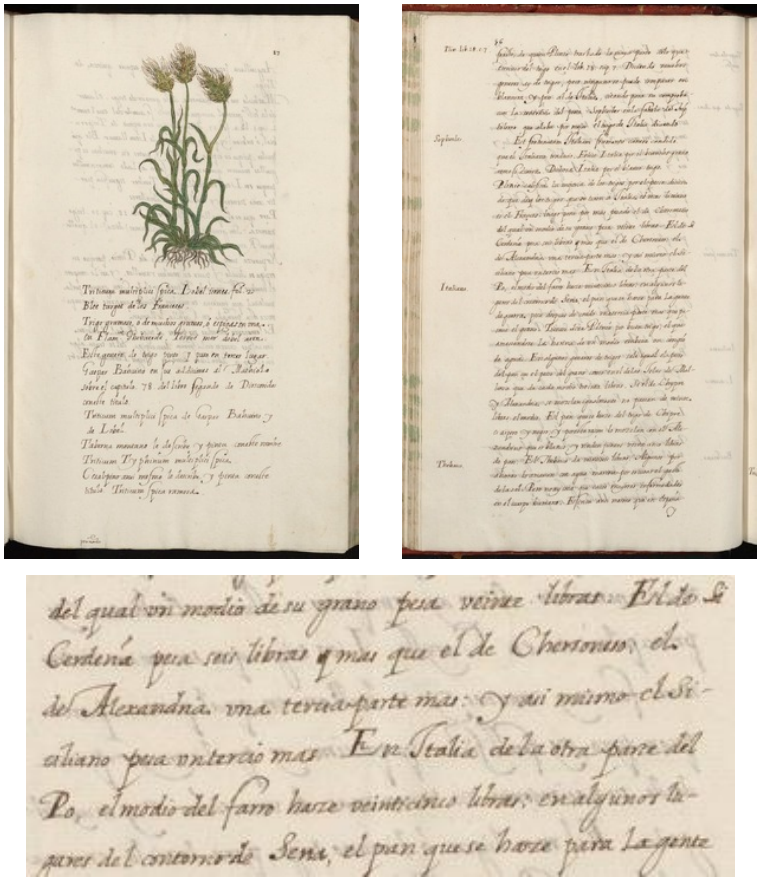


Figure A.7. Examples of a couple of pages and some text lines extracted from the Plantas dabatase.

It is important to mention that reference transcripts of Plantas are provided in two different versions: diplomatic and modernized. We have only used the diplomatic transcripts, which typographically transcribe as accurately as possible all significant characteristics of the original manuscript, including spelling, punctuation marks, abbreviations, crossed-out and inserted text, and other text alterations.

Table A.14. Statistics of the Plantas partition used in the experiments.

	Train	Valid.	Test	Total
# Pages	224	40	607	871
# Lines	6 788	955	11 801	19 544
# Words	67 912	9 753	117 029	194 694
Alphabet size†	73	73	73	73
Lexicon size	10 861	2 198	14 018	20 834

† Not including the whitespace or other auxiliary symbols.

Table A.15. Statistics of the total amount of text data used with the Plantas database.

# Words	Lexicon size
70 557	11 890

In addition to the transcripts from the training images, in order to estimate n-gram language models, we included the text from the glossaries and indexes. We did so to reduce the number of out-of-vocabulary words in the validation and test sets, since we performed only lexicon-based experiments on this data set. Table A.15 contains some basic statistics of the total amount of text used to train the language model.

The query set for Plantas was defined on the extended lexicon (last column in table A.15), excluding 958 words corresponding to numbers. This gives a total number of 10 932 different query keywords. The same set of queries was used for both validation and test experiments. Table A.16 includes some relevant statistics about the query set (on the test partition).

Table A.16. Statistics of the query set used in the test partition of the Plantas database.

# Queries	10 932
# Pairs	129 008 532
# Pertinent queries	4 888
# Pertinent pairs	86 304

List of Algorithms

List of Figures

List of Tables

Bibliography

[Aldavert et al., 2015] Aldavert, D., Rusiñol, M., Toledo, R., and Lladós, J. (2015). A study of Bag-of-Visual-Words representations for handwritten keyword spotting. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(3):223–234.

[Allauzen and Mohri, 2002] Allauzen, C. and Mohri, M. (2002). On the Determinizability of Weighted Automata and Transducers. In *In Proceedings of the workshop Weighted Automata: Theory and Applications (WATA*.

[Almazán et al., 2012] Almazán, J., Gordo, A., Fornés, A., and Valveny, E. (2012). Efficient Exemplar Word Spotting. In *Proceedings of the British Machine Vision Conference*, pages 67.1–67.11. BMVA Press.

