Language Modelling for Handwriting Recognition

Wassim Swaileh

Normandie Université

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de Laboratoire LITIS EA 4108
Université de Rouen

Language Modeling for Handwriting Recognition

Présentée et soutenue par
Wassim SWAILEH

Thèse soutenue publiquement le 4 octobre 2017 devant le jury composé de		
Mme. Laurence LIKFORMAN-SULEM	Professeur à Telecom ParisTech	Rapporteur
Mme. Véronique EGLIN	Professeur à l'INSA de Lyon	Examinateur
M. Antoine DOUCET	professeur à l'Université de La Rochelle	Examinateur
M. Christian VIARD-GAUDIN	professeur à l'Université de Nantes	Rapporteur
M. Thierry PAQUET	professeur à l'Université Normandie – Université de Rouen	Directeur

UNIVERSITÉ DE ROUEN NORMANDIE

MIIS

litis

"Je tiens tout d'abord à remercier le directeur de cette thèse, M. Theirry Paquet, pour m'avoir fait confiance malgré les connaissances plutôt légères que j'avais en novembre 2013 sur la dynamique des systèmes de la reconnaissance de l'écriture, puis pour m'avoir guidé, encouragé, conseillé, fait beaucoup voyager pendant presque quatre ans tout en me laissant une grande liberté et en me faisant l'honneur de me déléguer plusieurs responsabilités dont j'espère avoir été à la hauteur.

Je remercie tous ceux sans qui cette thèse ne serait pas ce qu'elle est : aussi bien par les discussions que j'ai eu la chance d'avoir avec eux, leurs suggestions ou contributions. Je pense ici en particulier à M.Clement Chatelain, M.Pierrick Tranouez, M.Yann Soulard, M.Julien lerouge et M.Kamel Ait Mhand "

Wassim Swaileh

Contents

List of Figures

List of Tables

Chapter 1

General Introduction to Handwriting Recognition

1.1 Introduction

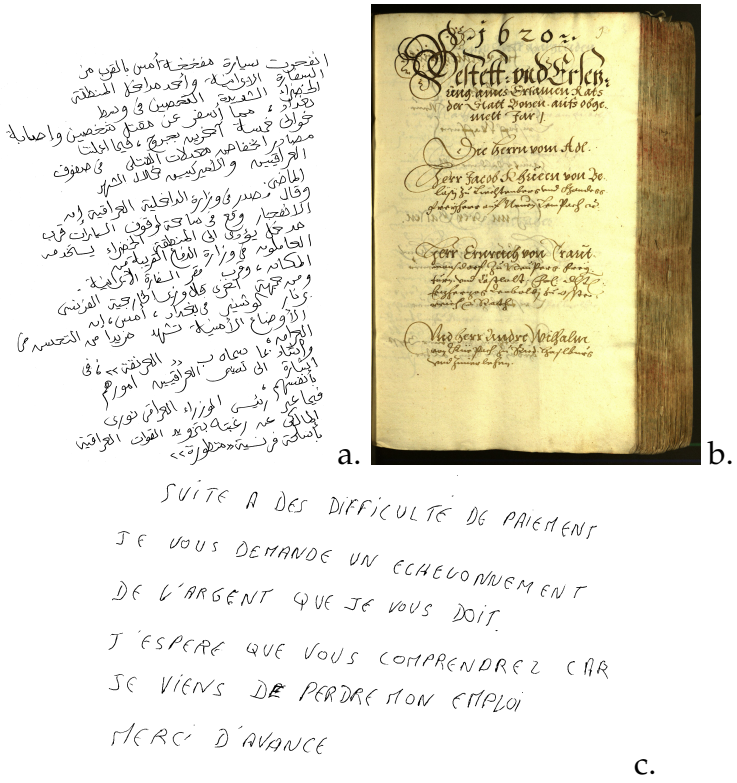
1.1.1 definition

With speech, cursive handwriting is still today the most natural way for human's communication and information exchanging. However, the digital age is asking for digital assistants capable to automatically translate or transcribe texts in different languages, spoken or written on different media such as paper or electronic ink. Automatic Handwriting recognition comes as a response to such a need.

Handwriting recognition is the task of transforming images of handwritten texts, into their ASCII or Unicode transcriptions, generally for further treatments such as text processing, translation, classification, indexing or archiving. Handwritten texts can be obtained from a real time capturing device while the user is writing (on-line), or from an off-line capturing process, once the information has been written on paper. In the on-line (dynamic) handwriting recognition case, one can use specific devices such as a digital pen which provides dynamically the recognition system with several informations such as pen position over writing time, pen displacement velocity, inclination, pressure and so on. Regarding off-line handwriting recognition, one can only exploit the handwriting in the form of a scanned document image. Thus, relevant handwritten text description features have to be extracted using specific image processing algorithms. Generally, the transcription process exploits the pixel values only, which makes the off-line handwriting recognition systems less accurate than the on-line ones. Printed or typewritten text recognition fall into the off-line recognition problem and are known as Optical Character Recognition (OCR). But the regularity of typewritten texts make their recognition easier than for cursive handwriting.

The main task of an off-line continuous handwriting recognition system is to obtain the transcriptions of characters and words in forms of human language sentences from an acquired document image. This implies a processing chain that starts by an appropriate text localization process when complex two-dimensional spatial layouts are present. Once text regions have been localized, text line images can be extracted using text localization algorithm. Then text line image pass thought a feature descriptor such as HoG (Histogram

of Oriented Gradient) descriptor in order to represent the text line image via a suitable numerical description vectors (feature vectors). These feature vectors represent the raw materials of the recognition engine. Given a text line description vectors, the recognition engine proposes the most likely transcription for the given text line descriptions with the guidance of a lexicon and a language model. Figure 1.1, shows three examples of document images written in French, German and Arabic languages which are taken from the French Rimes, German READ and Arabic OpenHaRT2013 data sets.



a. b. c.

FIGURE 1.1: Examples of handwritten document taken from Arabic OpenHaRT13, Germanic READ & French Rimes data sets

1.1.2 History

The history of OCR systems started at the dawn of computer technology and evolved gradually in three ages during the past and current century. First age extended from 1915 to 1980, it is characterized by the advent of character recognition methodologies such as template matching that was used for recognizing machine-printed texts or well-separated handwritten symbols, for examples ZIP codes (Arica and Yarman-Vural, 2001; Bluche, 2015). In 1915, Emmanuel Goldberg filed a patent for his machine controller which controlled in real time a machine via on-line recognition of digits written by the human operator. For this, an electrical sensor pen was used to trace the written digits on a tablet surface (the patent of the sensor pen dated to 1888). The current principle of OCR systems was first proposed by Gustav Tauschek in 1929. Basically, the idea was to find the best matching between the character shape and a set of

metallic templates. By setting a photo-sensor behind the metallic templates which represented the set of the alphabetic characters, the printed character which totally obscures the light from the photo-sensor is identified to be the associated metallic template character class (Tang, Lee, and Suen, 1996). After the birth of the first computers in 1940, some modern versions of handwriting recognition systems appeared (Bledsoe and Browning, 1959; Mantas, 1986; Govindan and Shivaprasad, 1990). The basic idea was standing on some low-level image processing techniques applied on binary image in order to extract a set of feature vectors which represent the inputs of the recognition engine. At this age, recognition engines based on statistical classifiers were introduced (Suen, Berthod, and Mori, 1980; Mori, Yamamoto, and Yasuda, 1984; El-Sheikh and Guindi, 1988)

The second age represents the period between 1980 and 1990. It is characterized by the rapid and wide development of handwriting recognition methodologies thanks to the advent of more powerful computer hardware and data capturing devices. In addition to statistical classification methods, structural methods (see section 1.4.3.3) are introduced in many character recognition applications. The applications were limited to two fields; the recognition of addresses for automatic mail sorting and delivery (Kimura, Shridhar, and Chen, 1993; Srihari, 2000; El-Yacoubi, Gilloux, and Bertille, 2002; Bluche, 2015), and the bank cheque reading systems (Le Cun, Bottou, and Bengio, 1997; Guillevic and Suen, 1998; Gorski et al., 1999; Paquet and Lecourtier, 1993).

The third and current age extends from 1990 till nowadays. This age is the age of the big achievements in handwriting recognition research field. New methods and technologies were proposed by efficiently combining pattern recognition techniques with artificial intelligence methodologies such as neural networks (NNs), hidden Markov models (HMM), fuzzy set reasoning, and natural language processing (Arica and Yarman-Vural, 2001).

Nowadays applications of handwriting recognition tend to process less constrained documents layouts with very large vocabularies, for example, the transcription of handwritten mail (Grosicki et al., 2009), information extraction (Chatelain, Heutte, and Paquet, 2006), ancient document text recognition (Romero, Andreu, and Vidal, 2011; Sánchez et al., 2013)

1.1.3 Challenges

Handwriting recognition methods evolved gradually over years in order to tackle different types of recognition problems with divers degrees of complexity. The simplest task is to recognize in isolation, a character or an alphanumeric symbol regardless of its role in the word or the sentence it belongs to. In this case, the segmentation problem is supposed to be solved prior to the recognition step. Nowadays, handwriting recognition targets the recognition of cursive and unrestricted continuous handwriting where words or sub-word's portions are written in a single stroke by connecting adjacent characters. In this case, the handwriting recognition uses implicit or explicit character segmentation methods. Handwriting recognition represents a challenging problem because of the writing styles (the calligraphy) diversity and the variability of

FIGURE 1.2: Character "r" writing style diversity according to its position in the word, text example taken from the IAM database

the image capturing conditions such as illumination, contrast and resolution. One can write same character with different shapes according to its position in the words (begin, middle & end). Furthermore, a writer can write two different characters using one single stroke with no care about the separation between the two characters (Bluche, 2015).

Figure 1.2, illustrate the different shapes of the Latin character "r" according to its position in the word. In fact, it is difficult to recognize the character "r" in total isolation without looking to its neighboring characters and its semantic context.

Before delving into the handwriting recognition processing techniques, the text in the document image should be localized in rectangular zones, and must be distinguished from other residual graphical components within the same document, see (Mao, Rosenfeld, and Kanungo, 2003) for a survey and more details about document analyses. Then text lines should be extracted from the localized text region. Regarding to Ghosh et al. (Ghosh, Dube, and Shivaprasad, 2010) the organized contests for text line extraction, this task is assumed as a challenging task. The major problem behind this task is the localization of the inflected (curvature) or maybe broken and overlapped text lines where the inter-space between the text lines are not tractable along the text zone width. For more details about the state of art of text line extraction techniques see chapter 4.

In fact, humans recognize a text trough a top level global view of the whole text document, which makes them able to recognize characters and words in their context. Similarly, computer vision try to mimic this ability by using a hierarchical structure in order to take account of contextual information above handwriting optical models during handwriting recognition.

The challenge behind a handwriting recognition system is the ability of the system to recognize the entire texts block regardless to the handwriting style, writing conditions and the text block image capturing condition.

1.2 handwriting processing chain

The handwriting processing chain consists of three main stages:

1. pre-processing stage: includes all the necessary faults correcting processes such as text localisation, line detection, image denoising, image inclination and slant correction.

2. Recognition stage: consists of matching text line extracted features to a set of character labels by using the couple of mathematical representations called optical models and language model.

3. Post-recognition stage: aims to enhance the recognition rate by correcting the sequence of characters produced in the recognition stage by using more sophisticated language models.

It is necessary to apply a set of processes to enhance the document image by correcting the introduced defects during the capturing process. Such defects include:-

a. Poor image resolution.
b. Noisy image.
c. Illumination variations due to clarity faults.
d. Image inclination and slant.
e. Black boarders of image.

In between the pre-processing stage and the recognition stage of the handwriting there is an intermediate process called feature extraction process. Feature extraction process is the operation by which the segmented part of text (line, word or character) can be described by certain significant measures called descriptors. Every descriptor represents one dimension of the feature space, the extracted features represent the input of the classification operation. Selection of a relevant feature extraction method is probably the single most important factor in achieving high recognition performance with much better accuracy in handwriting recognition systems (Sahu and Kubde, 2013).

In the recent years, we have seen the emergence of deep neural networks. The strength of these new architectures of networks lies in its ability to learn representation of features from the raw input data (pixels) directly, thus allowing to proceed without the need to define a feature space.

The recognition stage gives as an output a sequence of characters (class labels) which are most likely the handwritten text on the document image. The recognition process consists of a search procedure which maps the optical models (which can be either characters, sub-words or words) with the observed sequences of features. The idea is to decode the character hypotheses space in order to find the best mapping path represented by the maximum likelihood between the optical models and the observed features, taking into account some lexicon and language constraints.

There are two types of recognition errors: non-word errors and real word errors. A non-word error is a word that is recognized by the recognition engine; however, it does not correspond to any entry in the language vocabulary for example, when "how is your day" is recognized by the recognition engine as "Huw is your day" it is said to be a non-word error because "Huw" is not defined in the English language. In contrast, a real-word error is a word that is recognized by the recognition system and does correspond to an entry in the language vocabulary, but it is grammatically incorrect with respect to the sentence in which it has occurred, for example, "How is your day" is recognized by the OCR system "How is you day" the "you is considered as a real-word error (Bassil and Alwani, 2012). At the post-recognition stage, the recognition errors

can be corrected by using two linguistic information sources: dictionaries and language models.

1.3 Problem formulation

In order to automatically retrieve the handwritten text from an image, several methodologies of patterns recognition have been introduced in the literature standing on certain artificial intelligence technologies. The core function of such technologies is to represent the image properties by probabilistic models that represent the optical models, constrained by natural language grammar represented by the language model. Through a learning process, the optical models are optimized on a set of annotated examples in order to achieve the best possible character or word sequence transcription. On the other hand the language model has to be representative of the text in the document image to be recognized in order to get better performance.

Handwriting recognition can be achieved using one or two pass decoding algorithms which are based on Bay's rule. By decoding, we seek to retrieve the optimal sequence of words $\hat{W}$ that maximizes the *a posteriori* probability $P(W|S)$ among all possible sentences W (see equation 1.1. Two important parameters guide the decoding algorithm: they are the language model scaling parameter γ and the word insertion penalty parameter β that controls the insertion of too frequent short words. These two parameters need to be optimized for optimum coupling of the optical models with the considered language model, because these two models are estimated independently from each other during training.

By introducing the coupling hyper-parameters within the Bay's formula, we obtain the general formulation of the handwriting recognition problem as stated by equation 1.1.

$$\hat{W} = argmax_w P(W|X) \approx argmax_w P(X|W)P(W)^{\gamma}\beta^{length(W)} \tag{1.1}$$

In this formula, X represents the sequence of observations (features) extracted from the image and $P(X|W)$ represents the likelihood that the features X are generated by the sentence W, it is deduced from the optical model. $P(W)$ is the prior probability of the sentence W, it is deduced from the language model.

From this general formula, we note that language model contribution has the same effect as the optical model contribution during the recognition. The effect of language model contribution on the handwriting recognition process represent the core study of this thesis.

1.3.1 Optical models

An optical model can be an optical template such as the primitive models that were used during the early studies of OCR, or a statistical model such as a Hidden Markov model (**HMM**) or a Recurrent Neural Networks (**RNN**) or structural model that can model the variation of patterns after a specific learning

process. The OCR recognition problem is presented as a supervised classification or categorization task, where the classes are pre-defined by the system designer. According to the literature, Optical models can be classified into word, sub-word (BenZeghiba, Louradour, and Kermorvant, 2015), character and sub-character (Bluche, 2015) optical models.

The goal of using word optical models is to recognize a word directly as a whole, without relying on a segmentation or an explicit representation of the parts (Bluche, 2015). Simple representation of the word is extracted from the image and matched against a lexicon as presented in (Madhvanath and Govindaraju, 1996; Madhvanath and Krpasundar, 1997) and applied for the check amount recognition task. The disadvantage of the word optical model is the limitation of the vocabulary size because the number of optical models grows linearly with the number of words in the vocabulary. Despite, the limited size of training vocabulary cannot guarantee to provide enough example per model, so that they can not properly generalise to unseen examples. Therefore word optical models are limited to small vocabulary applications such as bank chicks.

To our knowledge, there was no attempts of using sub-word units as optical models in the literature. However in speech recognition, may be used as accoustic models in placeof the full word model. Usually sub-word models like demi-syllables, syllables, phonemes, or allophones are used instead of full-word models (Mousa and Ney, 2014). The advantage of using sub-word units is that they reduce the model complexity, which allows a reliable parameter estimation. Since the set of sub-words is shared among all words, the search vocabulary does not need to be equal to the training vocabulary. The acoustic model of any new words that is not present in the pronunciation dictionary can be assembled from the corresponding sequence of sub-word units.

Usually, the modern Large Vocabulary Continuous Speech Recognition (L VCSR) systems use the so-called context-dependent phoneme models which are models of phonemes with some left and right context. For example, a triphone is a phoneme joined with its predecessor and successor. In fact, characters in the written language meets phonemes in the spoken language. The contextual modelling of the language characters can be viewed as a sub-word optical model, that because each character optical model is considered to be conditionally related to its neighbouring characters. The authors of (Ahmad and Fink, 2016) proposed a class-based contextual modelling for handwriting Arabic text recognition based on HMM optical modelling.

The character optical models are used by almost all Large vocabulary continuous handwriting recognition systems. The advantage of using character optical models is their reduced number of classes compared to the sub-word and word optical models. Thus a moderate number of training examples is sufficient to efficiently train the character optical models. Using the embedded training techniques, there is no need to segment the text image into characters for training the optical models. Conventionally, the optical models represent the characters set of the language of interest.

For the sub-character optical models, the image is divided into sub-regions corresponding to at most one character (Bluche, 2015). The idea is to find all

possible image segmentations which is often derived from the character curvatures heuristically. A survey of character segmentation techniques was presented by (Casey and Lecolinet, 1996). We will discuss in more details the theoretical bases of the HMM and NN optical models in chapter 2.

1.3.2 language models

Sometimes the recognition engine decide for the recognition of a sequence of tokens (words) which may be grammatically or semantically incorrect. Many techniques from the natural language processing field have been developed for checking the validity of a sentence of words. A set of grammar rules can be applied on a dataset of tokenized text in order to constrain the search space on a grammatically valid paths. For example, (Zimmermann, Chappelier, and Bunke, 2006) re-scored a list of sentence hypotheses by using a probabilistic context-free grammar.

Generally speaking, language modelling for handwriting recognition usually consists in giving a score to different word sequence alternatives (Bluche, 2015). The common language modelling approach used for handwriting recognition (e.g. Rosenfeld, 2000 & Marti and Bunke, 2001) is the statistical approach. It consists of n-gram language models or connectionist language models based on neural network. This type of approaches estimates the *a priori* probability of observing a word sequence W from a large amount of training text samples (Bluche, 2015).

The perplexity measure is used to measure the capability of a language model to predict a given corpus. It is derived from the entropy of the probability model and can be expressed as the following:

$$PPL = 2^{\frac{-1}{N_w}\sum_{k=1}^{N_w} log_2 P(w_k|w_{k-1},\ldots)} \tag{1.2}$$

where N_w is the number of words in the text. Language models with smaller perplexities are generally better at predicting a word given the history. However, better perplexity does not mean better recognition performance according to (Klakow and Peters, 2002). A state of the art of language modelling will be presented in chapter 3.

1.3.3 System vocabulary

When the recognition system introduces words optical models as well as language models, a predefined lexicon is already embedded in the recognition engine, and limited to the set of words modelled (Bluche, 2015). However for large vocabulary it is necessary to introduce optical character models. The representation of words by their character sequences in a dictionary reduces the size of the search space and help to alleviate the recognition ambiguities.

Indeed, the recognizable words are limited only to the dictionary words and the chance to encounter out of vocabulary words (OOV) during the recognition increases when dictionary size is small. The OOV represents the ratio of non-covered words by the lexicon dictionary and thus the language model on

an evaluation data set. On the other hand, as the dictionary size increases, the search space grows accordingly which leads to higher recognition complexity. Furthermore, the competition between words increases proportionally to the dictionary size. Consequently, the selection of the smallest possible vocabulary achieving the highest coverage rate on unknown dataset is a real challenge.

The dictionary can be organized in such ways to enhance the recognition speed. The simplest way consists in computing scores for each word in the dictionary independently and decide for the word with the highest score. This solution is not satisfactory because the complexity increases linearly with the vocabulary size while many words share the same characters at same positions. Therefore, another way to organize a vocabulary is called prefix tree. In this way, the vocabulary is organised as a tree, and the root is considered to be the beginning character of a word. Each branch indicates another character of the word and the terminal nodes contains the word made of the characters along the path. By this way the shared prefix characters will be examined only once. More interested readers about the large vocabulary reduction and organization techniques can see (Koerich, Sabourin, and Suen, 2003).

Finite-state Transducer (FSTs) (Mohri, 1997) is another interesting and efficient way for representing vocabulary. It consists of a directed graph with a set of states, transitions, initial and final states. The transitions from one state to another are labeled with an input symbol from an input alphabet, and an output symbol from an output alphabet. When a sequence of input symbols is provided, the FST follows the successive transitions for each symbol, and emits the corresponding output symbol. A valid sequence allows to reach a final state with such sequence of transitions.

Vocabulary representation using FST defines the input alphabet as the set of characters, and the output alphabet matches the vocabulary. The compact structure of the vocabulary FST is similar to the prefix tree vocabulary representation. FST representation has the advantage of its ability to integrate the language model within the search graph. FST representations of language models are popular in speech recognition (Mohri, Pereira, and Riley, 2002) and are applied to handwriting recognition (e.g. in Toselli et al., 2004) and in several recognition toolkits such as Kaldi (Povey et al., 2011) or RWTH-OCR (Dreuw et al., 2012).

1.4 Handwriting recognition systems

In the literature, various structures have been proposed to perform handwriting recognition (Plötz and Fink, 2009). According to their lexical structure, the recognition systems can be classified into three main categories:-

1.4.1 Closed vocabulary systems

The first category includes the closed vocabulary systems that are optimized for the recognition of words in a limited and static vocabulary. This kind of system is used for specific applications such as bank checks reading (Gorski

et al., 1999). In this case the optical models may be word models. These approaches are often mentioned as global or holistic recognition.

1.4.2 Dynamic vocabulary systems

The second category includes dynamic vocabulary systems that are able to recognize words never seen by the system during training. In this category, the optical models are character models, and the approach is often referred as analytical recognition approaches that are guided by the knowledge of a lexicon (lexicon driven) and constrained by a language model during the recognition phase. With this capability, these systems are used for general purpose applications, such as recognition of historical documents, for example (Pantke et al., 2013) but they can not deal with unknown words (OOV) which words that do not appear in the vocabulary of the system.

1.4.3 Open vocabulary systems

The third category of approaches includes systems without vocabulary (lexicon free) that perform recognition in lines of text by recognizing sequences of characters. To improve their performance, these systems may use character sequences models in the form of n-gram statistical models (Plötz and Fink, 2009) by considering the space between words as a character (Brakensiek, Rottland, and Rigoll, 2002). The advantage of these systems is their ability to recognize any sequence of characters, including out of vocabulary words (OOV) such as named entities. Hence, they have the disadvantage of being less efficient than previous models in the absence of the sentence level modelling. Kozielski & al. (Kozielski et al., 2014b) have explored the use of character language models (for English and Arabic) using 10-gram character models estimated using the Witten-Bell method. They compared this lexicon free approach with a lexicon based approach associated with a 3-gram language model of words estimated using the modified Kneser-Ney estimation method. They have also combined the two models (characters and words) by using two approaches. The first one by building a global interpolated model of the two models, the second one by using a combination of back-off models. The results show the effectiveness of the combination of the two language models using interpolation.

1.5 Metrics

The error rates (ER) commonly used for the evaluation of continuous text recognition are the word error rate (WER) and character error rate (CER). Both are calculated based on the number of substitutions, insertions and deletions of words, respectively characters, and the number of words or characters in the reference text using the following equation:

$$ER = \frac{\#substitutions + \#insertions + \#deletions}{\#enforcements} \tag{1.3}$$

This ER can be deduced from the Levenstein edit distance (Levenshtein, 1966), which is the minimum edit distance between two strings (recognition hypotheses and ground-truth) and can be retrieved efficiently with a dynamic programming algorithm (Bluche, 2015).

Note that although it is generally expressed in percentage, it may goes beyond 100% because of the potential insertions. In this thesis, the reported WERs are computed with the SClite (Fiscus, 1998) and Kaldi (Povey et al., 2011) implementation. Similarly, we can consider an even finer measure of the quality of the output sequence in terms of characters, which penalizes less words with a few wrong characters and is less dependent on the distribution of word lengths: the Character Error Rate (CER). It is computed like the WER, with characters instead of words. The white space character should be taken into account in this measure, since this symbol is important to separate words.

1.6 conclusion

In this chapter, we have introduced the general problem of off-line handwriting recognition, including the related history, processing chain and systems. We introduced the general formula of handwriting recognition systems and related hyper-parameters which require an optimisation stage in order to get the optimal coupling of the optical model with the language model.

The main goal of this thesis has been to develop improved language modelling approaches for performing efficient large vocabulary unconstrained and continuous handwriting recognition for French and English handwritten languages.

The work of this thesis has been focused in two major directions: the first is to investigate the use of language dependent (syllables) and language independent (multigrams) types of sub-lexical units during the creation of the language models (LMs), the second is to investigate the enhancement of the handwriting recognition performance by unifying the recoginition system components: the optical and language models. The use of sub-lexical units has lead to a significant increase in the overall lexical coverage indicated by a considerable reduction in the out-of-vocabulary (OOV) rates measured on the test datasets. This has introduced one step towards the solution of data sparsity and the poor lexical coverage problems. As a result, significant improvements in the recognition performance have been achieved compared to the traditional full-word based language model. The unification of the recognition systems produce a generalized recognition whose performance outperform the performance of its parents for similar language such as the French and English languages. Experiments have been conducted on the French RIMES and the English IAM handwriting datasets.

This thesis consists of two main parts; theoretical part and experimentation part. on the one hand, theoretical part includes the first three chapters which covers the state of the arts and the bibliographical and literature reviews. The experimentation part include the last three chapters which discuss the architecture of the developed recognition systems and the contributions of the sub-lexical units based models to the handwriting recognition.

In chapter two, we present a state of the art of the handwriting optical models stating the theoretical architecture and the related training algorithms for each of Hidden Markov Models and Neural Network based optical models. We conclude this chapter with a comparative overview of the available toolkits used for training and decoding the optical models.

A state of the art on the language modelling approaches and techniques are presented in chapter three including language grammars and statistical language models. In addition to an overview of spoken and written language building blocks, we discuss different aspects related to n-gram and conectionnist language model training and smoothing techniques.

The developed recognition processing chain of two different recognition systems is described in chapter four. The processing chain illustrates the different architectures of optical models (HMM or BLSTM-CTC) and the n-gram language models of different lexicon types (words, syllables, multigrams and characters) in addition to the decoding process. Some primary results are presented to justify some choices regarding the decoding and language modelling approaches.

The contribution of the sub-lexical units of syllables and multigrams are presented in chapter five. The utilised supervised and unsupervised word decomposition into syllables and multigrams approaches are introduced justified with some statistical analyses of the generated lexicons coverage rates on the test datasets lexicons. State of the art recognition performance are obtained by the multigrams language models on the RIMES and IAM test datasets.

In chapter six, we studied the possible ways to unify the French recognition system with the English one. We studied the benefit of unifying sub-lexical units of both languages as well as unifying the optical character models. We observed that combining multigram sub-lexical units of both languages does not degrade the recognition performance while at the same time maintaining moderate complexity in the language model due the use of sub-lexical units.

Chapter 2

Theoretical bases of handwriting optical models

2.1 Introduction

Traditional handwriting recognition systems (Schambach, Rottland, and Alary, 2008) use discrete (Kaltenmeier et al., 1993) or continuous (Marti and Bunke, 2001) Hidden Markov Models (HMM) for modelling each building unit of the language of interest with respect to the writing time frame on the writing direction (Jonas, 2009). Conventionally, the HMM models represent the optical models of the character set of the language of interest.

An HMM consists of a set of dependent states associated with one or more Gaussian mixture density functions. The number of states in a sequential HMM model defines its length. The number of states depends on the topology of the HMM and the resolution used during the feature extraction process. In the literature, the variable length models show an improved recognition performance over the fixed length one because of their ability to adapt to the variable length graphical representation of the characters. The Maximum Likelihood Estimation (MLE) algorithm is used to estimate the optimal number of states per HMM character model (Zimmermann and Bunke, 2002).

For several years ago, neural networks have emerged in the handwriting recognition field as a fast and reliable classification tool. Neural networks are characterized by their simple processing units namely called neurons and their intensive weighted interconnections. The weights of the interconnections between neurons are learned from training data. The neurons are organised into hierarchically "stacked" layers which consist of initial or input layer, intermediate or hidden layer (these can be one or more stacked layers) and a final or output layer.

The information processing flows from the input layer through intermediate layers towards the output layer which predicts the recognized characters or words. The network layers are mutually interconnected from the input layer towards to output layer to form multi-layer neural networks.

In the literature, neural network architectures are classified into two specific categories: feed-forward and recurrent networks. Recurrent neural networks have the advantage over feed-forward neural networks of their ability

to deal with sequences over time. In spite of the different underlying principles, it can be shown that most of the neural networks architectures are equivalent to statistical pattern recognition methods (Ripley, 1993; Arica and Yarman-Vural, 2001).

Several types of NNs are used for handwriting recognition such as handwritten digits recognition (LeCun et al., 1989) using Convolutional Neural Network or handwritten sentence recognition using Bidirectional Long-short term memory recurrent neural networks (Graves et al., 2006) (BLSTM-RNN) or multi - dimensional long-short term memory recurrent neural networks (MDLSM - RNN) (Bluche, 2015).

In this chapter, we present an overview of the theory of optical models in two parts before concluding with a list of recent available platforms. The Hidden Markov Model is the topic of the first part and the neural network models are the topic of the second part.

2.2 Hidden Markov Models (HMM)

2.2.1 Introduction

Since several decades, Hidden Markov models are considered one of the most efficient statistical tools for solving sequential pattern recognition problems, such as spoken language processing, DNA sequence alignment, handwriting recognition. The name of this model holds the family name of a Russian scientist called Andrei Markov (1856-1922). Before delving deeper in the Hidden Markov Models (HMM), let us recall the foundation of these models. These models fall into the probability theory by addressing the description and modelisation of stochastic process (random process). Briefly, stochastic process are described by a collection of random variables (states) subject to change over time. The Markov property has been introduced as one simplifying assumption so as to describe the evolution over time by making the variables only depend on some other random variables at the precedent time frame. Any stochastic process that obeys the Markov property is often called a Markov process. Such an hypothesis neglects the long term dependencies.

2.2.2 Discrete Hidden Markov models

Hidden Markov Models are generative stochastic models representing the generation of an observed sequence $O = (o_1, \ldots, o_t, \ldots, o_T)$ from a hidden state sequence $Q = (q_1, \ldots, q_t, \ldots, q_T)$

The observation emission probability represents the possibility of generating a certain observation from a certain model state. This distribution explains the relation between the hidden states of the model and the observations (model outputs).

In case of discrete observations (which are related to Discrete HMM), the observation emission probabilities are represented by a $N \times M$ matrix (the number of model states N cross of the number of distinct observation symbols) known in the literature by $B = (b_j(v_k) = P[o_t = v_k | q_t = s_j])$ where s_j

represents the model state j and v_k represents the distinct observation k. So the observation emission probability $b_j(o_t = v_k)$ represents the probability of generating the observation element $o_t = v_k$ when the model state at time t is $q_t = s_j$.

Briefly, a HMM can be characterized by the following parameters:

1. N, the number of states in the model.
2. M, the number of distinct observation symbols.
3. A set of observation symbols $V = \{v_k, k = 1 \dots M\}$
4. A set of states $S = \{s_j, j = 1 \dots N\}$
5. The state transition probability distribution $A = \{a_{ij}\}$ having the following properties:-

$$a_{ij} = P\big(q_t = s_j | q_{t-1} = s_i\big) \qquad 1 \leq i, j \leq N \tag{2.1}$$

$$a_{ij} \geq 0$$

$$\sum_{i=1}^{N} a_{ij} = 1$$

6. The observation emission probability $N \times M$ matrix, in the literature known by $B = \big(b_j(o_t = v_k)\big)$ when the model state is s_j,

$$b_j(o_t = v_k) = P\big(o_t = v_k | q_t = s_j\big) \qquad \begin{matrix} 1 \leq j \leq N \\ 1 \leq k \leq M \end{matrix} \tag{2.2}$$

   So the observation emission probability $b_j(o_t = v_k)$ represents the probability of generating the observation element o_t equal to the observation symbol v_k when the model state is s_j at time step t.

7. The initial state distribution $\pi = \{\pi_i\}$, where

$$\pi_i = P\big(q_1 = s_i\big) \qquad 1 \leq i \leq N \tag{2.3}$$

The stochastic model is decomposed in two parts. First, the model assumes Markov dependency between hidden states, which writes:

$$P(Q) = \prod_t P(q_t | q_{t-1}) \tag{2.4}$$

Second, the model assumes conditional independences of the observations with respect to the hidden states, which writes:

$$P(O|Q) = \prod_t P(o_t | q_t) \tag{2.5}$$

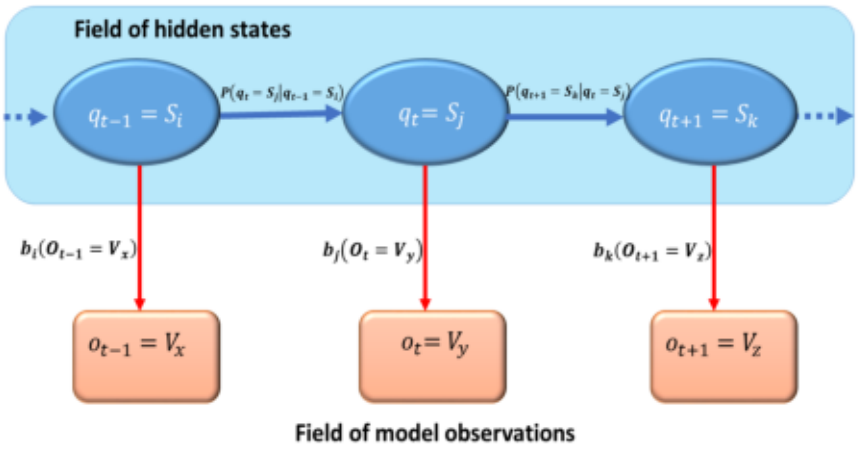


FIGURE 2.1: HMM Illustration

These two hypotheses allow writing the joint probability of the observation and state sequence as follows:

$$P(O,Q) = P(O|Q)P(Q) \tag{2.6}$$
$$= \prod_t P(o_t|q_t)P(q_t|q_{t-1}) \tag{2.7}$$

where $P(q_1|q_0) = \pi(q_1)$. The model can be represented through the temporal graph shown in figure 2.1 where arrows indicates the conditional dependency induced by the model. Finally, a HMM model λ can be characterized by the three parameters $\lambda = (A, B, \pi)$ which have the following representation:

$$\pi = \begin{bmatrix} \pi_1 \\ \vdots \\ \pi_N \end{bmatrix} \quad A = \begin{bmatrix} a_{11} & \dots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NN} \end{bmatrix} \quad B = \begin{bmatrix} b_1(v_1) & \cdots & b_1(v_M) \\ \vdots & \ddots & \vdots \\ b_N(v_1) & \cdots & b_N(v_M) \end{bmatrix}$$

In the literature an observation are commonly represented by real values. Practically, a quantization process can be introduced so as to assign a discrete label to the continuous observation. Such assignment can be made by introducing a clustering stage such as K-means. This fall into the semi-continuous Markov models. In fact, there is a risk of introducing quantization error with such quantization process. Such risk can be eliminated by introducing the continuous HMM (see 2.2.6. below).

2.2.3 Hidden Markov model structures

In the main application areas of HMM-based modelling, the input data to be processed exhibits a chronological or linear structure (Fink, 2014). Therefore, it does not make sense for such applications to allow arbitrary state transitions within a HMM. Based on this fact, we can distinguish five different topologies of HMM which are most often used in the literature:

1. linear HMM: stands on the idea that states are sequentially structured one after another without feedback loop and one forward jump from a model state to the next, in other word every state of the model can be reached only from its previous neighbour state as seen in figure 2.2.

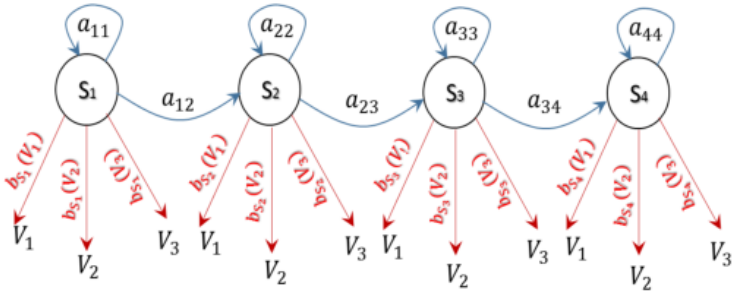


FIGURE 2.2: Linear HMM example with 3 discrete observation symbols

2. Bakis HMM: Allows the state to transit linearly similarly as in linear model but contains one additional forward jump from current state q_t to q_{t+2} as seen in figure 2.3.

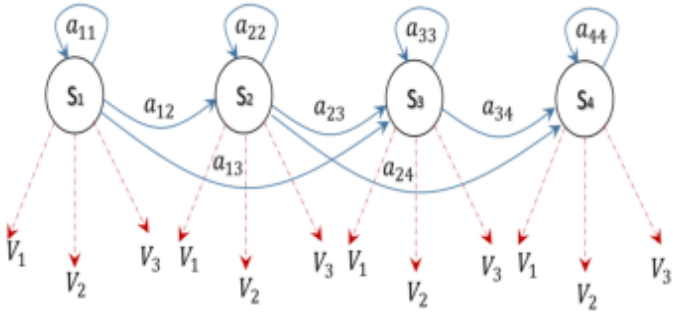


FIGURE 2.3: Bakis model example with 3 discrete observation symbols

3. Left to right HMM: Every possible forward jumps are allowed (no backward transition) as seen in figure 2.4.

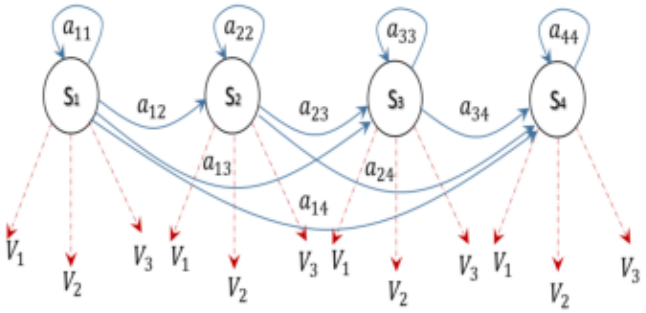


FIGURE 2.4: Left to right HMM example with 3 discrete observation symbols

4. parallel left to right HMM: represents a special case of left to right HMM where the middle states of the model has a side transition route to the parallel row of states which give more flexibility for this model. Figure 2.5 illustrates this model.

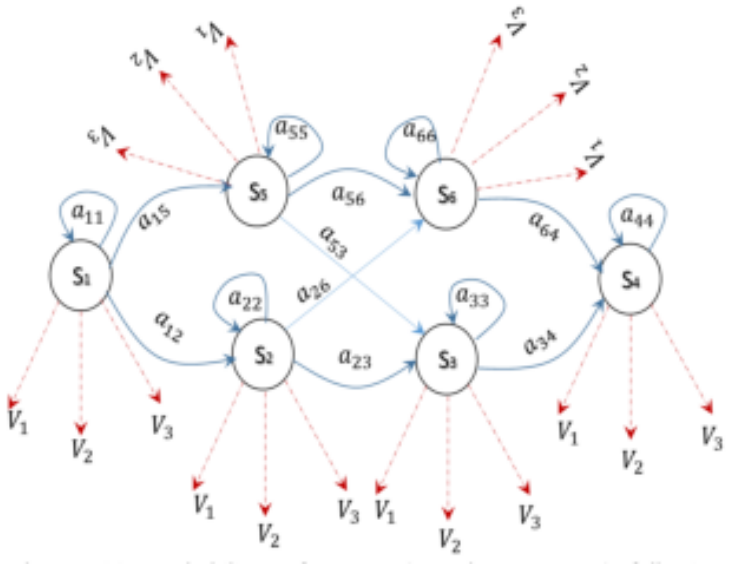


FIGURE 2.5: Parallel Lift To Right HMM example with 3 discrete observation symbols

5. Ergodic HMM: All state transition paths along the model states are allowed as seen in figure 2.6 .

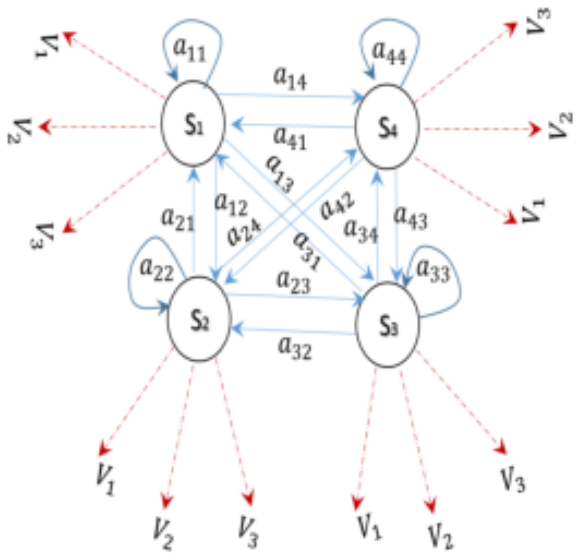


FIGURE 2.6: Ergodic HMM example with 3 discrete observation symbols

2.2.4 HMM design problems

We are considering the probability of producing a certain observation sequence by a HMM. This probability is called the production probability $P(O|\lambda)$ (Fink, 2014). It indicates the ability of a HMM to generate the given observation sequence, thus it can be used as a basis for a classification decision (Fink, 2014). We are interested in determining which model can produce the observation with the highest probability or which is the most probable model to produce the observation. When multiple HMM models are competing, the winner HMM λ_j will be the model for which the posterior probability $P(\lambda_j|O)$ becomes the maximum (Fink, 2014):

$$P(\lambda_j|O) = \max_i \frac{P(O|\lambda_i)P(\lambda_i)}{P(O)} \tag{2.8}$$

When evaluating this expression, the probability $P(O)$ of the data itself represents a quantity irrelevant for the classification — or the maximization of $P(\lambda_i|O)$ — because it is independent of λ_i and, therefore, constant. Thus for determining the optimal class, it is sufficient to consider the numerator of equation 2.8 only:

$$\lambda_j = arg\max_{\lambda_i} P(\lambda_i|O) = arg\max_{\lambda_i} \frac{P(O|\lambda_i)P(\lambda_i)}{P(O)} = arg\max_{\lambda_i} P(O|\lambda_i)P(\lambda_i) \tag{2.9}$$

In general, one considers equally likely models and $P(\lambda_i)$ can be omitted in equation 2.9. Therefore, the classification decision depends only on the emission probability $P(O|\lambda_i)$. In order to fit a HMM model to a real world application we need to find solutions to three main problems, which are the following:

First problem: Probability Evaluation

For a sequence of observations given a specific HMM, how do we compute the probability that the observed sequence was produced by that HMM model. In other words, the question to be answered is: *what is the probability $P(O|\lambda)$ that a particular sequence of observations O is produced by a particular model λ*? When multiple models are competing together, the target is to associate the model with the highest probability to generate the observed data. Computing the likelihood of the observation sequence allows to rank the models. Following the definition of a HMM, we can compute the emission probability $P(O|\lambda)$ by summing the contribution of every state sequence Q and write:

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) \tag{2.10}$$

but for an observation of length T and a model with N hidden states, the number of hidden state sequences Q is N^T which makes this computation untractable.