[Almazán et al., 2014] Almazán, J., Gordo, A., Fornés, A., and Valveny, E. (2014). Word Spotting and Recognition with Embedded Attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566.

[Asadi et al., 1991] Asadi, A., Schwartz, R., and Makhoul, J. (1991). Automatic modeling for adding new words to a large-vocabulary continuous speech recognition system. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 305–308.

[B. et al., 2014] B., Bluche, T., Knibbe, M., Benzeghiba, M. F., Messina, R., Louradour, J., and Kermorvant, C. (2014). The A2iA Multilingual Text Recognition System at the Second Maurdor Evaluation. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 297–302.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

[Bahl et al., 1986] Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52.

[Baum and Eagon, 1967] Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363.

[Baum et al., 1970] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded Up Robust Features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bazzi, 2002] Bazzi, I. (2002). *Modelling out-of-vocabulary words for robust speech recognition*. PhD thesis, Massachusetts Institute of Technology.

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554, USA. Curran Associates Inc.

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

[Biau and Devroye, 2015] Biau, G. and Devroye, L. (2015). Weighted k-nearest neighbor density estimates. In *Lectures on the Nearest Neighbor Method*, pages 43–51. Springer.

[Bird et al., 1996] Bird, C. L., Chapman, S. G., and Ibbotson, J. B. (1996). Content-driven navigation of large databases. In *IEEE Colloquium on Intelligent Image Databases*, pages 13/1–13/5.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

[Bluche, 2015] Bluche, T. (2015). *Deep Neural Networks for Large Vocabulary Handwritten Text Recognition*. PhD thesis, Université Paris Sud - Paris XI.

[Bluche and Messina, 2017] Bluche, T. and Messina, R. (2017). Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 646–651.

[Bourlard and Wellekens, 1990] Bourlard, H. and Wellekens, C. J. (1990). Links between markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

[Bridle, 1990] Bridle, J. S. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In Soulié, F. F. and Hérault, J., editors, *Neurocomputing*, pages 227–236, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Brown et al., 1990] Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A Statistical Approach to Machine Translation. *Comput. Linguist.*, 16(2):79–85.

[Brown and Frederking, 1995] Brown, R. and Frederking, R. (1995). Applying statistical English language modeling to symbolic machine translation. In *Proceedings of the 6th International Conference*

on Theoretical and Methodological Issues in Machine Translation (TMI-95), pages 221–239.

[Bukhari et al., 2012] Bukhari, S. S., Breuel, T. M., Asi, A., and El-Sana, J. (2012). Layout Analysis for Arabic Historical Document Images Using Machine Learning. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 639–644.

[Cambria et al., 2013] Cambria, E., Schuller, B., Xia, Y., and Havasi, C. (2013). New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2):15–21.

[Chen et al., 1993] Chen, F. R., Wilcox, L. D., and Bloomberg, D. S. (1993). Word spotting in scanned images using hidden Markov models. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 1–4.

[Chen et al., 2017] Chen, K., Seuret, M., Hennebert, J., and Ingold, R. (2017). Convolutional Neural Networks for Page Segmentation of Historical Document Images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 965–970.

[Chen et al., 2015] Chen, K., Seuret, M., Liwicki, M., Hennebert, J., and Ingold, R. (2015). Page segmentation of historical document images with convolutional autoencoders. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1011–1015.

[Chen and Goodman, 1999] Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078.

[Cruz and Terrades, 2018] Cruz, F. and Terrades, O. R. (2018). A probabilistic framework for handwritten text line segmentation. *CoRR*, abs/1805.02536.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

[Defferrard et al., 2016] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

[Dey et al., 2016] Dey, S., Nicolaou, A., Lladós, J., and Pal, U. (2016). Local Binary Pattern for Word Spotting in Handwritten Historical Document. In Robles-Kelly, A., Loog, M., Biggio, B., Escolano, F., and Wilson, R., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 574–583. Springer International Publishing.

[Diem et al., 2017] Diem, M., Kleber, F., Fiel, S., Grüning, T., and Gatos, B. (2017). cBAD: ICDAR2017 Competition on Baseline Detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1355–1360.

[Digalakis et al., 2000] Digalakis, V., Tsakalidis, S., Harizakis, C., and Neumeyer, L. (2000). Efficient speech recognition using subvector quantization and discrete-mixture HMMs. *Computer Speech & Language*, 14(1):33–46.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numer. Math.*, 1(1):269–271.

[Doetsch et al., 2014] Doetsch, P., Kozielski, M., and Ney, H. (2014). Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 279–284.

[D'Orazio, 2012] D'Orazio, D. (2012). Oxford and Vatican libraries to digitize 1.5 million pages of ancient texts. *The Verge*. Visited on 17-Jun-2018.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal Machine Learning Research*, 12:2121–2159.

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience, New York, NY, USA, 2nd edition edition.

[Eilenberg, 1974] Eilenberg, S. (1974). *Automata, Languages, and Machines*. Academic Press, Inc., Orlando, FL, USA.

[El-Yacoubi et al., 1999] El-Yacoubi, A., Sabourin, R., Suen, C. Y., and Gilloux, M. (1999). An HMM-Based Approach for Off-Line Unconstrained Handwritten Word Modeling and Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8):752–760.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

[Espana-Boquera et al., 2011] Espana-Boquera, S., Castro-Bleda, M. J., Gorbe-Moya, J., and Zamora-Martinez, F. (2011). Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779.

[Fischer et al., 2013] Fischer, A., Frinken, V., Bunke, H., and Suen, C. Y. (2013). Improving HMM-Based Keyword Spotting with Character Language Models. In *2013 12th International Conference on Document Analysis and Recognition*, pages 506–510.

[Fischer et al., 2012] Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2012). Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7):934 – 942. Special Issue on Awards from ICPR 2010.

[Frinken et al., 2012] Frinken, V., Fischer, A., Manmatha, R., and Bunke, H. (2012). A Novel Word Spotting Method Based on Recurrent Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):211–224.

[Fréchet, 1935] Fréchet, M. (1935). Généralisation du théorème des probabilités totales. *Fundamenta Mathematicae*, 25(1):379–387.

[Fukushima and Miyake, 1982] Fukushima, K. and Miyake, S. (1982). Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. In Amari, S.-i. and Arbib, M. A., editors, *Competition and Cooperation in Neural Nets*, pages 267–285, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Garz et al., 2012] Garz, A., Fischer, A., Sablatnig, R., and Bunke, H. (2012). Binarization-Free Text Line Segmentation for Historical Documents Based on Interest Point Clustering. In *2012 10th IAPR International Workshop on Document Analysis Systems*, pages 95–99.

[Gers et al., 2000] Gers, F. A., Schmidhuber, J. A., and Cummins, F. A. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Comput.*, 12(10):2451–2471.

[Giménez and Juan, 2009] Giménez, A. and Juan, A. (2009). Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition. In *2009 10th International Conference on Document Analysis and Recognition*, pages 896–900.

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

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

[Graves et al., 2004] Graves, A., Eck, D., Beringer, N., and Schmidhuber, J. (2004). Biologically Plausible Speech Recognition with LSTM

Neural Nets. In Ijspeert, A. J., Murata, M., and Wakamiya, N., editors, *Biologically Inspired Approaches to Advanced Information Technology*, pages 127–136, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Graves et al., 2006] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA. ACM.

[Graves et al., 2013a] Graves, A., Jaitly, N., and Mohamed, A. (2013a). Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278.

[Graves et al., 2013b] Graves, A., Mohamed, A., and Hinton, G. (2013b). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

[Graves and Schmidhuber, 2009] Graves, A. and Schmidhuber, J. (2009). Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc.

[Grüning et al., 2017] Grüning, T., Leifert, G., Strauss, T., and Labahn, R. (2017). A Robust and Binarization-Free Approach for Text Line Detection in Historical Documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 236–241.

[Guérin, 1896] Guérin, P. (1896). *Recueil des documents concernant le Poitou contenus dans les registres de la chancellerie de France*, volume 11. Archives historiques du Poitou.

[Gómez et al., 2017] Gómez, L., Rusiñol, M., and Karatzas, D. (2017). LSDE: Levenshtein Space Deep Embedding for Query-by-String Word Spotting. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 499–504.

[Günter and Bunke, 2004] Günter, S. and Bunke, H. (2004). HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37(10):2069–2079.

[Hahnloser et al., 2000] Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951.

[Hazen et al., 2009] Hazen, T. J., Shen, W., and White, C. (2009). Query-by-example spoken term detection using phonetic posteriorgram templates. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 421–426.

[He et al., 2017] He, D., Cohen, S., Price, B., Kifer, D., and Giles, C. L. (2017). Multi-Scale Multi-Task FCN for Semantic Page Segmentation and Table Detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 254–261.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[Holmes et al., 1998] Holmes, J., Johnson, G., and Vine, B. (1998). *Guide to the Wellington corpus of spoken New Zealand English*. School of Linguistics and Applied Language Studies, Victoria University of Wellington Wellington.

[Hopcroft et al., 2006] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition.

[Hu et al., 2014] Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc.