For evaluating the production probability $P(O|\lambda)$, we use two algorithms: the forward algorithm or the backwards algorithm (do not confuse them with the forward-backward algorithm) (Rabiner, 1989).

Forward algorithm

The Forward Algorithm is a recursive algorithm for calculating the emission probability for the observation sequence of increasing length t using the partial and accumulative probability forward variable $\alpha_t(i)$ of observing the partial sequence $(o_1, \ldots, o_t)$ and ending with state $q_t = s_i$. let us define:

$$\alpha_t(i) = P(o_1, o_2 \ldots o_t, q_t = s_i|\lambda) \tag{2.11}$$

We can observe that there are N different ways of arriving in state $q_t = s_j$ from state q_{t-1}. They correspond to the N possible previous states $q_{t-1} = s_i \ ; \ i \in 1, \ldots, N$ which leads to the recursion formula, taking account of the HMM independent emission.

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i)a_{ij}]b_j(o_{t+1}) \ \ 1 \leq t \leq T-1 \tag{2.12}$$

Then we can derive the emission probability of O by

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.13}$$

Backward algorithm

We can define the Backward variable $\beta_t(i)$ which represents the partial emission probability of the observation sequence from o_{t+1} to the last time step T, given the state at a time t is $q_t = s_i$ and the model λ.

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T|q_t = s_i, \lambda) \tag{2.14}$$

For the same reasons as for computing the forward variable we write the following recursion formula

$$\begin{array}{ll} \beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(o_{t+1})\beta_{t+1}(j), & t = T-1, T-2, \ldots, 1 \\ & 1 \leq i \leq N \end{array} \tag{2.15}$$

and finally, we can compute the emission probability by

$$P(O|\lambda) = \sum_{j=1}^{N} \beta_1(j)b_j(o_1) \tag{2.16}$$

Obviously both Forward and Backward algorithms must give the same results for total probabilities $P(O|\lambda) = P(o_1, o_2, \ldots, o_T|\lambda)$. The complexity of

computing the emission probability has reduced to $O(N^2 \times T)$ which is acceptable in practical applications.

Second problem:computing the optimal state sequence (Decoding)

We try here to recover the hidden part (hidden model states) of a HMM by finding the most likely hidden state sequence. The task, unlike the previous one, asks about the joint probability of the entire sequence of hidden states that generates a particular sequence of observations (Rabiner, 1989). This problem is called decoding problem (Fink, 2014).

Decoding is predicting the hidden part of the model (the state sequence Q) given the model parameters $\lambda = (A, B, \pi)$ by choosing the optimal state sequence Q which maximizes the emission probability $P(O, Q|\lambda)$ for a given observation sequence O. The optimal emission probability $P^*(O|\lambda)$ for generating the observation sequence O along an optimal state sequence Q^* can be determined by maximization over all individual state sequence Q (Fink, 2014).

$$P(O, Q^*|\lambda) = \max_{Q} P(O, Q|\lambda) \tag{2.17}$$

Similarly as for computing the emission probability, the number of individual state sequences Q is too large to allow a direct computation of each probability, and decide for the maximum one. Fortunately, the Viterbi algorithm can compute the optimum state sequence by using dynamic programming principle with a complexity of $T \times N^2$.

Viterbi algorithm

To find the best state sequence $Q = (q_1, q_2, \ldots, q_T)$ which can approximately generate the given observation sequence $O = (o_1, o_2, \ldots, o_T)$ knowing the model $\lambda = (A, B, \pi)$, let us define the probability of the best path ending at time t in state $q_t = s_i$.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(o_1, o_2, \ldots, o_t, q_1, q_2, \ldots, q_{t-1}, q_t = s_i|\lambda) \tag{2.18}$$

As the probability of a partial path is monotonically decreasing while t is increasing, the optimal path is composed of the optimal partial path. therefore, the dynamic programming optimisation policy can be applied, and we can write.

$$\delta_{t+1}(j) = \max_{i} \left(\delta_t(i) a_{ij} b_j(o_{t+1})\right) \tag{2.19}$$

The Viterbi algorithm can be achieved by the following steps:-

1- Initialization:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1) \qquad 1 \leq i \leq N \\ \varphi_1(i) &= 0 \end{aligned} \tag{2.20}$$

2- Recursion:

$$\begin{aligned} &\delta_{t+1}(j) = \max_i[\delta_t(i)a_{ij}]b_j(o_{t+1}) \\ &\qquad\qquad For 1 \leq t \leq T \quad 1 \leq j \leq N \\ &\varphi_{t+1}(j) = argmax_i[\delta_t(i)a_{ij}] \end{aligned} \tag{2.21}$$

3- Termination:

$$P^*(O|\lambda) = P(O, Q^*|\lambda) = \max_{1 \leq i \leq N} \delta_T(i) \tag{2.22}$$

$$Q_T^* = arg \max_{1 \leq i \leq N} \delta_T(i) \tag{2.23}$$

4- Back-Tracking of the optimal state sequence path:

For all times t where $t = T-1, T-2, \ldots, 1$

$$Q_t^* = \varphi_{t+1}(Q_{t+1}^*) \tag{2.24}$$

The difference between the forward algorithm and the Viterbi algorithm is that the forward algorithm takes the sum of all previous paths into the current cell calculation while the Viterbi algorithm takes the max of the previous paths into the current cell calculation, but they are very similar in nature and complexity.

2.2.5 Third problem: Parameter Estimation (model training)

Here we want to optimize the model parameters so as to best describe a given observation sequence or a set of observation sequences. The observation sequences used to adjust the model parameters are called training sequences. The training problem is the crucial one for most applications of HMM. It allows to adapt model parameters to create best models of real phenomena (Rabiner, 1989).

For most of applications of HMM, we seek to use the ideal model which has approximately the same statistical properties as those of the data. To this aim, we have to adapt or estimate a certain pre-selected model by an iterative procedure that is called model training. Normally, the pre-selected model has to be selected by an expert.

Given an observation sequence O or a set of such sequences, the model learning task is to find the best set of model parameters (A, B, π) which can generate approximately the same given observation sequence. No tractable algorithm is known for solving this problem exactly, but a local maximum likelihood can be derived efficiently using the Baum–Welch algorithm or the Viterbi algorithm (Rabiner, 1989).

In Viterbi training algorithm, only the probability $P(O, Q^*|\lambda)$ of the respective optimal state sequence is considered (Fink, 2014). Instead of that, in the Baum-Welch training algorithm the emission probability calculated over all model states $P(O|\lambda)$ is considered. In general, the parameter estimation

method guarantee a monotonic increase of the modelling quality only (Fink, 2014):

$$P_{\tilde{\lambda}} \geq P_{\lambda} \tag{2.25}$$

where $\tilde{\lambda}$ is the modified model version of the current HMM model λ.

Viterbi training Algorithm

For Viterbi Training method, the target is to choose the best scoring path of state sequence Q^* by which the observation sequence O can be generated. Based on that, some iterative transformations on the pre-selected model parameters $\lambda = (A, B, \pi)$ will optimize the model parameters to its optimal values, then we can get a trained model $\widehat{\lambda}$ which has approximately the same statistical properties of the given observation sequence. The Viterbi training algorithm can be summarized in the following few steps.

step 1: Initialization step.

Choose a suitable initial model $\lambda = (\pi, A, B)$ (the pre-selected model by an expert) with the initial estimates of π_i for starting and a_{ij} for the transition probabilities as well as the emission probabilities $b_j(o_k)$.

step 2: segmentation step

Calculate the optimal state sequence Q^* which can generate the given observation sequence O given the model λ. We are looking to calculate the $\delta_t(i)$ which maximizes the state score in order to determine the optimal state sequence Q^*. Let $X_t(i)$ be the state of the optimal state sequence which is obtained by the Viterbi algorithm.

$$X_t(i) = \begin{cases} 1 & \text{if} \quad Q_t^* = \text{i and } Q_t^* = arg\max_Q P(Q, O|\lambda) \\ 0 & \text{otherwise} \end{cases} \tag{2.26}$$

step 3: Parameters estimation step.

Once the optimal state sequence has been determined in step 2, the computation of updated estimates of $\tilde{\lambda} = (\tilde{\pi}, \tilde{A}, \tilde{B})$ is the following:

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} X_t(i) X_{t+1}(j)}{\sum_{t=1}^{T-1} X_t(i)} \tag{2.27}$$

$$\tilde{b}_j(o_k) = \frac{\sum_{t:o_t=o_k} X_t(j)}{\sum_{t=1}^{T} X_t(j)} \tag{2.28}$$

$$\tilde{\pi}_i = \frac{\sum_{NB=1}^{NS} X_1(i)}{Number of sequences} \tag{2.29}$$

where NS is the number of sequences, this step is called optimization step (Fink, 2014)

step 4: Termination step.

The termination criteria for this method is considered by determining a threshold of improvement of the estimated probability $P^*(O|\tilde{\lambda})$ Also a maximum number of iterations can be fixed.

As a conclusion, Viterbi algorithm depends on the partial production probability $P^*(O|\lambda)$ which is calculated by taking into account only the optimal state sequence Q^* by which the highest state score $\delta_t(i)$ can be obtained.

Baum-Welch training Algorithm

The Baum–Welch algorithm is a special case of the expectation maximization algorithm (EM). The methodology of this algorithm stands on the optimization criterion of the total production probability $P(O|\lambda)$ for which

$$P(O|\tilde{\lambda}) \geq P(O|\lambda) \tag{2.30}$$

The mechanism of this algorithm is represented in the following steps.

step 1: Initialization step

Choose randomly or by an expert initial parameter values for a HMM model $\lambda = (\pi, A, B)$.

step 2: expectation step (E)

1. Forward inference procedure: compute $\alpha_t(i)$
2. Backward inference procedure:compute $\beta_t(i)$:-
3. Deduce the posterior probability γ to be in state i at time t given the observation sequence O is computed using α and β as follows:

$$\gamma_t(i) = P(q_t = i|O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=i}^{N} \alpha_t(i)\beta_t(i)} \tag{2.31}$$

4. Computing $\xi_t(i, j)$ the probability of being in state i at time t, and state j at time $t + 1$ which reflects the effect of the state transition probability.

$$\xi_t(i, j) \quad = P(q_t = i, q_{t+1} = j|O, \lambda) \tag{2.32}$$

step 3: Maximisation step (M)

At this stage we compute the updated estimates of the model parameters $\tilde{\lambda}(\tilde{\pi}, \tilde{A}, \tilde{B})$ by using the $\gamma_t(i)$ and $\xi_t(i, j)$ calculated in step 1 as the following:-

$$\tilde{\pi}_i = \gamma_1(i) \tag{2.33}$$

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.34}$$

$$\tilde{b}_j(o_k) = \frac{\sum_{t=1;o_t=v_k}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{2.35}$$

step 4: Termination step.

the termination of this algorithm depends of a stoping criterion that measures the achieved enhancement in the total production probability $P(O|\tilde{\lambda})$ at every update of the model parameter. For example, we can choose a percentage value ϵ = 0.01 as a threshold and compare the measured enhanced production probability as a likelihood probability η, where $\eta(\lambda)$ is the likelihood probability calculated by the old model parameter estimates and $\eta(\tilde{\lambda})$ is the likelihood probability for the updated model parameter estimates which can be expressed as the following:

$$\frac{\eta(\tilde{\lambda}) - \eta(\lambda)}{\eta(\lambda)} \leq \epsilon \tag{2.36}$$

The algorithm jumps to step 3 while the criterion is not satisfied. ***otherwise terminate the procedure;***

2.2.6 Continuous Hidden Markov model architecture

The continuous nature of the observation vector (symbol v_k) introduced by the nature of the raw observation (pixels values or more elaborate features extracted from the pixels) requires to introduce an appropriate calculation of the observation emission probability $b_j(o_t)$. The estimation procedure for the continuous HMM parameters π and A is the same as the discrete HMM ones, the difference is in the estimation of the parameter B which describes the observation emission probabilities $b_j(o_t)$.

The choice of a Gaussian mixture density function to represent the observation emission probability $b_j(o_t)$ comes from its ability to approximate any probability distribution function (Huang et al., 2001). Figure 2.7 shows a representation of the observation via multivariate Gaussian density functions.

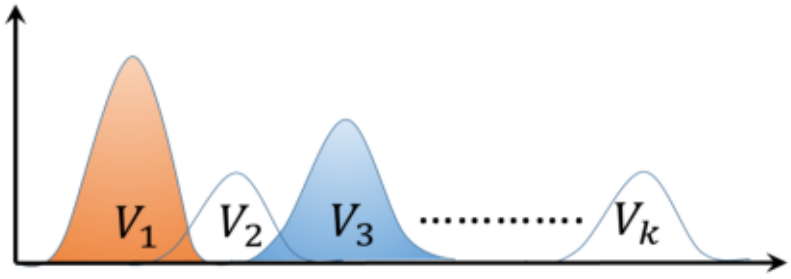


FIGURE 2.7: Representation of the observation by Gaussian density functions i.e Gaussian clusters

For M Gaussian density functions, the model writes:

$$b_j(o_t) = \sum_{k=1}^{M} C_{jk} N(o_t, \mu_{jk}, U_{jk}) = \sum_{k=1}^{M} C_{jk} b_{jk}(o_t) \qquad 1 < j < N \tag{2.37}$$

$$\sum_{k=1}^{M} C_{jk} = 1 \qquad 1 < j < N$$

$$C_{jk} \geq 0 \qquad 1 \leq j \leq N \ , \ 1 \leq k \leq M$$

where $N(o_t, \mu_{jk}, U_{jk})$ denotes a single Gaussian density function with mean vector μ and covariance matrix U for the state j, M denotes the number of mixture-components and C_{jk} the weight for the k^{th} mixture-component (Huang et al., 2001).

The Gaussian density function with mean vector μ_j and covariance matrix U_j is calculated as the following:

$$b_{jk}(o_t) = N(o_t, \mu_{jk}, U_{jk}) = \frac{1}{\sqrt{(2\pi)^n |U_{jk}|}} e^{-}1/2(o_t - \mu_{jk})' U_{jk}^{-1}(o_t - \mu_{jk}) \tag{2.38}$$

Where n represents the dimension of the observation vector o_t and the prime in $(o_t - \mu_{jk})'$ denotes vector transpose.

The estimation of those probabilities is based on the estimation of the parameters of the mixture Gaussian density functions C_{jk}, μ_{jk} and U_{jk}. When the observation elements are assumed independent, the covariance matrix U is reduced to a diagonal covariance (Huang et al., 2001).

In order to take into consideration the weight of each Gaussian mixture component the calculation of the posterior probability $\gamma_t(j,k)$ of being in state j at time t with k^{th} mixture component accounting for o_t is as the following:

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{C_{jk} b_{jk}(o_t)}{\sum_{m=1}^{M} C_{jm} b_{jm}(o_t)} \right] \tag{2.39}$$

The estimation formulas for the coefficient of the Gaussian mixture density functions C_{jk}, μ_{jk} and U_{jk} can be summarized as the following:-

$$\tilde{C}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)} \tag{2.40}$$

$$\tilde{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k).o_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{2.41}$$

$$\tilde{U}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k).(o_t - \mu_{jk})(o_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{2.42}$$

Where $\gamma_t(j,k)$ is the probability of being in the state j at time t with k^{th} mixture component accounting for o_t.

2.3 Recurrent neural network (RNN)

2.3.1 introduction

In general, neural networks (NNs) are commonly used for building pattern recognition system with very good performance. The scientific term "(*Artificial*) *Neural Network*" comes from the similarity of these models with the biological neurons. One of the most useful and powerful features of neural networks is their ability to learn and generalize from a set of training data.

Neural networks were introduced for the first time by Mc Culloch an Pitts (McCulloch and Pitts, 1943). The *Perceptron* notion was introduced by Rosenblatt (Rosenblatt, 1958) and after years the Multi-Layer Perceptrons (**MLP**, (Rumelhart, Hinton, and Williams, 1988) was introduced. Recurrent neural networks are the most convenient neural network type for solving handwriting recognition problem thanks to their ability to process sequences (Graves et al., 2006).

2.3.2 Perceptron as a building block

The basic building block of an artificial neural network is the artificial neuron which is know by *Perceptron*. The artificial neuron consists of a processing unit with weighted inputs and outputs. In fact, biological neurons receive signals

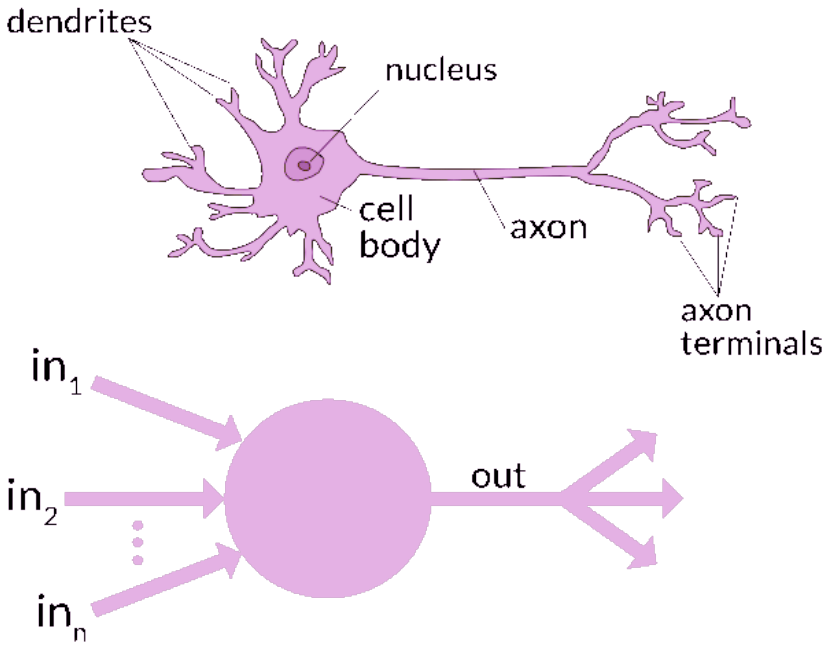


FIGURE 2.8: Biological (upper figure) versus artificial neuron (lower figure)

to be processed in their cell body via a set of dendrites and send processed signals out to other neurons via an axon. Similarly, the artificial neuron has a number of inputs and a processing unit, and one output that can be fed to multiple other neighbouring neurons at the next processing stage (next layer) as illustrated in figure 2.8.

The j^{th} neuron with $j \in \{1, 2, \ldots, n\}$ of an artificial neural network of n neurons consists of an activation function $f(x)$ (also called transfer function) which receives vector of inputs in_i with $i \in \{1, 2, \ldots, m\}$ (where m is the number of inputs) and outputs out_j which represent the j^{th} binary decision output where $out_j \in \{out_1, out_2, \ldots, n\}$. By definition, the neuron which classifies its input into two output categories is called *Perception* (Freund and Schapire, 1999). It can be formulated as the following

$$out_j = f(\sum_{i=1}^{n} in_i w_{ij} - \theta_j) = step(w_{1j} in_1 + w_{2j} in_2 + \ldots + w_{nj} in_n - \theta_j) \quad (2.43)$$

where, $w_{ij} \in \{w_{1j}, w_{2j}, \ldots, w_{nj}\}$ are the input vector weights and θ_j is the decision threshold associated to the neuron j, where $step(x) = 1$ if $in_i > \theta$ and 0 otherwise.

Training a single perceptron refers to determining the appropriate weights and threshold that leads to produce the expected output once certain inputs are given to the perceptron. It is useful to simplify the mathematics by treating the neuron threshold as if it were just another input connection weight by either assuming that $-\theta = w_0$ is the weight of the input $in_0 = 1$ for the neuron j (Bullinaria, 2004) or by representing the neuron j threshold by what is called the bias b_j. Thus we can rewrite equation 2.43 as the following:

$$out_j = step\left(\sum_{i=1}^{n} w_{ij} in_i + b_j\right) = step\left(\sum_{i=0}^{n} w_{ij} in_i\right) \quad (2.44)$$

For practical reasons, a continuous and differentiable function is preferred such as the hyperbolic tangent function (**tanh**), rectified linear unit (**ReLU**) and *sigmoid* function. The common choice is the *sigmoid* function because of its non-linearity and derivative nature.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d\sigma(x)}{dx} = \sigma(x) \times (1 - \sigma(x))$$

The sigmoid function can never return a 0 or a 1 due to its asymptopic nature.

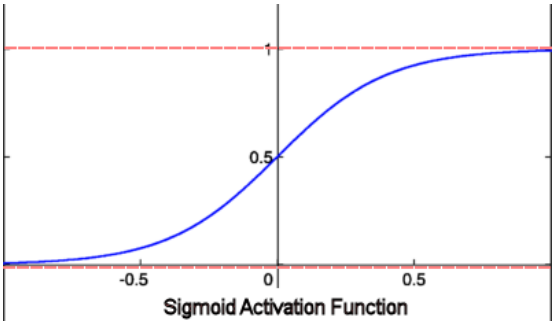


FIGURE 2.9: Sigmoid Activation function

Although, a binary classification decision can be taken from the output of the sigmoid such as:

$$out_j = \begin{cases} 0, & \text{if} \quad \sigma\left(\sum_{i=1}^{n} w_{ij} in_i + b_j\right) < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{2.45}$$

The thresholding of the sigmoid function output defines the binary classifier decision boundary as a hyperplane according to the following equation:

$$b_j + w_{1j} in_1 + \ldots + w_{nj} in_n = 0 \tag{2.46}$$

An optimal perceptron is the one which can separate linearly the classification problem space.

Practically, the bias b value shift the hyperplane left and right on the input space and the weights W rotate it. In graphical terms the threshold translates the hyperplane while the weights rotate it. This threshold also need to be updated during the learning process as well as the inputs' weights.

Perceptron Learning rule

Perceptron decision boundaries are hyperplanes, and we can think of learning as the process of shifting around the hyperplanes until each training example is classified correctly. The learning process starts with random initial weights and adjust them in small number of steps until the required outputs are produced by using an iterative learning algorithm.

Let the network weights at time t be $w_{ij}(t)$, then the shifting process corresponds to moving them by a small amount $\Delta w_{ij}(t)$ so that at time $t+1$ we have weights as following:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{2.47}$$

Suppose the target output of the perceptron unit j is $targ_j$ and the actual output is $out_j = step(\sum in_i w_{ij})$, where in_i are the activations of the input. The shifting amount $\Delta w_{ij}(t)$ can be calculated from the difference between the traget output $targ_j$ and the actual output out_j of the perceptron unit j as the following equation:

$$\Delta w_{ij} = \eta(targ_j - out_j).in_i \tag{2.48}$$

where η represents the *learning rate* or step size that determines how smoothly we shift the decision boundaries. The weight update equation 2.47 is called the *Perceptron Learning Rule* for which we consider three main cases:-

1. If $out_j = targ_j$ that leads to $targ_j - out_j = 0$ so then no need to update the weight w_{ij}.

2. If $out_j = 1$ and $targ_j = 0$ that leads to $targ_j - out_j = -1$ which means that $\sum_{i=0}^{n} in_i w_{ij}$ is too large so then we need to decrease w_{ij} in case $in_i = 1$ by the learning rate value
$$w_{ij} = w_{ij} - \eta.in_i \tag{2.49}$$

3. If $out_j = 0$ and $targ_j = 1$ that leads to $targ_j - out_j = 1$ which means that $\sum_{i=0}^{n} in_i w_{ij}$ is too small so then we need to increase w_{ij} in case $in_i = 1$ by the learning rate value
$$w_{ij} = w_{ij} + \eta.in_i \tag{2.50}$$

The weight changes Δw_{ij} need to be applied repeatedly for each weight w_{ij} in the network, and for each training example in the training set. One pass through all the weights for the whole training set is called one *epoch* of training. The training process converges to a solution, when all the network outputs match the targets for all the training patterns so all Δw_{ij} are zeros.

There are two important aspects of the network's operation to consider; (i) The network must learn decision boundaries from a set of training examples until these training examples are classified correctly. (ii) After training, the network must be able to generalize and correctly classify test examples that it has never seen before. In fact, there is an important trade-off between learning and generalization that arises quite generally (Bullinaria, 2004).

Learning by Error Minimisation

The general requirement for learning is an algorithm that adjusts the network weights w_{ij} to minimise the difference between the actual outputs out_j and the desired outputs $targ_j$. The Error Function or Cost Function E is used for quantifying this difference. One of theses quantification functions commonly used for learning neural networks is the *Sum Squared Error* (SSE) function which represents the total squared error summed over all the output units j and all the training examples p.

$$E_{SSE}(w_{ij}) = \frac{1}{2} \sum_{p} \sum_{j} (targ_{jp} - out_{jp})^2 \tag{2.51}$$

Another common used cost function for multiple class classification problem is called *Cross Entropy* cost function (E_{CE}). If we have network output out_j representing the probability of class j, and $targ_j$ is the binary target output, the probability of observing the whole training data set is $\prod_p \prod_j out_j^{targ_j}$ and by minimizing the negative logarithm of this likelihood, the cost function

becomes

$$E_{CE} = -\sum_{p}\sum_{j} targ_{jp}.\log\left(out_{jp}\right) \tag{2.52}$$

By the training process we aim to minimize such error function. The network training by minimizing the Sum Squared Error means that the derivative of the error with respect to each network weight must be zero at the minimum

$$\frac{\partial}{\partial w_{ij}}\left[\frac{1}{2}\sum_{p}\sum_{j}\left(targ_{jp} - f(\sum_{i} in_{ip}w_{ij})\right)^{2}\right] = 0 \tag{2.53}$$

If we want to change given x value by Δx to minimise a function $f(x)$, what we need to do depends on the gradient of $f(x)$ at the current value of x according to

$$\Delta x = x_{new} - x_{old} = -\eta\frac{\partial f}{\partial x} \tag{2.54}$$

where η is a small positive constant specifying how much we change x, and the derivative $\partial f/\partial x$ tells us which direction to go in. By iteration with equation 2.54, the function $f(x)$ will keep descending towards its minimum.This iterative procedure is known by *gradient descent minimisation*.

The idea is to apply a series of small updates to the weights $w_{ij} \leftarrow w_{ij}+\Delta w_{ij}$ until the cost $E(w_{ij})$ is "small enough". For this we need to determine the direction that the weight vector need to change to best reduce the cost function. This can be achieved by knowing the effect of varying the weights w_ij on the gradient of the cost function E. By adjusting repeatedly the weights by small steps against the gradient, the cost function will move through a weight space and descends along the gradients towards the minimum value.

2.3.3 Neural Networks Topologies

Multi-layer perceptron (MLP)

The multi-layer notion of the perceptrons has been inspired from the biological neurons structures to form what is called multi-layer perceptron (MLP, (Rumelhart, Hinton, and Williams, 1988)). In this analogy, the inputs in_i are received from other previous activated neurons outputs, that forms a neural network which consists of layers of mutually connected neurons with different connecting weights, starting form the input (first) layer through intermediate layers towards the output (last) layer as illustrated in figure 2.10 .

Let $W_{ij}^{(l)}$ denotes the weights of each neuron j of layer l applied on each input i which is the output of the neuron i of layer $l-1$. Thus the output (vector) of layer l namely $out_j^{(l)}$ can be computed by the multiplication of the vector of inputs $in^{(l)}$ with the weight matrix $W^{(l)}$ considering the bias as one of the input weights and the element-wise application of a non-linear function

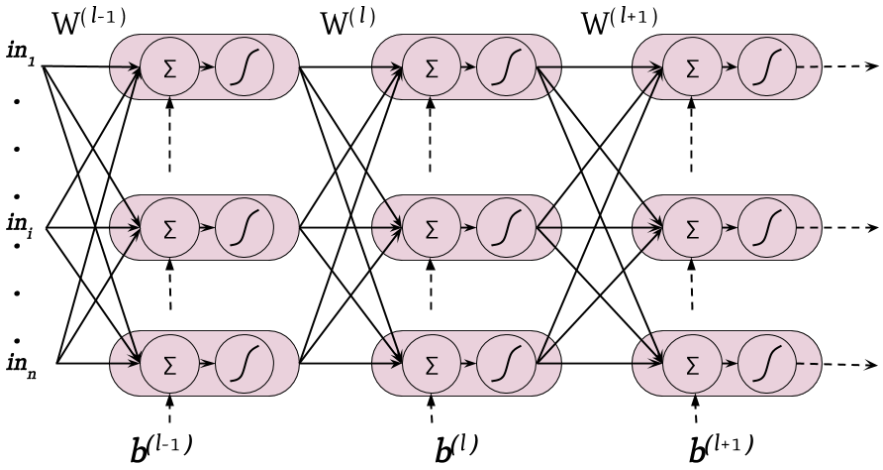


FIGURE 2.10: Multi-layer perceptron (MLP) illustration

f_l.

$$\begin{aligned} out_i^{(l-1)} &= in_i \\ out_j^{(l)} &= f_l\left(\sum_i W_{ij}^{(l)}.out_i^{(l-1)}\right) \\ out_k^{(l+1)} &= f_{l+1}\left(\sum_j W_{jk}^{(l+1)}.out_j^{(l)}\right) \\ &\vdots \end{aligned} \tag{2.55}$$

The advantages of such organisation of layers is in the computation of the layer output in a single *feed-forward pass*. The last layer neurons of the MLP are linear binary classifiers sharing the same input features (Bluche, 2015). Therefore, an MLP with several outputs is a multi-class classifier. (Bourlard and Wellekens, 1990) interpreted the network outputs as posterior probabilities. Posterior probabilities can be obtained by the application of a *softmax* function instead of a *sigmoid* function at the output (last) layer. By denoting $h = \sum_i w_{ij} in_i$ as the activation of the neuron j (before the non-linear activation function), and z its output, the *softmax* function can be defined for n neurons with $h_1, \ldots, h_n$ activations as follows:

$$z_i = softmax(h_i) = \frac{e^{h_i}}{\sum_{k=1}^{n} e^{h_k}} \tag{2.56}$$

where the output z_i is satisfying two conditions, *(i)* $z_i \in [0, 1]$, (ii) $\sum_{i=1}^{n} z_i = 1$. By this, a probability distribution over all classes is defined which allows to decide for the class with the highest probability, i.e highest output.

With one single layer perceptron, it is not possible to deal with non-linearly separable problems. However, Multi-Layer Perceptrons (MLPs) are able to cope with non-linearly separable problems.

Learning single layer perceptron

Single layer perceptron means that the network consists of one layer of perceptrons for which we apply the learning process and one layer of inputs as shown in figure 2.11.

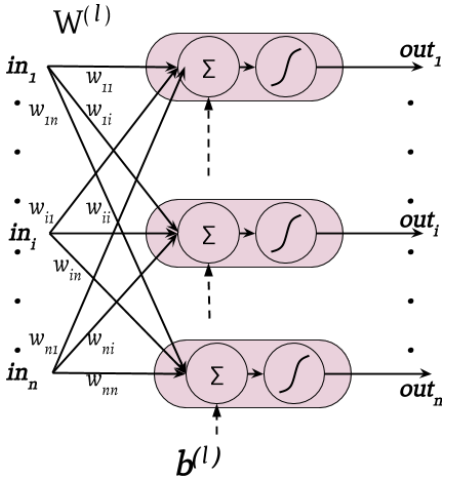


FIGURE 2.11: Multi-layer perceptron (MLP) illustration

We assume that we want to train a single layer network by adjusting its weights w_{ij} to minimise the SSE (2.51) by making a series of gradient descent weight updates according to

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{2.57}$$

If the transfer function for the output neurons is $f(x)$, and the activations of the previous layer k of neurons are in_i , then the outputs are $out_j = f(\sum_i in_i w_{ij})$, and:

$$\Delta w_{ij} = -\eta \frac{\partial}{\partial w_{ij}} \left[\frac{1}{2} \sum_p \sum_j \left(targ_{jp} - f(\sum_i in_{ip} w_{ij}) \right)^2 \right] \tag{2.58}$$

where $p \in p_1, p_2, \ldots, p_N$ are the training pattern. For training Single Layer Perceptrons, the *delta rule* of equation 2.58 and the *perceptron learning rule* have exactly the same weight update equation obtained by two algorithms which are theoretically different. The Perceptron Learning Rule was derived from a consideration of how we should shift around the decision hyper-planes for step function outputs, while the Delta Rule emerged from a gradient descent minimisation of the Sum Squared Error for a linear output activation function (Bullinaria, 2004). The *Perceptron Learning Rule* will converge to zero error and no weight changes in a finite number of steps if the problem is linearly separable, but otherwise the weights will keep oscillating. On the other hand, the *Delta Rule* will (for sufficiently small η) always converge to a set of weights for which the error is a minimum, though the convergence to the precise target values will generally proceed at an ever decreasing rate proportional to the output discrepancies $delta_j = targ_j - out_j$ (Bullinaria, 2004). The output error

$delta_j$ at the unit j represents the difference between the target output $targ_j$ at the unit j and the unit output out_j.

Learning MLP network by Back-propagation

Learning Multi-layer Perceptron network is motivated by the same principle as for single layer networks learning. The learning process aims to adjust the network weights $w_{ij}^{(l)}$ of each layer l in order to minimize an output cost function such as Cross Entropy Error function E_{CE} Unfortunately, the error at each intermediate layer in not known which prevent from using the gradient descent rule on each layer directly. This can be achieved by a series of gradient descent weight updates.

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E(\{w_{ij}^{(l+1)}\})}{\partial w_{ij}^{(l)}} \tag{2.59}$$

The linear final outputs can be written:

$$out_j^{(N)} = \sum_i out_i^{(N-1)} w_{ij}^{(N)} = \sum_i f\left(\sum_k in_k w_{ki}^{(N-1)}\right) w_{ij}^{(N)} \tag{2.60}$$

It is only the outputs $out_j^{(N)}$ of the final layer N ($N \in \{0, 1 \ldots, n, \ldots, N\}$) that appear in the output error function E_{CE}. Meanwhile, the final layer outputs will depend on all predecessor layers of weights, and the learning algorithm will adjust them all also, by backpropagation the error on each layer and updating the parameters simultaneously on each layer.

When implementing the Back-Propagation algorithm it is convenient to define

$$delta_j^{(N)} = (targ_j - out_j^{(N)}) \tag{2.61}$$

which is the output error. One the delta of the output layer N is computed according to equation 2.61, then the delta is back-propagated to earlier layers using

$$delta_j^{(l)} = \left(\sum_k delta_k^{(l+1)}.w_{jk}^{(l+1)}\right).f'\left(\sum_i out_i^{(l-1)} w_{ij}^{(l)}\right) \tag{2.62}$$

Then each weight update equation can be written as:

$$\Delta w_{hj}^{(l)} = \eta \sum_p delta_{pj}^{(l)}.out_{ph}^{(l-1)} \tag{2.63}$$

$$\Delta w_{hj}^{(l)} = \eta \sum_p \left[\left(\sum_k delta_{pk}^{(l)}.w_{jk}^{(l)}\right).out_{pj}^{(l-1)}.\left(1 - out_{pj}^{(l-1)}\right)\right].in_{ph} \tag{2.64}$$

So the weight $w_{hj}^{(l)}$ between units h and j is changed in proportion to the output of unit h and the $delta$ of unit j. The weight changes at the earlier layer

$(l-1)$ now take on the same form as the layer (l), but the "error" delta at each unit j is back-propagated from (l) each of the output units k via the weights $w_{jk}^{(l)}$.

Recurrent Neural Network (RNN)

The network contains at least one feed-back connection that makes its activation flow round in a loop over time as shown in figure 2.12. Thanks to

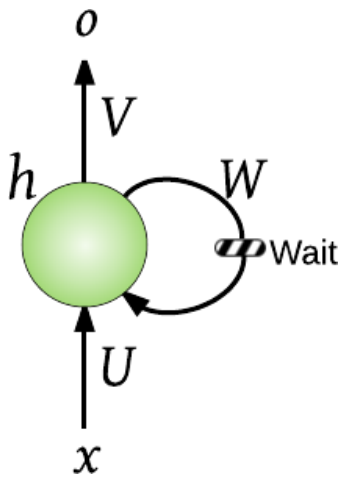


FIGURE 2.12: Recurrent neural network (RNN) illustration Graves et al., 2006

this property, the network is able to achieve temporal processing tasks such as handwriting recognition and learn sequences like in language modelling tasks. Figure 2.13 shows a RNN being unrolled (or unfolded) into a full one hidden layer network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 3 events, the network would be unrolled into a 3-layer neural network, one layer for each event.

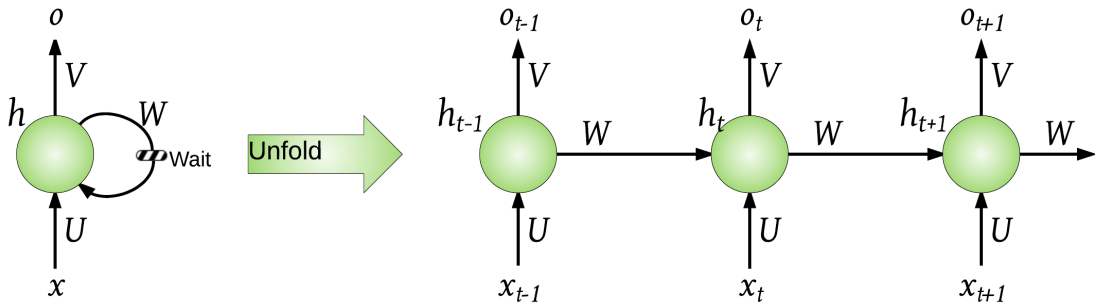


FIGURE 2.13: Unrolled recurrent neural network (RNN) illustration

Let x_t and o_t be the input and output vectors of events associated with two connection weights matrices U and V respectively. Furthermore, let denote h_t as the hidden state (the memory unit) of the network which depends on

the previous hidden state h_{t-1} and the input event at the current step x_t considering their associated weights W and U respectively. The hidden and output unit activation functions f_h and f_o are used for controlling the recurrent unit behaviour. Unlike a traditional deep neural network, which uses different weights at each layer, the RNN network shares the same weights (U, V, W seen in figure 2.13) across all time steps. This reflects the fact that we are performing the same task at each time step, just with different inputs. This greatly reduces the total number of weights we need to learn.

The behaviour of the recurrent network can be described as a dynamical system by a pair of non-linear matrix equations:

$$h_t = f_h\left(\mathbf{U}.x_t + \mathbf{W}.h_{t-1}\right) \tag{2.65}$$

$$o_t = f_o(\mathbf{V}.h_t) \tag{2.66}$$

The function f usually is a non-linearity such as *tanh* or *ReLU*. Theoretically, the state h_t captures information about what happened in all the previous time steps. The output at step o_t is calculated solely based on the memory at time t. As briefly mentioned above, it's a bit more complicated in practice because h_t typically can't capture information from too many time steps ago.

Recurrent Neural Networks unit (RNNs) are popular models that have shown great and promising results in many sequence related tasks such as handwriting recognition and language modelling. The idea behind the language modelling using RNNs is to predict the next event on the line of events with keeping in mind the event which comes before it. However, for handwriting recognition, the RNN is concerned by the identification of input events with regards to its history in some form of memory. Bidirectional RNNs (BRNNs,

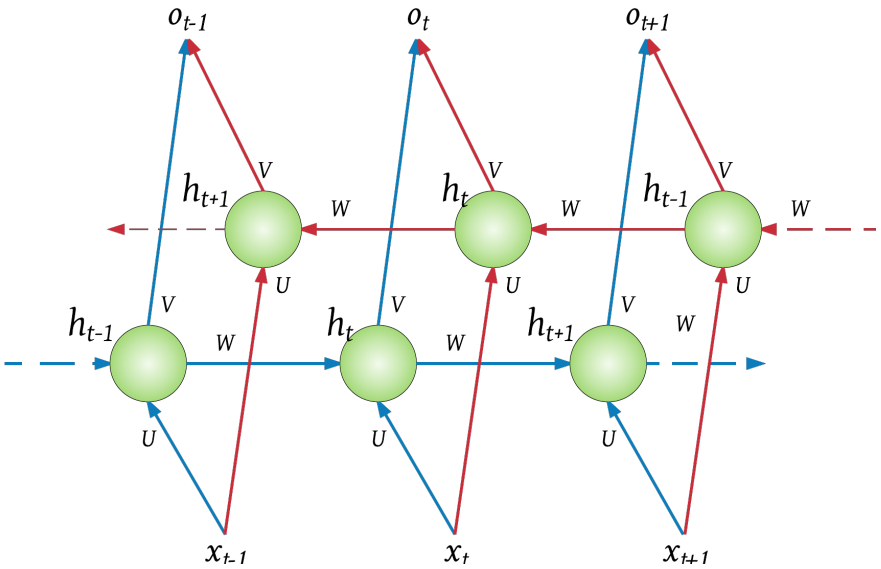


FIGURE 2.14: Recurrent neural network (bidirectional-rnn) illustration

Schuster and Paliwal, 1997) are based on the idea that the output at time t may

not only depend on the previous elements in the sequence, but also future elements. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs. They compute the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence o by iterating the backward the forward layer from $t = 1$ to T, layer from $t = T$ to 1 and updating the output layer as the following:

$$\begin{aligned} \overrightarrow{h} &= f_h(\overrightarrow{\mathbf{U}} x_t + \overrightarrow{\mathbf{W}} \overrightarrow{h}_{t-1}) \\ \overleftarrow{h} &= f_h(\overleftarrow{\mathbf{U}} x_t + \overleftarrow{\mathbf{W}} \overleftarrow{h}_{t-1}) \\ o_t &= f_o(\overrightarrow{\mathbf{V}} \overrightarrow{h}_t + \overleftarrow{\mathbf{V}} \overleftarrow{h}_t) \end{aligned} \tag{2.67}$$

Figure 2.14 shows an illustration of the bidirectional RNN.

Multi-Directional RNNs (MDRNNs) was introduced by (Graves and Schmidhuber, 2009). MDRNNs network is able to process an input image in four directions with recurrent layers. (Mikolov et al., 2011b) present freely available open-source toolkit for training recurrent neural network based language models. Simple recurrent neural networks was proposed by (Robinson, 1994) for speech recognition and (Senior, 1994; Lee and Kim, 1995; Senior and Robinson, 1998) for handwriting recognition.

Learning RNNs using back-propagation through time algorithm BPTT

The Backpropagation through Time (BPTT) learning algorithm is a natural extension of standard backpropagation that performs gradient descent on a complete unfolded network.

When a network training starts at time t_0 and ends at time t_1, the total cost function is the sum over time of the standard error function $E(t)$ at each timestep is:

$$E_{total}(t_0, t_1) = \sum_{t=t_0}^{t_1} E(t) \tag{2.68}$$

Moreover, the total error must be summed over every sequence example in a way similar to a multi-layer perceptron but by considering a shared layer over time. Thus the name backpropagation through time of the training algorithm.

Vanishing Gradient issue

Training certain *Artificial Neural Networks* with gradient based methods (e.g Back Propagation) encounters a difficult problem known by *Vanishing Gradient* caused by stacking too many layers and also due to the activation function (e.g *sigmoid* or *tanh*) which squeeze their input into a very small output range in a very non-linear fashion. When stacking multiple layers, even a large change in the input will produce a small change in the output - hence the gradient is small. This means that large time dependencies cannot by processed by such method; this explains why ANN have not been considered useful in sequence modelling during time.

CTC learning criterion

Graves et al., 2006 introduced the Connectionist Temporal Classification (CTC) criterion basically for mapping an input sequence x to an output sequence of labels L (ex. characters) within RNN network by using N-to-N predictions with no need for further post-processing. The problem is that the standard neural network objective functions are defined separately for each frame in the training sequence; in other words, RNNs can only be trained to make a series of independent label classifications. This means that the training data must be pre-segmented, and that the network outputs must be post-processed to give the final label sequence. (Graves et al., 2006).

The proposed method defines the network outputs to be the set of possible labels sequence introduced to get a better character representation, plus a blank output ($\emptyset$). By this way, a sequence of predictions is mapped to the target sequence of labels by removing repeated labels and the blanks for example:

$$aa\ \emptyset\ bb\ \emptyset\ a \quad \Rightarrow \quad aba$$

Training a RNN network using CTC criterion aims to maximize the probability of the label sequence given the input sequence. Several prediction sequences yield the same label sequence $(e.g. aabb, aaab, a\emptyset bb, ...)$. To simplify the analogy with the methods presented previously, let $Q_n(L)$ be the set of all labels (prediction) mapping the groundtruth target sequence L.

$$p(L|x) = \sum_{q \in Q_{|x|}(L)} \prod_{t=1}^{|x|} p(q_t|L) \tag{2.69}$$

This quantity can also be efficiently computed with a forward procedure in a way similar to the HMM. The mapping defines the allowed transitions between labels: one can either continue to predict the same label, jump to the next one if it is different, or jump to a blank. The forward and backward variables are defined as follows, with $L = l_1, \ldots, l_n$ and $L' = l'_1, \ldots, l'_n = \emptyset l_1 \emptyset, \ldots, \emptyset l_n \emptyset$

$$\alpha_t(l'_s) = p(q_{1:t} \in Q_t(L_{1:s/2}, q_t = l'_s|x) \tag{2.70}$$

$$\beta_t(l'_s) = p(q_{t+1:T} \in Q_{T-t}(L_{s/2+1:|L|}), q_t = l'_s|x) \tag{2.71}$$

and the recurrences are:

$$\alpha_t(l'_s) = p(q_t = l'_s|x) \sum_{n=0}^{k} \alpha_{t-1}(l_{s-n}) \tag{2.72}$$

$$\beta_t(l'_s) = \sum_{n=0}^{k} p(q_{t+1} = l'_{s+n}|x) \beta_{t+1}(l'_{s+n}) \tag{2.73}$$

where $k = 1$ when $l'_s = l'_{s-2}$ or $l'_s = l'_{s+2}$ for forward / backward variables, and k=2 otherwise. The α and β variables allow to compute

$$p(L|x) = \sum_{q \in Q_{|x|}(L)} \alpha_t(q)\beta_t(q) \tag{2.74}$$

and the derivation of the cost $-\log p(L|x)$ leads to the following back-propagated error:

$$\frac{\partial E}{\partial a_k^t} = y_k^t - \sum_{s:\mu(s)=k} \frac{\alpha_t(s)\beta_t(s)}{\sum_r \alpha_t(r)\beta_t(r)} \tag{2.75}$$

were y_k^t is the output of the neural network at time t for label k, and a_k^t are the activations before the softmax.

Long Short-Term Memory Units (LSTM)

In RNNs, the *vanishing gradient* issue prevents the neural network to learn long time dependencies (Bluche, 2015; Hochreiter, 1991; Bengio, Simard, and Frasconi, 1994). The solution proposed by (Hochreiter and Schmidhuber, 1997) to tackle this problem introduced a gating mechanism in form of a Long Short-Term Memory unit neural network (LSTM). Unlike the recurrent unit which simply computes a weighted sum of the input signal and applies a nonlinear activation function, each LSTM unit maintains a memory C_t at each time t. The temporal recurrent memory cell C_t is controlled by forget, input and output gates which serve to flush, change or retrieve data from the memory cell respectively. The gating mechanism is what allows LSTMs to explicitly model long-term dependencies. By learning the parameters for its gates, the network learns how its memory cells should behave. The gate consists of an activation function (like, sigmoid) and a point wise multiplication operation (Bluche, 2015). The gate outputs numbers between zero and one, deciding either to let the whole information go when the output value is one or total information blocking once the output value is zero.

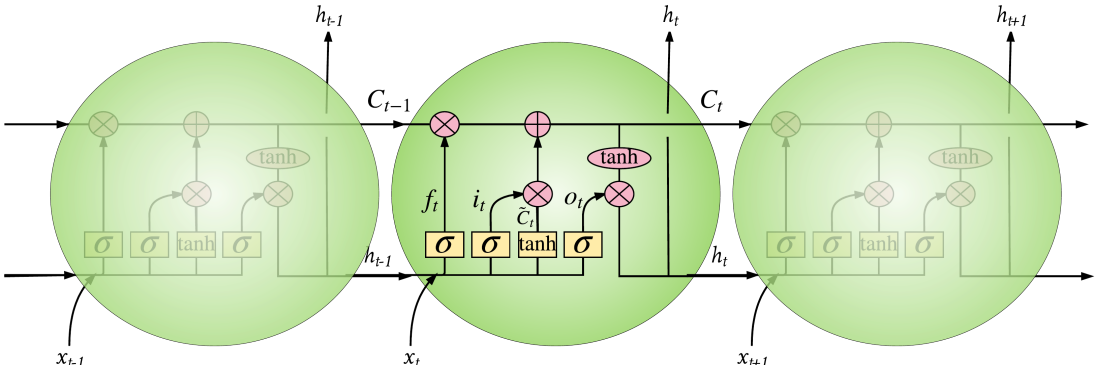


FIGURE 2.15: LSTM functional illustration

Looking at figure 2.15, an LSTM unit is described by the following terms:-

The ***Output Gate*** o_t controls whether the LSTM unit emits its memory activation output h_t.

$$\begin{aligned} o_t &= \sigma(W_o.[x_t, h_{t-1}] + b_o) \\ h_t &= o_t \odot tanh(C_t) \end{aligned} \tag{2.76}$$

The ***forget gate*** f_t controls by interruption, the participation of the previous state forward to the next one.

$$f_t = \sigma(W_f.[x_t, h_{t-1}] + b_f) \tag{2.77}$$

The ***Input Gate*** i_t decide whether the current cell input $\tilde{C}_t$ has a contribution to the memory cell state at the next time slot C_{t+1}.

$$i_t = \sigma(W_i.[x_t, h_{t-1}] + b_i) \tag{2.78}$$

The next cell state C_t can be obtained by.

$$\begin{aligned} \tilde{C}_t &= \tanh(W_c.[x_t, h_t - 1] + b_c) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \end{aligned} \tag{2.79}$$

Unlike the traditional recurrent unit which overwrites its content at each time-step, an LSTM unit is able to decide whether to keep the existing memory state via the introduced gates. Intuitively, if the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information (the existence of the feature) over a long distance, hence, capturing potential long-distance dependencies (Chung et al., 2014).

The LSTM unit is more powerful than the standard RNN's one by the ability of learning simple grammars (Hochreiter and Schmidhuber, 1997; Gers and Schmidhuber, 2001) and music composition (Eck and Schmidhuber, 2002). LSTM-RNN's was successfully used for phonemes and speech recognition by (Graves and Schmidhuber, 2005; Graves, Mohamed, and Hinton, 2013).

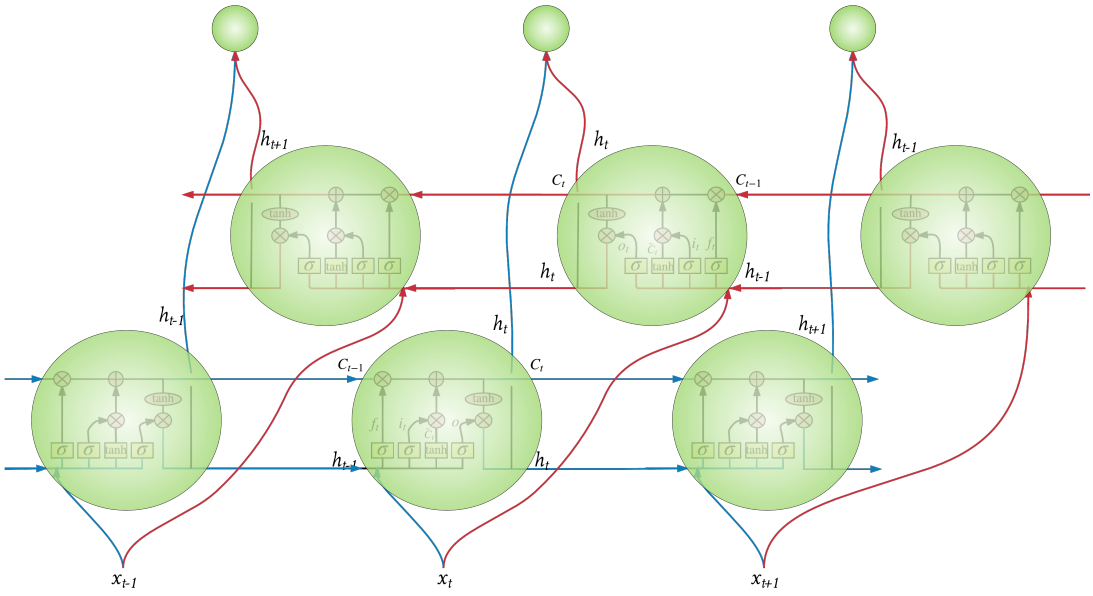


FIGURE 2.16: BLSTM functional illustration

For many sequence labeling tasks such as handwriting and speech recognition tasks, it is beneficial to have access to both the past (left) and the future (right) contexts with a Bi-directional LSTM network (BLSTM). However, the LSTM's hidden state h_t takes information only from past and knows nothing about the future. Recurrent layers of BLSTM neurons structured for working on Bi-directions achieve the state of the art in the handwriting recognition with best obtained results (Doetsch, Kozielski, and Ney, 2014; Graves and Schmidhuber, 2009; Bluche et al., 2014). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output as illustrated in figure 2.16.

Convolutional Neural Networks CNN

Convolutional Neural Networks (CNN or ConvNets) has the same MLP structure with one essential structural difference (LeCun et al., 1989). The neurons of the CNN neural network are not fully interconnected to each other from one layer to the next. The traditional structure of the ConvNets consists of

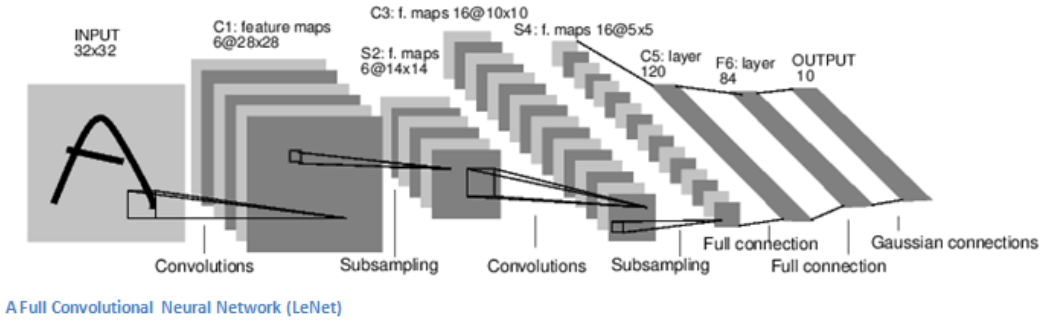


FIGURE 2.17: ConvNets illustration example

four main operations as seen in figure 2.17:-

1. *Convolution*: aims to learn the input image features by preserving the spatial relationship between pixels.

2. *Non linearity (ReLU)*: ReLU is an element wise operation (applied per pixel) and replaces all negative values in the feature map by zero.

3. *Polling or Sub Sampling*: Spatial Pooling (also called sub-sampling or down-sampling) reduces the dimensionality of each feature map but retains the most important information.

4. *Classification (Fully connected layer)*: The Fully Connected layer is a traditional Multi Layer *Perceptron* that uses a *softmax* activation function in the output layer.

The network learning process is similar to MLP network, it uses the back-propagation gradient decent algorithm. Convolutional Neural Networks have

a good reputation in computer vision application field because of their excellent achievements with best performance results in various challenging tasks such as object recognition (Szegedy et al., 2015), image segmentation (Farabet et al., 2013), speech recognition (Abdel-Hamid et al., 2012). They are also used for handwritten digits recognitions by (LeCun et al., 1989), then for whole handwritten text (sentence) recognition by(LeCun et al., 1989; Bengio et al., 1995; Le Cun, Bottou, and Bengio, 1997; Bluche, Ney, and Kermorvant, 2013a; Bluche, Ney, and Kermorvant, 2013b).

2.3.4 Handwriting recognition platforms and toolkits

Several toolkits have been used for speech recognition or for handwriting recognition. Some of these toolkits support implementation of Hidden Markov model or neural network based HWR systems and provides sophisticated facilities for speech analysis, HMM training, testing and results analysis.

We present a non-exhaustive list of HMM based platforms and toolkits that help to build HMM based HWR systems is the following:

1. HTK (Young and Young, 1993) consists of a set of library modules and tools available in C source form. The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems. initially, HTK is a HMM toolkit but recently, it has the ability to deal with deep neurl network architecture.

2. Juluis (Lee, Kawahara, and Shikano, 2001) is a high-performance, two-pass LVCSR decoder written in C and context-dependent HMM decoding (deal with HTK HMM formates).

3. Sphinx-4 (Walker et al., 2004) is a modular (written in Java) and pluggable framework that incorporates design patterns from existing systems, with sufficient flexibility to support emerging areas of research interest.

4. RWTH Aachen University speech recognition toolkit (Rybach et al., 2009) is written in C++ and consists of speaker adaptation, speaker adaptive training, unsupervised training, a finite state automata library, and an efficient tree search decoder components.

5. Kaldi (Povey et al., 2011) is an open-source toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. It provides the tools to train HMM / NN optical models and intergrates Open-FST library (Allauzen et al., 2007) for decoding with Finite State Transducers techniques.

Common NN based platforms and toolkits used for building HWR are presented in the following list:-

1. RETURNN (Doetsch et al., 2016) is extensible and easily configurable neural network training software. It support training different recurrent neural network topologies on multiple GPUs. RETURNN stands on Theano as an top of layer for the Theano library which represents a bridge between research oriented software packages and application driven machine learning software like Caffe (Jia et al., 2014). RETURNN extends RASR (Wiesler et al., 2014) to support various recurrent neural networks architectures in ASR systems.

2. Theano (Bastien et al., 2012) is a Python based framework that support the automatic differentiation for symbolic mathematical tensor expressions. Expressions modelling represents a computational dependency graph which can be augmented by an automatic optimization procedure. The abstract of each graph node can be defined for various types of hardware like CPUs or GPUs. By these properties, Theano is particularly useful for neural network training tasks.

3. Keras (Chollet, 2015) is a high-level data-driven machine learning framework based on Theano similar to RETURNN, Lasagne (Lasagne, 2017) and Blocks (Van Merriënboer et al., 2015). Recently, Keras starts to support TensorFlow in addition to Theano as back-end with minimal restrictions.

4. TensorFlow is the most recent open source machine learning package by Google (Abadi et al., 2016). It is actively developed and comes already with many predefined solutions such as LSTMs, end-to-end systems and others. TensorFlow is similar to Theano as it also works with symbolic computation graphs and automatic differentiation.

5. Torch (Collobert, Bengio, and Mariéthoz, 2002) have its own programming language "Lua" and consists of many flexible and modular components that were developed by the community. In contrast to Theano, Torch does not use symbolic expressions and all calculations are done explicitly.

6. EESEN (Miao, Gowayyed, and Metze, 2015) is a speech recognition toolkit which can be considered as an extension to Kaldi tool kit by adding rudimentary support for LSTMs networks. We used this toolkit for implementing our LSTM based handwriting recognition system.

There are some other notable NN-based frameworks which are written in C++ such as Caffe (Jia et al., 2014), some others are Python-based such as Neon (Neon, 2017) and Brainstorm (Brainstorm, 2015). A comparison between Caffe, Neon, Theano, and Torch was presented by (Bahrampour et al., 2015).

2.4 conclusion

In this chapter we aimed to present an overview of the theoretical bases of the optical models as one of two essential components for any handwriting recognition system (the optical models and language models). An optical models can by introduced either by HMM models or neural network model. Discrete, continuous or semi-continuous HMM model can be used as an optical model, which represent generative model for sequence of observations.

Nowadays, MLP , RNN and their extension, represent the mile-stone of a wide range of state of the art handwriting recognition systems thanks to their discriminative nature in decision making. Such optical models fall in the category of discriminative optical models, as opposed to HMM.

For the HMM based optical models, training the optical models can be achieved using the expectation maximization dynamic algorithms like Viterbi and forward-backward algorithms. Meanwhile, neural networks optical models can be trained by using gradient descent algorithms such as back-propagation, back-propagation through time and CTC learning algorithm. The RNN based recognition system perform better than those base on HMM as mentioned by the literature on the handwriting and speech recognition. $\tanh$

Chapter 3

State of the art of language modelling

3.1 Introduction

Language modelling is the art of determining the probability of a sequence of words (Joshua, 2001). Applications that deal with natural language data, (Written or spoken) require a representation of the characteristics of the treated language. For example, in case where the data are sentences, this representation gives the possible associations of words. One of the important language representation is the language model which is able to capture language regularities.

Spoken language and written language differ by their low level structures (phonemes for spoken language and characters for written language) but they follow the same concatenation rules for higher level structures (ex. when constructing sentences from "words").

Phonemes (ex. /ə/) are the primary units of speech in a language. Conventionally, the term ”phone” designates the acoustic realization of a phoneme. Phonemes, are classified into two main categories: vowels and consonants. Vowels are linguistically characterized with respect to the pitching position in the oral cavity [front - back], [high - low]. Consonants are characterized by constriction or obstruction of important pitching in the oral cavity. In English 40 phonemes are used to code the language, while 35 make up the French language, and only 25 phonemes are sufficient for coding the Italian language. One important thing to be considered here, is that vowels and consonants phonemes in speech do not have corresponding written representation. Therefore, the low level representation of spoken and written languages do not match. Some language specific rules allow converting the spoken representation to the written representation of that language.

In this chapter, we first give an overview of the spoken and written language building units which represent the bricks that must be defined before building any language model. Then we present a comparison between structural modelling of the language (representation that stands on a set of rules for the composition of sentences) and statistical language modelling that is based on the frequency computations of word occurrences in a sentence. These two approaches represent the two main approaches for natural language modelling. Then, we focus more in depth on the statistical modelling approach of

language, since it is more adapted to our problem, and we compare in particular different techniques of language modelling before finishing by introducing some applications.

3.2 Language building units

According to the literature (Shaik et al., 2011; Xu et al., 1996), we can classify the spoken / written language building units from the lexical point of view into three main categories (see Tables 3.1): lexical units, sub-lexical units and primary units. In the following sections we try to illustrate and explain this taxonomi of the human spoken and written language.

<table>
<tr><th>language units</th><th>lexical units</th><th>sub-lexical units</th><th>primary units</th></tr>
<tr><td rowspan="3">Speech</td><td rowspan="3">Words</td><td>morphemes</td><td rowspan="3">phonemes</td></tr>
<tr><td>phonotic syllables</td></tr>
<tr><td>graphones</td></tr>
<tr><td rowspan="4">Writing
htop</td><td rowspan="3">Words</td><td>morphemes</td><td rowspan="3">Characters</td></tr>
<tr><td>graphemic syllables</td></tr>
<tr><td>Hyphens based</td></tr>
<tr><td colspan="3">Alphanumeric characters</td></tr>
</table>

TABLE 3.1: spoken & written language components

3.2.1 Primary units

Both spoken and written languages are based on unambiguous primary units which are either phonemes of spoken languages or characters and alphanumeric symbols of written languages. Phonemes are represented ambiguously using a phonetic alphabet (Alphabet Phonetique International (International-Phonetic-Association, 1999) but written languages do not use this alphabet, they use one of the many writing systems that have been introduced through history.

The type of writing system used for a language is the most important factor for determining the nature of the language primary units. Writing systems can be logographic, where a large number (often thousands) of individual logograms (symbols) represent words (Indurkhya and Damerau, 2010). A logogram or logograph is a written symbol that represents a word or a phrase. Chinese characters and Japanese kanji alphabet are logograms; some Egyptian hieroglyphs and some graphemes in cuneiform scripts are also logograms. In contrast, writing systems can be syllabic, in which individual symbols represent syllables, or alphabetic, in which individual symbols (more or less) represent sounds; unlike logographic systems, syllabic and alphabetic systems typically have fewer than 100 symbols.

Syllabic alphabets, are writing systems in which the main element is the syllable. Syllables are built up of consonants, each of which has an inherent vowel. For example in Devangari writing system like "माझे लेटेक् देवनागरीचे प्रयोग.", diacritic symbols are used to change or mute the inherent vowel, and separate vowel letters may be used when vowels occur at the beginning of a syllable or on their own. The majority of the written languages use an alphabetic or syllabic system (Comrie, Matthews, and Polinsky, 1997). However, in practice, no modern writing system employs symbols of only one kind, so no natural language writing system can be classified as purely logographic, syllabic, or alphabetic (Indurkhya and Damerau, 2010).

In writing, the alphanumeric characters and symbols such as punctuation marks sometimes attach the beginnings or/and the ends of the meaningful language words like in the French sentence *"C'est intéressant,"* or in the English sentence "*I 'd like to jump!!*". The punctuation characters (’), (,) and (!!) defines the reading style of the handwriting. For example, the couple of characters (’d) represents an appreviation to the sentence "I would", while the punctuation character (’) does not make sense in a sentence if it was written alone. However, there are some other alphanumeric characters that are considered as a meaningful language word such as digits, mathematical symbols and question marks. Thus, meaningful alphanumeric characters and symbols can be considered as lexical, sub-lexical and primary building units the written language.

3.2.2 Lexical units

For the spoken and written language, the word is the lexical unit of a language that hold a significant meaning. Regarding to the writing systems, it is not easy to determine what is meant by "word". Typically people initially think of items in a sentence separated by silence or white spaces or certain types of punctuation marks. In languages such as French, English, German or Arabic such concepts can be applied. In languages such as Bengali, chines, Japanese or Thai, there are no separation mark between words, because a sequence of adjacent characters can represent multiple words.

The notion of word is very important for most of the evaluation protocols, a word being defined as a sequence of characters sided by two white space for writing or by silence for speech.

3.2.3 Sub-lexical units

Sub-lexical units are used in speech recognition as the acoustic model building blocks rather than phonemes (Choueiter, 2009). Sub-word units are also used for language modelling (Shaik et al., 2011; Mousa et al., 2010; El-Desoky et al., 2009). For highly inflected languages such as German (Shaik et al., 2011) or Arabic (Märgner and El Abed, 2012), a large amount of words are derived from the same root due to multiple factors such as inflection, derivation and compounding. This morphological flexibility leads to high OOV rates and large language model perplexity. For such languages, sub-word units allow

modelling OOV with lower perplexity language models. The perplexity is defined in information theory (Shannon, 1948) as a measurement of how well a probability distribution or probability model predicts a sample. It is used for comparing probability models such as language models. A low perplexity indicates the probability distribution is good at predicting the sample.

One important issue of language modelling with sub-lexical units is the proper choice of the sub-word units. According to Shaik (Shaik et al., 2011), sub-word units are classified into three main types:- morpheme based sub-word units, syllable based sub-word units and graphone based sub-words units. While lexical units are defined with no ambiguity as the language words, sub-lexical units have at least five main definitions based on two main decomposition strategies:

Spoken language dependent word decomposition into sub-lexical units

The strategy of word decomposition into sub-lexical units relies on some spoken language aspects such as word morphology and word pronunciation that generate the following three types of sub-lexical units:-

1. **Morpheme sub-lexical units**: the decomposition of a word according to its morphological structure produces sub-lexical semantic units called *morphemes* which represent the prefix, root and suffix parts of the word (ex. **non-**, **perish**, and **-able** as prefix, root and suffix respectively).

   The morpheme is known as the smallest linguistic component of a word that has a semantic meaning. A word can be decomposed into a sequence of morphemes by applying either supervised or unsupervised approaches. Supervised approaches (Adda-Decker and Adda, 2000) use a set of linguistic rules designed by an expert. Unsupervised approaches are almost statistical based data driven approaches (Adda-Decker, 2003). The proposed decompounding algorithm is corpus-based and language independent in the sense of no linguistic knowledge is required, excepting a large text corpus. Word-specific decompounding rules are automatically extracted based on the branching factor of successor characters.

   Morphemes have been used by Shaik (Shaik et al., 2011) for speech recognition and by Hamdani et al (Hamdani, Mousa, and Ney, 2013 for handwriting recognition. Hamdani & al. proposed an Arabic handwriting recognition system based on HMM models using a mixed language model of words and sub-word units. The vocabulary contains words and sub-word units produced by a specific morphological decomposition method of Arabic words which is based on the semantic decomposition of words into roots, suffixes and prefixes (Creutz et al., 2007). The results show the improvement provided by this system, notably to cope with OOV, compared to a lexicon driven recognition system.

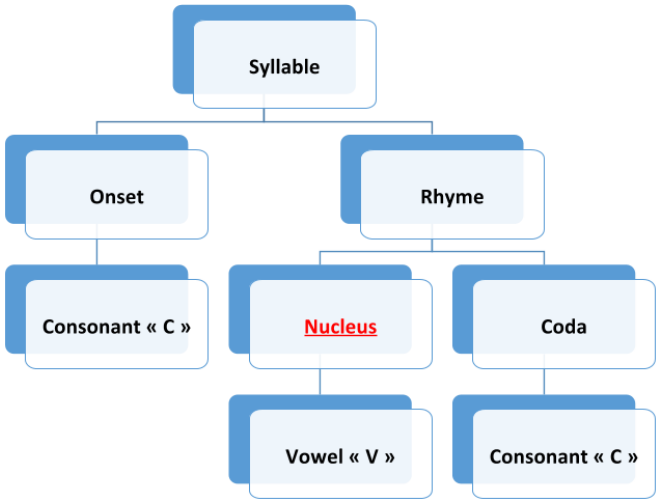
2. **Phonetic syllable sub-lexial units**: Syllables are sub-lexical units that are based on a phonetic decomposition. They are sometimes defined physiologically as a continuous unit of the spoken language which consists of

a sound or a group of sounds uttered in one breath (Ryst, 2014; Brown, 2006) for example the french word (bibliothécaire) has the phonological syllable representation (**bi - bli - ɔ - te - k**ɛʁ).

Syllables are generally centered around the vowels in English (Huang et al., 2001), French (Ryst, 2014) and German (Shaik et al., 2011). Acoustic models of syllables and syllable-like units are implemented with success in (Choueiter, 2009; Ganapathiraju et al., 2001) and successful implementation of syllabic language models are presented in (Xu et al., 1996) for Chinese, (Majewski, 2008) for Polish, (Schrumpf, Larson, and Eickeler, 2005) for English and German language (Shaik et al., 2011). Syllabic models based on hidden Markov model (HMM) are examined in (Jones, Downey, and Mason, 1997) and compared with mono-phones HMM based acoustic models using a bigram language model. A Hybrid syllable and phonetic based model was proposed by (Sethy, Ramabhadran, and Narayanan, 2003) and the system shows better performance than systems that use only one of these two models.

The syllable plays an important role in the organization of speech and language (Kozielski, Doetsch, and Ney, 2013; Ryst, 2014). The syllable consists of one or more written letters representing a unit of speech called phoneme. The segmentation of speech into syllables can be achieved using acoustic units or phonological units (Ridouane, Meynadier, and Fougeron, 2011), and syllables produced by these two models are not always compatible (Ryst, 2014). As illustrated in figure 3.1, most phoneticians agree that a syllable is composed basically of a rhyme that is preceded by an onset (one or more consonants "**C**" optionally comes at the beginning of the syllable). Inside a rhyme, the nucleus (usually a vowel "**V**") is the constitutive element of the syllable. This is followed by a coda (one or more consonants "**C**" at the end of the syllable) (Ryst, 2014). The languages differ from each other with respect to topological parameters as optionality of the onset and admissibility of the codas. For example, the onsets are mandatory in German while the codas are prohibited in Spanish (Bartlett, Kondrak, and Cherry, 2009). In French and English, the nucleus is always considered as a vowel. Thus, counting the number of syllables pronounced in a French or English, should be equivalent to counting the number of pronounced vowels (Ryst, 2014). The phonetic structure does not match the orthographic structure because of the significant difference between the vowel phoneme and the vowel characters as illustrated in figure 3.1

Considering written languages, and according to (Flipo, Gaulle, and Vancauwenberghe, 1994), in French the orthographic syllable differs from the phonetic syllable because it retains all "e" silent placed between two consonants or placed at the end of a word. Hyphenation rules separate the double consonants even if they are pronounced as a single consonant. For example, graphically there are three syllables in the French word ***pu-re-té*** even if we pronounce it as [***pyr-te***] (two phonetic syllables). The



Words	Syllables	phonetic structure	orthographic structure
bonnement	bO-n°-m@	CV-CV-CV	CVC-CV-CVCC
récente	Re-s@t	CV-CVC	CC-CVCCV

FIGURE 3.1: Syllable phonetic components

authors of (Roekhaut, Brognaux, and Beaufort, 2012) have classified the syllables in three different categories:

- A ***phonetic syllable*** is composed of a combination of phonemes that are pronounced in a single breath.
- A ***graphemic syllable*** represents a faithful transposition of phonetic syllabification in the spelling of the word.
- An ***orthographic syllable*** applies hyphenation rules that must belongs to writing.

It seems difficult to integrate these different views of specialists into a recognition system, but in any case, only graphemic or orthographic syllables provide a decomposition of writing that is likely to have an impact on a recognition system.

The Syllable based n-gram language model shows through our experiments good language modelling capacity with reduced language model complexity as demonstrated in chapter 5, in addition to its ability to cover a wide range of Out-Of-Vocabulary words. The syllable based n-gram model suffer of losing word context with lower values of n-gram orders. Generally, it represents a good compromise between the word based language model and the character based language model for sufficiently hight order n-gram language models.

3. **Graphemic syllable sub-lexial units**: derived from phonological syllables, graphemic syllables have been introduced for written languages. The same example "bibliothécaire" has the following graphemic representation (**bi - bli - o - thé - caire**) . Thus, the graphemic syllable always

respects the boundaries of the phonetic syllable. The graphemic syllable seems to occupy a prominent place in the learning process of the language. Several researchers believe that it is one of the basic units from which to build gradually the recognition of the words of the language (Lecocq, 1991). A graphemic syllabic algorithm that relies on the strict link between graphemes and phonemes of French was introduced by (Roekhaut, Brognaux, and Beaufort, 2012). In this context, we introduced a supervised syllabification algorithm (Swaileh and Paquet, 2016b) that allow to decompose any French or English word into its syllable decompositions. This approach will be developed in chapter 5.

4. **Hyphen based sub-lexical units**:

   The hyphen $(-)$ is a punctuation mark used to join words and to separate syllables of a single word. The use of hyphens is called hyphenation. In writing, the hyphen is a single entity. In terms of character encoding and display, this entity is represented by any of several characters and glyphs (including hard hyphens, soft or optional hyphens, and nonbreaking hyphens), depending on the context of use. Hyphens are occasionally used to denote syllabification, as in (**syl-la-bi-fi-ca-tion**).

   There are a few hyphenation rules that will let one hyphenate almost all English words properly as presented in (Beeton, 2010):

   - Break words at morpheme boundaries (inter-face).
   - Break words between doubled consonants (bat-tle).
   - Never separate an English digraph (e.g., th, ch, sh, ph, gh, ng, qu) when pronounced as a single unit (au-thor but out-house).
   - Never break a word before a string of consonants that cannot begin a word in English (jinx-ing and not jin-xing).
   - Never break a word after a short vowel in an accented syllable (rapid but stu-pid).

   The rules above leave more than one acceptable break between syllables, use the Maximal Onset Principle: If there is a string of consonants between syllables, break this string as far to the left as you can (mon-strous).

   Sometimes the rules conflict with each other. For example, ra-tio-nal gets hyphenated after a short vowel in an accented syllable because ti acts as a digraph indicating that the 't' should be pronounced 'sh'.

   Sometimes it's not clear what constitutes a morpheme boundary: why ger-mi-nate and not germ-i-nate?

   These rules are not mandatory and other rules can be used like the simplest rule of Mikolove (Mikolov et al., 2012) which split words at vowels and limit the minimum size of the sub-lexical units to 2 characters.

Spoken language independent word decomposition into sub-lexical units

Some heuristic and probabilistic approaches are used to produce sub-lexical units with no care about semantic or syntactic aspects of the language of interest. These kind of decomposition into sub-lexical units approaches have been used with success by Shaik (Shaik et al., 2011) for speech recognition based on a grapheme-to-phoneme (G2P) conversion model as described in (Bisani and Ney, 2008), Benzeghiba (BenZeghiba, Louradour, and Kermorvant, 2015) for Arabic handwriting recognition using the definition of the Part-Of-Arabic word (PAW) and Feild (Feild, Learned-Miller, and Smith, 2013) for text scene recognition using a probabilistic context free grammar (PCFG) model.

1. **Graphone sub-lexical units**: One of speech sub-lexical units is the graphone. The graphone units are derived from a grapheme-to-phoneme (G2P) conversion model (Bisani and Ney, 2008). It is a combination of the graphemic sub-word units with their context dependent pronunciation forming one joint unit (Shaik et al., 2011), in other words, it is the joint character-phoneme sub-lexical unit (Wang, 2009) as illustrated in figure 3.2.

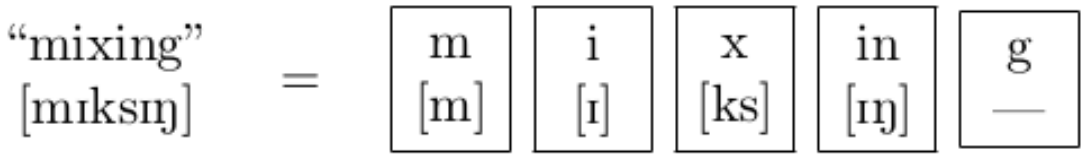


FIGURE 3.2: Illustration of a sequence of graphones (Mousa and Ney, 2014)

A model estimated over such units represents an implicit combination of language model and pronunciation model in one joint probability distribution (Shaik et al., 2011). Graphones are mainly used for modelling OOV. They can be derived directly from Grapheme-to-phoneme conversion as illustrated in (Bisani and Ney, 2008), or by using fixed-length subwords units without any linguistic restrictions like in (Bisani and Ney, 2005). A combination of the three types of sub-lexical units was studied in (Shaik et al., 2011) for German and a hybrid speech recognition system has been implemented using a hybrid lexicon. In this system, the most frequent words of the language are used normally as full-words without decomposition into sub-word units. Less frequent in-vocabulary words are decomposed into syllables. Finally, the OOV words are decomposed into graphones.

2. **Other spoken language independent decomposition into sub-lexical units**: In the literature, there are few researchers who are using sub-lexical units in handwriting. Few of them used the Part-Of-Arabic word (PAW) sub-lexical units in order to tackle the problem of recognition confusion between inter-word and intra-words white space in Arabic handwriting recognition system as illustrated by BenZeghiba (BenZeghiba, Louradour, and Kermorvant, 2015).

BenZeghiba & al. (BenZeghiba, Louradour, and Kermorvant, 2015) proposed a hybrid model for the Arabic language, which is designed according to the observed frequency of words. The idea is to keep the most frequent words as is (without decomposition), and to decompose only the least frequent words into sub-word units or PAWs (Part of Arabic Words). By taking advantage of the specific property of the Arabic language and script, a PAW is a sequence of characters that can be connected together. A character that cannot be connected with its following neighbor defines the end of a PAW (AbdulKader, 2008). The advantage of a hybrid word - PAW model lies in the trade off between its modeling and generalization ability and its reduced size. The two models (hybrid word / PAW) and PAW alone perform almost similarly on OOV words but the hybrid system is less complex. Statistical models such as Probabilistic Context Free Grammar (PCFG) and multigrams model are used for decomposing words into sub-lexical units as illustrated by Feild & al. (Feild, Learned-Miller, and Smith, 2013).

Regarding English language, Jacqueline & al. (Feild, Learned-Miller, and Smith, 2013) proposed a probabilistic syllable model for scene text recognition such as advertisements panels. They used a probabilistic context free grammar (PCFG) that models each sub-lexical unit as a sequence of characters generating a consonant or a vowel. The proposed model shows good performance on OOV words such as proper nouns, under the condition that these words should all be pronounceable in English. The PCFG model was built by learning a set of parsing rules of characters using a syllabified dictionary. As a defect of this model, the method does not use any information about successive syllables which leads to a loss of context modelization of syllables.

The multigram definition can be considered as an extension to the n-gram definition where the suffix "gram" can be defined as a language unit of word, syllable, multigram or character. While the n-gram is defined as the sequence of language units of fixed and limited size n, the mulitgram can be defined as a sequence of language units with variable length. Table 3.2 shows some examples of the n-grams and multigrams of order $n = (1, 2, 3, 4, 5)$ for the French word "Merci" and the english word "Darling".

In the field of speech recognition and language processing, the choice of sub-lexical units that structure the data stream affects massively the efficiency of the recognition algorithms. On the one hand, sub-lexical units may be determined by relying on some linguistic knowledge but with the risk of being inadequate to the particular task at end, due to the specificity of the data. On the other hand, it can be determined directly from the data using machine-learning approaches.

Former works in speech recognition such as (Chou and Lookabaugh, 1994; Sagisaka and Iwahashi, 1995; Bacchiani et al., 1996; Ries, Buo, and Wang, 1995) show the impact of determining the sub-lexical units from

	Lexicon type	French examples	English examples
n-grams of characters	uni-gram	M e r c i	D a r l i n g
	bi-gram	Me, er, rc, ci	Da, ar, rl, li, in, ng
	tri-gram	Mer, erc, rci	Dar, arl, rli, lin, ing
	4-gram	Merc, erci	Darl, arli, rlin, ling
	5-gram	Merci	Darli, arlin, rling
Multigrams decompositions	m2gram	Me, r, ci	d, ar, li, ng
	m3gram	Mer, ci	dar, l, ing
	m4gram	Merc, i	dar, ling
	m5gram	Merci	dar, ling

TABLE 3.2: n-grams versus multigrams examples

the data directly, and yielded to the multigram model (Deligne and Bimbot, 1997). The multigram model was first presented in (Bimbot et al., 1994) for language modelling by taking into account the variable length dependencies of words in a sentence. Theoretical formulation and evaluation of multigrams is presented in (Deligne and Bimbot, 1997; Deligne, Yvon, and Bimbot, 1996). It assumes the language produces a stream of words that flow off a memoryless source (Deligne and Bimbot, 1995). The language words are not independent, and their dependencies are of variable length. It can be opposed to the popular n-gram model (Jelinek, 1990) which assumes that the statistical dependencies between words are of fixed length (n) along the whole flow of words (Deligne and Bimbot, 1995), thus assuming an n-order Markov models of word streams. In the n-multigram model (Bimbot et al., 1994), a sentence is considered as the concatenation of independent (zero order Markov source) sequences of words of length k (with $k \leq n$). Such concatenation of words (multigram) is commonly known as phrase in the natural language processing literature. Then, the likelihood of the sentence is computed as the product of the individual phrases's likelihood corresponding to each possible segment (multigram) (Deligne and Bimbot, 1995).

More formally, the multigram model is defined as a stochastic generative model of variable length units (Deligne and Bimbot, 1997). These units can be word phrases when considering language modelling, or they can be character or phoneme sequences when considering handwriting or speech recognition. Let us assume that a lexical source produces a set of M sub-lexical units (multigrams) which are sequences of length k characters ($k \leq n$). While each data stream produced by the source can be viewed as a sequence of T characters which are the primary units of the observation sequence $O = (c_1, c_2, \ldots, c_t, \ldots, c_T)$, it can also be viewed as a sequence $S = (m_1, m_2, \ldots, m_l, \ldots, m_L)$ of multigrams of length $L(L \leq T)$, each multi-gram m_l being the concatenation of k characters $m_l = (c_t, \ldots, c_{t+k-1})$. As depicted on figure 3.3 below, the multi-gram decomposition corresponds to a particular segmentation of the input character string.

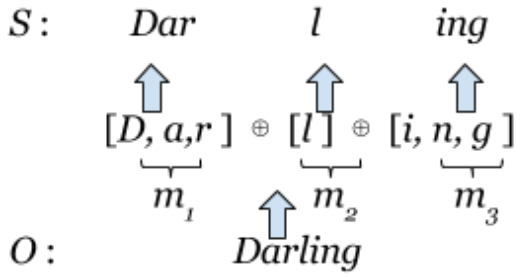


FIGURE 3.3: Multigram structure and an example of word decomposition into multigrams

In the literature, this generative process of multigrams from the discrete observation stream of characters has been modeled using variable length Hidden Markov Models, assuming time independent multigrams (Deligne and Bimbot, 1997). Then, the likelihood of a particular multigram decomposition of the input string is computed as follows

$$P(O,S) \triangleq P(O|S)P(S) = \prod_{l} P(c_t c_{t+1} \dots c_{t+k-1}|m_l)P(m_l) \tag{3.1}$$

or more precisely

$$P(O,S) = \prod_{l} P(c_t c_{t+1} \dots c_{t+k-1}|d_l = k)P(d_l) \tag{3.2}$$

where d_l is the character duration of multigram m_l. The main interest of such modelling is to allow finding the optimal decomposition into multigrams of a training corpus of text, thus providing an appealing approach for the definition of sub-lexical units for any language without relying on linguistic expertise for each dedicated language, the optimization process being unsupervised. In Deligne and Bimbot, 1997 the authors have applied the multigram model to derive sub-words phonetic units from phonetic transcriptions of utterances using continuous observations sequences. They also applied the model to derive the graphemic structure of text, considering discrete character observations. In their *EM* iterative optimization process, they introduced a heuristic by selecting the more frequent multigrams at each iteration, so as to optimize not only the data likelihood but also the complexity of the model so as to get better generalization capabilities of the model, following the Minimum Description Length (MDL) principle. In Deligne and Sagisaka, 2000 the same authors apply the multigram paradigm to language modelling (sequences of words) and they report lower perplexity for the multigram model compared to the n-gram model.

3.2.4 Definitions

We present by the following, some essential definitions that help to identify and evaluate the relationship between a training vocabulary / dataset and a

test dataset.

- The ***word***: standing on the standard handwriting recognition performance evaluation protocol of the Word Error Rate (WER), a word is defined as a sequence of characters sided by two white spaces. With defining the character "|" as the word delimiter character, we consider in the following coloured sentence six different words The French sentence

  |La| |journée| |TALADOC| |s'organise| |le| |2/6/2017|

  above contains an example of an entity name words, abbreviation word and a date word. These words represent real problem while training the language model, that because they may have rare occurrences in the language model training corpora. For Building robust language model, the choice is to tokenize the training corpora and splitting the rare occurrence words into sequences of alphanumeric words and then use the token building unit in place of the words for building the language models.

- The ***token***: it defines the language units used for building a language model that can be a word or a smaller language units than a word, such as sub-lexical, characters or orthographic symboles (e.g. ?,!,$,@,",...etc). Looking at the sentence above, the token delimiter character "|" defines

  |La|_|journée|_|TALADOC|_|s|'|organise|_|le|_|2|/|6|/|2|0|1|7|

  twenty tokens that are derived from six words.