[Huang et al., 1993] Huang, X., Hon, H., Hwang, M., and Lee, K. (1993). A comparative study of discrete, semicontinuous, and continuous hidden Markov models. *Computer Speech & Language*, 7(4):359–368.

[Hull, 1998] Hull, J. J. (1998). Document image skew detection: Survey and annotated bibliography. In *Document Analysis Systems II. Word Scientific*, pages 40–64. World Scientific.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

[Jaderberg et al., 2014] Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Deep Features for Text Spotting. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 512–528. Springer International Publishing.

[Jelinek, 1976] Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

[Ji et al., 2013] Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231.

[Jimenez, 2007] Jimenez, C. (2007). British Library books go digital. *BBC News*. Visited on 17-Jun-2018.

[Johansson et al., 1978] Johansson, S., Leech, G., and Goodluck, H. (1978). Manual of Information to Accompany the Lancaster-Olso/Bergen Corpus of British English, for Use with Digital Computers. Technical report, Department of English, University of Oslo.

[Juang et al., 1997] Juang, B.-H., Hou, W., and Lee, C.-H. (1997). Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265.

[Katz, 1987] Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

[Keaton et al., 1997] Keaton, P., Greenspan, H., and Goodman, R. (1997). Keyword spotting for cursive document retrieval. In *Document Image Analysis, 1997. (DIA '97) Proceedings., Workshop on*, pages 74–81.

[Khoubyari and Hull, 1993] Khoubyari, S. and Hull, J. J. (1993). Keyword location in noisy document image. In *2nd Annual Symposium on Document Analysis and Information Retrieval*, pages 217–231.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

[Kneser and Ney, 1995] Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184.

[Kozielski et al., 2012] Kozielski, M., Forster, J., and Ney, H. (2012). Moment-Based Image Normalization for Handwritten Text Recognition. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 256–261.

[Kozielski et al., 2013] Kozielski, M., Rybach, D., Hahn, S., Schlüter, R., and Ney, H. (2013). Open vocabulary handwriting recognition using combined word- level and character-level language models. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 8257–8261.

[Kumar and Govindaraju, 2014] Kumar, G. and Govindaraju, V. (2014). A Bayesian Approach to Script Independent Multilingual Keyword Spotting. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 357–362.

[Kundu et al., 1989] Kundu, A., He, Y., and Bahl, P. (1989). Recognition of handwritten word: First and second order hidden Markov model based approach. *Pattern Recognition*, 22(3):283–297.

[Lavrenko et al., 2004] Lavrenko, V., Rath, T. M., and Manmatha, R. (2004). Holistic word recognition for handwritten historical documents. In *First International Workshop on Document Image Analysis for Libraries, 2004. Proceedings.*, pages 278–287.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

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

[Ljolje et al., 1999] Ljolje, A., Pereira, F., and Riley, M. (1999). Efficient general lattice generation and rescoring. In *Proceedings of the Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99)*, pages 1251–1254.

[Louloudis et al., 2009] Louloudis, G., Gatos, B., Pratikakis, I., and Halatsis, C. (2009). Text Line and Word Segmentation of Handwritten Documents. *Pattern Recogn.*, 42(12):3169–3183.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157.

[Luján-Mares et al., 2008] Luján-Mares, M., Tamarit, V., Alabau, V., Martínez-Hinarejos, C. D., Pastor, M., Sanchis, A., and Toselli, A. H. (2008). iATROS: A speech and handwritting recognition system. In *V Jornadas en Tecnologıas del Habla (VJTH'2008)*, pages 75–78.

[Maas et al., 2013] Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 30, page 3.

[MacLeod, 2004] MacLeod, R. (2004). *The Library of Alexandria: centre of learning in the ancient world*, chapter Part III, pages 70–74. I.B. Tauris.

[Madrigal, 2013] Madrigal, A. C. (2013). Norway Decided to Digitize All the Norwegian Books. *The Atlantic*. Visited on 16-Jun-2018.

[Manmatha et al., 1996a] Manmatha, R., Han, C., and Riseman, E. M. (1996a). Word spotting: a new approach to indexing handwriting. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 631–637.

[Manmatha et al., 1996b] Manmatha, R., Han, C., Riseman, E. M., and Croft, W. B. (1996b). Indexing Handwriting Using Word Matching. In *Proceedings of the First ACM International Conference on Digital Libraries*, DL '96, pages 151–159, New York, NY, USA. ACM.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[Mao et al., 2003] Mao, S., Rosenfeld, A., and Kanungo, T. (2003). Document structure analysis algorithms: a literature survey. volume 5010, pages 5010 – 5010 – 11.