We need to deal with these two definition differently because we need to quantify the ability of the training corpus to cover the test corpus in terms of words and then we can study the relation between the training corpus coverage with the handwriting recognition performance measured by the Word Error Rate. We need also to quantify the ability of a language model of tokens to cover the tokenized test corpus in order to study the effect of language model missing words on the recognition performance, specially when the missing words depend on the training corpus tokensiation strategy: lexical tokens, sub-lexical tokens and primary tokens. Based on the word and token definitions, we introduced two measures:

- The ***effective coverage rate (ECR)***: The proportion of words of the test dataset that belongs to the training dataset as illustrated in figure 3.4. It measures the lexical proximity between two datasets of texts.

- The ***Out-Of-Vocabulary (OOV)***: The proportion of tokens of the test dataset that do not belong to the language model lexicon (figure 3.4). It evaluate the capacity of the language model to cover the test dataset.

The lexicon represents the essential linguistic source of information on which the language model training is based. Furthermore, the recognition systems complexities are determined standing on their lexicon sizes. The lexicon and

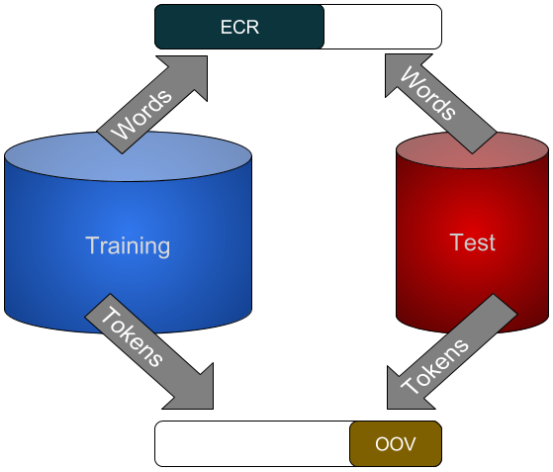


FIGURE 3.4: Illustration of the Effective Coverage Rate (ECR) versus the Out-Of-Vocabulary

language models building units (lexical, sub-lexical and primary units) define their architecture type.

In the literature, the recognition systems are sometimes characterised by their vocabulary type which can be classified in two vocabulary types. Closed vocabulary systems and open vocabulary systems each of which can be small or large in size. The small vocabularies are the vocabulary whose size is less than 10k token and the large vocabulary is the one which include than 10k token.

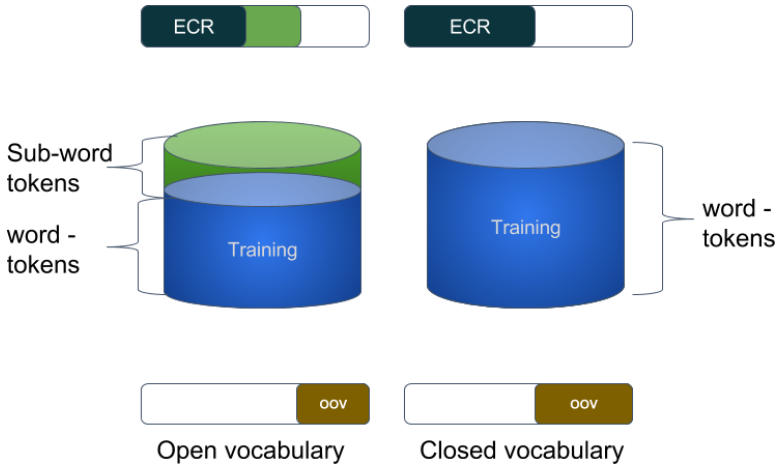


FIGURE 3.5: Illustration of the closed and open vocabularies

- ***closed vocabulary***: in this case the vocabulary tokens are words of a specific lexicon. The language model that is trained with this vocabulary can only identify the words of this lexicon, any other word is an Out-Of-Vocabulary word.

  This is convenient for small vocabulary recognition tasks which are restricted to the recognition of a limited set of words with almost no OOV.

- ***open vocabulary***: this vocabulary language models is composed of tokens that are words and of some other tokens which represent a portion of the OOVs. By these tokens some words that do not belong to the lexicon can be reconstructed, thus the effective coverage rate increases as a result of the OOVs reduction. In figure (figure 3.5) the ECR in blue colour represents the effective coverage rate obtained by the word tokens of the language model and the green one represents the ratio of the reconstructed words thanks to the other sub-words tokens. These tokens can be produced by replacing the words with rare occurrence in the training corpus by the OOV token (<unk>) or by their set of characters (the green part of figure 3.5 represent the portion of the training lexicon which hold the OOV tokens). In the literature, the recognition system that make use of the open vocabulary show slight performance degradation in comparison with the closed vocabulary based system performance.

In the following section, we give an overview on the natural language modelling approaches and some of their applications in real world applications.

3.3 Why statistical language models for handwriting recognition?

Natural language modelling can be achieved with two main approaches: structural and statistical approaches. In this section, we present these two approaches and compare their advantages and disadvantages, to justify our choice to work with the statistical approaches for the handwriting recognition.

In the structural approach, a language model is often defined as a set of rules that build the language grammars. The set of such rules defines the possibilities of word association, according to their lexical categories (noun, verb, adjective,.. etc), and makes it possible to model the structure of a given sentence (Chomsky, 2002). Therefore, these rules determine the validity of a sentence: if a sequence of rules can describe the sentence under examination then the sentence will be accepted, otherwise it will rejected by the model.

In the statistical approach, the language is represented by the occurrence frequencies of each possible sequence of words. For this purpose, a representative corpora (consists of blocks of texts about the domain of interest) are used for training the language model by the statistical inference algorithm (Manning and Schütze, 1999).

One of the advantages of statistical approaches is their independence from the language experts, as the language models are learned automatically from corpora. Modelling of the language following this approach is therefore dependent on the data used and can also attribute a non-zero probability to syntactically incorrect sentences which represent a drawback point for the statistical language modelling approaches. One of the advantages of models based on structural approach is their ability to express the grammatical nature of the sentences. Nevertheless, one of the main difficulties of these methods remains the conception of the grammars, which means the definition of rules. Indeed,

they must be robust enough to take into account all the syntactic hypotheses of a language.

It seems difficult for continuous handwriting recognition to develop a set of rules that describes the possible writing sentences specially when such sentences includes punctuation marks and written symbols associated with the language meaningful words.

Due to the statistical nature of the language models constructed according to the statistical approach, their integration into a recognition system is more effective. Statistical language models have a great margin of flexibility in decision making while they depend on the associated probability of the language sentences. This probability evaluates whether or not the sentence is correct, in contrast to the structural approaches that associate a boolean decision of accepting or rejecting the sentence.

However, there are language models that combine both structural and statistical approaches, such as Probabilistic-Context-Free-Grammars (PCFG) which allow more flexible response by expressing the sentence validity in a probabilistic way. The use of PCFG models is more appropriate for tasks whose domain is limited in vocabulary (because of the structural approach nature), such as address recognition (Srihari and Kuebert, 1997) or speech recognition with a specific and limited vocabulary (Jurafsky et al., 1995).

In the case of large vocabulary domains, these structural language models can be used in a second recognition pass to rearrange a list of sentences (Hacioglu and Ward, 2001) or in combination with a statistical language model (Zimmermann, Chappelier, and Bunke, 2006). In the latest cases, a statistical model of the language is used jointly with the structural PCFG model.

Due to, the efficiency of integrating language models based on the statistical approaches into the handwriting recognition system and the simplicity of their learning procedures, we focused our work on the study of the statistical language modelling approaches, specially the back-off n-gram language modelling approach of the sub-lexical units.

3.4 Statistical language modelling

The goal of statistical language modelling is to predict the next word in textual data given passed context; thus we are dealing with sequential data prediction problem when constructing language models (Mikolov et al., 2010). We first present a general overview of the statistical language modelling principle. Then, we describe different measures to evaluate the language models quality. Finally, we focus on different language models and techniques for combining language models.

3.4.1 General principles

Statistical modelling of language aims to capture the regularities of a language by statistical inference on very large training corpora of text, composed of a set of topics in the language of interest (Manning and Schütze, 1999).

A statistical language model makes it possible to evaluate the a *priori* probability of a sequence of N words $W = w_1 \dots w_N$. This probability is equal to the product of the probabilities of the words within its context as the following:

$$P(W) = \prod_{i=1}^{N} P(w_i|h_i) \tag{3.3}$$

Where $P(w_i|h_i)$ is the occurrence probability of the word w_i knowing that it comes directly after the sequence of words $h_i = w_1, \dots w_{i-1}$. These probabilities $P(w_i|h_i)$ constitute the parameters of the language model. The sequence of words h_i is called the history of the word w_i. So for $P(w_i|h_i)$ to be computed for $i = 1$, a start-of-sentence marker $< s >$ is added. We thus have $h_1 =< s >$. Moreover, an end-of-sentence marker $< /s >$ is added and included in the product of the probabilities of the words, then we can re-write equation 3.3 as follow:

$$P(W) = \prod_{i=1}^{N+1} P(w_i|h_i) \tag{3.4}$$

where $w_{N+1=</s>}$

3.4.2 Language model quality evaluation

The role of a language model is to estimate the *a priori* probability $P(W)$ of a sequence of words W. Thus, as much as a word sequence W conform to the language model, its probability will be higher. The simplest measure for evaluating the quality of a language model is by measuring the affected probability upon a test data set T which is considered to be representative of the language of interest and consists of a set of sentences $(W_1 \dots W_N)$ where N is the number of sentences in T. From the probabilities $P(w_i|h_i)$ given by the language model LM, the probability of the sentences can be calculated using equation 3.4 and the probability of the test dataset T is then given by:

$$P(T) = \prod_{i=1}^{N} P(W_i) \tag{3.5}$$

Thus, the language model assigning the highest probability to the test dataset T will be considered as the best model. This probability comes from the relationship between prediction and compression, resulting from information theory (Shannon, 1948): given a language model LM which assigns a probability $P(T)$ to a test dataset T, a compression algorithm can be created to encode the text to $-\log_2(P(T))$ bits. Thus, the cross entropy H_{LM} of a language model LM, on a test dataset T, is defined by:

$$H_{LM}(T) = -\frac{1}{|T|_w} \log_2(P(T)) \tag{3.6}$$

Where $|T|_w$ is the number of words in T The cross entropy evaluates how different the language model is with the empirical distribution of the dataset.

This value can be interpreted as the average number of bits needed to encode each of the $|T|_w$ words of the test dataset, using the compression algorithm associated with the model LM. Therefore, the best language model will be the one which has the smallest cross entropy.

The *Perplexity* is a measure derived from cross entropy, which is in fact the most used measure to evaluate the performance of a language model. The perplexity PP_{LM} of a language model LM on a test dataset S corresponds to the reciprocal of the mean probability given to each of the words of the test dataset T. It is related to the cross entropy by the equation:

$$PP_{LM} = 2^{H_{LM}(T)} \tag{3.7}$$

This quantity accounts for the predictive power of the model LM. Thus, $PP_{LM}(T) = k$ means that the language model LM hesitates on average between k words. Thus, like cross entropy, the lower perplexity the best language model.

In the context of handwriting recognition, it may also be advantageous to measure the recognition rates obtained on a test set, using different language models in a recognition system. In fact, we will rather evaluate the error rate on words obtained on the test set, this indicator being well correlated with perplexity (Klakow and Peters, 2002): the lower the error rate, the better performance of the recognition system.

The main problem of equation 3.4 is the high number of different historical data, which leads to a very large number of probabilities to be estimated. Moreover, most of these word histories appear so few times in the training corpus which does not help in estimating their probability in a sufficiently reliable way. One solution to solve this problem is to group word histories into equivalence classes called n-grams:

$$P(w_i|h_i) \approx P(w_i|\Phi(h_i)) \tag{3.8}$$

While $\Phi(h_i)$ associates a class of equivalence to each word history h_i. An n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles (Broder et al., 1997). An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (2-gram) (or, less commonly, a "digram"; size 3 is a "trigram" (3-gram). Larger sizes are sometimes referred to by the value of n-gram.

Unit	Sample sequence	2-gram sequence
character	...to_be_or_ not_to_be...	..., to, o_, _b, be, e_, _o, or, r_, _n, no, ot, t_, _t, to, o_, _b, be, ...
word	... to be or not to be ...	..., to be, be or, or not, not to, to be, ...

TABLE 3.3: n-gram examples

3.4.3 N-gram language models

N-gram language models are the most widely used language models (Manning and Schütze, 1999). They are based on sequences of n words, the value n is called the order of the n-gram model: the word histories that end with the same $n-1$ words are considered to be equivalent. The name of the "n-gram models" comes from the domain of the Markov models of which these models are instantiated. Thus, the n-gram language model can be seen as a Markov model of order $n-1$. From Equation 3.8 we obtain:

$$P(w_i|\Phi(h_i)) = P(w_i|w_{i-n+1}^{i-1}) \tag{3.9}$$

Where $w_{i-n+1}^{i-1} = w_{i-n+1} \dots w_{i-1}$ is the word history reduced to $n-1$ words that are the predecessor of the word w_i. The probabilities $P(w_i|w_{i-n+1}^{i-1})$ of the equation 3.9 are generally estimated by the maximum likelihood estimation (MLE) on the training set:

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{N(w_{i-n+1}^{i-1} w_i)}{\sum_{w_j \in V} N(w_{i-n+1}^{i-1} w_j)} \tag{3.10}$$

Where $N(.)$ is the number of occurrences of the word sequence of interest, in the training corpus, and V the vocabulary.

Let's take a practical application of a bigram language model on the following text corpus:

<s> JOHN READ MOBY DICK </s>
<s> OPEND READ BOOK LIBRARY </s>
<s> MARY READ A DIFFERENT BOOK </s>
<s> SHE READ A BOOK BY CHER </s>

The probability of the sentence "JOHN READ A BOOK" according to equation 3.10 is the following, assuming $n = 2$.

p(<s> JOHN READ BOOK </s>)

$= p(\text{JOHN} \mid \text{<s>}) p(\text{READ} \mid \text{JOHN}) p(\text{BOOK} \mid \text{READ}) p(\text{</s>} \mid \text{BOOK})$

$$= \frac{N(\text{<s> JOHN})}{\sum_w N(\text{<s> w})} \cdot \frac{N(\text{JOHN READ})}{\sum_w N(\text{JOHN w})} \cdot \frac{N(\text{READ BOOK})}{\sum_w N(\text{READ w})} \cdot \frac{N(\text{BOOK </s>})}{\sum_w N(\text{BOOK w})}$$

$$= \frac{1}{4} \frac{1}{1} \frac{1}{4} \frac{1}{3} \approx 0.021$$

While the probability of another sentence like "CHER READ BOOK" will be zero because the number of occurrence of the bigram CHER READ is equal to zero. In practice, the commonly used order n of a n-gram language model of words (lexical units) is generally equal to 2 (bi-gram model) or 3 (trigram

model) or even 4 or 5 but rarely beyond since the performances of the language model slightly improve for higher order (Goodman, 2001) while the number of parameters to be estimated increases exponentially with n. Indeed, for a trigram model with a vocabulary of 20000 words, the number of probabilities to be estimated is $8.10^{12} = (20000)^3$. This limitation of the order n makes it difficult to take into account long dependencies of the language sentences. Another limitation of this approach is that admissible n-grams from a linguistic point of view may have zero probability if they do not appear in the training corpus.

3.4.4 Lack of data problem

Despite the reduction in the number of probabilities to be estimated, by regrouping the word histories in classes of equivalence (n-gram models), some possible word sequences may not appear in the training corpus: their probability of occurrence will be estimated as zero while it should not. To tackle this problem, one technical solution is to assign a non-zero probability to the sequences that do not appear in the learning corpus, using a smoothing model. A general solution is based on the estimation of the conditional probability $\hat{P}(w_i|w_{i-n+1}^{i-1})$ by combining two components: a discounting model and a redistribution model which represent together the smoothing model. The occurrence probabilities of the word sequences found in the training set are to be discounted, then the discounted probability is redistributed among the n-grams (equivalence classes) of null occurrences.

Discounting probabilities

The discounting model is used to deduct a small amount of the probabilities allocated to the word sequences that are present in the training set. Two techniques make it possible to extract this mass of probabilities.

Linear discounting consists in taking fraction of the probability of a n-gram, according to its number of occurrences in the training set. The discounted probability,$\hat{P}_{LeanDisc}(w_i|w_{i-n+1}^{i-1})$, of the n-gram w_{i-n+1}^{i} is expressed as the following:

$$\hat{P}_{LeanDisc}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{N(w_{i-n+1}^{i})\times r}{N(w_{i-n+1}^{i-1})} & \text{if } N(w_{i-n+1}^{i}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

Where r is the discounting factor relative to the number of occurrences of the n-gram ($0 < r < 1$). Then the mass of probabilities kept aside is given by:

$$\sum_{N(w_{i-n+1}^{i})>0} P_{MLE}(w_i|w_{i-n+1}^{i-1}) - \sum_{N(w_{i-n+1}^{i})>0} \hat{P}_{LeanDisc}(w_i|w_{i-n+1}^{i-1}) = 1 - r \tag{3.12}$$

where $N(w_{i-n+1}^{i})$ is the number of occurrences of the n-gram $N(w_{i-n+1}^{i})$ in the training set.

The Absolute Discounting consists in extracting a fixed value from the probability of n-gram, independently from its number of occurrences. The reduced

probability P_{DecAbs} is defined by:

$$\hat{P}_{AbsDisc}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{N(w_{i-n+1}^{i})-D}{N(w_{i-n+1}^{i-1})} & \text{if } N(w_{i-n+1}^{i}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

Then the mass of probabilities kept aside is given by:

$$\sum_{N(w_{i-n+1}^{i})>0} P_{MLE}(w_i|w_{i-n+1}^{i-1}) - \sum_{N(w_{i-n+1}^{i})>0} \hat{P}_{AbsDisc}(w_i|w_{i-n+1}^{i-1}) = \frac{|\{w_{i-n+1}^{i-1} : N(w_{i-n+1)}^{i}) > 0\}| \times D}{N(w_{i-n+1}^{i-1})} \tag{3.14}$$

where $|\{w_{i-n+1}^{i-1} : N(w_{i-n+1)}^{i}) > 0\}|$ is the number of distinct word histories w_{i-n+1}^{i-1} preceding the word w_i and D the fixed discounting factor on the n-grams.

Probability re-distribution

Once the mass probability has been extracted from the present n-grams, it must be then redistributed. This redistribution is usually done by affecting some values to unseen n-gram.

The back-off model redistribute the precedent extracted mass of probabilities over the absent n-grams only. The probability of the absent n-gram w_{i-n+1}^{i} defined by:

$$\hat{P}_{Back-off}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \hat{P}_{Disc..}(w_i|w_{i-n+1}^{i-1}) & \text{if } N(w_{i-n+1}^{i}) > 0 \\ \lambda(w_{i-n+1}^{i-1}) \times \hat{P}_{Back-off}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \end{cases} \tag{3.15}$$

where $\hat{P}_{Disc..}$ is one of the discounting models $\hat{P}_{AbsDisc}$ or $\hat{P}_{LeanDisc}$, $\lambda(w_{i-n+1}^{i-1})$ the back-off weights that depends of the words histories w_{i-n+1}^{i-1} and w_{i-n+2}^{i-1} the lower order n-gram that corresponds to the n-gram w_{i-n+1}^{i-1}. With the back-off probability re-distribution approach, the probability of un-estimated tri-gram is the estimated based on the correspondent bi-gram or uni-gram probability if the bi-gram probability is absent.

Then the interpolation redistributes the extracted mass of probabilities $\hat{P}_{Int}(w_i|w_{i-n+1}^{i-1})$ over all the n-grams, whether or not they have been estimated. The probability of the n-gram w_{i-n+1}^{i}

$$\hat{P}_{Int}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \hat{P}_{Disc..}(w_i|w_{i-n+1}^{i-1}) + \lambda(w_{i-n+1}^{i-1}) \times \hat{P}_{Int}(w_i|w_{i-n+2}^{i-1}) & \text{if } N(w_{i-n+1}^{i}) > 0 \\ \lambda(w_{i-n+1}^{i-1}) \times \hat{P}_{Int}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \end{cases} \tag{3.16}$$

In the interpolation redistribution approach, and unlike the back-off approach, the probability of a trigram, for example, represents the combination of the

corresponding bigram and unigrm probability estimations if the probability of the trigram is not estimated in the language model.

smoothing techniques

Whenever data sparsity is an issue, smoothing can improve performance, and data sparsity is almost always an issue in statistical modelling. In the extreme case where there is so much training data that all parameters can be accurately trained without smoothing, one can almost always expand the model, by moving to a higher n-gram model, to achieve improved performance. With more parameters data sparsity becomes an issue again, but with proper smoothing the models are usually more accurate than the original one. Thus, no matter how much data one has, smoothing can almost always improve performance, and for a relatively small effort (Chen and Goodman, 1996).

We present a set of smoothing methods that consists of the combination of the discounting model and the redistribution model.

Katz back-off

This model (Katz backing-off) relies on the estimation of Good-Turing (linear discounting model). The Good–Turing frequency estimation is a statistical technique for estimating the probability of encountering an object of an unseen species, given a set of past observations of objects from different species (one can refer to(Good, 1953) for more details). This is based on the fact that an n-gram appearing r times appears in fact $r^\star$ times, where $r^\star$ is defined by:

$$r^\star = (r+1)\frac{n_r+1}{n_r} \tag{3.17}$$

where r is the number of the n-gram w_{i-n+1}^{i} occurrences and n_r the number of the n-grams appearing exactly r times in the training dataset. The used redistribution model is of back-off type. The n-gram probability calculated using this method is given by

$$\hat{P}_{Katz}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{d_r \times r}{\sum_{w_i \in V} N(w_{i-n+1}^{i-1} w_i)} & \text{if } N(w_{i-n+1}^{i}) > 0 \\ \gamma(w_i|w_{i-n+1}^{i-1}) \times \hat{P}_{katz}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \end{cases} \tag{3.18}$$

where d_r represents the discounting ratio used for reducing the occurrence number of the n-grams found in the training dataset and $\gamma(w_i|w_{i-n+1}^{i-1})$ represents the back-off weights calculated by:

$$\gamma(w_i|w_{i-n+1}^{i-1}) = \frac{1 - \sum_{w_i \in V: N(w_{i-n+1}^{i-1} w_i)>0} \hat{P}_{katz}(w_i|w_{i-n+1}^{i-1})}{\sum_{w_i \in V: N(w_{i-n+1}^{i-1} w_i)=0} \hat{P}_{katz}(w_i|w_{i-n+2}^{i-1})} \tag{3.19}$$

In fact, the n-gram that appear more than k times in the training data set are considered as reliable and their occurrence number not to be modified. Katz suggests to set the value of k to 5. The discounting ratio d_r can be calculated as

follow:

$$d_r = \begin{cases} \frac{r\star}{r} & \text{if } r \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.20}$$

Absolute Discounting back-off

This model is characterised by an absolute discounting model with a back-off redistribution model. It calculates the n-gram probability by the following equation:

$$\hat{P}_{Abs}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{\max\{N(w_{i-n+1}^{i})-D,0\}}{\sum_{w_i \in V} N(w_{i-n+1}^{i} w_i)} & \text{if } N(w_{i-n+1}^{i}) > 0 \\ \gamma(w_i|w_{i-n+1}^{i-1}) \times \hat{P}_{Abs}(w_i|w_{i-n+2}^{i-1}) & \text{otherwise} \end{cases} \tag{3.21}$$

where the back-off weights is computed by

$$\gamma(w_i|w_{i-n+1}^{i-1}) = \frac{D}{\sum_{w_i \in V} N(w_{i-n+1}^{i} w_i)} N_{1+}(w_{i-n+1}^{i} \bullet)$$

and $N_{1+}(w_{i-n+1}^{i} \bullet) = |\{w_i : N(w_{i-n+1}^{i}) > 0\}|$ is the number of words that follow the word history w_{i-n+1}^{i-1} and occurred once or more, considering D a fixed count whose value is estimated by:

$$D = \frac{n_1}{n_1 + 2n_2} \tag{3.22}$$

where n_1 and n_2 are the total number of n-grams that occurred once and twice time respectively in the training dataset.

Modified Kneser-Ney interpolation

This successful method was proposed by (Chen and Goodman, 1996). It consists of some modifications applied on the original Kneser-Ney smoothing method (Kneser and Ney, 1995) that combines the absolute discounting model and interpolation redistribution models.

Let us take an example of a bi-gram in which the word *Francisco* is very frequent but it occurs after the word *San* in the training corpus most of the time. The probability of the *Francisco* uni-gram is high and then the absolute discount will assign a high probability to the word *Francisco* appearing after any new word (different from *San*). Maybe, it is more reasonable not to give it such high probability for each possible occurrence, whereas it systematically follows the word *San*. In such case its probability is well modelled by the bi-gram model.

For Kneser and Ney, the uni-gram probability should not be calculated proportionally to the number of occurrences of a word but to the number of different words following it; the number of accounts assigned to each of the uni-grams will be the number of words it follows simply. Therefore, the probability

of a word is given by the following:

$$\begin{aligned}\hat{P}_{KN}(w_i|w_{i-n+1}^{i-1}) &= \gamma(w_{i-n+1}^{i-1}) \times P_{MLE}(w_i|w_{i-n+1}^{i-1}) \\ &\quad +(1-\gamma(w_{i-n+1}^{i-1})) \times \frac{N_{1+}(\bullet w_{i-n+2}^{i})}{\sum_{w_i \in V} N_{1+}(\bullet w_{i-n+2}^{i} w_i)}\end{aligned}$$

where $P_{MLE}(w_{i-n+1}^{i-1})$ is the initial probability of w_{i-n+1}^{i} and $N_{1+}(\bullet w_{i-n+2}^{i} w_i)$ represents the number of words w_{i-n+1} where w_{i-n+1}^{i} appeared once or more time in the training corpus. the back-off weight is calculated by

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D}{\sum_{w_i \in V} N(w_{i-n+1}^{i-1} w_i)} N_{1+}(w_{i-n+1}^{i-1} \bullet) \tag{3.23}$$

The introduced modification to this method consists of using three discounting parameters, D_1, D_2 and D_{3+}, applied to the n-grams appearing one, two and three times or more respectively. Thus, the parameter γ of equation 3.23 is rewritten as:

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1 \times N_1(w_{i-n+1}^{i-1}\bullet) + D_2 \times N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} \times N_{3+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i \in V} N(w_{i-n+1}^{i-1} w_i)} \tag{3.24}$$

where $N_1(w_{i-n+1}^{i-1}\bullet), N_2(w_{i-n+1}^{i-1}\bullet)$ and $N_{3+}(w_{i-n+1}^{i-1}\bullet)$ are defined by the same way for the occurrence frequencies of once, twice, three times or more respectively. The optimum values of the parameters D_1, D_2 and D_{3+} are calculated by the following equations:

$$\begin{cases} D_1 = 1 - 2Y \times \frac{n_2}{n_1} \\ D_2 = 2 - 3Y \times \frac{n_3}{n_2} \\ D_{3+} = 3 - 4Y \times \frac{n_4}{n_3} \end{cases} \tag{3.25}$$

where $Y = \frac{n_1}{n_1+2n_2}$ and n_1, n_2, n_3 and n_4 are the total number of n-grams appearing once, twice, three or four times in the training corpus.

3.4.5 Hybrid language models vs. language model combination

Language model combination has the interest of collecting the advantageous specificities of each individual model and surpasses the best of them (Goodman, 2001). The simplest way to combine two or more homogeneous language models (defined on the same language units e.g. words, syllables or characters) is known by the *linear interpolation* method. The produced language model from the linear interpolation of K models is obtained by the weighted sum of the K models as the following:

$$\tilde{P}_{LinInterpol}(w_i|\Phi(h_i)) = \sum_{k=1}^{K} \lambda_k P_k(w_i|\Phi(h_i)) \tag{3.26}$$

where λ_k is the weight of the language model ML_k and $0 \leq \lambda_k \leq 1$. The values of these weights are a matter of optimization problem that leads to an optimal interpolated language model $ML_{LinInterpol}$ under the condition of $\sum_{k=1}^{K} \lambda_k = 1$.

The combination of language models that are built with different language units type (for example, words+characters or words+sub-lexical units) produces a non-uniformed language model which is called hybrid language model.

3.4.6 Variable length dependencies language models

The order n of a n-gram model of words usually is less than 4 or 5, which does not allow for long dependencies in sentences to be modelled. When the order of n-gram model increases, the probability of meeting exactly each existing word history decreases.

Based on this observation, skipping language model (Huang et al., 1993) stands on ignoring certain word histories (grams) that might be defined in the normal n-gram language model. These models are used generally in combination with n-gram language models in order to consider the short and the long word context. One of the advantages of these language models is that they take into account similar word histories but not strictly identical and regular context. However, the increment of the n-gram language model increases the amount of the model parameters which represent a draw-back for these models.

Another extension of the n-gram models is based on the variation of the length of the word history by keeping only the long word histories whose relative information are present (Siu and Ostendorf, 2000). These models are called poly-gram models. The n-grams probabilities of poly-gram model (Kuhn, Niemann, and Schukat-Talamazzini, 1994) are estimated by interpolating the relative frequencies of all k-grams, while $k \leq n$. By this interpolation procedure, polygram model accounts for variable length dependencies of words, thus this method represents an alternative way to count for variable-length dependencies of language words sequences.

Their principle is thus to ensure a compromise between the relevance of the language model (keeping the most informative histories) and its number of parameters. The advantage of the poly-gram models is the reduction of the number of their parameters, while maintaining similar performances.

There are other language models coping with variable length word history such as Permugram language model (Schukat-Talamazzini et al., 1995) in which the considered word histories corresponds to a permutation of the words of the current context. Another variable length word histories (variable length dependencies) in which we are specifically interested is the multigram language models (Deligne and Bimbot, 1995) in which the word histories are considered as a sequence of groups of words, more details about these models is presented in the next section.

3.4.7 Multigrams language models

The multigram model provides a statistical tool to retrieve sequential variable-length regularities within streams of data (Deligne and Bimbot, 1997). The multigram model was first presented by (Bimbot et al., 1994) for language modeling by taking into account variable-length dependencies of language words sequences. Theoretical formulation and evaluation of multigrams is presented in (Deligne and Bimbot, 1997; Deligne, Yvon, and Bimbot, 1996). It assumes the language as a stream that flow off a memoryless source of variable length sequence of words (Deligne and Bimbot, 1995). The language words are not independent, and their dependencies are of variable length. Conversely, the *n-gram* model (Jelinek, 1990) assumes that the statistical dependencies between words are of fixed length n along the whole sentence (Deligne and Bimbot, 1995).

In the m-multigram model (Bimbot et al., 1994), a sentence is considered as the concatenation of independent variable-length sequences of words (up to length m) commonly known as phrases and the likelihood of the sentence is computed as the sum of the individual likelihood corresponding to each possible segmentation (Deligne and Bimbot, 1995). The multigram model can be characterized as a stochastic phrase-based model (Deligne and Sagisaka, 2000).

Let $W = (w_1, \ldots, w_t, \ldots, w_T)$ denotes a sequence of T words, and let S denote one possible segmentation of W into q sequences of words, formally, $S = (m_1, m_2, \ldots, m_l, \ldots, m_q)$ and $m_l = (w_t, \ldots, w_{t+k-1})$ where m_l is the sequence of words of length k with $k \leq n$ and n is the maximum length of the sequence of words m_l. The n-multigram model (n-mgr) computes the joint likelihood $P_{mgr}(W, S)$ of the corpus W associated to segmentation S as the product of the probabilities of the successive sequences of maximum length n:

$$P(W, S) = \prod_{l=1}^{l=q} p(m_l) \tag{3.27}$$

Assuming that $\{S\}$ represents the set of all possible segmentation of W into sequences of words, the likelihood of W is :

$$P_{mgr}(W) = \sum_{S \in \{S\}} P(W, S) \tag{3.28}$$

The model parses the sequence of words W according to the most likely segmentation, thus yielding the approximation :

$$P^{\star}_{mgr}(W) = \max_{S \in \{S\}} P(W, L) \tag{3.29}$$

For example, let $T = 4$, $n = 3$ and $W = abcd$, considering the sub-sequence of words boarders are the brackets :

$$P^{\star}_{3-mgr}(abcd) = \max \left\{ \begin{array}{l} \text{p([a]) p([bcd])} \\ \text{p([abc]) p([d])} \\ \text{p([ab]) p([cd])} \\ \text{P([ab]) p([c]) p([d])} \\ \text{p([a]) p([bc]) p([d])} \\ \text{p([a]) p([b]) p([cd])} \\ \text{p([a]) p([b]) p([c]) p([d])} \end{array} \right\} \tag{3.30}$$

Where $P^{\star}_{3-mgr}(abcd)$ is the likelyhood of the sequence of words $abcd$ calculated by multigram language model of order 3. In the other hand, n-gram language model of order 3 calculates this probability as the following:

$$P_{3-gram}(abcd) = p(a)p(b|a)p(c|ab)p(d|bc) \tag{3.31}$$

3.4.8 Conctionest neural network language models

In recent years, neural networks language models (NNLMs) are becoming increasingly popular for a wide range of tasks such as handwriting / speech recognition & machine translation, due to their inherently strong sequence modelling ability and generalization performance. For several decades, N-grams language models have been dominating the area of statistical language modelling. However, n-gram models suffers of data sparsity problem. Contrarily to conventional statistical N-gram LMs, the generalization ability of NNs and the continuous space representation of words allow the NN LMs to perform an implicit smoothing (Zamora-Martinez et al., 2014).

Feedforward Neural Network Based Language Model

The neural network language models was first introduced by (Bengio et al., 2003). The NNLM are defined as statistical n-gram language model that use the capability of the NN to estimate the probabilities that satisfies equation (3.4). Thus its input is a fixed length sequence of $n-1$ words where each of the previous $n-1$ words is encoded using 1-of-V coding, where V is the size of the vocabulary. By this, every word in the vocabulary is associated with a vector of length V where only one value that correspond to the index of the given word is 1 while the other components are set to 0.

The orthogonal representation of words 1-of-V is projected linearly to a lower dimensional space using a projection matrix P. The matrix P is shared among all the word of the word history during the projection. A hidden layer follows the projection layer which consists of non-linear activation functions with a dimensionality of 100-300 units. The output layer follows, with a size equal to the size of the full vocabulary. Figure 3.6 illustrate the NN language model structure. The NNLM have the advantage over the n-gram language models that the word history is no longer constrained to an exact sequence of $n-1$ words of history, but rather as a projection of the history into some lower

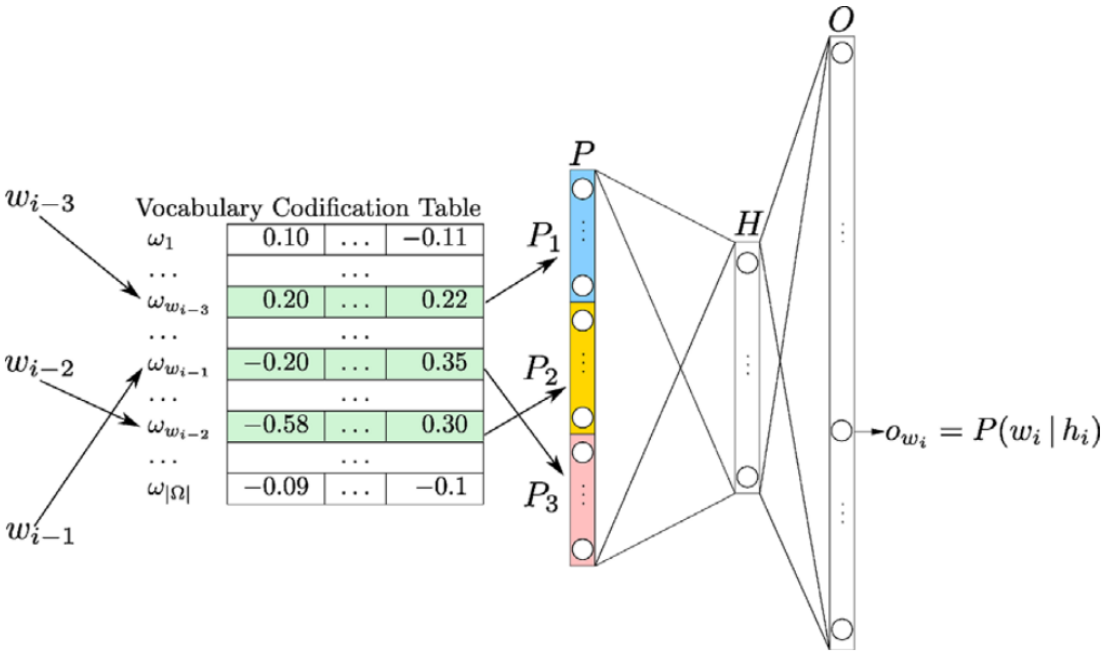


FIGURE 3.6: The architecture of a NN LM after training, where the first layer is substituted by a look-up table with the distributed encoding for each word $\omega \in \Omega$(Zamora-Martinez et al., 2014)

dimensional space. As a result, the automatic clustering of the similar word histories, reduces the number of parameters. The weak point of these models is their very large computational complexity during training, which usually prohibits to train these models on full training set using the full vocabulary (Mikolov, 2012).

The applicability of neural network language models in unconstrained off-line handwriting recognition has been studied by (Zamora-Martinez et al., 2014). They also compared the performance of conventional n-gram language models across of NNLMs on IAM handwriting database using two different recognition engines under different vocabulary sizes.

Recurrent Neural Network Based Language Model

The difference between the feed forward neural network language model NNLM and the recurrent neural network language model (RNNLM) architecture is in the representation of the word history, in feed forward NNLM the word history is still limited to the several previous words, meanwhile in RNNLM an effective representation of word history is learned from the data during training. The hidden layer of RNN represents all previous history and not just the $n-1$ previous words, thus the model can theoretically represent long context patterns (Mikolov, 2012). In addition, RNNLM have the advantage over feed-forward one of representing more advanced patterns in the sequential data such as the patterns that rely on words that could have occurred at variable position in the history, the recurrent model can remember some specific words in the state of the hidden layer, while the feed-forward model would need

to use parameters for each specific position of the word in the history, which lead to an increased number of parameters that required equivalent increased number of examples for training the given pattern.

The architecture of RNNLM consists of consecutive input , hidden and output layers. The input layer consists of a vector $\mathbf{w}(t)$ that represents the current word w_t encoded as 1 of V where V is the vocabulary (thus the size of $\mathbf{w}(t)$ is equal to the size of V) and of vector $s(t-1)$ that represents output values in the hidden layer from the previous time step. After the network is trained, the output layer $\mathbf{y}(t)$ represents $P(w_{t+1}|w_t, s(t-1))$.

Training the network can be achieved by stochastic gradient desent using the *backpropagartion through time* (BPTT) algorithm (Hinton, Rumelhart, and Williams, 1985). The network can be represented by its input, hidden and output layers associated with their corresponding weight matrices. The matrices **U** and **W** represent the weights between the input and the hidden layer, and matrix **V** between the hidden and the output layer as shown in figure 3.7.

Output values in the layers are computed as follows:

$$s_j(t) = f\left(\sum_i w_i(t)u_{ji} + \sum_l s_l(t-1)w_{jl}\right) \tag{3.32}$$

$$y_k(t) = g\left(\sum_j s_j(t)v_{kj}\right) \tag{3.33}$$

where g(z) and f(z) are sigmoid and activation function like softmax function. The softmax function is used in the output layer to guarantee that all outputs are greater than 0 and their sum is 1, thus:

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{3.34}$$

Alternatively, equation 3.33 and 3.33 can be rewritten in vector representation:

$$\mathbf{s}(t) = f(\mathbf{U}w(t) + \mathbf{W}s(t-1)) \tag{3.35}$$

$$\mathbf{y}(t) = g(\mathbf{V}s(t)) \tag{3.36}$$

The network weight matrices are trained by using the stochastic gradient descent calculated by an objective function such as the cross entropy criterion. The gradient of an error vector in the output layer is then propagated to the hidden layer and in case of BPTT backwards in time through the recurrent connections. During training, the learning rate is controlled by observing the enhancement rates on a validation dataset. For more details about the architecture and the training scheme of RNNLM. Yousfi et *al*. (Yousfi, Berrani, and Garcia, 2017) use simple RNN models that are learned with a Maximum Entropy language model decoded jointly with a BLSTM-CTC optical models for Arabic text recognition in videos. These models have reached an improvement of almost 16 points in Word Recognition Rate (WRR) compared to baseline BLSTM-CTC OCR system. RNNLMs are widely used in the speech recognition field as in (Mikolov et al., 2011a,Mikolov et al., 2012).

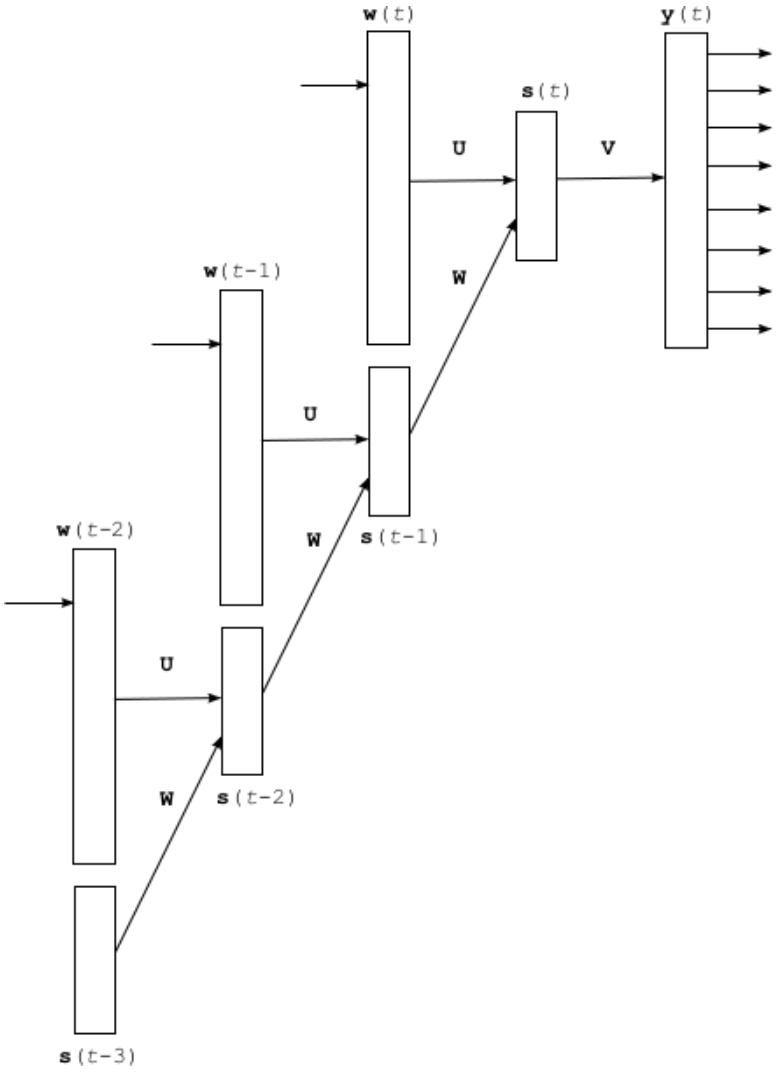


FIGURE 3.7: Recurrent neural network unfolded as a deep feed-forward network, here for 3 time steps back in time(Mikolov, 2012)

3.4.9 caching language models

One of the obvious disadvantage of n-gram language models is their limited capacity to represent long term patterns. It has been empirically observed that many words, especially the rare ones, have significantly higher chance of occurring again if they did occur in the recent history (Mikolov, 2012). Cashing language model (Jelinek et al., 1991) consider a cash component for the word occurrences beside the training corpus for calculating its probabilities which are estimated dynamically from the recent history (it usually represents few hundreds of words) and interpolated with the main (static) n-gram model.

The advantage of the cache model is its highly reduced perplexity compared to the conventional n-gram models, that's why this modelling approach is very popular in the language modelling related papers (Mikolov, 2012). The main drawback of such model is in the correlation between the perplexity improvements and word error rate reductions. Advanced versions of caching

models like trigger models or LSA models have reported interesting WER reduction. Recently, these models are not widely in practice (Mikolov, 2012).

3.4.10 Maximum entropy language models

The maximum entropy language models ME (Berger, Pietra, and Pietra, 1996) allow to incorporate many feature functions (for example, n-gram, n-class and skip-gram) using triggers and word features to obtain highly improved perplexity, as well as significant word error rate reduction. The ME model is expressed as an exponential model with a form

$$P(w|h) = \frac{e^{\left(\sum_i \lambda_i f_i(w,h)\right)}}{Z(h)} \tag{3.37}$$

where w is a word in context h and $Z(h)$ is used for normalizing the probability distribution:

$$Z(h) = \sum_{w_i \in V} e^{\left(\sum_j \lambda_j f_i(w_i,h)\right)} \tag{3.38}$$

This model is reported to have a state-of-the-art performance on a broadcast news speech recognition task, when applied to a very well tuned system that is trained on large amounts of data and uses state of the art discriminatively trained acoustic models. An alternative name of maximum entropy models used by the machine learning community is logistic regression (Mikolov, 2012) and Conditional Random Field (Sutton and McCallum, 2006).

3.4.11 Probabilistic Context free Grammar

A Probabilistic Context-Free Grammar (PCFG, Manning and Schütze, 1999) is simply a Context-Free Grammar with probabilities assigned to the rules such that the sum of all probabilities for all rules expanding the same non-terminal is equal to one. Context-free grammar (CFG) is a certain type of formal grammar: a set of production rules that describe all possible strings in a given formal language. Production rules are simple replacements just like replacing A by α or β. In context-free grammars, all rules are one-to-one, one-to-many, or one-to-none. Rules can also be applied in reverse to check if a string is grammatically correct according to the grammar. The PCFG grammar is defined by

- A set of terminals, related to the language words: $\{w_k\}$ where $w_k \in V$ and V the language vocabulary.
- A set of non-terminals, related to grammatical objects: $\{N_i\}$.
- A starting symbol: N_1.
- A set of derivation rules: $\{N_i \rightarrow \zeta_j\}$ where ζ_j is a sequence of terminals and non-terminals.

- A set of probabilities evaluated automatically from a labelled corpus governed by rules such as

$$\forall i, \sum_{j} P(N_i \rightarrow \zeta_j) = 1 \tag{3.39}$$

Every derivation of a phrase is associated with a probabilistic score that represents the product of the rules probability related to that derivation (phrase).

3.5 Language models application domains

Statistical Language Models estimate the distribution of various natural language phenomena for the purpose of speech and handwriting recognition and other language technologies. A First practical model was proposed in 1980 (Della Pietra et al., 1992), since then many attempts have been made to improve the state of arts. Language models are commonly used in several domains beside speech and handwriting recognition such as:

1. Spelling correction: In the field of text processing application language models are used to correct spellings (Kukich, 1992). Two types of errors can be distinguished, the first is the non lexical word, where the error in writing a word that does not belong to the language words (for example the word "bi y" in place of "bye"). These errors can be corrected by using a lexicon and computing edit distance that can find the nearest word in the lexicon. The second type of errors represents the miss-placed words which are valid words but not coherent with the phrase structure (for example, the French word "lange" in place of "langue". A language model can correct such type of errors by detecting the incorrect word in the sentence and correct it.

2. Language translation: The automatic translation of a sentence from a source language to a target language corresponds to interfering two types of models (Brown et al., 1990). One model expresses the source language sentence ideas using the words of the destination language, the second model regulate the output of the first model to be structured correctly from the grammar point of view of the destination language.

3. Corpora labelling: One of the most recent tasks is the corpora labelling task which consists of labelling each corpus word with its grammatical class (Manning and Schütze, 1999). Almost, any word may belong to multiple classes (for example the French word "souris" may be a noun and verb), so a language model is necessary to reduce the ambiguity in labelling words by taking into account their context.

4. Information searching: With the growing mass of information in the internet, searching of an information becomes an important field. In order to respond effectively to the user query, the language model can be used to incorporate the linguistic knowledge to the searching motors by using some sort of interpretations of the user natural lingual structured query.

5. Language detection: Human language has some words which are used more than others. For example you can imagine that in a document the word "the" will occur more frequently than the word "aardvark". Moreover there is always a small set of words which dominates most of the language in terms of frequency of use. This is true both for words in a particular language and also for words in a particular category. Thus words which appear in a sporty document will be different from the words that appear in a political document and words which are in the English language will obviously be different from words which are in the French language. In order to classify a sample document, n-gram models can be used to detect the language in which a texts was written, where one computes models for different languages and then compare a test text to each model using the cross-entropy (actually, perplexity) between the text and the models. one classify the text into the language for which the perplexity was minimal (Ramisch, 2008).

3.6 conclusion

In this chapter we have presented a set of natural language modelling approaches commonly applied for speech and handwriting recognition tasks. However, most sophisticated modelling approaches found in the literature enhance lightly the handwriting recognition systems performances compared to the conventional statistical n-gram language models and this enhancement comes with a valuable increment in the models complexities and / or calculation time and models storing volumes.

For our handwriting recognition systems, we looked for utilizing language models that have lower complexities. For that our choice was to utilise the conventional n-gram modelling approach for lexical, sub-lexical, and character language units.

Chapter 4

Design and evaluation of a Handwritting recognition system

4.1 Introduction

In this chapter, we present the design of the different handwriting recognition systems that have been tested all along this thesis. The first experimentations have been carried out using the standard HMM optical models. Of course this was a limitation regarding the state of the art performance obtained on the reference datasets, even if our primary objective was to study the contribution of language models to the performance of the whole system. This is the reason why we decided to introduce LSTM Neural Networks in the system in place of the HMM stage. This chapter is intended to give a comprehensive, although light, description of every stages required in the design of handwritten text recognition systems. While chapter 2 and chapter 3 have been dedicated to the fundamental aspects of optical and language models, here we concentrate on describing how to integrate these models together, in order to build an operational reading system that can take a full handwritten text image as input and produce its transcription in electronic format at the output. Performance evaluation as well as training of the various systems implemented have been carried out using two popular datasets : the French RIMES dataset and the English IAM dataset. They are described briefly in section 4.2 regarding their optical as well as linguistic content in terms of number of writers, lexicon size and coverage. The main components of a reading system are shown on figure 4.1.

One of the primary components of the system is the preprocessing stage that is design to determine the target text line images locations in the document image using some image processing or machine learning techniques. Once the text line images have been extracted, some image correction such as skew and slant correction algorithms can be applied for reducing the variability in the data and possibly enhance the recognition performance. Section 4.3 is intended to describe this very important component of our whole reading system. The second important component of a text recognizer is the optical model of characters. In section 4.4 we will describe our implementations and training of the two statistical optical models that have been tested during this thesis: HMM and BLSTM-RNN optical models. Optical models provide likelihood scores denoted $P(S|W)$ of observing the image, or a representation of the image denoted S, assuming a certain text transcription W. This section

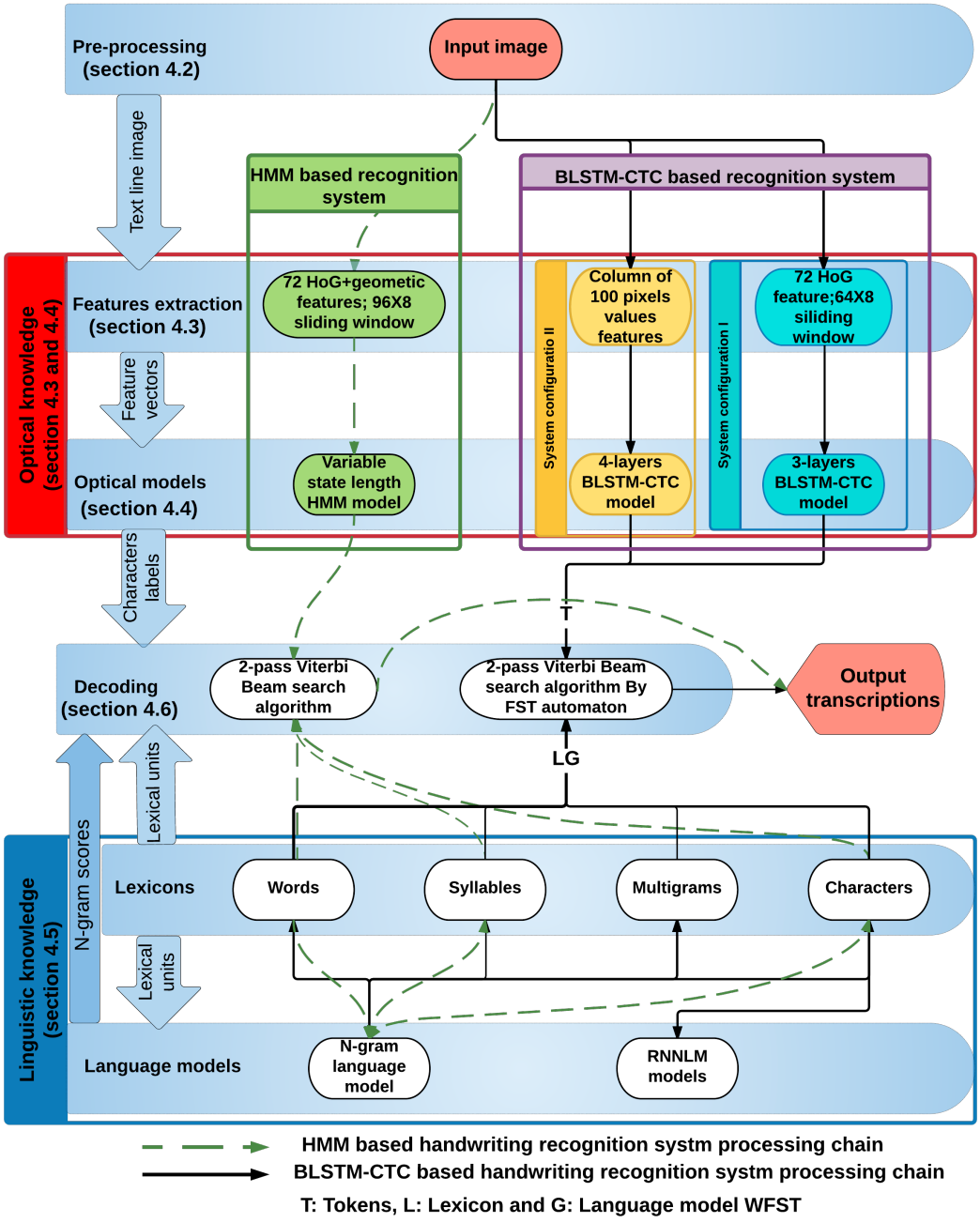


FIGURE 4.1: General architecture of the reading systems studied

also includes a brief description of the features used as input of the optical models. Section 4.5 is dedicated to the linguistic component of the system : the language model, that is design to estimate the probability of a certain transcription W, denoted by $P(W)$. All along this thesis we have studied the contribution of this component and we will briefly present the practical aspects and problems related to combining lexicons and language models and have them optimized on the appropriate training datasets. Most of the discussion will rely on the linguistic properties of the two datasets used for the evaluation (RIMES and IMA datasets). Last but not least is the decoding component of the

recognition system. During recognition, the two system components (optical model and language model) are combined together to build a hypotheses lattice called search space. The contribution of each model to the final recognition result is governed by equation 4.1. The two essential coupling parameters are the language model scale parameter γ and word insertion penalty β. These two coupling parameters require careful optimization in order to obtain optimal recognition performance. We will discuss these questions in section 4.6. By the end of this presentation of each system component, we will give some primary recognition results in section 4.7 and draw the main guidelines for our contributions in the next chapter (Chapter 5).

$$\hat{W} = arg \max_{w} P(S|W)P(W)^{\gamma}\beta^{Length(W)} \tag{4.1}$$

4.2 Datasets and experimental protocols

The experiments have been carried out on publicly available databases, in order to compare our work to other approaches and research groups (see Chapter 5). The scope of the experiments is limited to the French RIMES and the English IAM databases. In addition, we make use of some large and divers corpora for training the language models in same cases. In the next section, we describe the used datasets for training and evaluating the systems performances.

4.2.1 Training and evaluation databases

A conventional practice in handwriting recognition is to divide the databases into three different sets. The evaluation set (or test set) contains images never seen during training, with which we evaluate our systems performances. Training algorithms have some parameters themselves (the hyper-parameters), that should be set to train the systems. The development (or validation) set is another test set, on which we can check the best choice of values for these hyper-parameters.

RIMES dataset

The Rimes dataset (Grosicki and El-Abed, 2011) consists of images of handwritten letters from simulated French mail. We followed the setup of the ICDAR 2011 competition. The available data are a training set of 1,500 paragraphs (11333 text lines), manually extracted from the images, and an evaluation set of 100 paragraphs (748 text lines). We selected randomly 1333 of the training set text lines as a validation set, and trained the systems on the remaining 10000 text lines as illustrated in table 4.2.

IAM dataset

The IAM database (Marti and Bunke, 2002) consists of images of handwritten pages. They correspond to English texts extracted from the LOB corpus

(Johansson, 1980), copied by different writers. The database is split into 747 images for training, 116 for validation, and 336 for evaluation. Note that this division is not the one presented in the official publication or of the distribution website [1]. The database has been used by different researchers with different setup. In table 4.1 we list three different setup of the IAM dataset introduced by different groups. For our experiments we are using the setup used by (Bluche, 2015).

Dataset Setup	# LINES		
	Training	Validation	Test
site, 2016	6161	900+940	1861
Zamora-Martinez et al., 2014	6161	920	2781
Bluche, 2015	6482	976	2915

TABLE 4.1: IAM dataset setups

Databases	Handwriting recognition system	Number of text line examples per data set		
		training	validation	test
RIMES	HMM based	10963	382	382
	BLSTM-RNN based	9947	1333	778
IAM	HMM based	11349	971	1033
	BLSTM-RNN based	6482	976	2915

TABLE 4.2: RIMES and IAM database divisions used for training, validation and test the HMM based and BLSTM-RNN based handwriting recognition systems

The HMM based and the BLSTM-RNN experiments have been carried out on different divisions of RIMES and IAM dataset for training, validation and test. Table 4.2 shows of the annotated text line examples of RIMES and IAM databases used for training, validation and test the two proposed handwriting recognition systems.

4.2.2 Language model training corpora

The language models have been learned following two different configurations: the first configuration corresponds to using the training sets of texts of each of the RIMES or IAM training datasets. This means that the associated lexicon is limited to the vocabulary of the training sets. The second configuration corresponds to using large and divers text corpora such as Wikipedia French web pages or the English LOB, Brown and Wellington corpora and then evaluating these language models on the French RIMES and the English IAM test datasets.

[1] http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

French extended Wikipedia corpus

This corpus was collected from the Wikipedia French version pages [2] that includes diverse subjects. This corpus includes all the collected text data which consists of $529,299$ passage of text, $702,097$ word and $84,236,612$ character, considering that a word is defined by a sequence of characters sided by two white spaces characters.

French Wikipedia corpus

The corpora represents a portion (10%) of the French extended Wikipedia pages, it consists of $52,930$ passage of text, $182,735$ words and $12,921,198$ character. By analysing the effective coverage of the whole and partial Wikipedia corpuses on the RIMES test dataset, we noticed that the difference in the effective coverage rate reported for the $50,000$ most frequent words of the extended Wikipedia corpus (83.28%) and for the $50,000$ most frequent words of the (partial) Wikipedia corpus (82.71%) on the RIMES test data set is around 0.57% which is not significant compared to the size of corpus lexicon size which increases from $182,735$ word to $702,097$ word for the partial and the whole corpuses respectively. Therefore, we decided to carry out all our experiments with this reduced size corpus.

English LOB corpora

The Lancaster-Oslo/Bergen Corpus (LOB Johansson, 1980) was compiled by researchers at Lancaster, Oslo and Bergen universities. It consists of one million words of British English texts from 1961. The texts for the corpus were sampled from 15 different text categories. Each text is just over 2,000 words long (longer texts have been cut at the first sentence boundary after 2,000 words) and the number of texts in each category varies.

English Brown corpus

The Brown Corpus (Francis and Kučera, 1979) was the first computer-readable general corpus of texts prepared for linguistic research on modern English. It was compiled by W. Nelson Francis and Henry Kučera at Brown University in the 1960s and contains over 1 million words (500 samples of 2000+ words each) of running text of edited English prose printed in the United States during the calendar year 1961. There are six versions of the corpus available: the original Form A, Form B from which punctuation codes have been omitted, the tagged Form C, Bergen Forms I & II and the Brown MARC Form.

English Wellington corpus

The Wellington Corpus (Bauer, 1993) of Written New Zealand English (WWC) consists of one million words of written New Zealand English collected from

[2] https://dumps.wikimedia.org

writings published in the years 1986 to 1990. The WWC has the same basic categories as the Brown Corpus of written American English (1961) and the Lancaster-Oslo-Bergen corpus (LOB) of written British English (1961). The WWC consists of 2,000 word of different topics. Text categories include press material, religious texts, skills, trades and hobbies, popular lore, biography, scholarly writing and fiction.

English extended Wikipedia corpus

The Wikipedia English version pages was collected from the same source of the French one in order to build the English extended Wikipedia corpus. The corpus consists of $540,385$ passage of text, $994,545$ word and $128,558,999$ character.

English Wikipedia corpus

The corpora represents a portion (10%) of the English extended Wikipedia pages, it consists of $53,999$ passage of text, $240,659$ words and $18,193,128$ character. Again, the difference in the effective coverage rate reported for the $50,000$ most frequent words of the extended Wikipedia corpus (87.75%) and for the $50,000$ most frequent words of the (partial) Wikipedia corpus (87.12%) on the RIMES test data set is around 0.63% which is not significant compared to the size of corpus lexicons size which increases from $240,659$ word to $994,545$ word.

Softwares and tools

The introduced handwriting recognition systems in this theses are implemented and evaluated by using open source and free of charge toolkits so they are easy to be replicated with no cost. We can categories the utilized softwares and tools according to the systems tasks into four main categories: Optical model training toolkits, language model training toolkits, decoding tools and evaluation tools. Table 4.3 illustrates each toolkits used jointly with the system task that required it for the two systems considered:

Tasks/ toolkits	optical model training tools	language model training tools	decoding tools	evaluation tool
EESEN	BLSTM-RNN			
HTK	HMM		HMM	
MITLM		BLSTM-RNN		
SRILM		HMM & BLSTM-RNN		
Kaldi (open FST)			BLSTM-RNN	BLSTM-RNN
Sclite (NIST)				HMM

TABLE 4.3: Resume of the toolkits used for each systems realization task

One can refer to (Miao, Gowayyed, and Metze, 2015) for more details about EESEN toolkit, (Young and Young, 1993) for HTK toolkit, (Povey et al., 2011)

for Kaldi toolkit, (Stolcke, 2002) for SRILM toolkit and (Fiscus, 2015) for Sclite scoring toolkit from National Institute of Standards and Technology (NIST).

4.3 Pre-processing

In their full denomination, pre-processing includes image enhancement prior to image segmentation to get the image components localized, and allowing each component to be passed to the recognition stage. During this thesis our contribution has been focused on the detection of text lines in images of texts blocks solely.

4.3.1 Related works

The difficulty of the line extraction task depends on the regularity of the handwriting: length and height of text lines, straightness of baselines... The lines in handwritten documents are often irregular and have many defects which makes the detection of text lines delimiters difficult. The most problematic defects being the overlapping lines (mainly due to the occurrence of the ascenders and descenders), the touching characters across two consecutive lines which require special treatment (proper cutouts and assignment to the proper line) and the skew variability between lines and within the same line. Therefore, line segmentation in unconstrained handwritten documents is a very challenging task and is still an open research area. This is why, in the recent years, many papers have been published, and several contests have been organized during ICDAR2009 (Gatos, Stamatopoulos, and Louloudis, 2011), ICDAR2013 (Stamatopoulos et al., 2013), ICFHR 2010 (Gatos, Stamatopoulos, and Louloudis, 2010).

Three distinct approaches for line segmentation exist in the literature (Katsouros and Papavassiliou, 2011): - bottom-up approaches try to merge the base units (mostly connected components) according to their spatial proximity to form horizontal lines - top-down approaches first locate the limits of the different lines before assigning the base units to their respective lines – smearing methods apply a blurring filter to the document image in order to reveal the structure of the underlying lines through a "smearing" of the horizontal alignments of black pixels. Proposed filters are either anisotropic Gaussian filters (Li et al., 2008; Bukhari, Shafait, and Breuel, 2009; Bukhari, Shafait, and Breuel, 2009) or ellipsoid filters (Shi, Setlur, and Govindaraju, 2009). To be effective, the filters must be more elongated along the horizontal axis otherwise the blurring may merge close lines.

4.3.2 Proposed method

Because of the great variability between writer's styles and the different scripts (latin, arabic, chinese, hindi...), existing methods do not generalize well to any types of documents (Katsouros and Papavassiliou, 2011). Therefore, it seems that the most robust methods will be those that do not use any *a priori* and

are able to adapt to the characteristics of each document (Papavassiliou et al., 2010). From the analysis of the results of international contests that have been organized few years ago, we came to the conclusion that smearing methods may have a great potential to generalize well to different scripts. Inspired by (Shi, Setlur, and Govindaraju, 2009), we designed a generalized smearing method that does not require any prior setting. An iterative approach is used as depicted on figure 4.2 below, in which the algorithm parameters (filter size) are adjusted at each iteration to detect new lines, with different size. Lines detected at each iteration are added to the final line detection result. As a main advantage, the method explores a wide range of potential line sizes and integrates a detection criterion allowing to validate or reject the line hypothesis, thus being robust with respect to the variability in size of the lines.

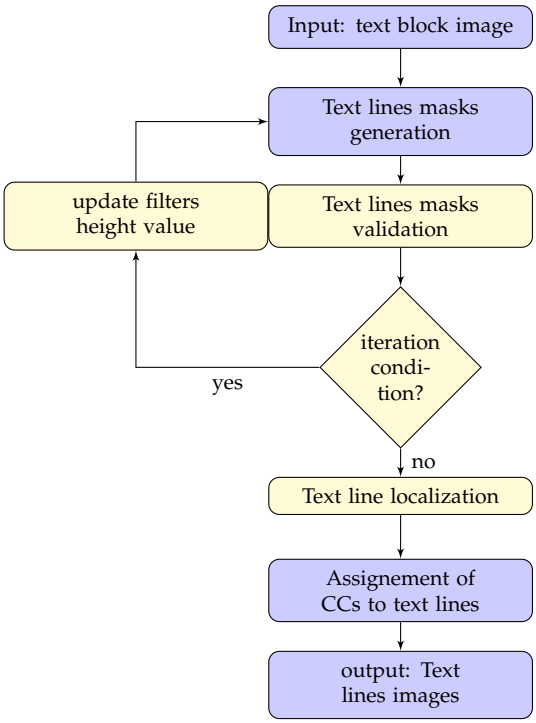


FIGURE 4.2: The proposed generalized smearing approach for line detection.

The method is based on a generalized adaptive local connectivity map (ALCM) generated by a set of steerable filters (Shi, Setlur, and Govindaraju, 2009) that process the image in different orientations. First the method locates the text lines through their masks that are the result of the ALCM. The adaptive local connectivity map is defined as the result of the following filtering :

$$A(x,y) = \int_{R^2} f(x,y) G^{\theta_o}_{FH,FW}(x-t, y-s)dtds \tag{4.2}$$

where

$$G^{\theta_0}_{FH,FW}(x,y) = \begin{cases} 1 & if(x,y) \in E^{\theta_o}_{FH,FW} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

and

$$E^{\theta_o}_{FH,FW} = \left\{(x,y)| \begin{array}{rcl} x & < & FH\cos(\theta-\theta_o) \\ y & < & FW\cos(\theta-\theta_o) \end{array} \; and \;\; 0 \leq \theta < 2\pi \right\} \tag{4.4}$$

E^{θ_o} is an ellipse with semi-minor axis FH, FW and rotated by the angle θ_o. For effective performance, 11 filters with different directions are used in the range $[-20, 20]$ degrees, for each filter size. Filters elongation has been set to a constant ratio of 5 with the following equation : $FW = 5FH$. The iterative process starts with a large filter height parameter (FH) set to a fraction of the estimated line height (ELH). At each iteration the filter height is decreased by 15%. The iteration process stops once the current (FH) is lower than a fraction of ELH. Because the filtering process has no blurring significant effect on the image below this value. On Figure 4.3 we can see the cumulated output of the steerable filters obtained for different filter sizes (from left to right). The potential line masks are obtained by binarizing these images as depicted in figure 4.4. This result highlight the capacity of the method to detect thin as well as thick lines where necessary in the image.

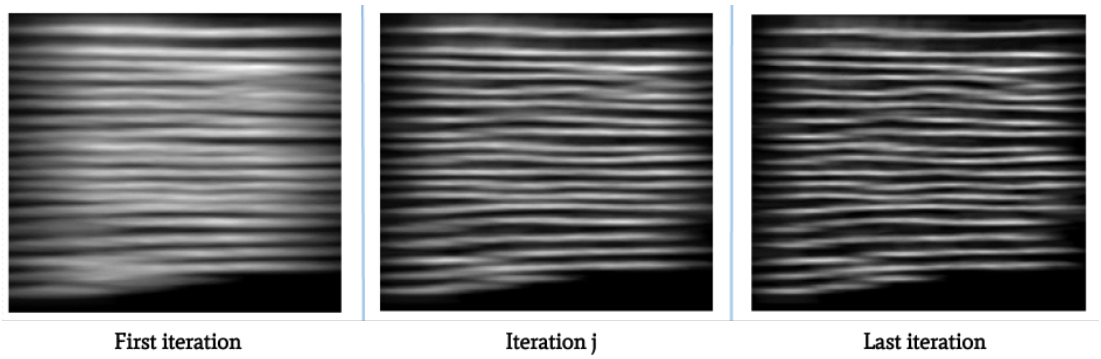


FIGURE 4.3: Steerable filters output for different values of the filter height (large, medium, low) from left to right.

At this stage, a text line mask may represent a group of lines, a part of a line or a single complete text line. Thus we need to check if the generated masks are valid before adding them to the validated text line masks image. Each connected component of the potential text line mask image is considered a valid line mask if its maximum vertical thickness is lower than twice its most frequent vertical thickness and lower than twice its average thickness. The underlying idea is that a line mask has almost a constant height so that its maximum, average and more frequent vertical thickness are relatively close to each other. Once a CC is validated as a text line mask, we add it to the final validated text lines masks image. The result of the validation process is illustrated in Figure 4.5 below.

The validated text line masks image obtained should ideally content each line of the document image by a single mask. However, when a text line of a document image overlaps other text lines or has a sharp skew variation or a very wide inter-word spaces, it may be decomposed into several masks, each covering a portion of that text line. To improve the final masks image, we a

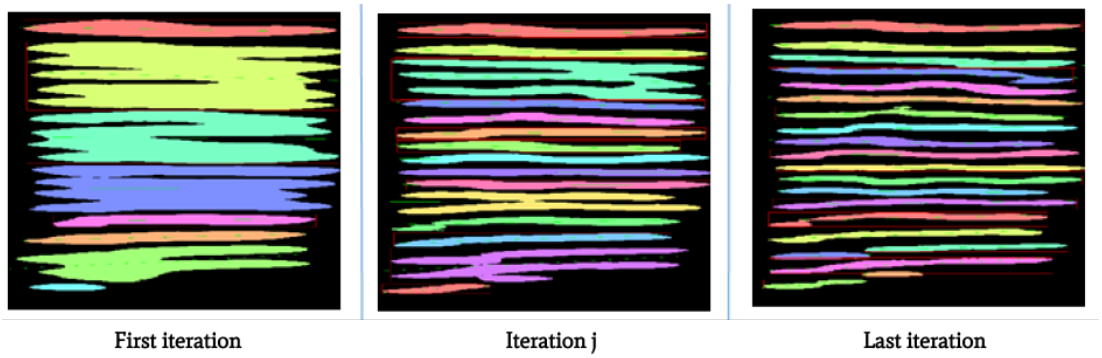


FIGURE 4.4: Potential line masks obtained for the corresponding steerable filters of figure 4.3 above

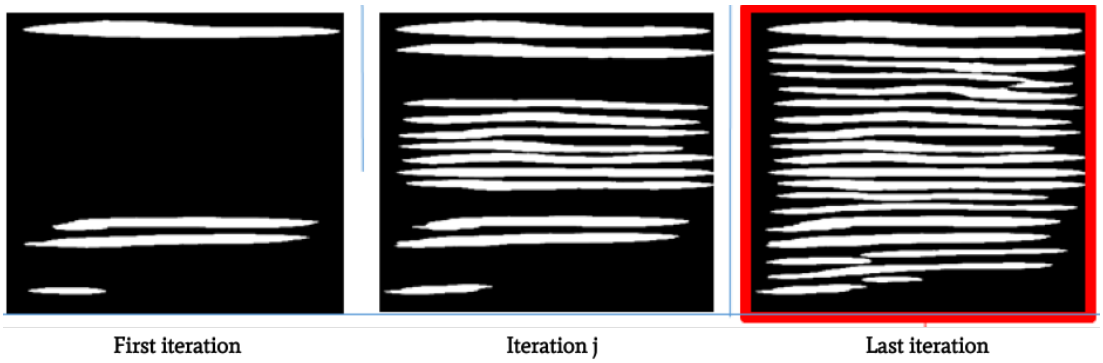


FIGURE 4.5: Results of the validation process of text line masks of figure 4.4 above

post-processing stage is applied which merges masks that appear to be horizontally overlapping. Then, text lines boundaries are determined using line masks positions. Finally the connected components are assigned to their respective lines. The text line masks obtained by this process are used to determine the boundaries between the text lines of the document. For this purpose, we compute an anisotropic distance transform (the distance along the horizontal axis has half the weight of that of the vertical axis) of the mask image. The resulting distance map allows to assign each pixel to its nearest mask and to obtain the line delimiters and then the desired line images as depicted on figure 4.6.

The line delimiters obtained are the default separators between lines of text but there are some overlapping situations where a stroke may extend below or above the delimiter, and we wish to affect the whole stroke to its proper line rather than cutting the stroke in two parts and affect each part to two different lines. Each connected component (CC) of the document image is assign to its proper text line using the following assignment rule. There are two possible situations: either a CC is totally included inside a line container or it is localized in between two lines. In this later case, we have to determine whether to assign CC to a single line or if CC must be splited. Let H_i be the percentage of the height of CC that is included inside a line container L_i. If H_i is greater than a given threshold (which we empirically fix to 66%), then CC is affected to L_i. Otherwise, we split CC at the Line delimiter and assign each

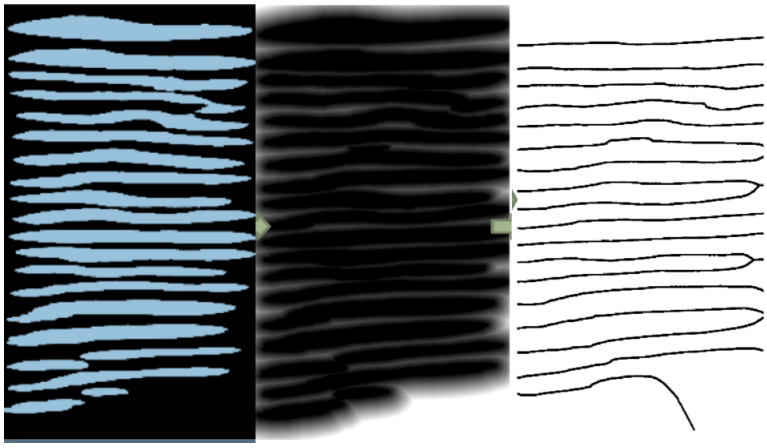


FIGURE 4.6: Text line mask, distance transform, line delimiters detected (left to right)

part of CC to its respective line.

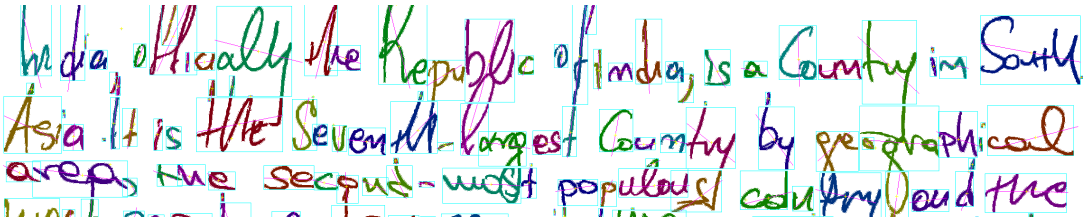


FIGURE 4.7: Connected components splitting according to the boarders image

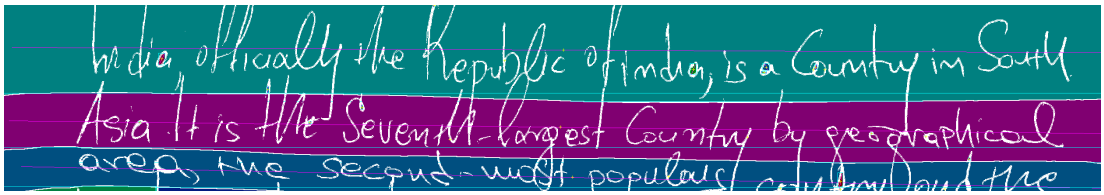


FIGURE 4.8: Connected components assign to their corresponding text lines

Evaluation and results

The performance of the proposed method was evaluated on Arabic, Latin and Bangla handwritten documents. The test datasets provided by ICDAR 2013 (containing 2,649 text lines), ICFHR 2010 (1,629 text lines), ICDAR 2009 (4,034 text lines) were used for evaluating the performance on Latin and Bangla handwritten texts, while we extracted from the OpenHart 2010 training dataset (provided by DARPA MADCAT) a subset of 10,445 text lines for testing the performance on Arabic handwritten documents. All tests have been conducted following the standard protocol defined by the ICDAR and ICFHR contests.

ICDAR and ICFHR data sets are originally oriented towards text line segmentation competitions, their documents do not include any non-text elements such as scratches, vertical/horizontal lines and random noise. In addition, the provided pixel-wise ground truth represents exactly the text lines delimiters even in the case of overlapping lines.

In contrast, the DARPA MADCAT training dataset was originally design for handwriting text recognition competition. Therefore, it includes many kinds of non-text elements such as custom lines background, scratches, etc... For the tests we selected the document images that do not contain custom lines (such information is provided with the MADCAT dataset). However, the provided ground truth gives only the bounding box position of words and text lines. This is insufficient for a precise segmentation of overlapping text lines. In order to get more precise line delimiters, we have used the word bounding boxes provided in the ground truth data so as to build a more precise line localization ground truth. Combining the word bounding boxes with the inter-words white spaces between bounding boxes allows to extract a more precise line image, as depicted in figure 4.9 below.

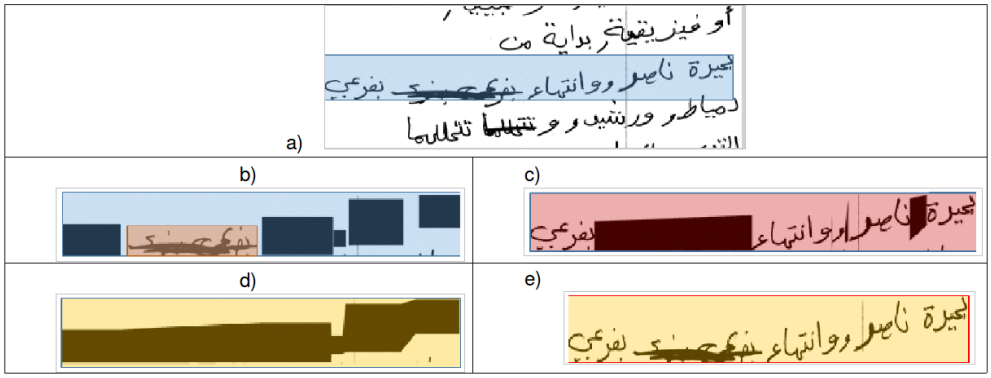


FIGURE 4.9: MADDCAT line delimiter ground truth e) derived from word delimiters ground truth b) c)

We can see that a small lower part of the line is still missing because it is not included in the bounding box of the text line. Following the ICDAR evaluation protocol, this could cause the rejection of a correctly detected text lines. The obtained ground truth is still too imprecise for an exact evaluation of text line segmentation. We should remember this limitation when comparing the line segmentation performance of the method on several datasets.

Test and validation protocols

We developed the test application as described in (Stamatopoulos et al., 2013,Gatos, Stamatopoulos, and Louloudis, 2010,Gatos, Stamatopoulos, and Louloudis, 2011). A $MatchRate(i, j)$ table is used to store the intersection values of the ON pixel sets of the result and the ground truth images. Let I denotes the set of all image pixels, G_j denotes the set of all pixels inside the ground truth image,R_i denotes the set of all pixels inside the result image and $T(s)$ is a counting function for the elements of the set s. The matching results between the j^{th} ground

truth image and the i^{th} result image is represented by the $MatchRate(i, j)$ table as shown from equation 4.5.

$$MatchRate(i, j) = \frac{T(G_j \cap R_i \cap I)}{T((G_j \cup R_i) \cap I)} \tag{4.5}$$

We validate a one-to-one match between images j and i only if the matching score is equal to or above 95%.

The global performance of the method FM is evaluated as shown in equation 4.7, by computing the detection rate DR that reflects the ability of the method to detect the text lines and the detection accuracy rate RA that reflects the method detection accuracy. If N is the number of the ground truth elements and M is the number of elements in the result, and $O2O$ is the number of validated one-to-one matches, then we calculate DR and RA using equation 4.6.

$$DR = \frac{O2O}{N} \quad , \quad RA = \frac{O2O}{M} \tag{4.6}$$

$$FM = \frac{2\ DR\ RA}{DR + RA} \tag{4.7}$$

Results

The test results of our method (that we called "LITIS") are shown in table 4.4. Compared to methods that have the best performance on each contest data set, our method has similar but slightly lower performance. We noticed that very few errors are due to a bad detection of line borders. The majority of invalid lines are due to a wrong CC split. There are two cases of wrong splits: either the ascender or descender of a character across two overlapping text lines was cut, or touching characters were segmented at the wrong position (Figure 4.10). Therefore, the weak point that affects the performance of our method lies precisely in the assignment of the connected component when they belong to more than one text line. As an advantage, the CUBS method is able to split the touching characters by using a specific algorithm that decides for the best cutting position (Shi, Setlur, and Govindaraju, 2009). The results obtained on the Arabic MADCAT dataset show a significant difference in the segmentation performance compared with the Latin script. Most of this difference comes from the low quality ground truth, as discussed above. One can reasonably suppose that the score obtained by our method on the MADCAT dataset is actually much closer to the scores obtained on other datasets. Nevertheless, to the best of our knowledge, these results are the first attempt to evaluate Arabic text line segmentation using the ICDAR competition protocol.

We searched for the optimal initial value of the filter's height parameter for each dataset by testing the various performance of the system for each value of the filter's height ranging between $0.5 \times MDH$ up to $1.5 \times MDH$ as reported in table 4.4. Figure 4.11 shows two examples of the text line segmentation results (different colour for each localised text line) applied on document images taken from the READ handwriting recognition competition 2017.

a. b. c.

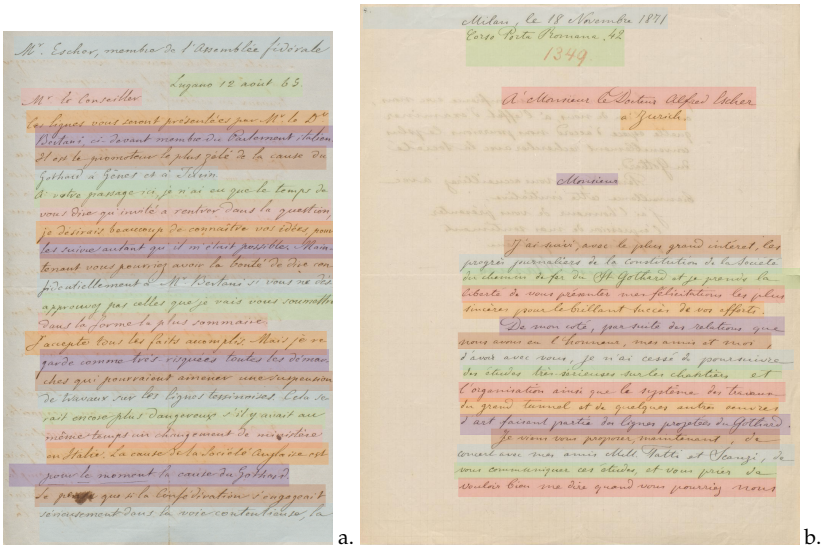
FIGURE 4.10: Illustration of a touching characters split error. (a) Two touching characters shown in green colour. (b) LITIS splitting decision level according to the lines delimiter found by the method. (c) Red dotted line represents the correct splitting position.

	ICDAR 2013	ICFHR 2010	ICDAR 2009	MADCAT 2010
Optimal filter's Height	0.8× MDH	MDH	MDH	1.5× MDH
LITIS	97.18 %	96.75 %	98.86 %	92.21 %
CUBS	97.45 %	97.63 %	99.53 %	——
Best ONE	INMC 98.66%	CUBS	CUBS	——

TABLE 4.4: Method evaluation results

4.4 Optical models implementations

During this thesis we have implemented various configurations of character optical models that have been combined with various language models. As depicted in figure 4.12 two types of models have been developed. We started our primary experiments with using the standard and popular generative Hidden Markov Models. But the current state of the art has led us to introduce LSTM-RNN optical models. Both models take as input some predefine features that describe 1 dimensional (1D) streams of real valued feature vectors. The strength of recurrent neural networks lies in their capacity to learn their own internal feature representation of the input data by the introduction of hidden layers. As a consequence it is possible to introduce simple features such as pixels values at the input of recurrent neural networks whereas hidden Markov Models will benefit of having some more elaborated features to model the character classes. In the following subsections we first give a brief overview of the used features descriptors, before giving some insights of the design and optimization (training) of the optical models.



a. b.

FIGURE 4.11: Illustration of smearing method resulting text line segmentations applied for two images taken from READ 2017 handwriting competition training set

4.4.1 Feature descriptors

A- Simple pixel based features

Some pixel based features can be extracted very easily from the gray level input images. The height of the input image is first normalized to a fixed height (possible values range from 64 to 100 pixels height). The width of the image is normalized accordingly, so as to keep constant the aspect ratio of the image (width / height). Then the 256 gray level pixel values are normalized in the range $[-1, 1]$. Finally, each column of pixels is the retained vector of feature.