[Marti and Bunke, 2000] Marti, U. V. and Bunke, H. (2000). Handwritten sentence recognition. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 463–466 vol.3.

[Marti and Bunke, 2001a] Marti, U. V. and Bunke, H. (2001a). On the influence of vocabulary size and language models in unconstrained handwritten text recognition. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 260–265.

[Marti and Bunke, 2001b] Marti, U. V. and Bunke, H. (2001b). Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. In *Hidden Markov Models*, pages 65–90. World Scientific.

[Marti and Bunke, 2002] Marti, U.-V. and Bunke, H. (2002). The IAM-database: an English sentence database for offline handwriting

recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[McLuhan, 1962] McLuhan, M. (1962). *The Gutenberg Galaxy: The Making of Typographic Man*. University of Toronto Press.

[Miller et al., 2007] Miller, D. R., Kleber, M., Kao, C.-L., Kimball, O., Colthurst, T., Lowe, S. A., Schwartz, R. M., and Gish, H. (2007). Rapid and accurate spoken term detection. In *8th Annual Conference of the International Speech Communication Association*.

[Mohri, 2002] Mohri, M. (2002). Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

[Mohri, 2004] Mohri, M. (2004). *Weighted Finite-State Transducer Algorithms. An Overview*, pages 551–563. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Mohri et al., 2008] Mohri, M., Pereira, F., and Riley, M. (2008). *Speech Recognition with Weighted Finite-State Transducers*, pages 559–584. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Mohri et al., 1996] Mohri, M., Pereira, F. C. N., and Riley, M. (1996). Weighted Automata in Text and Speech Processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96). Workshop on Extended finite state models of language.*, pages 1–5.

[Mohri and Riley, 2002] Mohri, M. and Riley, M. (2002). An Efficient Algorithm for the N-Best-Strings Problem. In *Proceedings of the International Conference on Spoken Language Processing 2002 (ICSLP '02)*.

[Mondal et al., 2016] Mondal, T., Ragot, N., Ramel, J.-Y., and Pal, U. (2016). Flexible Sequence Matching technique: An effective learning-free approach for word spotting. *Pattern Recognition*, 60:596–612.

[Mondal et al., 2018] Mondal, T., Ragot, N., Ramel, J.-Y., and Pal, U. (2018). Comparative study of conventional time series matching techniques for word spotting. *Pattern Recognition*, 73:47–64.

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. The MIT Press, Cambridge, MA.

[Nelson and Kuçera, 1964] Nelson, F. W. and Kuçera, H. (1964). Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers. Technical report, Department of Linguistics, Brown University.

[Ortmanns et al., 1997] Ortmanns, S., Ney, H., and Aubert, X. (1997). A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech & Language*, 11(1):43–72.

[Paniagua, 2018] Paniagua, E. (2018). Así se digitaliza la Biblioteca Nacional de España. *El País*. Visited on 16-Jun-2018.

[Papandreou et al., 2016] Papandreou, A., Gatos, B., and Zagoris, K. (2016). An Adaptive Zoning Technique for Word Spotting Using Dynamic Time Warping. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 387–392.

[Pastor et al., 2004] Pastor, M., Toselli, A., and Vidal, E. (2004). Projection Profile Based Algorithm for Slant Removal. In Campilho, A. and Kamel, M., editors, *Image Analysis and Recognition*, pages 183–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Perronnin et al., 2009] Perronnin, F., Liu, Y., and Renders, J. (2009). A family of contextual measures of similarity between distributions with application to image retrieval. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2358–2365.

[Pesch et al., 2012] Pesch, H., Hamdani, M., Forster, J., and Ney, H. (2012). Analysis of Preprocessing Techniques for Latin Handwriting Recognition. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 280–284.

[Povey et al., 2011] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y.,

Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

[Povey et al., 2012] Povey, D., Hannemann, M., Boulianne, G., Burget, L., Ghoshal, A., Janda, M., Karafiát, M., Kombrink, S., Motlíček, P., Qian, Y., Riedhammer, K., Veselý, K., and Vu, N. T. (2012). Generating exact lattices in the WFST framework. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4213–4216.

[Povey and Woodland, 2002] Povey, D. and Woodland, P. C. (2002). Minimum Phone Error and I-smoothing for improved discriminative training. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 105–108.

[Pratikakis et al., 2014] Pratikakis, I., Zagoris, K., Gatos, B., Louloudis, G., and Stamatopoulos, N. (2014). ICFHR 2014 Competition on Handwritten Keyword Spotting (H-KWS 2014). In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 814–819.

[Puigcerver, 2017] Puigcerver, J. (2017). Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 67–72.