B- HoG based features

More elaborated features can be extracted from the normalized input image. After height normalization of the image, skew and slant correction is applied on the binarized image in order to get the lines of text as much horizontal as possible and with handwritten components with as much vertically oriented as possible. The Histogram of oriented Gradient (HoG) is computed within a sliding window of 8 pixels width, by considering 8 orientations of the gradient in 8 sub-regions of the window. 64 real values features encode the HoG, while 8 geometrical feature descriptors are added. The horizontal sliding window has a 2 pixels step size in the horizontal direction.

4.4.2 Design and training optical models

The two models that we have been designed in this thesis are characterized by their generative and discriminative modelling ability respectively. Following equation 4.1, the optical models provide the recognition system with the likelihood estimations $P(S|W)$ of the observed feature vectors $S = (s_1, s_2, \ldots, s_T)$

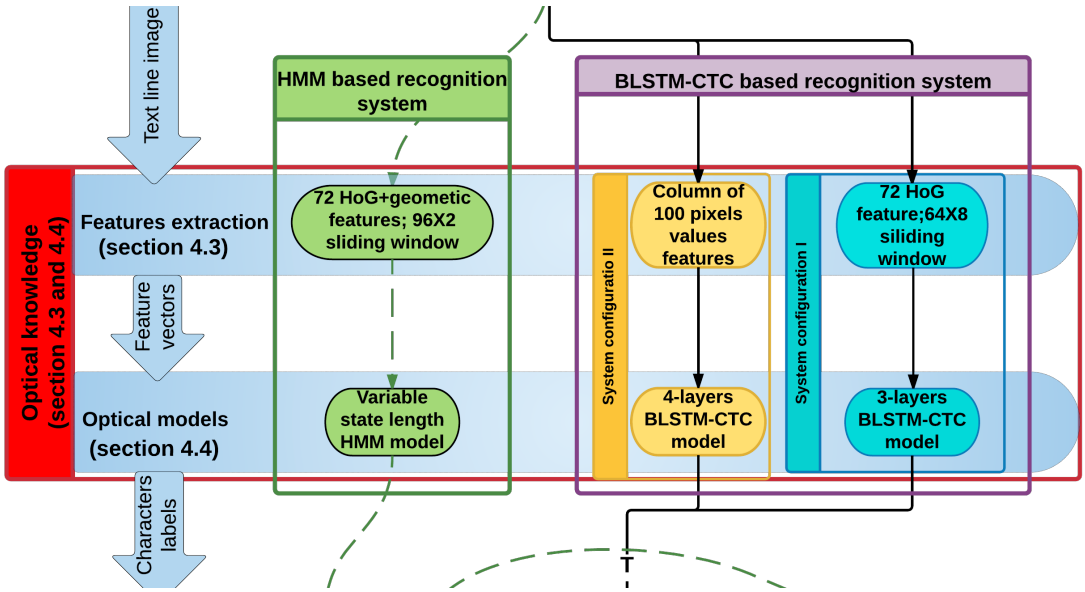


FIGURE 4.12: overview of the various optical models implemented in this thesis

towards a set of character classes. As presented in chapter 2, Hidden Markov Models are generative models that are optimized using the expectation maximization (EM) algorithm. This algorithm uses the likelihood criterion as training criterion. The BLSTM-RNN optical models are trained using a discriminative approach that uses the posterior probability $P(W|S)$ as training criterion and the Back-Propagation-Through-Time algorithm. In the following paragraphs we give some more details about the optical models architectures and their optimization procedure.

A- HMM model optimization

The internal structure of the HMM optical Models of characters is defined by a variable number of hidden states organized sequentially from left to right, and for each of them, a fixed size Gaussian mixture is also determined. We chose to use mixtures with 20 Gaussians, which guarantee a description ability fairly accurate for each frame of feature descriptors. Determining the number of hidden states is an optimization problem. An overestimated number of hidden states leads to over-trained models. An underestimated number of states leads to inadequate specialized models. This problem has been addressed in (Zimmermann and Bunke, 2002; Ait-Mohand, Paquet, and Ragot, 2014). We have been inspired by the proposed method in the first reference that is based on the Bakis method in order to optimize the number of states of each character model.

The principle of this method is as follows. Once a first training of one initial set of character models has been carried out with a fixed number of states (5 states per model), we compute the average number of frames F per optical model using a forced alignment decoding process of the corresponding models

on the ground truth of each image. The number of states E of the corresponding model is then defined as a fraction of F ($E = \alpha.F$ with α to be optimized). The figure 4.13 and figure 4.14 show the histogram of the number of states per HMM model for the RIMES and IAM character sets respectively. We can observe that character model structures are very similar for the two datasets.

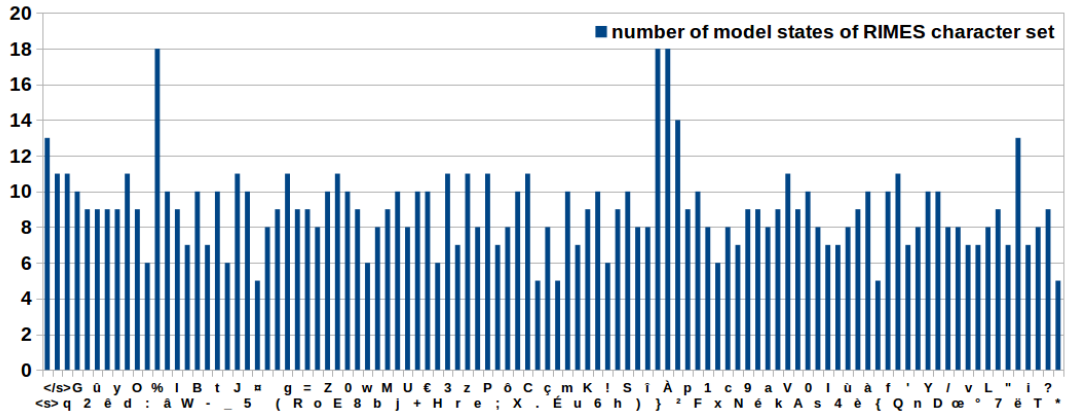


FIGURE 4.13: Number of states per HMM models of RIMES character set

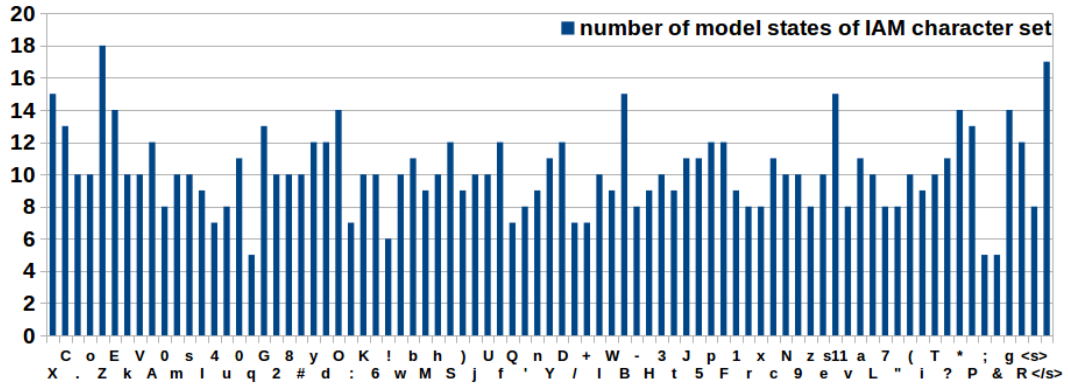


FIGURE 4.14: Number of states per HMM models of IAM character set

A new training process (Baum-Welch) is performed with the new variable length models that have been created, according to the parameter α. Then a final decoding without ground truth (no forced alignment) on the training dataset using the trained models is performed and the character error rate (CER%) is computed. The operation is repeated for different values of α (increasing values between 0.1 and 0.9). Finally we select the most accurate models based on a criterion combining the average CER and the misalignment rates of the models on the training examples. Indeed, excessively long character models tend to maximize the recognition rate but at the expense of misalignments on shorter examples. Misalignment can happen when a small number of frames are presented to a model with too many states during the training process which lead to eliminate the training example (technically the training examples elimination is called "over pruning"). The best model has to achieve the

minimum CER and the minimum misalignment rate (MAR%). This criterion is tested at each iteration of the Baum-Welch training procedure. Training is stopped after 30 iterations, with more iterations, we noticed that the best CER is reported within the 30 iteration. We then obtain optimized optical models. For the values of α listed in table 4.5, the selected the models which meet the selection criterion that minimize the CER% with minimum MAR% where the models that are trained with 0.75 and 0.65 values of α on RIMES and IAM test databases respectively that realise 38.12% CER with 3.60% MAR on the RIMES test dataset, however, 45.57% CER with 0.77% MAR on IAM test dataset.

Databases	Measures	α values [0,1]				
		0.55	0.65	0.75	0.85	0.95
RIMES	CER %	49,64	39,05	38,12	39,1	40,37
	MAR%	0.34	1.17	3.60	8.40	17.43
IAM	CER %	53,47	45,57	49,9	50,87	47,76
	MAR%	0.27	0.77	2.85	7.75	16.99

TABLE 4.5: Optical models alignment optimization for training on RIMES and IAM databases

B- BLSTM architecture design

The internal structure of the BLSTM-RNN optical model of characters is defined by a fixed number of neurons distributed and interconnected with each other along a certain number of stacked BLSTM layers ending at the top layer of the stack by a fully connected CTC layer. For the BLSTM-RNN based handwriting recognition system, we have design two optical model configurations as illustrated in figure 4.12. Configuration (I) : consists of HoG features extracted from the binarized input images that are used for training a BLSTM-RNN network composed of three layers, within each layer, there are 200 LSTM units. Configuration (II) : consists of columns of pixels feature descriptors extracted from the Gray level colour of the input image. The BLSTM-RNN network is characterized by four layers, each of which with 200 LSTM units.

During the training process the examples have been sorted according to their number of frames in order to apply the principle of the curriculum learning algorithm (Bengio et al., 2009), by starting the optical models training with first learning the short sequences and step by step go towards longer ones. For the two system configurations, the training by backpropagation through time of the neural network starts with a learning rate of 10^{-5} and continues with no change for 100 epochs. Then the learning rate starts decaying by a ratio of 0.5 and checking the enhancements of the network performance on the validation set. The training process stops, if the performance enhancement between two successive epochs is less than 0.0001. At every training epoch, the recognition performance of the optical models are reported and the best trained models are selected according to the character error rate (CER) on the validation datasets. The recognition performance on RIMES dataset for the two optical model configurations are illustrated in table 4.6, as well as for the English IAM

dataset. Configuration (II) performs much better than configuration (I) with a CER lower by 2.95% on the IAM dataset, and lower by 10.7% on the RIMES dataset. As a conclusion of this section we can highlight the performance of

Datasets	Optical models configurations	Training epoch #	Learning rates	CER% on validation dataset
RIMES	configuration (I)	200	$5.72643e^{-07}$	11,97
	configuration (II)	137	$6.04662e^{-08}$	9,03
IAM	configuration (I)	153	$2.1937e^{-10}$	22
	configuration (II)	120	$2.16e^{-06}$	11,3

TABLE 4.6: Optical model performance measured by character error rate CER during the training process on the training and validation datasets for the two configurations (I & II)

recurrent neural networks compared to those of HMM. BLSTM RNN achieve a 9.03% CER on the RIMES validation dataset and 11.3% on the IAM validation dataset, whereas the HMM models obtain a 38.12% CER on RIMES and 45.57% on the IAM dataset. Moreover, the BLSTM-CTC optical model with the pixel values shows a significant improvement of 2% CER on the RIMES validation dataset and 10.7% CER on the IAM validation dataset.

4.5 Language models and lexicons

The language model contribution to the final recognition likelihood score is the a priori probability $P(W)$ of the sentence as given in equation 4.1. This probability helps the recognition process to favour sequences of words that are linguistically valid and the most probable. In between the optical model hypotheses which provides sequence of characters' hypotheses with their likelihood score, and the language model input which are sequences of lexicon tokens, the recognition process has to integrate character concatenation rules given by the recognition lexicon.

As shown on figure 4.15, the language component of the system must integrate both the recognition lexicon and the language model. Following chapter 3, and as can be shown on this figure 4.15, closed vocabulary systems are defined by a working lexicon of words which prevent them to recognize out of vocabulary words (OOV), thus the systems will lack generalization capabilities. One way to increase the lexicon generalization capabilities is to increase the size of the working lexicon. However, large lexicons lead up to higher complexity (and perplexity) language models with which the recognition task has more difficulty due to the increased number of competing hypothesis which may content the target to be recognized. Reversely, small lexicons with high

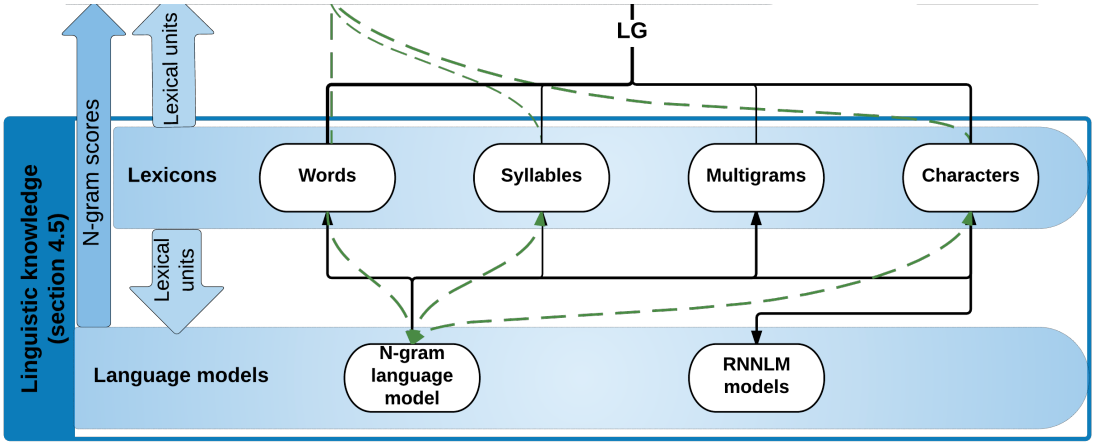


FIGURE 4.15: Illustration of the linguistic knowledge system components

OOV rates make the language models unable to guide effectively the recognition process. Therefore, it is necessary to find a compromise between the lexicon size and the Out-Of-Vocabulary rate when determining the language model lexicon. One alternative is to introduce sub-lexical units in place of lexical units. From the lexicon of word tokens, sub-lexical units can be derived in different ways to provide syllables or multi-grams, and this will be the topic of chapter 5.

Simple models using characters only have also emerged as one possible way to deal with OOV words. The open vocabulary systems can reach a nearly 100% coverage rates of words of the test datasets and consequently we can expect them to have lower Word Error Rates than closed vocabulary systems. This will however depend on the language model design and most importantly on the properties of the training corpus compared to those of the test corpus. Whatever the strategy adopted (closed or open vocabulary) the architecture of the system is the same as shown on figure 4.15. First a token lexicon of words, sub-words units, or characters is defined, then a statistical language model is trained on some training data in the form of a n-gram or a recurrent neural network language model based on the tokens that have been defined.

4.5.1 Lexicon properties

All along this thesis, the experimentations have been conducted on the two reference datasets available: the French RIMES dataset, and the English IAM dataset. It is worth mentioning here their respective properties in terms of lexicon sizes, coverage rates, and introduce some additional external linguistic resources that have demonstrated to be necessary to achieve state of the art performance, as they allow reaching much higher lexicon coverage rate, and improve the linguistic context generalization capability of the language model.

A- Tokenisation

Beside meaningful language words (a word is a sequence of characters sided by a white space character), plain text corpus may contain sequences of alphanumeric characters and symbols like bank account numbers, date or time formats (such as 02/08/2000 or 10:30'PM') which are counted as words according to the definition of a word, and the Word Error Rate metric, contrary to the spoken languages. Moreover, written languages include many punctuation marks that have to be identified as they count as a word in the WER metric for most of the competitions. At last we have to pay much attention on modelling the inter word white space. Indeed, most of the written languages include a blank separator between words that serve as an important visual word delimiter, whatever the word is a word of the working lexicon or an alphanumeric expression, whereas the ASCII space character can serve as a natural word delimiter when encoding texts in electronic format (and therefore space character is not introduced in most language models devoted to Natural Language Processing (NLP) Technics, here it is highly necessary that our optical model includes one class for the blank space (word delimiter). During decoding (see section below) as the system combines the optical model with the language model, it is necessary to integrate this optical model similarly to the integration of other special characters. As a consequence of integrating punctuation marks and blank character classes we must notice that the order of n-gram language models will be at least twice as the order of standard language models used by NLP technics.

The tokenization process is introduced as a necessary pre-processing step for preparing the training data according to the targeted type of language model. During the experimentations, we have applied the following tokenization process on the training text corpora, before training the language model. The training text corpora are cleaned from any character not found in the RIMES or the IAM datasets. Recall that the RIMES dataset contains 100 alphanumeric symbol classes and IAM dataset contains 80 alphanumeric symbol only. Then, token separators are explicitly introduced in the training corpus so as to allow training the language model with the desired lexicon entries and the specific alphanumeric characters, including punctuation marks and the white space character (word delimiters). By default our tokenizer includes vocabulary word tokens, punctuation marks token, numeric characters and symbols (excluding simple characters). When testing the sub-lexical language models introduced in Chapter 5, the tokenization process will be modified so as to decompose words into their sub-lexical units as well.

Some additional pre-processing steps have demonstrated to be effective during the experimentations. Indeed, the text datasets are generally provided with line breaks. By removing line breaks in the training corpus we aimed to improve the performance of the language models by giving them the ability of modelling long sequence dependencies of tokens (words, syllables, multigrams and characters). Experiments have demonstrated a decrease between 1% and 2% of the WER by removing line breaks from the training corpus when using the Wikipedia+RIMES corpora. Finally, we should not forget that

whereas we try to build the best suited language model to get the best open-vocabulary recognition system by introducing a specific tokenization process that allow training the desired model, the evaluation process is conducted without no consideration about the internal representation of the language model of the system. For the evaluation process, only the Word Error Rate is computed, which evaluates the capacity of the system to recognize any word (in lexicon or OOV) delimited between two white spaces, and by counting punctuation marks.

B- Lexicon statistics

A first group of lexicons is derived from the vocabularies of the RIMES and IAM datasets only. In table 4.7 we see that the lexicon of RIMES training and validation datasets achieves a 93.7% effective coverage rate (ECR) on the test dataset with a small lexicon size of 5867 word tokens. This demonstrates the homogeneity of the RIMES datasets in terms of lexicon coverage. In table 4.8

Lexicon type	ECR% on the test dataset	Lexicon size (words)	Vocabulary size (tokens)
RIMES train	93.29	8107	5484
RIMES valid	84.42	2354	1810
RIMES train + valid	93.72	8760	5867

TABLE 4.7: RIMES dataset lexicon properties.

below, we see that the training and validation IAM datasets do not allow reaching sufficiently high lexicon coverage rates on the IAM test dataset to allow getting interesting recognition performance. This raises the important question of choosing some external resources that could compensate for the lack of available data in the training corpus. This is one of the most important difference between the two RIMES and IAM datasets.

Lexicon type	ECR% on the test dataset	Lexicon size (words)	Vocabulary size (tokens)
IAM train	75.45	9956	7414
IAM valid	67.17	2992	2219
IAM train + valid	78.92	11615	8291

TABLE 4.8: IAM dataset lexicon properties.

It is only by using some additional external resources that we can expect having high lexicon coverage rates of the IAM test datasets and then have the recognition performance sufficiently high. This is observed in the recent references (Voigtlaender, Doetsch, and Ney, 2016; Bluche, 2015; Bunke, Bengio, and Vinciarelli, 2004) also, which have introduced the LOB corpus, the Brown corpus, as well as the Wellington corpus as additional resources. This lead us to consider a second group of lexicons which are derived from the training dataset lexicons and by adding some external resources. The statistics of these lexicons are presented in table 4.9 for the RIMES dataset, and in table 4.10 for the IAM dataset. For the RIMES dataset, we considered a subset of the most frequent words or the whole French Wikipedia resource. For the IAM dataset we considered the English Wikipedia (or a subset of it) or the original resources that have served to build the IAM dataset (the Lob, Brown and Wellington corpora).

Lexicon type	ECR% on the test dataset	Lexicon size (words)	Vocabulary size (tokens)
50K,French Wikipedia	82.71	50000	29096
50K,French Wikipedia ,+ RIMES train	94.27	50000	29111
French Wikipedia	87.29	182735	88959
French Wikipedia,+ RIMES train	96.66	185860	90519
French extended Wikipedia	93.91	702097	285985
French extended Wikipedia,+ RIMES train	97.70	704191	287056

TABLE 4.9: External resources lexicon properties on the RIMES test dataset.

Regarding the RIMES dataset, table 4.9 above shows that low coverage rates of the test dataset are achieved by considering only the external resource for whatever configuration (whole Wikipedia or most frequent words of it). In the best case, a 87% coverage rate is achieved but with a lexicon of more than 88K word tokens. Whereas a 82% coverage rate is achieved with a lexicon size of $29K$ word tokens. The coverage rate of the test dataset is only increased by 0.5% by considering together the RIMES training dataset and the most frequent words of the French Wikipedia. By extending the Wikipedia corpora, an equivalent coverage rate (93.91%) to the RIMES training dataset (93.7%) is obtained at the expense of using a large lexicon of $286k$ word tokens which is not manageable by the recognition system.

Regarding the IAM dataset, we can observe from table 4.10 above that, the English Wikipedia achieves a 90% coverage rate of the IAM test dataset but with a lexicon of more than $240,000$ words ! The LOB corpus achieves a better ECR of 92% but with a lexicon of $90,000$ words, which shows that this resource is more adapted to the IAM dataset, as was expected. Brown and Wellington

Lexicon type	ECR% on the test dataset	Lexicon size (words)	Vocabulary size (tokens)
50K,English Wikipedia	84.6	50000	29473
50K,English Wikipedia,+ IAM train	84.84	50000	31308
English Wikipedia	91.85	240659	103943
English Wikipedia,+ IAM train	92.19	242928	104871
English extended Wikipedia	95.76	994545	311712
LOB corpus	99.33	98716	87777
LOB corpus including IAM train only	92.7	95526	87501
Brown corpus,+ IAM train	90.1	102775	51196
Brown corpus	89.35	100236	51165
Wellington corpus,+ IAM train	91.47	98254	49234
Wellington corpus	91.08	95781	48122
50K,LOB+Brown+Wellington	91.12	50000	29697
LOB+Brown+Wellington	95.13	199521	87501

TABLE 4.10: External resources lexicon properties on the IAM test dataset.

corpora achieve very similar ECR. Finally, by considering the most frequent $50,000$ words of both LOB, Brown, and Wellington corpora, a 91.12% ECR is achieved but with a lexicon of $29,697$ word tokens only; whereas a ECR of 95% is obtained when considering the whole three corpora of $199,521$ words. An equivalent hight coverage rate is obtained by using the English extended Wikipedia corpora (95.76%) with a lexicon of $994,545$ words which is not manageable by the recognition system. From these lexicon statistics, we see that the two popular datasets exhibit different properties in terms of lexicon coverage of their test dataset by their training datasets. As the lexicon coverage rate achieved by the language model is the upper bound of the WER of the recognition system, improving the coverage rate of the language model is one necessity for improving the WER, whatever the performance of the optical model, this is particularly necessary for the experimentations that will be conducted on the IAM dataset. A first strategy is to increase the lexicon size, but this is at the expense of having a large language model of relatively high order. A second strategy is to introduce sub-lexicon units in the language model. Chapter 5 will address this issue.

4.5.2 Language model training

All along this thesis we have conducted most of our experimentations by integrating n-gram statistical language models. However, some primary experiments have been conducted by using Recurrent Neural Networks Language Models. In this section, we give some details about training these two models and their first evaluation.

A- N-gram language models

During this thesis, we implemented various n-gram language models with different toolkits. We started our primary experiments by combining HMM optical models with n-gram LM models. At this time the language models were estimated by the constant discounting smoothing method with using the SRILM toolkit (Stolcke, 2002). Later on, when introducing the BLSTM-RNN optical model, we used modified Kneyser-Ney smoothing provided by the MIT language modelling toolkit (Hsu, 2009) for training n-gram language models. Modified Kneyser-Ney smoothing achieved similar or better performance compared to other estimation methods. We favoured the use of MITLM toolkit because it offers the ability to tune the model (Kneyser-Ney) parameters using the Limited-memory BFGS (Broyden-Fletcher-Goldfarb-Shanno) optimization algorithm using the validation datasets which consequently minimizes the language model perplexity.

As mentioned in the previous section, due to the presence of white spaces and punctuation marks, high order n-gram language models are required to capture long distance dependencies. Remember that a n-gram of order n will in fact account for modelling word dependencies of order $n/2$. For example, $n = 10$ (which at first sight may appear a rather long history in the sentence) accounts for the last 4 words back in the past only. A value which is not that large in the field of statistical NLP. For validating the necessity of having high order n-gram language models for handwriting recognition, we evaluated the contribution of various n-gram language models of increasing order to the recognition performance of the system. This was repeated for various optical models (HMM and RNN) and using word n-gram language models. Language models have been estimated from the RIMES train and Wikipedia + RIMES train training corpus. Table 4.11 below report the performance of two systems.

Training datasets	n-gram	2	3	4	5	6	7	8	9
RIMES train	HMM	22.7	17	17	16.8	16.9	16.9	17	17
	BLSTM	21.7	17.01	17.27	16.99	16.81	16.42	16.47	16.36
Wikipedia+ RIMES train	HMM	36	29.8	29.6	29.5	29.3	—	—	—
	BLSTM	22.88	16.74	16.68	15.72	16.06	15.49	15.45	15.76

TABLE 4.11: Influence of word language model order on the RIMES dataset, using a closed lexicon of RIMES train and Wikipedia + RIMES train tokens, and for two optical model (WER%).

Best recognition performance of the HMM based recognition system are reported when using 6-gram language models. Whereas the BLSTM-RNN recognition system achieved its best performance with 9-gram language models of words. These results show that high order n-gram language models (in the range 6 to 10) are to be preferred to low orders. Of course, there are some well known drawback of using high order models, the most important one being the complexity of the model and number of parameters. What we can notice here looking at these results is that parameter estimations using smoothing techniques such as modified Kneyser-Ney are robust to the data sparsity problems that can be encountered on too small training datasets.

Nevertheless, there are several problems with using n-gram language models, the most important one being probably the problem of context (or topic) coverage. In fact, like the notion of lexicon coverage (Effective Coverage Rate), we can never guarantee that a n-gram language model trained on a certain corpus will generalize well on a specific test corpus. As we will see in the next chapters, the linguistic context coverage is not limited to the lexicon coverage, we expect that using a language models trained on a certain resource will help recognizing words of another resource. Having generalizable language models require the models to be trained on sufficiently general and large resources. Another limitation of statistical n-gram LM is that they don't introduce some more elaborated syntactical or grammatical knowledge. For example, n-gram don't use any notion of word classes or categories. Finally, a compromise between the n-gram order, the recognition performance, and the computing capacity of the machine in terms of time and memory space is to be determined for real applications. Such considerations have been mostly out of the scope of this study as our systems have been implemented on machines with huge memory capacity (256 GB).

B- Recurrent neural network language models (RNNLM)

The advantage of the connectionist language models such as RNNLM over the n-gram language models is their capacity to model the language variable length dependencies of words. The primary experiments that we conducted was using RNNLM during a second decoding pass (see next section "decoding" below) dedicated to rescoring the sentence recognition hypotheses. Due to lack of time, the primary experiments were only conducted using character language models. 5-gram and 10-gram character language models were used in the first decoding pass to generate the possible recognition hypotheses and their associated weights represented by a token-lattice. The RNNLM effective parameter is the number of units in the hidden layer of the neural network. Three RNNLM language models have been trained on the Wikipedia + RIMES training datasets with 50,100 and 1000 units in the hidden layer respectively. The system with RNNLM at the second decoding pass overcomes the n-gram models, as can be seen in table 4.12. It must be noticed however that these experimentations have been carried out using a three layers BLSTM with HOG features, which later on demonstrated to perform lower than other architectures. Therefore these results are only provided to attest the capacity

of RNNLM on language modeling task. However, as reported in the literature, we also observed that training RNNLM language models is time consuming, and requires multiple trials for choosing the proper number of hidden units in the network (Yousfi, Berrani, and Garcia, 2017). This is also the reason why we did not continue our experimentations in this direction further.

1^{st} pass decoding	Character RNNLM			Character n-gram LM
	50 units	100 units	1000 units	
5-gram	28.64	25.27	22.28	25.96
10-gram	22.06	18.67	18.31	18.92

TABLE 4.12: performance of a second pass character RNNLM on the RIMES dataset compared to a traditional n-gram LM.

4.5.3 Decoding

Decoding is the process of combining the optical model character hypotheses together in order to build admissible character sequences (i.e. words / tokens) that belongs to the working lexicon and that are the most likely with respect to the language model. Through the decoding step, we aim to determine the most probable word sequence W given the observed optical feature sequence O using the information produced by the optical model and the language model. Decoding is the optimization process depicted by equation 4.1 introduced at the early beginning of this chapter. It can be solved using Dynamic Programming. The time synchronous beam search Viterbi algorithm is the traditional decoding algorithm used for achieving speech and handwriting recognition tasks. Ideally, a search algorithm should consider all possible hypotheses based on a unified probabilistic framework that integrates all knowledge sources (optical and linguistic knowledge) (Xu et al., 1996).

When the explored search space becomes unmanageable, due to the increasing size of the vocabulary or the language models (high order), the search might be infeasible to implement or too long run. Then the solution is to restrict the search space to a sub-set of hypotheses using what is called a beam. During Viterbi decoding at every time step, search paths that are unlikely to succeed (less probable than the best path by a factor δ) are removed from the list of hypotheses; thus the name Time Synchronous Beam Search ($beam$ search parameter used for limiting the search space and $beam_lattice$ parameter which influences speed of the lattice extraction). But this pruning strategy alone cannot cope with the complexity involved by managing high order language models. In practice, time Synchronous Beam Search is generally implemented using a limited history of two words at most (a 3-gram language model). This constitutes a first decoding pass during which optical models, lexicon entries and a low order language model are combined following equation 4.1. Outputs of the first pass decoding is a word lattice (token lattice in general) that contains the N-best hypotheses as depicted on figure 4.16. A word-lattice (or

token-lattice) consists of the possible interconnected recognition hypotheses weighted by the optical models and language model's weights. A second decoding pass analyses the hypotheses network (word-lattice) using a n-gram language model of higher order, which allows re-weighting the hypotheses. Alternatively, some RNN LM can be used during the second decoding pass.

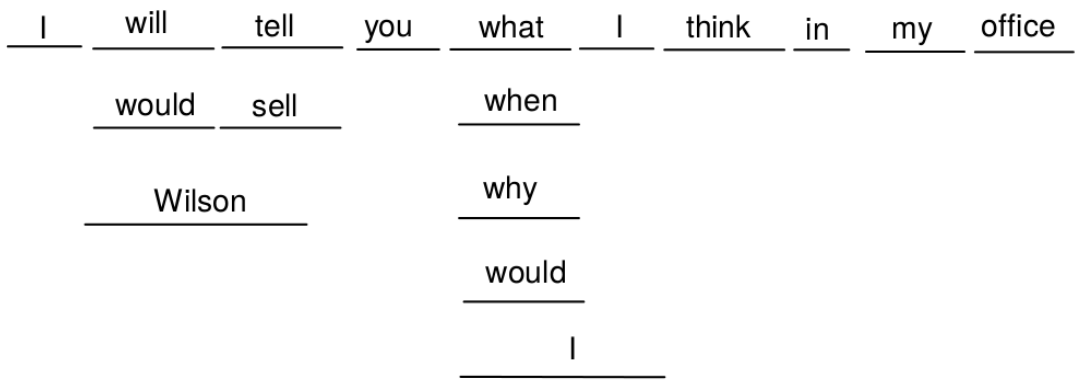


FIGURE 4.16: Example of a word lattice produced by the N-best time synchronous Viterbi beam search algorithm. Each word occupies an explicit time interval.

Two important parameters guide this first decoding pass: they are the language model scaling parameter γ and the word insertion penalty parameter β that controls the insertion of too frequent short words. These two parameters need to be optimized for optimum coupling of the optical model with the language model, because these two models are estimated independently from each other during training. Both the HMM based and the BLSTM-CTC based recognition systems have been implemented using a two pass decoding algorithm. In practice, multipass search strategy using progressive knowledge sources can generate better results than a search algorithm forced to use less powerful models due to computation and memory constraints (Xu et al., 1996). We now give some details about optimizing the coupling parameters of the first decoding pass, before giving some more information on the algorithmic implementations.

A- Parameter optimization algorithm

The language model and optical model coupling parameters have a direct effect on the recognition performance. With the HMM optical models, HTK's decoding toolkit was used. The HTK decoding toolkit uses the token passing paradigm to find the best path and / or multiple alternative paths. In the latter case, it generates a lattice containing the multiple hypotheses which can if required be converted to an N-best list. For the token passing algorithm, a token represents a partial path through the recognition network extending from the time 0 through to the time t of decoding.

The word insertion penalty is a fixed value added to each token when it transits from the end of one word to the start of the next. The language model scale parameter is the amount by which the language model probability is

scaled before being added to each token as it transits from the end of one word to the start of the next (Young and Young, 1993). During the first pass decoding, the HTK decoding toolkit make use of the optical model and language model coupling parameter in addition to the beam search parameter and build a token-lattice according to the equation 4.1. Then, during the second pass decoding, the produced token-lattice is rescored by tuning the language model scale parameter using the same language model used in the first decoding pass or with using higher order n-gram language model.

The simplified way proposed in the Kaldi toolkit for optimizing the coupling of the two models is to introduce one single coupling parameter defined by equation 4.8 below, in which the *optical-scale* parameter α stands for the inverse of the language model scale parameter γ; thus we can writer $\alpha = 1/\gamma$. By comparing equation 4.1 and 4.8. we can see that optimizing the insertion penalty parameter is not possible in the standard first pass decoding with Kaldi.

$$\hat{W} = arg\max_{w} P(S|W)^{\alpha} P(W) \tag{4.8}$$

Thus, we developed our own optimization algorithm 1 within the Kaldi framework aiming to optimize the language model parameter and the word insertion penalty parameter during the second decoding pass as was possible with the HTK toolkit (during the first decoding pass). The parameter optimiza-

Input: validation set examples, n-gram language models FSTs
Output: best language model order, best language model scale, best word insertion penalty
$\gamma \leftarrow$ the best language model scale parameter;
$\beta \leftarrow$ the best word insertion penalty parameter;
$n-gram \leftarrow$ the best n-gram language model $init_{\gamma} \leftarrow$ initial language model scale used for token-lattice generation = 0.1 ;
$beam \leftarrow$ the beam search parameter used for limiting the search space = 16.0;
$beam_latt ice \leftarrow$ the beam used for reducing the token-lattice size = 10 ;
$WIP \leftarrow$ a set of word insertion penalty values [-2,2] with a step value of 0.5 ;
`/* token-lattice generation through` 1^{st} `pass decoding */`
forall *n-gram models* **do**
  forall *WIP* **do**
    forall *validation set examples* **do**
      * Building a world lattice by decoding the validation set examples using the initial parameters and the n-gram language model;
      * Add insertion penalty value WIP to the generated token-lattice;
      * Store penalized token-lattice at each value of WIP
    end
  end
end
`/* token-lattice rescoring during 2nd pass decoding */`
$LMS \leftarrow$ a set of language model scales [0.1, 0.6] with a step value of 0.01 **forall** *Penalized token-lattices* **do**
  forall *LMS* **do**
    * Rescoring the token-lattice with the language model scale LMS. * Decoding the tokenl-attice returning the best bath as the recognized text line. * Computing the word error rate WER over all the validation set and getting the WER scores as a results.
  end
end
`/* Best parameter setting of n-gram language model, language model scale and word insertion penalty */`
$\gamma, \beta, n-gram \leftarrow$ min(of all WER scores) ; `/* retrieving the best WER score which indicates to the best parameters */`
return $\gamma, \beta, n-gram$;

Algorithm 1: Coupling parameters optimization algorithm introduced in the Kaldi toolkit.

tion algorithm is applied on the validation set and once the optimum values

of the coupling parameter have been determined, we carry out the decoding algorithm on the test dataset with using the optimum values of the coupling parameters.

B- Decoding configurations / implementations

For the HMM based recognition system and during the first pass, we used a bigram language model to produce a token-lattice using the initial optical and language models parameters in addition to an important beam search parameter to limit the space of search within a reasonable limits of decoding time with regards to the recognition accuracy. In the second decoding pass, we re-scored the generated word-lattice using a 6-gram language model of words, syllables or characters according to the decoding scenario. HTK **HVite** decoding tool was used for the first decoding pass and the word-lattice scoring was achieved using **lattice-score** tool from SRILM during the second decoding pass.

First the character hypotheses are arranged in a lexical tree where the first character of a word corresponds to the root of the lexical tree and the last one corresponds to the leaf, this representation has the advantage to be compact by sharing the common prefixes of the whole lexicon on a single branch of the tree. One other advantage is that lexicon trees can be efficiently encoded using a Finite State Transducer (FST).

We designed the BLSTM-RNN based recognition system, through the use of Weighted Finite State Transducers (WFST) automaton of Kaldi toolkit. These methods are recently applied in continuous handwriting recognition (Bluche, Ney, and Kermorvant, 2014). By definition, a finite state transducer is a finite automaton whose transiting states are labelled with a couple of input and output symbols. A weighted finite state transducer adds a weight to each transition i.e. to each pair of input and output symbols. Weights represent probabilities, penalties or any other quantitative values that accumulates over the transducer paths in order to compute the overall weight (cost) of mapping the input symbol string to the output symbol string. Thus, a path through the transducer encodes the matching between an input symbol sequence or string and an output symbol sequence or string. The reader can refer to (Mohri, Pereira, and Riley, 2008) for more details about the weighted finite state transducer theory and algorithms.

By using the general framework of WFST it is possible to design a specific transducer for each component of the system : an optical model transducer, a lexicon transducer and finally a language model transducer. Then it becomes easy to combine these three components into a single WFST that will guide the first pass decoding. EESEN speech recognition implementation (Miao, Gowayyed, and Metze, 2015) uses Kaldi's toolkit (Povey et al., 2011) and highly-optimized FST libraries such as OpenFST (Allauzen et al., 2007). The WFST decoding stage considers the CTC labels, lexicons and language models as separate weighted finite state transducers denoted by T, L and G respectively. After compiling the three WFST individually, they are composed gradually into a global compact search graph S by using two special WFST operations known by Weighted determinization and minimization algorithms

that optimize their time and space requirements. For more details about the determinization and minimization algorithms one can return to (Mohri, Pereira, and Riley, 2002). The overall order of the search graph generation FST operations is given by:

$$S = T \circ min(det(L \circ G)) \tag{4.9}$$

where $\circ, det$ and min denote composition, determinization and minimization respectively.

Optical model transducer T: The first WFST component maps a sequence of frame labels to a single character unit. The character WFST is designed to subsume all the possible label sequences at the frame level that correspond to a character. Therefore, this WFST allows occurrences of the blank label ϕ, as well as any repetition of one specific character non-blank labels. For example, after processing 5 frames, the RNN model may generate 3 possible label sequences "AAAAA", "ϕ ϕ A A ϕ". The token WFST maps all these 3 sequences into a singleton character unit "A". We denote the token WFST by T.

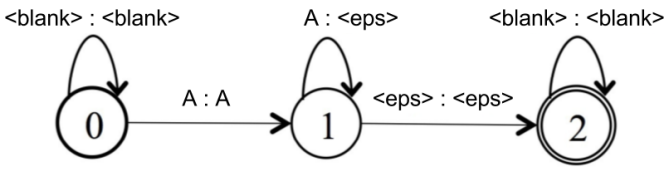


FIGURE 4.17: An example of the token WFST which depicts the phoneme "A". We allow the occurrences of the blank label "<blank>" and the repetitions of the non-blank label "IH"

Lexicon transducer L: A lexicon WFST encodes the mapping from sequences of character units to lexicon units. The input string is composed of characters, while the output is a lexicon token with the necessary epsilon (nul output token). The lexicon WFST is denoted as L. Figure. 4.18 illustrates the structure.

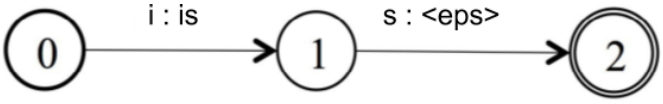


FIGURE 4.18: The WFST for the lexicon entry "is i s". The "<eps>" symbol means no inputs are consumed or no outputs are emitted.

Language model transducer G: A language model WFST encodes the permissible word sequences in a language/domain. The WFST shown in Figure. 4.19 represents a trivial language model which accepts two sentences "how are you" and "how is it". The WFST input symbols are the word tokens, and the outputs are the words with the arc weights are the language model probabilities. With this WFST representation, CTC decoding in principle can leverage any language models that can be converted into WFSTs.

During the decoding process higher order language models are used for maintaining long distance dependency. For that, one need to take into account the token dependency between the first token of a text line and its history of token defined by the last few tokens located in the previous text line. Therefore,

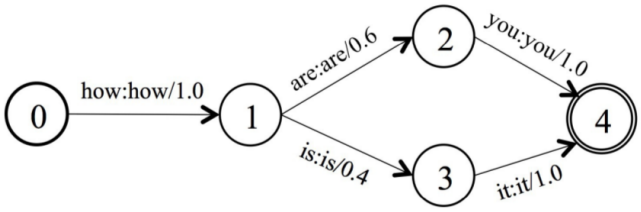


FIGURE 4.19: A trivial example of the grammar (language model) WFST. The arc weights are the probability of emitting the next word when given the previous word. The node 0 is the start node, and the double-circled node is the end node (Miao, Gowayyed, and Metze, 2015)

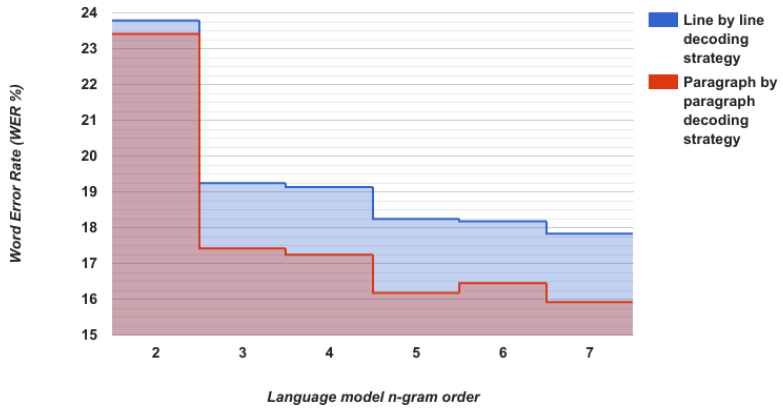


FIGURE 4.20: Line by line versus paragraph by paragraph decoding effect on the recognition performance in WER

we decided to adapt our decoding algorithms of the two system to decode paragraph by paragraph rather than the traditional way of line by line decoding. Figure 4.20 shows that we gain around 2% of recognition performance enhancement when using the paragraph by paragraph decoding strategy compared with the recognition performance obtained by the line by line decoding strategy. The primary recognition results illustrated in figure 4.20 was obtained on RIMES test dataset by using the BLSTM-RNN based system with HoG features and language models of words of order 2-gram upto 7-gram trained on Wikepedia + RIMES training dataset.

4.5.4 Primary recognition results

The conventional language models of words are evaluated and primary results is obtained by the handwriting recognition system making use of the HoG features. With the HMM models we built a handwriting recognition system prototype and studied the relation between the lexicons size and the OOV rate and the consequence of the OOV rate on the recognition performance measured by the Word Error Rate (WER%).

Figure 4.21 shows the performance achieved on the RIMES and IAM dataset by HMM optical model, and using different lexicons. Notice that a 100% coverage rate is achieved with the RIMES/IAM dataset as the language model was

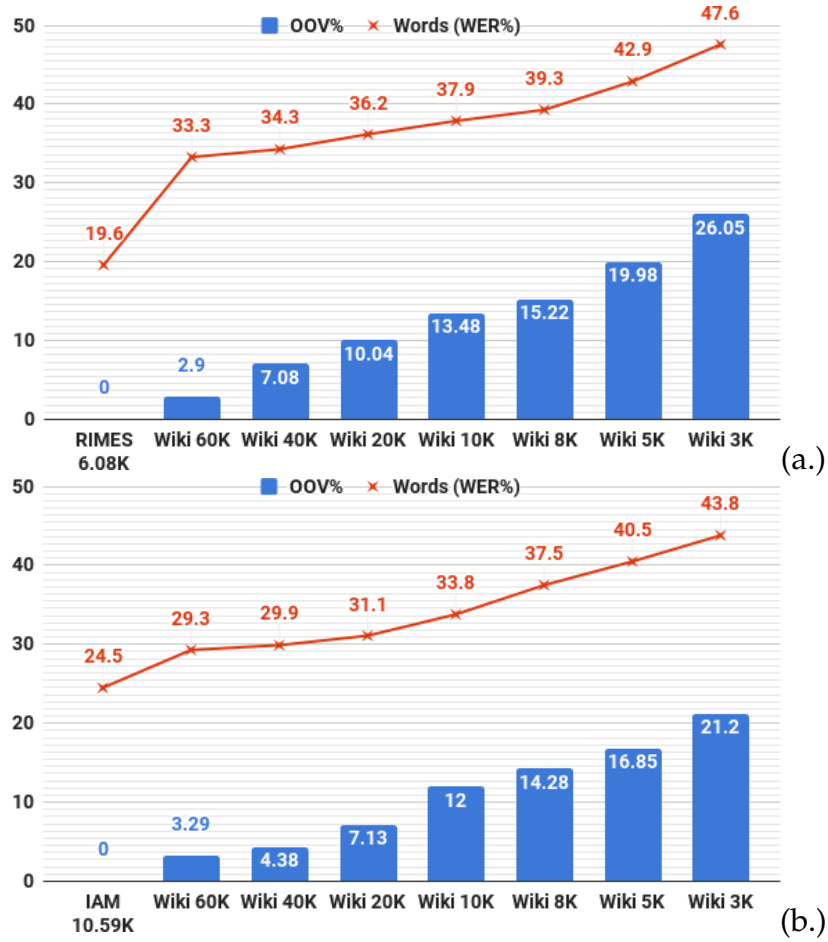


FIGURE 4.21: Primary recognition results with HMM optical models and language models of words; a. RIMES and French wikipedia 6-gram LMs and lexicons tested on RIMES test dataset, b. IAM and English Wikipedia 6-grams LMs and lexicons tested on IAM test datasets

trained using both training and test datasets. The other experiments have been conducted with the French Wikipedia corpora for training a set of language models of words for which a set of lexicons (wiki 60k down to wiki 3k word) was selected. The OOV rates reported on the RIMES test dataset increase proportional to the the lexicon size and the same phenomena is observed on the WER. This highlight the correlation between the lexicon size and the WER for the closed vocabulary systems.

The same phenomena was observed on the IAM and English Wikepedia language models of word tokens which are illustrated in the right side of figure 4.21. With the BLSTM-RNN based recognition system, we wanted to figure out the relationship between the effective coverage rate, OOV rate, lexicon size and the recognition performance in WER. The idea is to evaluate the recognition performance with language models of word tokens that are trained on the whole RIMES/IAM corpora (training, validation and test) with lexicons which cover effectively the test data set by $100\%, 95\%, 90\%, 85\%, 80\%$ (ECR). The lexicon was arbitrary selected from the whole RIMES/IAM corpora to attend the

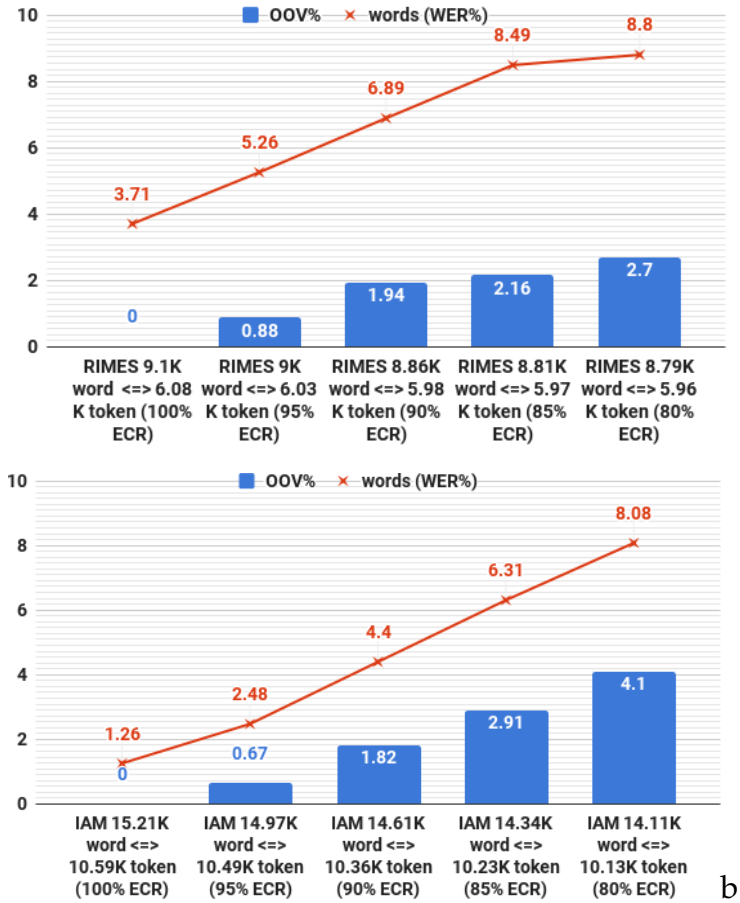


FIGURE 4.22: Primary recognition results with BLSTM optical models and language models of words; a. RIMES 9-gram LMs and lexicons whose ECR rates are decreasing from 100% to 80%, b. IAM 9-gram LMs and lexicons whose ECR rates are decreasing from 100% to 80%

specified ECR rate. Recalling the definitions discussed in chapter 3, one have to remember that the ECR rates are calculated at the word level (standing on the definition of a word as a sequence of character sided by two white spaces) and the OOV rates are calculated at token level which define the language model building blocks.

Looking at figure 4.22 which illustrate the result obtained by the BLSTM based recognition system on the RIMES and IAM test datasets, the extreme left points of result of curve (a) and (b) which shows the recognition performance with the ideal language model of word tokens which has no OOV% (100% ECR). These results illustrate the behaviour of the ideal language models of word tokens. By comparing the recognition performance of the HMM based and the BLSTM based recognition systems on the RIMES and IAM test dataset with the ideal language models of word tokens, the BLSTM based recognition systems outperforms the HMM base ones by 15.86% and 23.24% WER respectively. The next results on the right side of the ideal results of figure 4.22 illustrate the proportional relationship between the ECR rate, OOV rate, WER and the lexicon size. While the ECR is decreasing, the OOV increase and the WER

increase proportionally.

From these primary results, one can conclude that the recognition performance (in WER) with the conventional language model of word tokens is affected proportionally with the OOV rates. The OOV rates decrease by increasing the lexicon size. However, this increases the recognition system complexity and some times becomes unmanageable. Therefore, it is required to find a compromise between the language model vocabulary size and the coverage rate on the target dataset by that vocabulary. Our main contribution in this theses is to introduce the use of the language models of sub-lexical units in place of the conventional language model of words. The sub-lexical units represent the intermediate modelling units used for language modelling by which one can realize the intended compromise.

4.6 Conclusion

We introduced in this chapter the description of the main processing chain of our recognition systems which are based on HMM and BLSTM-RNN models. The main processing chain consists of the text line extraction, feature selection and extraction, optical models, language models and decoding process. The two systems differ from each other by their optical models types (HMM or BLSTM-RNN).

The two recognition systems (optical and language models) were trained and evaluated on the French RIMES and the English IAM datasets. Different large and external databases was used for training French and English language models with different lexicons size.

For the text line extraction process, we presented an iterative steerable filters based method. For the optical models training, we observed the recogntion performance enhancement due to using the BLSTM-RNN optical models in comparison to weak performance of the HMM optical models. The linguistic knowledge representation was persented in form of lexicons and language models of tokens (words, syllables, multigrams and characters). We defines the effective coverage rate and the out-of-vocabulary rates measures to evaluate the lexicons and language models capacity to cover the test datasets.

By experiments, it was evident that the recognition performance increased by 1% approximately when using the language model that trained on the corpus of text which include in addition to its text lines, one long text line that represents the concatenation of whole corpus text lines. This can be explained by the ability of the language model to capture long distance token dependencies through the long concatenation text line. In the same context, the paragraph by paragraph decoding strategy improves the recognition performance by 2% approximately in comparison with the line by line decoding strategy. Two language modelling methods are examined, n-gram modelling approach for general use in all our experiments and RNN language modelling approach in a special case of character based recognition system.

Chapter 5

Handwriting recognition using sub-lexical units

5.1 Introduction

The significant progress observed these last years in the speech and handwriting recognition field came from the advancement of deep learning techniques and recurrent neural networks. But some open issues still remain which are more oriented towards a better modelling of lexicons and languages. Whereas the role of language models has been intensively studied in speech processing, very few studies have been devoted to Handwriting recognition. Moreover, whereas speech and handwriting share the same properties in terms of lexicons and languages, they exhibit very different low level properties when looking at the phonetic structure of speech on the one hand, and the graphemic structure of text on the other hand. Indeed, there is a discrepancy between the way of coding a text and the way of coding its oral pronunciation depending on the nature of the language. In chapter 3, we have shown the very differences between oral languages and written languages regarding their primary units (characters versus phonemes) and sub-lexical units (phonetic syllables versus graphemic syllables). The speech recognition literature includes many studies where sub-lexical units have been successfully introduced as a mean to cope with Out Of Vocabulary (OOV) words. In comparison, to our knowledge, such a strategy has not been studied in the field of handwriting recognition. We can highlight the very few studies devoted to handwritten Arabic text recognition using PAW sub-lexical units (BenZeghiba, Louradour, and Kermorvant, 2015) or handwritten German text recognition, because German language allows the composition of words with some smaller words in a way similar to agglutinative languages (although German does not belong to agglutinative languages). Thus, German lexicons should be of very large size in order to get high Effective Coverage Rates (ECR). Similar properties are observed for Italian language as well.

Introducing sub-lexical units in a handwriting recognition system appears an interesting, even necessary, strategy to overcome the problem of OOV. In this chapter we investigate the use of sub-lexical units such as syllables and multigrams for handwriting recognition. The experiments are conducted on the French RIMES dataset, and on the English IAM dataset. A first necessary step towards this goal is the design of some automatic lexicon decomposition

algorithms able to provide the desired sub-lexical units. We propose two decomposition algorithms. The first one is a supervised syllabification method that exploits a linguistic datasets of word decomposition into syllables made by experts to propose a decomposition into syllables for any given word using a similarity measures. This method has the drawback of being dependent of linguistic expertise which is available for very few languages and which does not cover all language vocabularies. The idea of the second decomposition method consists of using a Hidden Semi-Markovian Model (HSMM) which learns the regularities of character sequences called multigrams from a lexicon of words. N-multigram sub-lexical units (character sequences of at most N characters) are finally using the Viterbi algorithm which provides the optimal segmentation of the words of the training lexicon. The first approach has been presented in (Swaileh and Paquet, 2016c, Swaileh, Lerouge, and Paquet, 2016) while the second one will be presented in (Swaileh et al., 2017). This chapter is organized as follows. Section 5.2 presents a supervised sub-lexical decomposition approache into syllable, and section 5.3 is dedicated to unsupervised decomposition into multigrams. In section 5.4 we present the experimental results obtained using sub-lexical units for the recognition of French and English Handwritten texts. These models compare favorably against the standard word lexicon based approaches and the lexicon free approaches made of character n-gram language models.

5.2 A supervised word decomposition methods into syllables

The word decomposition into sub-lexical units (syllables or multigrams) is expressed in the literature by the syllabification process. Usually, the phonetic syllables generation relies on linguistic and phonetic rules which form the basis of the orthographic syllables. The orthographic syllables can be obtained from a database of syllables designed by language experts such as the French Lexique3 database (New et al., 2004) and The Free English language Hyphenation dictionary (EHD) (Hindson, 2016). The Lexique3 database provides the orthographic syllabic decomposition of a lexicon of almost $142,695$ French words into $9,522$ syllables only. It therefore constitutes a linguistic knowledge base from which our French syllabic model is developed. The free English language Hyphenation Dictionary contains $166,280$ words decomposed into $21,991$ syllables. Some examples of these two resources are provided on table 5.1.

However, despite their relatively large size, it quickly becomes out that these linguistic resources by far do not cover the French or English vocabularies. For example, Lexique3 covers only 69.83% of RIMES vocabulary. Similarly EDH covers only 54.42% of the IAM vocabulary (see left side of figure 6.2). Right side of figure 6.2 shows the effective coverage rate (ECR) of Lexique3 and of EHD on RIMES and IAM datasets respectively, when using the syllabic decomposition. Although insufficient, we can see the increased coverage when using a syllabic decomposition of the RIMES or IAM data set. Therefore, we

	Words	Syllables
Lexique3 examples	connaître	con naî tre
	adhésion	ad hé sion
	norvégiennes	nor vé gien nes
English Hyphenation Dectionary	abandonment	a ban don ment
	abiotrophy	ab i ot ro phy
	zymosthenic	zy mos then ic

TABLE 5.1: Some examples of the Lexique3 and the English Hyphenation Dictionary.

have to find a general way for generating a syllabic decomposition of any corpus and learning a syllable based language models on it. For this purpose, we developed an automatic syllabification method (Swaileh and Paquet, 2016b) and its source code is open source[1].

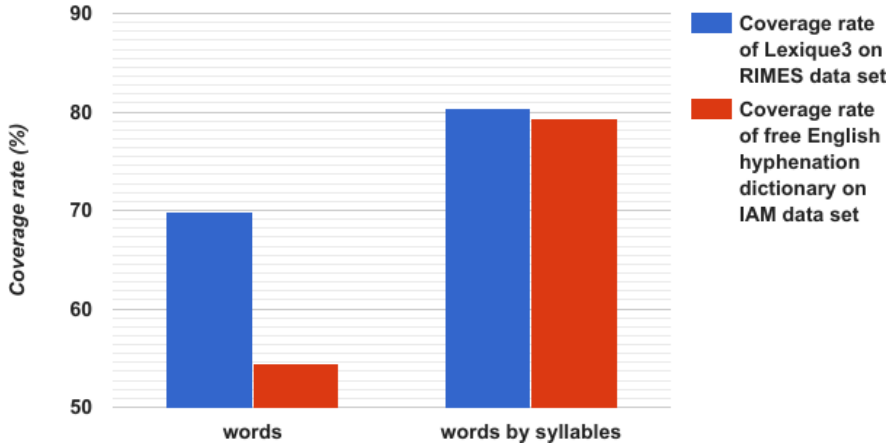


FIGURE 5.1: ECR (%) of Lexique3 on RIMES (blue) and EHD on IAM (red), for words (left) and syllables (right) decomposition.

The syllabification method exploits the lexical and phonetic similarities between the target word (to be decomposed into syllables) and the syllabified words that belong to the dictionary. The method that we propose is based on matching similar lexical and phonetic structures in order to provide a syllabic decomposition of any unknown word. Let us assume a word lexicon L, the n^{th} word entry being represented by its sequence of characters m_n associated with its syllabic decomposition denoted s_n, namely its sequence of syllables. Formally we can write $L = \{(m_1, s_1), (m_2, s_2), \ldots, (m_n, s_n), \ldots, (m_l, s_l)\}$. For any unknown word (not in L) represented by its character string m, we wish to determine its syllable representation s. The first idea is to search for the closest word m_n in the dictionary. But two very similar words may have different syllabic decomposition, especially if they differ by a vowel, which often marks the presence of a syllable. To take this information into account, we introduce an syllabic structure that represents the word by its consonants and vowels string. For example, the word "$Bonjour$" is encoded by its syllabic structure

[1]http://swaileh.github.io/Syllabifier

$ss = "CVCCVVC"$. Then we can define a similarity measure by combining both representations according to equation 5.1, where S_{lex} and S_{syl} are respectively two similarity measures on the lexical and the syllabic representations:

$$S_G\big((m, ss), (m_i, ss_i)\big) = \frac{S_{lex}(m, m_i) + S_{syl}(ss, ss_i)}{2} \tag{5.1}$$

Table 5.2 shows an illustration for the similarity measures obtained on one example. The similarity score between sequences of characters counts the average number of identical character pairs at the same positions between the two sequences. When sequences have different size, the shorter one is completed by space characters, so the two sequences have an identical size. By this way, we make the segmentation into syllables based on the prefix of the dictionary word, possible errors may occur at word's suffix that may be different due to of the completion of the words by spaces. When the lexicon entry that is most similar to the unknown word gets a similarity score exceeding a threshold T, its syllabic representation serves as a model to decompose the unknown word. The segmentation into syllables of the unknown word is made at the same positions as in the model.

Lexical similarity S_{lex} = 5/8								
Query word	*b*	*o*	*n*	*j*	*o*	*u*	*r*	-
Nearest lexicon entry	*t*	*o*	*u*	*j*	*o*	*u*	*r*	*s*
	-	*o*	-	*j*	*o*	*u*	*r*	-
Syllabic similarity S_{syl} = 6/8								
Query word	*C*	*V*	*C*	*C*	*V*	*V*	*C*	-
Nearest lexicon entry	*C*	*V*	*V*	*C*	*V*	*V*	*C*	*C*
	C	*V*	-	*C*	*V*	*V*	*C*	-
Syllable decomposition								
Query word	*b*	*o*	*n*	*j*	*o*	*u*	*r*	
Nearest lexicon entry	*t*	*o*	*u*	*j*	*o*	*u*	*r*	*s*

TABLE 5.2: Syllabification example on the query word ı*bonjour*ȷ

When the similarity score is below the threshold T, the decomposition into characters is chosen as the syllabic decomposition by default. Reasonable values of T can be chosen between $[0.5, 1]$. Between these two extreme values, the choice of an optimal value is crucial because when $T = 1$ the algorithm systematically will divide the unknown word into characters, which is not the goal. Conversely, when $T = 0.5$, the algorithm will systematically propose a syllabification even if the word query has a structure different from the closest word belonging to the lexicon. In this case there is a big chance to produce an erroneous syllabification. Such an example is the entity names which does not belong to the French or English spoken language. This margin of syllabification by mistake must be quantified by an linguist expert. It is therefore necessary to optimize the value of T to find the one that minimizes both the number of words decomposed into characters and the number of erroneous syllabification. A first possibility would be to optimize T by cross-validation A second approach consists in optimizing T with respect to the performance

of the recognition system. Figure 5.2 shows the evolution of the Word Error Rate of the recognition system on the two RIMES and IAM datasets. We notice a rather similar behaviour of the two systems with a turning point at certain value of T beyond which the recognition system loses its lexicon coverage power and simultaneously achieves lower recognition performance.

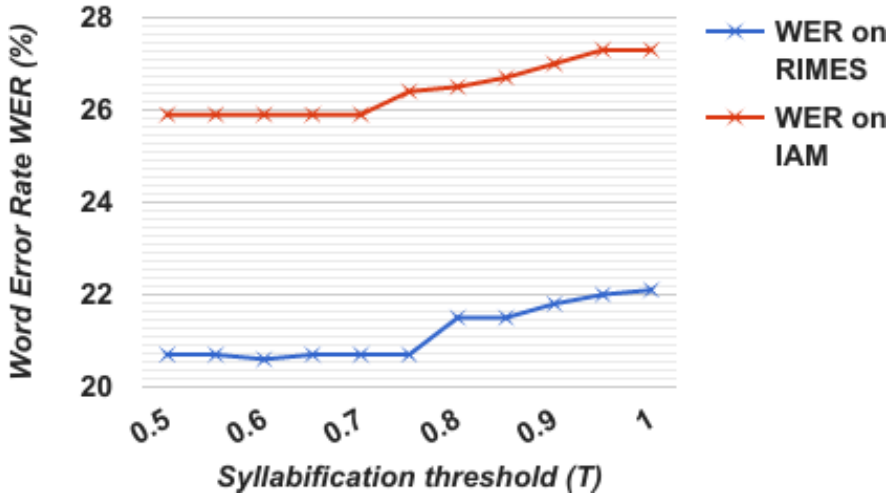


FIGURE 5.2: WER (%) as a function of T on the RIMES dataset (blue) and the IAM dataset (red), for an HMM based recognition system with HoG features

We chose $T = 0.6$ for the RIMES dataset, and $T = 0.7$ for the IAM dataset. In this situation only 0.25% of the words in the RIMES dataset are not syllabified. A close look at these errors shows that they correspond to out of French vocabulary words, as for example the sequence "$SXNHBOO$". The same phenomenon is observed on the IAM dataset, for example the word "$Lollobrigida$" is broken down into characters. by setting $T = 0.7$, only 0.37% of the words in the IAM dataset are not syllabified, and decomposed into characters by error. A complementary analysis of the supervised syllabification method is illustrated in figure 5.3 and figure 5.4. They represent the histograms of the number of words decomposed into n syllables for the RIMES and IAM lexicons, as a function of the threshold T. It can be noted that the majority of words are decomposed into 5 syllables at most on the RIMES dataset when $T = 0.6$. It is also noted that, beyond 0.6, the number of words abnormally decomposed into a large number of syllables increases, because the method favors the decomposition of words into characters beyond this threshold value. The same observations can be made on the IAM dataset from $T = 0.7$ (figure 5.5).

5.3 An unsupervised word decomposition into multigrams

During chapter 3 we have seen that multigrams are variable length sequences of elementary units that compose an observation stream, as opposed to fixed length n-gram (Deligne and Bimbot, 1995). They have been introduced first for

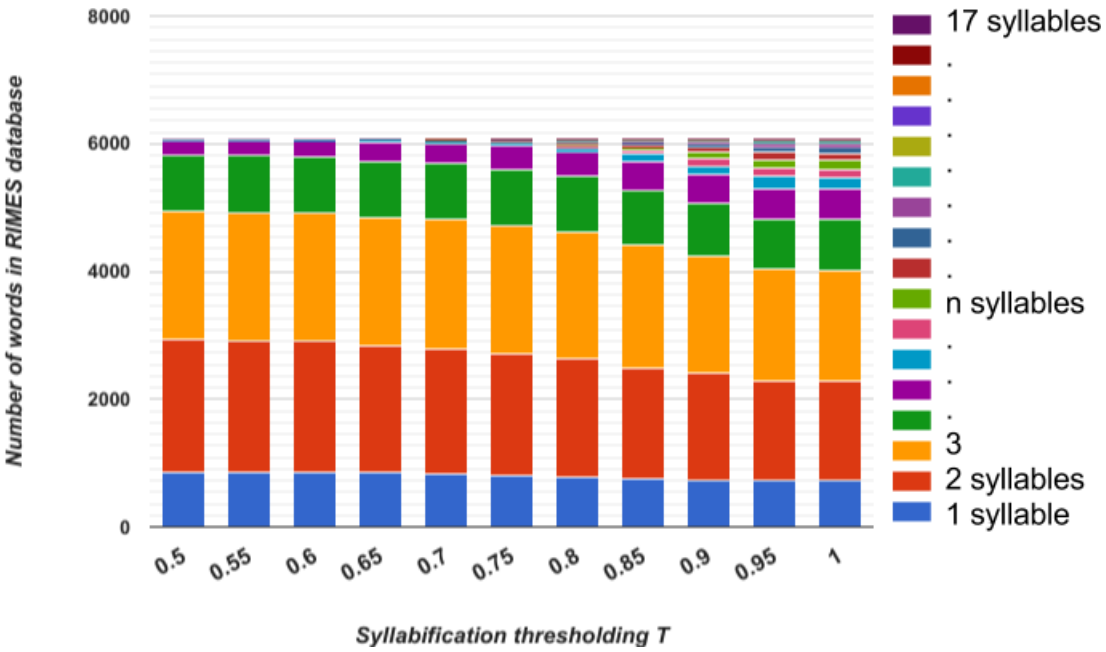


FIGURE 5.3: Number of words of the RIMES vocabulary decomposed into n syllables as a function of T.

language modelling of phrases, but later the same authors have studied their possible introduction as sub-lexical units for language modelling and speech recognition (Deligne and Bimbot, 1997). If one consider the character to be the primary units of a written language, then multigrams allow representing words as the concatenation of variable length sequences of characters. To our knowledge this study is the first application of multigram models to handwriting recognition. In the original paper the authors have proposed an unsupervised training approach based on variable length Hidden Markov Models. In the next paragraphs, based on Hidden semi Markov Models, we introduce the formal definitions underlying the multigram paradigm.

5.3.1 Hidden Semi-Markov Model General definition

Formally, Hidden Semi-Markov Models (HSMM) are considered as one extension of the traditional Hidden Markov Models (HMM) where the duration of each state is explicitly defined. Thus each state in the HSMM model can generate an observation sequence of duration d. Assuming a discrete-time Markov model with a set of M Hidden states $S = \{s_1, s_2, \ldots, s_M\}$, a state sequence of length T is denoted $Q_{1:T} = Q_1\ Q_2 \ldots Q_t \ldots Q_T$, in which Q_t is the state at time t. $Q_{[t1,t2]} = s_m$ accounts for state s_m starting at time t_1 and ending at time t_2 with a duration $d = t2 - t1 + 1$. Let $O_{1:T} = O_1\ O_2\ \ldots\ O_t\ \ldots\ O_T$ denotes the observation sequence of length T in which O_t is the observation at time t. The underlying unobservable (Hidden) state sequence that can generate the observation sequence $O_{1:T}$ can be any state segmentation path writing $Q_{[1:d1]}Q_{[d1+1:d1+d2]} \ldots Q_{[d1+\ldots+dn+1:T]}$ with $\sum_l d_l = T$.

One generally assumes that hidden state sequences with their respective duration are generated by a first order Markov source (Yu, 2010). Then the

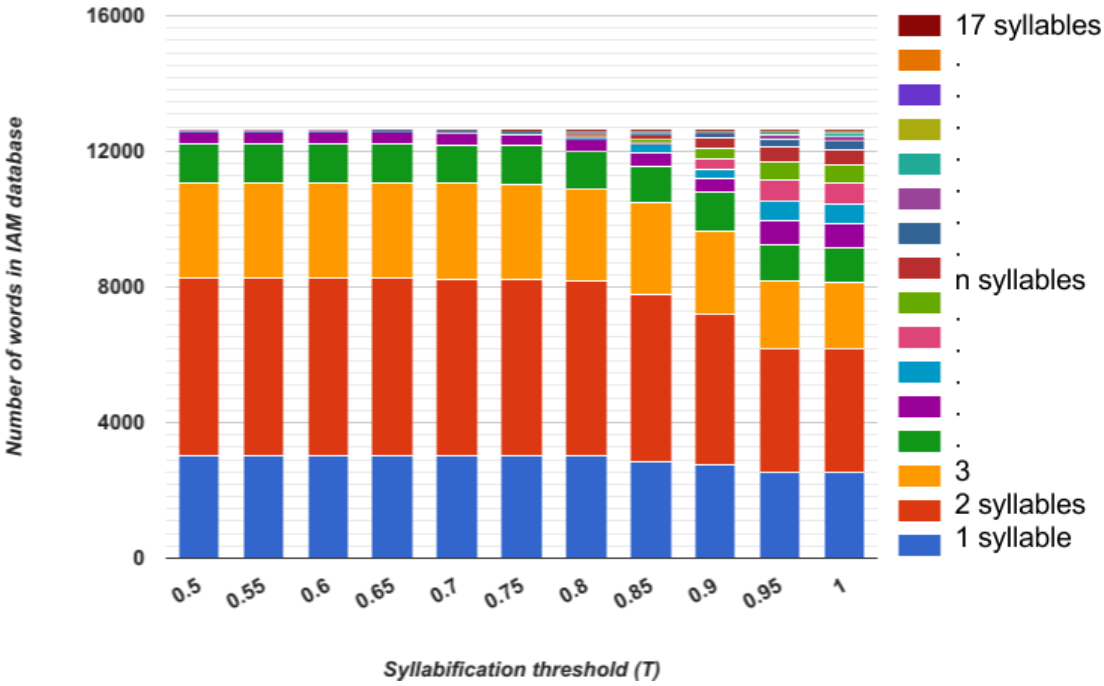


FIGURE 5.4: Number of words of the IAM vocabulary decomposed into n syllables as a function of T.

transition probability from one state to another writes:

$$a_{(i,d)(j,d')} = P(Q_{[t-d'+1,t]} = j | Q_{[t-d'-d+1,t-d']} = i) \tag{5.2}$$

with the following two properties :

$$\sum_{j \neq i} \sum_{d'} a_{(i,d)(j,d')} = 1 \tag{5.3}$$

$$a_{(i,d)(i,d')} = 0 \tag{5.4}$$

From this definition, we can see that state and duration are dependent on the previous state and duration. Then, beeing in state i with duration d, the model generates the observation sequence $O_{[t-d+1,t]}$ with the following emission probability, assuming time independence of the observations:

$$b_{(i,d)} = P(O_{[t-d+1,t]} | Q_{[t-d+1,t]} = i) \tag{5.5}$$

The definition of the initial state distribution which represents the probability of the initial state and its duration writes:

$$\pi_{(j,d)} = P(Q_{[1:d]} = j) \tag{5.6}$$

Hidden semi Markov Models are sometimes named segmental models, as

the model explicitly accounts for state duration, thus it accounts for the segmentation of the observation stream into consecutive segments of variable duration (or variable length). Figure 5.5 illustrates the general structure of the underlying segmentation of the observation sequence by a HSMM. Looking for

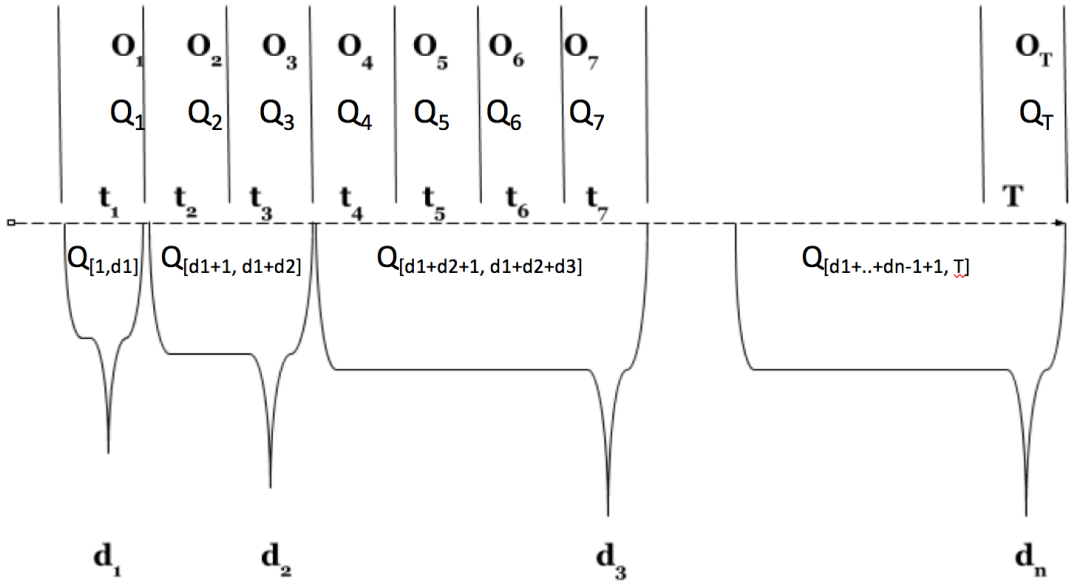


FIGURE 5.5: General structure of a Hidden semi-Markov model.

the best segmentation of the observation stream is achieved through a Viterbi decoding process that search for the state sequence Q^* that produces the highest data likelihood.

5.3.2 Generating multigrams with HSMM

In the literature (Deligne and Bimbot, 1995), multigrams have been first introduced using a generative memoryless source emitting variable length sequences of symbols or words. The memoryless source hypothesis is equivalent to assuming a zero order Markov source, while variable length observations typically fall into the Hidden Semi Markov paradigm. With these hypotheses, the data likelihood with respect to the model can be expressed by the following equation:

$$P(O,Q) = \prod_{l=1}^{n} P(O_{[t-d_l+1,t]}|Q_{[t-d_l+1,t]})P(Q_{[t-d_l+1,t]})$$

where d_l is the duration of multigram l in the sequence, and Q_l stands for the state of the hidden underlying segmentation process.

A multigram being a specific sequence of characters of length d, modelling the segmentation of a character string into multigrams can be accomplished by introducing hidden states, each one accounting for the emission of sub units of length d. In other words, each state of the model s_m is responsible for generating every multigram of length m. Such segmental model induces a $0/1$

duration probability of the hidden states i.e. $P(Q_{[t-d_l+1,t]} = s_m) = \delta(m, d_l)$. As a consequence the model also simplifies since the equality $s(m) = d$ holds for every value of d, which lead to the following equations:

$$P(O, Q) = \prod_l P(O_{[t-d_l+1,t]}|Q_{[t-d_l+1,t]})\delta(length(s_l), d_l)$$

which finally writes:

$$P(O, Q) = \prod_l P(O_{[t-d_l+1,t]}|d_l) \tag{5.7}$$

We see that variable length sequences of characters (multigram) can be represented by a zero order Hidden Discrete Semi Markov Source, with as much hidden states as the maximum duration of a multigram considered. Training a HSMM from textual data will allow to optimize a generative model to produce the observed sequences of character according to their underlying multigram structure, thus allowing to get an optimal multigram sublexical representation of words, as we will see in the next paragraph.

5.3.3 Learning multigrams from data

The estimation of the HSMM parameters would normally require knowing the segmentation of the observed character strings into their multigram decomposition. Unfortunately this is exactly what we expect to determine from the observation of the raw textual data. Therefore learning multigrams from data has to be performed considering segmentation is missing. This problem has received much attention in the literature and has been solved in an elegant manner by the Expectation Maximization (EM) algorithm (Deligne and Bimbot, 1997). This algorithm iterates the expectation step (E) during which statistics about the missing data are gathered, followed by the Maximization step (M) that estimates the parameters of the model according to the likelihood maximization principle. In a way which is very similar to the estimation of HMM parameters, the following Forward and Backward variables are first computed using the current model parameters λ following the above definition:

$$\alpha_t(j, d) = \alpha_t(d) \triangleq P(Q_{[t-d+1:t]} = d, O_{[1:t]}|\lambda)$$
$$\beta_t(j, d) = \beta_t(d) \triangleq P(O_{[t+1:T]}|Q_{[t-d+1:t]} = d, \lambda)$$

Assuming the zero order hypothesis the state transition probabilities can be ignored, and the following recurrences hold and allow efficient computation

of the two variables.

$$\alpha_t(d) = \sum_{d'} \alpha_{t-d}(d') b_d(O_{[t-d+1:t]}) \tag{5.8}$$

$$\beta_t(d) = \sum_{d'} b_{d'}(O_{[t+1:t+d']}) \beta_{t+d'}(d') \tag{5.9}$$

Once the Forward and Backward variables are determined, the following statistics are computed as follows:

$$\eta_t(d) \triangleq P(Q_{[t-d+1:t]} = d, O_{[1:T]}|\lambda) = \alpha_t(d)\beta_t(d) \tag{5.10}$$

$$\xi_t(d', d) \triangleq P(Q_t = d', Q_{t+1} = d, O_{1:T}|\lambda)$$

$$\xi_t(d', d) = \alpha_t(d') b_d(O_{[t+1:t+d]}) \beta_{t+d}(d) \tag{5.11}$$

$$\gamma_t(d) \triangleq P(Q_t = d, O_{1:T}|\lambda) = \sum_t \eta_\tau(d)$$

$$P(O_{1:T}) = \sum_d \eta_t(d) = \sum_d \alpha_T(d) \tag{5.12}$$

From these statistics, a new multigram HSMM model λ can be estimated during the maximization step using the following formulas which are the special multigram case of the general form of HSMM reestimation formulas Yu, 2010:

The initial transition probabilities estimation is:

$$\hat{\pi}_d = \frac{\eta_d(d)}{\sum_j \eta_j(j)} \qquad d = 1 \ldots d_{max} \tag{5.13}$$

The observation probabilities estimation is:

$$\hat{b}_d(v_{d,j}) = \frac{\sum_t \eta_t(d)\delta(O_{[t:t+d-1]}, v_{d,j})}{\sum_{t,i} \eta_t(d)\delta(O_{[t:t+d-1]}, v_{d,i})} \tag{5.14}$$

where $v_{d,j}$ is the j^{th} multigram of length d.

Finally, the global EM training algorithm of the model is depicted as follows considering one single observation sequence:

- Start with an initial model $\hat{\lambda} = \lambda_0$
- E step : Compute the auxiliary variables $\alpha_t(d)$, $\beta_t(d)$, $\eta_t(j, d)$ and $\xi_t(i, j)$
- M step : Estimate the new model $\hat{\lambda} = \{\hat{\pi}_d, \hat{b}_d(v_{d,j})\}$
- goto step E while $P(O_{1:T})$ increases

Training a HSMM with the EM algorithm does not require any segmentation information of the input stream into multigrams as the algorithm takes

account every segmentation path in the Forward Backward variables. The algorithm produces the optimal model $\hat{\lambda}$ considering any possible sequence of character (multigram) up to the length d_{max}. However, a certain amount of these multigrams have very low probability, while some others do not occur in the training data. The final step in training multigrams is to select the set of the most important ones. This is done by introducing a segmentation criteria so as to select the multigrams that take part in a segmentation path when segmenting the training data. We look for the most probable segmentation path Q^* for each training sample defined by:

$$Q^* = \underset{Q}{argmax} \quad P(O_{1:T}, Q_{1:T}) \tag{5.15}$$

This problem is solved with the well known Viterbi algorithm by introducing the partial optimal path up to time t with the probability $\delta_t(d) = P(O_{[1:t]}, Q_{[1:t]})$. Since this quantity is monotonously decreasing, the following recursion holds:

$$\delta_t(d) = \underset{d'}{max} \quad \delta_{t-d}(d')b(O_{[t-d+1:t]}) \tag{5.16}$$

The best path is stored using the auxiliary variable $\varphi_t(d)$ according to the following equation:

$$\varphi_t(d) = \underset{d'}{argmax} \quad \delta_{t-d}(d')b(O_{[t-d+1:t]}) \tag{5.17}$$

By repeating this optimal segmentation procedure for each training sample we can detect the most frequent and necessary multigrams that serve in the optimal decomposition of the training corpus. However, doing so, each multigram contributes with a weight proportional to its likelihood $b(O_{[t\breve{}d+1:t}$ to the score of the path, which tends to favour short multigrams and penalize longer ones that have relatively lower likelihood, as the number of potential multigrams increases exponentially with their length d. To remedy this phenomenon we introduce a penalizing factor so as to favor multigram paths with Minimum Description Length (MDL) over longer descriptions. This penalizing factor was chosen to be the inverse multigram length exponent, so as to balance the exponential decay of multigram likelihood. Finally the optimal decoding of multigrams obey the following equation:

$$\delta_t(d) = \underset{d'}{max} \quad \delta_{t-d}(d')b(O_{[t-d+1:t]})^{1/d} \tag{5.18}$$

Table 5.3 shows different examples of word decompositions for French and English words using the supervised syllabification method which produce the syllables in comparison with the multigram generation method which produce multigrams of two to five different orders. The order of the multigrame indicates the maximum number of the characters in a multigram denoted by 2-multigram for the second order multigram and so on for 3-mulitgram, 4-multigram and 5-multigram.

Table 5.4 shows the size of the lexicons for each type of sub-lexical unit, and allows comparison with the word lexicon size.

	words	syllables	2-multigram	3-multigram	4-multigram	5-multigram
FR	*Merci*	*Mer ci*	*Me r ci*	*Mer ci*	*Merci*	*Merci*
	indiqué	*in di qué*	*in di qu é*	*ind iqu é*	*ind iqué*	*ind iqué*
	Gambetta	*Gam bet ta*	*Ga mb et ta*	*Ga mb ett a*	*Gamb etta*	*Gam betta*
EN	*darling*	*dar ling*	*d ar li ng*	*dar l ing*	*dar ling*	*dar ling*
	Incredible	*in cre di ble*	*in cr ed ib le*	*in cre di ble*	*in cred ible*	*in credi ble*
	Geoffrey	*Geof frey*	*Ge of fr ey*	*Ge off rey*	*Geof frey*	*Geo ffrey*

TABLE 5.3: Syllabification results for French and English words using the supervised syllabification method and the multigram generation method

	words	syllables	2-multigram	3-multigram	4-multigram	5-multigram
RIMES training	5.5k	2.9k	1.1k	3k	4.4k	5.2k
IAM training	7.4k	4.6k	0.8k	2.9k	5.6k	6.9k

TABLE 5.4: Statistics of the various sub-lexical decompositions of the RIMES and IMA vocabularies.

As a conclusion, in this section we have proposed two approach dedicated to decompose a given lexicon into sub-lexical units. The supervised approach relies on the linguistic expertise to decompose words into their syllabic representation, it requires having such expertise beforehand for each language. Multigram sub-lexical units have the advantage that they are determined without the need for external expertise, by using a data driven optimization training algorithm only. The impact of these two approaches on a recognition system will be examined in the next section.

5.4 Recognition systems evaluation

In this section, experimental results are presented with the BLSTM-RNN optical model. This choice was made by considering the significant difference between the HMM based optical model and the BLSTM-RNN. From these observations is was clear that only the experimentations made using the RNN optical model can be compared to the state of the art reported in the literature. In the following sections, we report on the performance of the system obtained using sub-lexical units, and compare them with closed vocabulary systems made of word language models. Additionally, we also had a look at the performance of character language model.

5.4.1 Experimental protocol

The BLSTM-CTC system was evaluated on the French RIMES and the English IAM datasets with small and large vocabulary language models of multigrams, and with using closed and open vocabulary configurations. We compare multigrams models with language models of words or characters. All

recognition tasks were achieved on paragraph level to include more context in the language model as demonstrated in chapter figure 4.20.

A- Handwritten datasets

From the RIMES dataset, we randomly selected $1,333$ text lines from 882 paragraphs of the training set for the system validation. the RIMES dataset training, validation and test partitions are illustrated in table 5.5.

Database	Set	#Pages	#Lines	#of vocabulary words
RIMES	Train	*1350*	*9947*	*8.11k*
	Valid	*150*	*1333*	*2.35k*
	Test	*100*	*778*	*5.6k*

TABLE 5.5: RIMES dataset partitions

The IAM dataset (Marti and Bunke, 2002) has been used by different researchers with different setup. For our experiments we used the same subsets as in (Bluche, 2015) and table 5.6 which is composed of a training set of $6,482$ lines, a validation set of 976 line, and a test set of $2,915$ lines.