[Puigcerver et al., 2014a] Puigcerver, J., Toselli, A. H., and Vidal, E. (2014a). Word-Graph and Character-Lattice Combination for KWS in Handwritten Documents. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 181–186.

[Puigcerver et al., 2014b] Puigcerver, J., Toselli, A. H., and Vidal, E. (2014b). Word-Graph-Based Handwriting Keyword Spotting of Out-of-Vocabulary Queries. In *2014 22nd International Conference on Pattern Recognition*, pages 2035–2040.

[Puigcerver et al., 2015a] Puigcerver, J., Toselli, A. H., and Vidal, E. (2015a). A New Smoothing Method for Lexicon-Based Handwritten Text Keyword Spotting. In Paredes, R., Cardoso, J. S., and

Pardo, X. M., editors, *Pattern Recognition and Image Analysis*, pages 23–30, Cham. Springer International Publishing.

[Puigcerver et al., 2015b] Puigcerver, J., Toselli, A. H., and Vidal, E. (2015b). ICDAR2015 Competition on Keyword Spotting for Handwritten Documents. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1176–1180.

[Puigcerver et al., 2015c] Puigcerver, J., Toselli, A. H., and Vidal, E. (2015c). Probabilistic interpretation and improvements to the HMM-filler for handwritten keyword spotting. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 731–735.

[Puigcerver et al., 2017] Puigcerver, J., Toselli, A. H., and Vidal, E. (2017). Querying out-of-vocabulary words in lexicon-based keyword spotting. *Neural Computing and Applications*, 28(9):2373–2382.

[Rabin and Scott, 1959] Rabin, M. O. and Scott, D. (1959). Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, 3(2):114–125.

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

[Rath et al., 2004] Rath, T. M., Manmatha, R., and Lavrenko, V. (2004). A Search Engine for Historical Manuscript Images. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 369–376, New York, NY, USA. ACM.

[Retsinas et al., 2016] Retsinas, G., Louloudis, G., Stamatopoulos, N., and Gatos, B. (2016). Keyword Spotting in Handwritten Documents Using Projections of Oriented Gradients. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 411–416.

[Riesenhuber and Poggio, 1999] Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407.

[Robertson, 1977] Robertson, S. E. (1977). The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304.

[Rodriguez-Serrano et al., 2009] Rodriguez-Serrano, J. A., Perronnin, F., Llados, J., and Sanchez, G. (2009). A similarity measure between vector sequences with application to handwritten word image retrieval. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1722–1729.

[Rohlicek et al., 1989] Rohlicek, J. R., Russell, W., Roukos, S., and Gish, H. (1989). Continuous hidden Markov modeling for speaker-independent word spotting. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 627–630.

[Rose, 1995] Rose, R. (1995). Keyword detection in conversational speech utterances using hidden Markov model based continuous speech recognition. *Computer Speech & Language*, 9(4):309–333.

[Rose and Paul, 1990] Rose, R. C. and Paul, D. B. (1990). A hidden Markov model based keyword recognition system. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 129–132.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, pages 65–386.

[Rothacker et al., 2017] Rothacker, L., Sudholt, S., Rusakov, E., Kasperidus, M., and Fink, G. A. (2017). Word hypotheses for segmentation-free word spotting in historic document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1174–1179.

[Rothacker et al., 2012] Rothacker, L., Vajda, S., and Fink, G. A. (2012). Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 149–154.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[Rusiñol et al., 2011] Rusiñol, M., Aldavert, D., Toledo, R., and Lladós, J. (2011). Browsing Heterogeneous Document Collections by a Segmentation- Free Word Spotting Method. In *2011 International Conference on Document Analysis and Recognition*, pages 63–67.

[Saabni et al., 2014] Saabni, R., Asi, A., and El-Sana, J. (2014). Text line extraction for historical document images. *Pattern Recognition Letters*, 35:23–33. Frontiers in Handwriting Processing.

[Sanderson and Croft, 2012] Sanderson, M. and Croft, W. B. (2012). The History of Information Retrieval Research. *Proceedings of the IEEE*, 100(Special Centennial Issue):1444–1451.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Senior and Robinson, 1998] Senior, A. W. and Robinson, A. J. (1998). An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309–321.

[Sfikas et al., 2016] Sfikas, G., Retsinas, G., and Gatos, B. (2016). Zoning Aggregated Hypercolumns for Keyword Spotting. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 283–288.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

[Shannon, 1951] Shannon, C. E. (1951). Prediction and Entropy of Printed English. *Bell System Technical Journal*, 30(1):50–64.

[Shi et al., 2017] Shi, B., Bai, X., and Yao, C. (2017). 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.

[Siegelmann, 1995] Siegelmann, H. T. (1995). Computation Beyond the Turing Limit. *Science*, 268(5210):545–548.

[Silverman, 1986] Silverman, B. W. (1986). *Density estimation for statistics and data analysis*, volume 26. CRC press.

[Smeulders et al., 2000] Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-Based Image Retrieval at the End of the Early Years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380.

[Sudholt and Fink, 2016] Sudholt, S. and Fink, G. A. (2016). 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.

[Sudholt and Fink, 2017] Sudholt, S. and Fink, G. A. (2017). Evaluating Word String Embeddings and Loss Functions for CNN-Based Word Spotting. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 493–498.

[Sudholt and Fink, 2018] Sudholt, S. and Fink, G. A. (2018). Attribute CNNs for word spotting in handwritten documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 21(3):199–218.

[Sundermeyer et al.,] Sundermeyer, M., Schlüter, R., and Ney, H. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

[Szoke et al., 2008] Szoke, I., Burget, L., Cernocky, J., and Fapso, M. (2008). Sub-word modeling of out of vocabulary words in spoken term detection. In *2008 IEEE Spoken Language Technology Workshop*, pages 273–276.

[Sánchez et al., 2014] Sánchez, J. A., Romero, V., Toselli, A. H., and Vidal, E. (2014). ICFHR2014 Competition on Handwritten Text Recognition on Transcriptorium Datasets (HTRtS). In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 785–790.

[Tarafdar et al., 2013] Tarafdar, A., Pal, U., Roy, P. P., Ragot, N., and Ramel, J. (2013). A Two-Stage Approach for Word Spotting in Graphical Documents. In *2013 12th International Conference on Document Analysis and Recognition*, pages 319–323.

[Tay et al., 2001] Tay, Y. H., Lallican, P. M., Khalid, M., Knerr, S., and Viard-Gaudin, C. (2001). An analytical handwritten word recognition system with word-level discriminant training. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 726–730.

[Terasawa and Tanaka, 2009] Terasawa, K. and Tanaka, Y. (2009). Slit Style HOG Feature for Document Image Word Spotting. In *2009 10th International Conference on Document Analysis and Recognition*, pages 116–120.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

[Toselli et al., 2013] Toselli, A., Vidal, E., Romero, V., and Frinken, V. (2013). Word-graph based keyword spotting and indexing of handwritten document images. Technical report, Universitat Politècnica de València.

[Toselli et al., 2004] Toselli, A. H., Juan, A., González, J., Salvador, I., Vidal, E., Casacuberta, F., Keysers, D., and Ney, H. (2004). Integrated handwriting recognition and interpretation using finite-state models. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(04):519–539.

[Toselli et al., 2018a] Toselli, A. H., Leiva, L. A., Bordes-Cabrera, I., Hernández-Tornero, C., Bosch, V., and Vidal, E. (2018a). Transcribing a 17th-century botanical manuscript: Longitudinal evaluation

of document layout detection and interactive transcription. *Digital Scholarship in the Humanities*, 33(1):173–202.

[Toselli et al., 2015] Toselli, A. H., Puigcerver, J., and Vidal, E. (2015). Context-aware lattice based filler approach for key word spotting in handwritten documents. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 736–740.

[Toselli et al., 2016a] Toselli, A. H., Puigcerver, J., and Vidal, E. (2016a). Two Methods to Improve Confidence Scores for Lexicon-Free Word Spotting in Handwritten Text. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 349–354.

[Toselli et al., 2010] Toselli, A. H., Romero, V., Pastor, M., and Vidal, E. (2010). Multimodal interactive transcription of text images. *Pattern Recognition*, 43(5):1814 – 1825.

[Toselli and Vidal, 2013] Toselli, A. H. and Vidal, E. (2013). Fast HMM-Filler Approach for Key Word Spotting in Handwritten Documents. In *2013 12th International Conference on Document Analysis and Recognition*, pages 501–505.

[Toselli and Vidal, 2015] Toselli, A. H. and Vidal, E. (2015). Handwritten text recognition results on the Bentham collection with improved classical N-gram-HMM methods. In *Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing*, pages 15–22. ACM.

[Toselli et al., 2018b] Toselli, A. H., Vidal, E., Puigcerver, J., and Noya-García, E. (2018b). Probabilistic multi-word spotting in handwritten text images. *Pattern Analysis and Applications*.

[Toselli et al., 2016b] Toselli, A. H., Vidal, E., Romero, V., and Frinken, V. (2016b). HMM Word-Graph Based Keyword Spotting in Handwritten Document Images. *Information Sciences*, 370(C):497–518.

[Toshikazu et al., 1991] Toshikazu, K., Takio, K., and Hiroyuki, S. (1991). Intelligent Visual Interaction with Image Database Systems : Toward the Multimedia Personal Interface. *Journal of Information Processing*, 14(2):134–143.

[Tsoumakas et al., 2010] Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). *Mining Multi-label Data*, pages 667–685. Springer US, Boston, MA.

[Vidal et al., 2015] Vidal, E., Toselli, A. H., and Puigcerver, J. (2015). High performance Query-by-Example keyword spotting using Query-by-String techniques. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 741–745.

[Villegas et al., 2016a] Villegas, M., Müller, H., García Seco de Herrera, A., Schaer, R., Bromuri, S., Gilbert, A., Piras, L., Wang, J., Yan, F., Ramisa, A., Dellandrea, E., Gaizauskas, R., Mikolajczyk, K., Puigcerver, J., Toselli, A. H., Sánchez, J.-A., and Vidal, E. (2016a). General overview of imageclef at the clef 2016 labs. In Fuhr, N., Quaresma, P., Gonçalves, T., Larsen, B., Balog, K., Macdonald, C., Cappellato, L., and Ferro, N., editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 267–285, Cham. Springer International Publishing.

[Villegas et al., 2016b] Villegas, M., Puigcerver, J., Toselli, A. H., Sánchez, J.-A., and Vidal, E. (2016b). Overview of the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task. In *CLEF (Working Notes)*, pages 233–253.

[Villegas et al., 2016c] Villegas, M., Toselli, A. H., Romero, V., and Vidal, E. (2016c). Exploiting Existing Modern Transcripts for Historical Handwritten Text Recognition. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 66–71.

[Vinciarelli et al., 2004] Vinciarelli, A., Bunke, H., and Bengio, S. (2004). Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):709–720.

[Vinciarelli and Luettin, 2001] Vinciarelli, A. and Luettin, J. (2001). A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9):1043 – 1050.

[Vinyals et al., 2017] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2017). Show and Tell: Lessons Learned from the 2015 MSCOCO

Image Captioning Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663.

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

[Voigtlaender et al., 2016] Voigtlaender, P., Doetsch, P., and Ney, H. (2016). Handwriting Recognition with Large Multidimensional Long Short- Term Memory Recurrent Neural Networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 228–233.

[Wang et al., 2012] Wang, S., Uchida, S., and Liwicki, M. (2012). Part-based method on handwritten texts. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 339–342.

[Weintraub, 1993] Weintraub, M. (1993). Keyword-spotting using SRI's DECIPHER large-vocabulary speech- recognition system. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 463–466.

[Werbos, 1974] Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA.

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Wicht et al., 2016a] Wicht, B., Fischer, A., and Hennebert, J. (2016a). Deep learning features for handwritten keyword spotting. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3434–3439.

[Wicht et al., 2016b] Wicht, B., Fischer, A., and Hennebert, J. (2016b). Keyword Spotting with Convolutional Deep Belief Networks and Dynamic Time Warping. In Villa, A. E., Masulli, P., and Pons Rivero, A. J., editors, *Artificial Neural Networks and Machine*

Learning – ICANN 2016, pages 113–120, Cham. Springer International Publishing.

[Witten and Bell, 1991] Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

[Woodland et al., 2000] Woodland, P. C., Johnson, S. E., Jourlin, P., and Jones, K. S. (2000). Effects of out of Vocabulary Words in Spoken Document Retrieval. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 372–374, New York, NY, USA. ACM.

[Wshah et al., 2014] Wshah, S., Kumar, G., and Govindaraju, V. (2014). Statistical script independent word spotting in offline handwritten documents. *Pattern Recognition*, 47(3):1039–1050. Handwriting Recognition and other PR Applications.

[Yazgan and Saraclar, 2004] Yazgan, A. and Saraclar, M. (2004). Hybrid language models for out of vocabulary word detection in large vocabulary conversational speech recognition. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP)*, volume 01, pages 745–748.

[You et al., 2016] You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image Captioning With Semantic Attention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4651–4659.

[Young et al., 2002] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., and Povey, D. (2002). The HTK book. Technical report, Cambridge University Engineering Department.

[Young, 1994] Young, S. R. (1994). Detecting misrecognitions and out-of-vocabulary words. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume 2, pages 21–24.

[Zagoris et al., 2017] Zagoris, K., Pratikakis, I., and Gatos, B. (2017). Unsupervised Word Spotting in Historical Handwritten Document Images Using Document-Oriented Local Features. *IEEE Transactions on Image Processing*, 26(8):4032–4041.