Database	Set	#Pages	#Lines	#of vocabulary words
IAM	Train	*747*	*6,482*	*9.95k*
	Valid	*116*	*976*	*2.99k*
	Test	*336*	*2,915*	*6.2k*

TABLE 5.6: IAM dataset partitions (following Bluche, 2015)

5.4.2 B- Language models datasets and training

Regarding the French language resources, the tokenized RIMES training corpus vocabulary which contains $5.4k$ word tokens was used for training a closed vocabulary system with small lexicon. From the tokenized French Wikipedia corpus, we selected the $50k$ most frequent words before tokenization, which generated a large lexicon of $29k$ word tokens achieving an effective coverage rate 82.71%. We used this lexicon ($29k$ word token) for training two groups of large vocabulary language models : a closed vocabulary model by using the word tokens and open vocabulary model by using multigrams, syllables and characters.

Regarding the English language resources the IAM training dataset annotations are used for training a small vocabulary language model of words with a vocabulary of $7.3k$ word tokens. For training the large vocabulary language models, we selected the vocabularies of the combination of LOB (excluding the sentences that contain lines of the IAM test and validation sets), Brown and Wellington corpora which makes vocabulary of $86.5k$ words after tokenization.

During all the experimentations, the multigram segmentation algorithm defined above was used to generate the training corpus of multigrams from the tokenized corpus of text. Multigrams vocabularies are derived from their corresponding small and large vocabularies of words that are used to train respectively small and large vocabularies language models (LM) of words. The modified Kneser-Ney discounting method was used. We denote a multigram language models by $m-K$-multigram where $K \in 2, 3, 4, 5$ represents the maximum number of characters per multigram, and where m is the language model order.

As illustrated on figure 5.6, best recognition performance are achieved with 9-gram language models (higher n-gram order) of words and multigrams and by 10-gram language model of characters. As expected, the figure also shows that low order language models perform better with words and long multigrams (5-multigram), whereas 2-multigram are the worst model in this case. We see that as the language model order increases, the performance of the various models come closer to each other. The models perform almost similarly for orders 7 and higher. For all other experimentations, we chose to use 9-gram language models however, as the Kneser-Ney discounting method prevents from over-training the models. These results have been obtained on the RIMES dataset with the language model trained on the RIMES training dataset only (lexicon = $5,4K$ word tokens). The optical model was a BLSTM model with HoG features for this experiment.

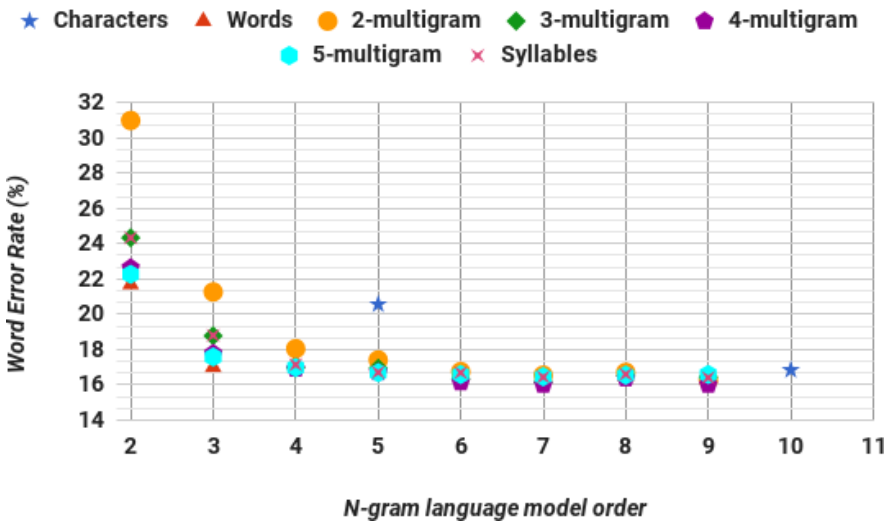


FIGURE 5.6: Performance obtained on RIMES as a function of the LM order and for various LM using Hidden Markov Models

As a prior experiment, we evaluated the contribution of the optical model only to the recognition task. Table 5.7 below shows the Character Error Rate (CER %) and the Word Error Rate (WER %) achieved by the optical model on the two datasets.

Test datasets	RIMES optical model		IAM optical model	
	CER	WER	CER	WER
RIMES	12.32	38.88	–	–
IAM	–	–	18.91	52.13

TABLE 5.7: BLSTM optical models performance only on the RIMES and on the IAM dataset.

5.4.3 Recognition performance

A- Using the RIMES and IAM resources only

A first series of experiment was conducted using the resources provided by the RIMES or IAM dataset only. These resources are used for training the optical models and the language models as well. Table 5.8 provides the performance of the various models, closed vocabulary word models and open vocabulary sub-lexical unit based models. We can see that all these models perform almost equally well with a small improvement with using sub-lexical units language models. Even a simple character language model is better that a word model. The syllable model seems to perform similarly to a multigram (MG) model.

Training data sets	Measure	French language models						
		words	5-MG	4-MG	3-MG	2-MG	characters	syllabes
RIMES training dataset	WER%	12.01	11.50	**11.12**	11.44	11.44	11.51	11.38
	CER%	6.43	6.43	5.79	5.90	5.87	5.63	5.82
	OOV%	3.10	1.61	1.16	0.53	0.16	0	0.63
	ECR%	93.29	97.43	98.01	98.84	98.98	100	98.6
	Lex	5.5k	5.2k	4.4k	3k	1.1k	100	2.9k

TABLE 5.8: Recognition performance of the various open and closed vocabulary systems trained on the RIMES resources only, (MG : multigram)

The same experiment was conducted on the IAM dataset. In table 5.9 below we can see that there is a small increase by using sub-lexical units compared word models. Sub-lexical units are to be preferred to the simple character models. Compared to the performance obtained on the RIMES dataset we see a large difference on WER and CER which can be explained by the low ECR of the training corpus on the test corpus. In other words, the linguistic training resource provided by the IAM training dataset in not sufficient to match the targeted test corpus. Some additional more general resources are needed. This was the motivation for conducting the experiments that we present in the next sub-section **D** below.

B- Using additional linguistic resources

This series of experiments was conducted to assess the capacity of sub-lexical units based language models when trained or more general resources. Notice that this was a necessity for the IAM dataset, as the training IAM dataset

Training data sets	Measure	English language models						
		words	5-MG	4-MG	3-MG	2-MG	characters	syllabes
IAM training dataset	WER%	31.88	29.13	28.8	28.59	**28.18**	30.6	28.63
	CER%	15.21	13.09	12.86	12.71	12.54	19.49	12.60
	OOV%	14.1	10.95	7.3	2.2	0.22	0	5.01
	ECR%	75.45	84.18	88.59	92.97	99.42	100	91.1
	Lex	7.4k	6.9k	5.6k	2.9k	0.8k	79	4.6

TABLE 5.9: Recognition performance of the various open and closed vocabulary systems trained on the IAM resources only, (MG : mulitgram)

achieves a moderate ECR on the test set of only 75.45%. On the contrary, by using one large external resource for training a French language model, we will analyse the generalization capability of this resource, and reversely we will have an estimation of the specificity of the RIMEs dataset in terms of language coverage.

Training data sets	Measure	French language models					
		words	5-MG	4-MG	3-MG	2-MG	characters
Wikipedia+ RIMES training dataset (a)	WER%	18.68	13.88	12.29	12.37	**10.78**	13.47
	CER%	9.49	6.82	6.33	6.17	5.90	7.07
	OOV%	8.3	4.1	2.3	1.4	0.22	0
	ECR%	82.71	93.73	96.37	97.32	99.48	0
	Lex	29k	19.3k	13.8k	6.3k	1.4k	100
Wikipedia dataset (b)	WER%	20.45	16.22	15.94	15.15	**15.11**	20.82
	CER%	10.62	8.30	9.10	8.33	9.25	10.10
	OOV%	8.3	4.1	2.3	1.4	0.22	0
	ECR%	82.71	93.73	96.37	97.32	99.48	0
	Lex	29k	19.3k	13.8k	6.3k	1.4k	100

TABLE 5.10: RIMES recognition performance of the various open and closed vocabulary systems trained on the French extended linguistic resources made of RIME train and wikipedia, (MG : mulitgram)

Table 5.10 shows that by augmenting the RIMES training corpus by the French Wikipedia allows a small decrease of the WER (1%) compared to training the language models on the RIMES corpus alone. The 2-multigram model performs a little much better than other sub-lexical units based language models. But when the language model are trained on the French Wikipedia only, although the models achieve similar ECR than the previous models, the recognition performance decrease by 5% at least for the best model, which again is a 2-multigram. This highlights the fact that 2-multigram is a good sub-lexical language model, but that ECR is only one indicator for having good language models, as two training resources may share the same lexicon, but have different language models, as the words or sub-lexical units do not appear in the same context. How to get better training corpus which should not only have high ECR but also provides similar contexts as the test corpus? This appears

to remain an open issue right now, or at least to be solved through trials and errors.

Regarding the English IAM dataset, we observe similar properties as for the RIMES dataset, but here there is a clear benefit of extending the language model training corpus by the LOB + Brown + Willington corpora. Indeed, in table 5.11 below, we see that adding theses resources allows reaching a 94.8% ECR, whereas we had only 75% using the IAM training corpus alone. This large improvement of the coverage rate has a clear effect on the recognition rate where we get a nearly 14% WER which was initially around 30%, a relative improvement of more than 50%. This improvement is observed for every language models either lexicon based or sub-lexicon units based language models, except the character language model. Multi-gram language models perform almost similarly for every of these experiments with high ECR rates. However we should highlight the fact that short multigram models based on $1.1k$ multigrams perform almost similarly than word models that require large vocabulary of $87.5k$ words dictionary (when looking at full size dictionaries). It is worth noticing the significant complexity reduction of the multi-gram language model compared to the equivalent word language model.

Training data sets	Measure	English language models					
		words	5-MG	4-MG	3-MG	2-MG	characters
LOB+BROWN +Wellington All vocabulary (a)	WER%	13.66	**13.47**	14.41	14.1	14.93	17.62
	CER%	7.63	7.40	7.96	7.67	8	9.09
	OOV%	1.4	0.82	0.33	0.06	0	0
	ECR%	94.82	98.35	99.46	99.55	99.99	100
	Lex	87.5k	52.7k	37.5k	10.7k	1.7k	79
LOB+BROWN+ Wellington 50k most frequent tokens (b)	WER%	14.57	14.78	14.26	**14.18**	14.89	17.62
	CER%	8.02	7.92	7.78	7.72	8.0	9.09
	OOV%	2.06	3.35	1.32	0.18	0	0
	ECR%	92.50	96.35	98.59	99.11	99.99	100
	Lex	50k	30.4k	20.2k	7.7k	1.3k	79
LOB+BROWN +Wellington 50k most frequent words (c)	WER%	16.06	14.98	14.78	**14.33**	14.75	17.62
	CER%	8.73	8.02	8	7.78	8	9.09
	OOV%	3.7	3.7	1.95	0.33	0.02	0
	ECR%	91.5	94.58	96.96	98.51	99.95	100
	Lex	29.7k	19.7k	13.7k	5.7k	1.1k	79

TABLE 5.11: IAM recognition performance of the various open and closed vocabulary systems trained on the English extended linguistic resources made of IAM train and LOB + Brown + Wellington corpora, (MG : mulitgram)

As a conclusion, we can highlight the strength of sub-lexical units such as multigram compared to the lexical units. They have the advantage to tackle the OOV problem while operating with reduced lexicons of sub-units. Our recognition system has a simple architecture than state of the art systems presented in table 5.12 (the state of art system defined in (Voigtlaender, Doetsch, and Ney, 2016) is based on combination of word and character language models), and achieves equivalent state of the art performance with a highly reduced lexicon size.

System	RIMES-WER%	IAM-WER%
Our system	10.78 (by m2gram)	13.47 (by m5gram)
Voigtlaender, Doetsch, and Ney, 2016	9.6	9.3
Bluche et al.Bluche, 2015	11.8	11.9
Doetsch et alDoetsch, Kozielski, and Ney, 2014	12.9	12.9
Pham et al.Pham et al., 2014	12.3	13.6
Zamora-Martínez et al.	—	16.1

TABLE 5.12: Performance comparison of the proposed system with results reported by other studies on the RIMES & IAM test datasets.

5.5 Conclusion

In this chapter, a set of sub-lexical units based language models recognition approaches have been investigated in order to deal with the challenges related to unconstrained continuous handwriting recognition. Experiments have been conducted on the French RIMES and the English IAM handwriting datasets. Different types of lexicons and language models have been investigated like words, syllables, multigrams and characters language models. A novel approach was proposed for word decomposition into sub-lexical units such into syllables and multigrams. To generate the syllabic model we stand on the Lexique3 dataset that provides an orthographic syllable modelling for the French language, and on the free English Hyphenation Dictionary (EDH), which proposes the orthographic modelling of English words by syllables decomposition. By training a Hidden semi-Markovian Model (HSMM) on a large lexicon of words of the language of interest (French or English), we were able to train a model that captures variable length dependencies between characters in an unsupervised way. For decomposing any word into a sequence of multigrams, it is sufficient to decode the the words' character sequence with the HSMM using Viterbi decoding algorithm. The sub-lexical (syllable and multigrams) based language models offer many advantages over character models and word models. The advantages of these models are twofold. On the one hand, it is of limited complexity, since it works with a reduced lexicon of syllables. It follows an n-gram model of syllables which is itself more compact, so better parametrized, and therefore easier to optimize. On the other hand it offers superior performance than a lexical model when working with out of vocabulary words, while achieving similar performance when trained on corpus with high ECR on the test corpus.

Chapter 6

A Multi-lingual system based on sub-lexical units

6.1 Introduction

For a few decades, most handwriting recognition systems were designed of recognizing each language individually, even if they share the same set characters such as French and English, or Arabic and Persian. Indeed, Languages of the same origins often share their character sets and/or glyph shapes. For example the Latin-root languages share at least 21 characters (Diringer, 1951), and Arabic, Persian and 12 other languages share at least 28 characters (Märgner and El Abed, 2012). Inspired from this fact, almost all multilingual or multi-script recognition systems are designed to work with a unified character set (Kessentini, Paquet, and Benhamadou, 2008,Moysset et al., 2014,Kozielski et al., 2014b,Lee and Kim, 1997).

In this chapter we introduce sub-lexical unified models for French and English handwriting recognition as demonstrated in chapter 5. For one single language a sub-lexical model exhibit a limited complexity in comparison to a word model, while recognition performance decrease very slightly. This property allows combining multiple languages in a unified sub-lexical model while maintaining an acceptable complexity in terms of lexicon size and statistical n-gram language models, maintaining equivalent recognition performance and even enhanced performance in some cases. We also show that unification of similar scripts in a single recognition system leads to better trained optical character models, due to the benefit of sharing training datasets between the languages, thus providing better performance of the unified system. This is observed for any kind of model used (characters, syllables, multigrams, words) on the experiments carried on the RIMES and IAM datasets.

This chapter has the following organisation: section 6.2 presents a literature review about the different approaches used for developing multilingual recognition systems and related previous work. Section 6.3 describes the character sets, the optical models, the lexicons and the language models unification methodologies that we have applied for realizing the unified French and English system. Section 6.4 discusses the evaluation protocols and the experimental results obtained by the unified recognition system using the specialised and unified optical models with the association of language models of words, syllables, multigrams and characters. Section 6.5 draws the perspectives of the unified sub-lexical models within the conclusion.

6.2 Brief literature overview

In the literature there are different possible approaches for developing multilingual recognition systems that can be classified into two categories: selective approaches and unified approaches.

6.2.1 Selective approach

The selective approach category includes two approaches. The first approach applies different individual recognition systems, in a competitive way, for the same samples and selects the one which provides the best confidence score. The second approach first detects the language of the samples before selecting an appropriate recognizer for the target language script detected (Lee and Kim, 1997). The comparison of the scores obtained by several recognizers, which have different error ranges, and the pre-detection of the language, which is a complex task, represent the major problems of these approaches. By exploiting the multilingual MAURDOR dataset, (Moysset et al., 2014) proposed an English, French and Arabic multilingual recognition system and (Kozielski et al., 2014a) proposed and English, French multilingual recognition system, which belong to this category.

6.2.2 Unified approach

The unified approach category uses a single system for recognizing any language, possibly using multiple scripts. Several multilingual or multiscript recognition systems are proposed in the literature. In (Kessentini, Paquet, and Benhamadou, 2008) the authors propose a multilingual system for Arabic and Latin handwriting recognition. In (Malaviya, Leja, and Peters, 1996) the authors propose a handwriting recognition system for intermixed language scripts such as Latin, Devanagari, and Kanji. A unified network-based handwriting recognition system for Hangul and English language was proposed by (Lee and Kim, 1997). These approaches have the advantage of having a single system for all recognition tasks, avoiding the problems envisaged in the selective approaches. The disadvantage of the unified approaches is the proportional increase of the system's complexity regarding to the number of languages due to their direct effect on lexicon size, which affects the overall system performance.

Almost all the proposed unified approaches are such that the unified multilingual system have lower performance in comparison to the specialised monolingual systems. This may be explained by the effect of the lexicon and the fast expansion of the language model, due to the absence of alphabetical and/or lexical shared units between the considered languages and scripts.

6.2.3 Multilingual with neural networks

Ghoshal et al. (Ghoshal, Swietojanski, and Renals, 2013) have been investigated multilingual modelling using a hybrid system of Deep Neural Network (DNN) and Hidden Markov Model (HMM) for achieving speech recognition task. They carried out the experiment on the GlobalPhone corpus using seven languages from three different language families: Germanic, Romance, and Slavic. This work focused on the multilingual acoustic modeling, and made use of the pronunciation dictionary and language model of GlobalPhone (ready for use). This work concluded that the deep neural networks perform a cascade of feature extraction, from lower-level to higher-level features, provides an explanation for the promised enhancement that comes from the training the hidden layers using data from multiple languages.

6.3 Multilingual unified recognition systems

The previous literature review has given us the motivation for designing a unified system, which has the great advantage to be easy to implement because, only one single system must be designed and optimized. By combining the strength of BLSTM used for the optical model, and the low complexity of sub-lexical units, we expect to get a lightly and powerful recognizer. To this end, this section describes the unified design approach that we conducted on the French and English systems components. The unification process includes the unification of the optical models, the lexicons and finally the language models, as illustrated in figure 6.1.

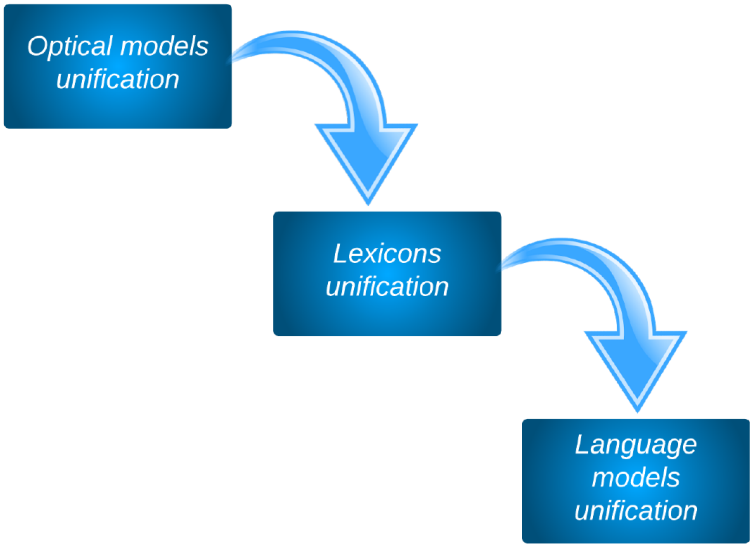


FIGURE 6.1: The three unification stages in the design process of a unified recognizer.

The benefit of using the sub-lexical units (syllables and multigrams) for building multilingual handwriting recognition systems has been studied through

two successive experiments. The first experiment aimed to examine the recognition performance of a unified handwriting recognition system with HMM optical models operated with language models of words, syllables and characters. The system configuration was prepared to simulate an ideal case study for which we considered language models with a null OOV rate (all language models fully covered the test dataset).

The objective of this experiment was to highlight the competitive performance of the syllable based model in comparison with the word and character models. We will not spend much lines to discuss this first experiment. The interested reader can refer to (Swaileh and Paquet, 2016a) to get more information, but the most important conclusion to us was the fact that we observed similar performance when running the unified system, to those of each specialized system. By the second experiment, we aimed to study the contribution of the unified optical and language models to the recognition performance of the unified handwriting recognition system which was based on state of the art BLSTM-RNN. Moreover, the experimental setup was design by considering different OOV rates of the unified lexicons and language models. In the following paragraphs we describe the properties of each unified model.

6.3.1 Character set unification

The RIMES dataset is composed of 100 character classes while the IAM dataset contains only 79 character classes. 77 character (including the white space character) classes are shared between the two datasets while in total the two datasets contain 102 character classes. The whole character set of the English IAM training dataset is included in the French RIMES training dataset except two characters:"#" and "&". Of course, most of the additional character classes in the RIMES dataset are accented character. By looking at the distribution

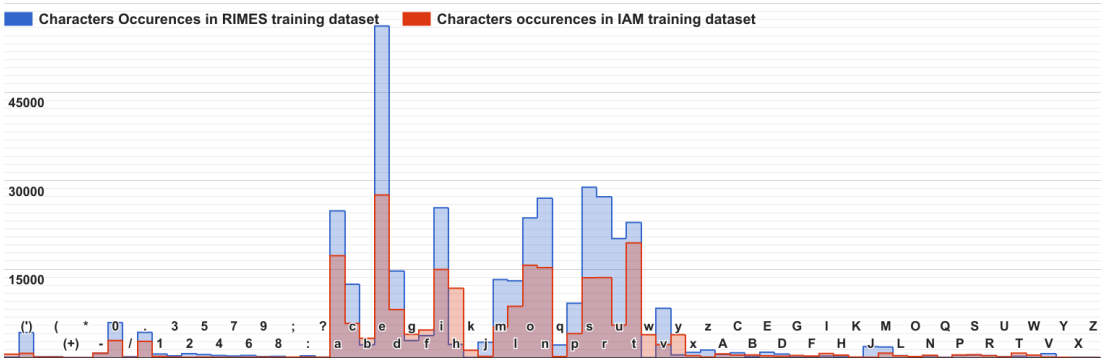


FIGURE 6.2: Statistics of the shared characters between the French RIMES and the English IAM database

of characters over the two datasets, we can see in figure 6.2. that lower case characters are more frequent in the RIMES training dataset that because it contains 9947 sentence, 8107 word and $470,317$ character. On the other hand, IAM training dataset consists of 6482 sentence, 9956 word and $287,727$ character. The total number of characters in IAM training dataset represents 61.18% of the RIMES training dataset total number of characters. We believe that this

difference in character frequencies may improve the unified optical model by having more training data for every classes.

6.3.2 Optical models unification

The unified French and English character set of the RIMES and IAM dataset are modelled by a BLSTM-RNN. Training the unified BLSTM-CTC optical model has been carried out by combining the RIMES and IAM training datasets. As already presented in chapter 5, a curriculum learning like algorithm was introduced by sorting the training examples according to their image width. Furthermore, the French and English RIMES and IAM validation datasets have been combined for controlling the training process. The training process starts with a learning rate of 10^{-5} and ends with a value of $2.79e^{-07}$ after 108 training epochs. Figure 6.3 shows the evolution of the network performance (CER) during the training epoch.

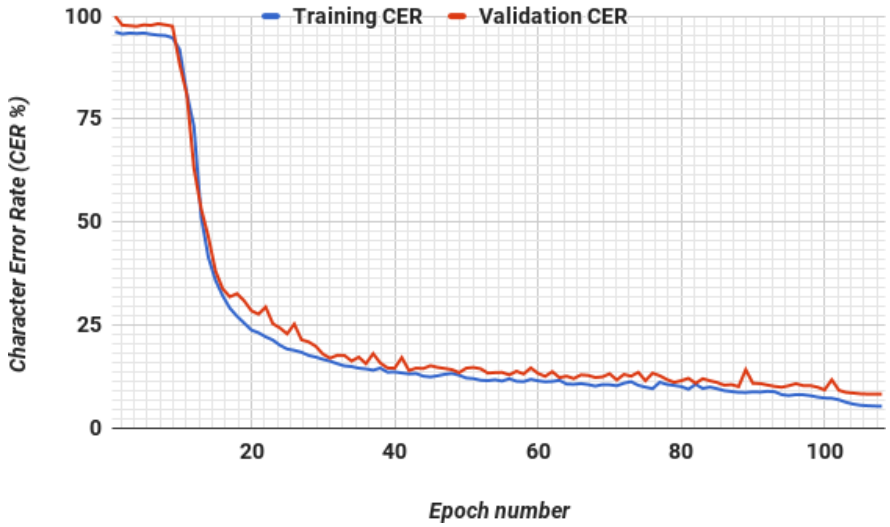


FIGURE 6.3: Evolution of the CER during the training epochs.

The unified optical model achieves a $8,33\%$ CER on the unified RIMES+IAM validation dataset which should be compared to CER obtained when training separately the network on the RIMES and on the IAM dataset. In this case the CER was $8,95\%$ on the RIMES dataset and $10,41\%$ on the IAM validation datasets. This phenomenon can be explained by the effect of having more training data as illustrated in figure 6.2.

6.3.3 Lexicons unification

The third step in building the multilingual handwriting recognition system is the unification of the French and English vocabularies (lexicons) and language models which are used by the system during the recognition process. Figure 6.4 shows the evolution of the various lexicons on the three datasets. It highlights the small increase of the 2-multigram lexicon and of the syllable lexicon when considering the unified corpus, while the word lexicon is the

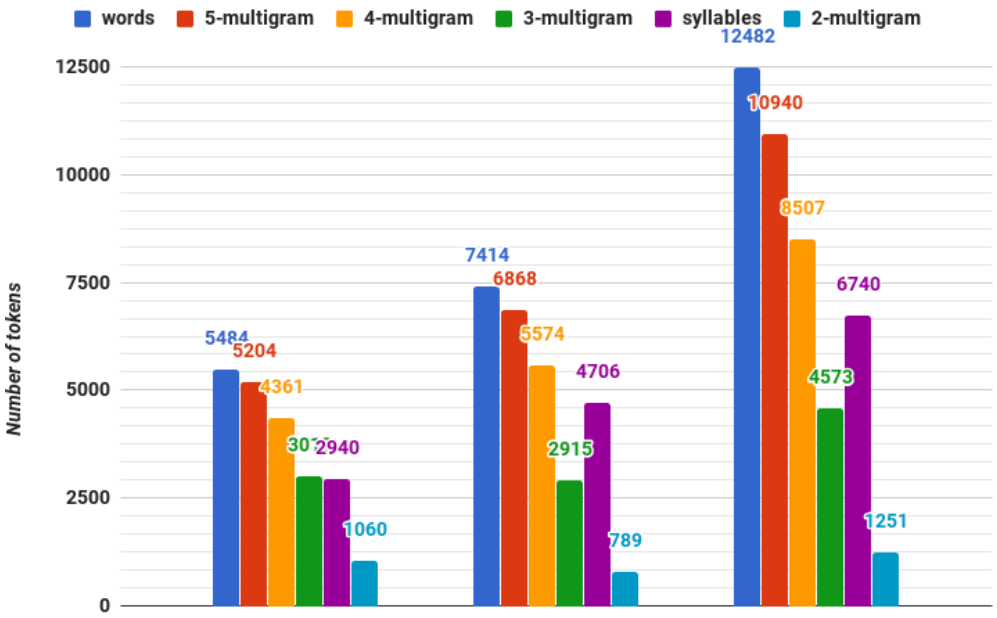


FIGURE 6.4: Size of the various lexicons on the RIMES, IAM, and the unified datasets (from left to right).

most affected in size. Similarly, We see that higher order multigrams lexicons are more affected in size when considering the unified corpus. This is a direct combinatorial affect when looking at higher order multigrams.

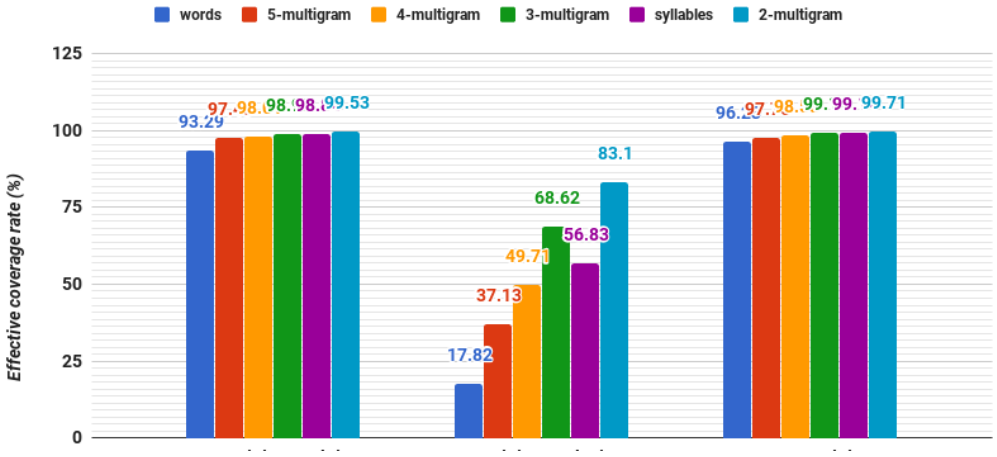


FIGURE 6.5: ECR of the RIMES test dataset by the words, syllables and multigrams of the three corpora(from left to right : RIMES, IAM, RIMES+IAM.

For a better understanding of the effect of unifying the two datasets, the effective coverage rates (ECR) of the words, syllables, and multigrams lexicons (5-multigram, 4-multigram, 3-multigram, 2-multigram) on the RIMES datasets is plotted on figure 6.5. We see that the unified lexicon has a very small effect on the ECR of the RIMES dataset. Figure 6.6 gives the ECR of the IAM dataset by the three corpora. We see that the unified lexicon improves the ECR for the higher order multigrams and the word lexicons, whereas for low order multigram the best ECR is obtained with a very small lexicon.

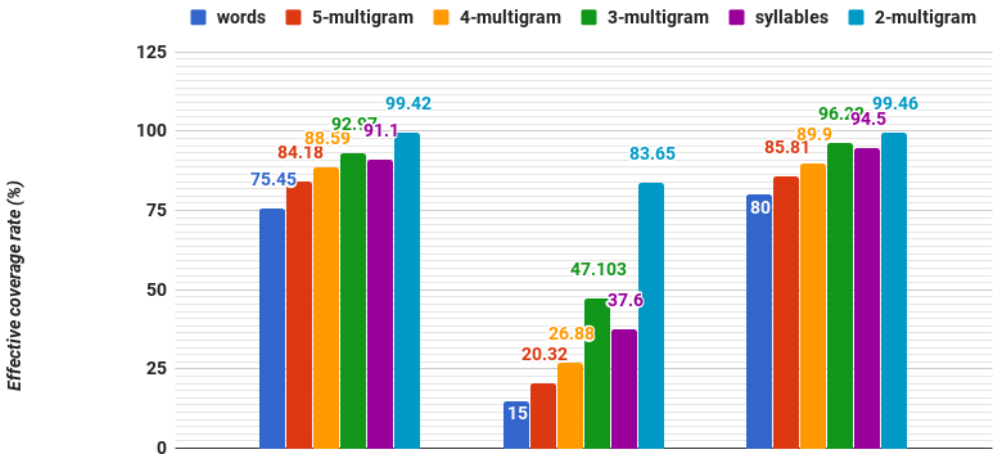


FIGURE 6.6: ECR of the IAM test datasets by the words, syllables and multigrams of the three corpora (from left to right : IAM, RIMES, RIMES+IAM).

To explore the impact of unifying external French and English linguistic resources on the same recognition task on RIMES and IAM test dataset, we unified the English LOB, Brown and Wellington corpora lexicons with the French Wikipedia + RIMES training dataset lexicons. Let EN denotes the English lexicons and language models that make use of the LOB, Brown and Wellington corpus and let FR denotes the French lexicons and language models that make use of Wikipedia + RIMES corpora. We have to recall here that the EN lexicon make use of all the corpus vocabulary while the FR lexicon make use only the 50k most frequent words (29k token) of the Wikipedia corpora. The lexicon evolution on the FR, EN and the unified dataset is illustrated by figure 6.7.

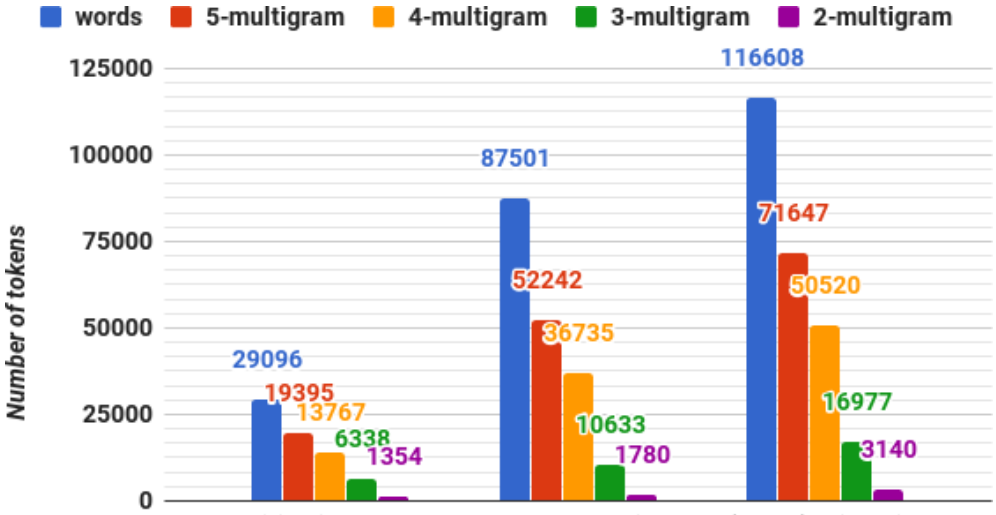


FIGURE 6.7: Size of the various lexicons on the FR, EN, and the unified datasets (from left to right).

In table 6.1 we report the ECR of the RIMES and IAM test dataset for the various lexicons derived from the three corpora (RIMES, IAM, IAM+RIMES). And we compare them to the ECR obtained with the specialized lexicons. We observe that the unified lexicons have higher ECR of the RIMES and the IAM

test datasets, for example the lexicon of word of IAM training dataset covers the IAM test dataset by 75.45% with $7.4k$ word tokens, while the same unified French and English lexicon of word covers the same test dataset by 80% with $12k$ word token.

Lexicon type	ECR of RIMES by RIMES	ECR of RIMES by IAM+RIMES	ECR of IAM by IAM	ECR of IAM by IAM+RIMES	Lexicon size of IAM+RIMES
words	93.29	96.23	75.45	80	12.5k
5-multigram	97.43	97.76	84.18	85.81	10.9k
4-multigram	98.01	98.53	88.59	89.9	8.5k
3-multigram	98.98	99.14	92.97	96.22	4.6k
syllables	98.84	99.14	91.1	94.5	6.7k
2-multigram	99.53	99.71	99.42	99.46	1.3k

TABLE 6.1: Comparing the ECR of RIMES and IAM achieved by the specialized corpora alone, and by the unified corpus.

Similarly, in table 6.2 we report the ECR of the RIMES and IAM test dataset for the lexicons derived from the FR, EN and FR+EN corpora. The same phenomena is repeating with these lexicons, the FR lexicon covers the RIMES test dataset by 82.27% with $29k$ word token while the unified FR+EN lexicon of words covers the same test dataset by 89.97% with $116k$ word token.

Lexicon type	ECR of RIMES by FR	ECR of RIMES by FR+EN	ECR of IAM by EN	ECR of IAM by FR+EN	Lexicon size of FR+EN
words	82.27	89.97	95.13	98.6	116k
5-multigram	93.73	95.22	98.34	98.58	71.6k
4-multigram	96.37	98.2	99.46	99.68	50.5k
3-multigram	97.32	98.48	99.48	99.8	16.9k
2-multigram	99.48	99.82	100	100	3.1k

TABLE 6.2: Comparing the ECR of RIMES and IAM achieved by the FR and EN corpora individually, and by the unified FR+EN corpus.

6.3.4 Unified language models

The unified French and English vocabulary of the RIMES / IAM training corpora of words, syllables, multigrams (2-multigram, 3-multigram, 4-multigram and 5-multigram) and characters are used to train their corresponding (open or closed) language model. The n-gram language models were trained with the modified Kneser-Ney smoothing method supported by the MIT language modelling toolkit. To consider the long distance dependencies while training the language models, we trained 9-gram language models of words and multigrams (5-multigram, 4-multigram, 3-multigram and 2-multigram) in addition to 10-gram language model of characters. Table 6.3 reports the perplexity values and OOV rates of these models and compares theme to the values obtained with the specialized corpora.

We observed for the tables 6.3 and 6.4 that the OOV % rate reported for the unified IAM+RIMES models on the RIMES and IAM test datasets are less

Language models	Unified IAM+RIMES models					
	perplexiy on RIMES	perplexity on IAM	perplexty on IAM+RIMES	OOV % on RIMES	OOV % on IAM	OOV % on IAM+RIMES
words	17.26	33.98	29.28	2.9	13.9	11.53
5-multigram	15.62	45.92	35.92	1.37	9.96	7.99
4-multigram	14.61	45.65	35.34	0.85	6.3	5.02
3-multigram	13.52	43.78	33.80	0.44	1.61	1.35
syllables	13.29	42.16	29.18	0.45	4.5	2.8
2-multigram	12.11	22.66	19.76	0.11	0.10	0.1
characters	11.96	13.55	13.17	0	0	0

TABLE 6.3: Unified language models perplexities and OOV rates, for the specialized and unified corpus.

Language models	Specialized RIMES models		Specialized IAM models	
	perplexity	OOV %	perplexity	OOV %
words	13.52	3.10	27.6	14.1
5-multigram	11.84	1.61	35.52	10.95
4-multigram	11.16	1.16	37.29	7.3
3-multigram	10.18	0.53	35.90	2.2
syllables	10.86	0.63	33.94	5.01
2-multigram	9.21	0.16	20.03	0.22
characters	9.6	0	12.1	0

TABLE 6.4: Specialized language models perplexities and OOV rates, for the specialized corpora.

than the ones reported for the specialized models on the same test datasets, for example the unified language model of word OOV% rate is less than the specialized one by 0.2% on the RIMES and IAM test datasets. On the other hands, the perplexity of the unified models are greater than the specialized ones, for example; the perplexity of the unified language model of words on the RIMES test data set is 17.26 where the specialized model of words have a perplexity of 13.52 on the same test dataset.

Language models	Unified FR+EN models					
	perplexity on RIMES	perplexity on IAM	perplexity on IAM+RIMES	OOV% on RIMES	OOV% on IAM	OOV% on IAM+RIMES
words	32.15	43.78	41.12	7.9	1.15	2.5
5-multigram	33.04	41.54	39.53	3.14	0.75	1.3
4-multigram	30.36	35.42	34.25	0.93	0.29	0.4
3-multigram	26.47	28.09	27.73	0.68	0.03	0.18
2-multigram	22.62	21.94	22.09	0.06	0	0.01
Characters	18.82	19.04	18.98	0	0	0

TABLE 6.5: Unified FR+EN language models perplexities and OOV rates, for the specialized (FR and EN) and unified (FR+EN) corpus.

With the FR and EN unified lexicons we trained the large vocabulary language models of words, muligrams and characters and their perplexities and

Language models	Specialized FR models		Specialized EN models	
	perplexity on RIMES	OOV %	perplexity on IAM	OOV %
words	35.82	8.3	40.12	1.4
5-multigram	22.38	4.1	38.30	0.82
4-multigram	20.78	2.3	32.88	0.33
3-multigram	18.55	1.4	25.81	0.06
2-multigram	16.83	0.22	19.53	0
Characters	16.97	0	17.03	0

TABLE 6.6: Specialized language models perplexities and OOV rates, for the specialized corpora.

OOV rates are reported in table 6.5. The FR and EN syllables lexicons are not available for training the unified large vocabulary language models of syllables, that because the unsupervised decomposition approach (see chapter 5) is time consuming. The specialized FR and EN language models perplexities and OOV rates are listed in table 6.6. For the large vocabulary unified language models one can observe through the tables 6.5 and 6.6 also the decrement in their OOV rates compared with the OOV rates reported for the specialized language models on the RIMES and IAM test datasets. Taking for example the 4-multigram models; the FR specialized 4-multigram model reported 2.3% OOV tokens on the RIMES test dataset while the unified FR+EN model of 4-multigram reported 0.93% on the same test dataset. The decrement in the OOV rates cab be explained by the compensation of the similar tokens between the French and English languages where the token which can be missed in one language training dataset can be learned from the other language training dataset.

6.4 Evaluation protocol and experimental results

The architecture of the BLSTM-RNN based unified recognition system allows us to evaluate the system components (language model and optical model) independently and together. For these experiments the optical model was trained using $10k$ text line images from the RIMES training dataset (which contains $11,333$ training example) combined with $6,482$ text line images that represent the IAM training dataset. The recognition system parameters are optimized using the unified RIMES+IAM dataset which consists of $1,333$ and 976 images of RIMES and IAM validation datasets respectively. The unified system performance was evaluated on the RIMES and IAM test datasets individually. We recall that the RIMES test dataset contains 778 text line images and the IAM test dataset contains $2,915$ text line images.

During the evaluation, we are interested in quantifying the contribution of each stage (optical model, and language model) to the performance of the unified system, and to compare these contributions with the performance of the specialized systems. Four experiments can be conducted in this purpose

as depicted on figure 6.8 below. The first setup (1=UU, in green) consists in combining the unified optical model with the unified language models. The second setup (2=US, in red) consists in combining the unified optical model with the specialized language model. The third setup (3=SU, in orange) combines the specialized optical model with the unified language model. Finally, the fourth setup (4=SS, in blue) combines the specialized optical model with the specialized language model. The four setups will be evaluated on the IAM dataset and the RIMES dataset.

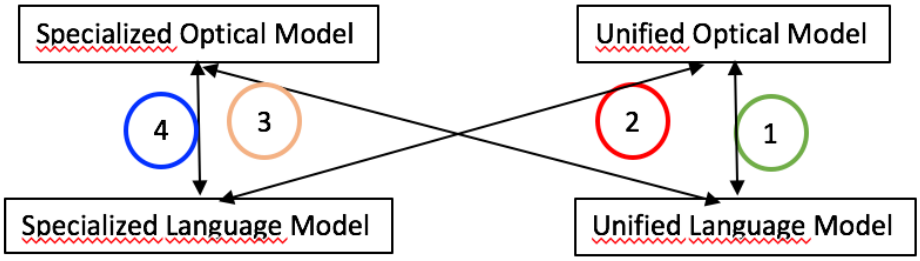


FIGURE 6.8: The four experimental setups for analyzing the respective contributions of the unified optical models and the language models.

For the specialized and unified optical models, we evaluated two different groups of language models on the RIMES and IAM test datasets; the first group contains the small vocabulary language models that are trained on the RIMES, IAM and the unified IAM+RIMES training datasets. The second group contains the large vocabulary language models which are trained with the FR, EN and FR+EN lexicons on their corresponding FR (Wikipedia + RIMES), EN (LOB, Brown and Wellington) corpora and the unified FR+EN training corpora.

A preliminary result is provided by analyzing the raw performance of the unified optical model alone. As reported in table 6.7 below, a slight improvement of the performance around 2% WER and CER on the RIMES datasets is observed, compared to the performance of the specialized optical model. A 3% improvement is observed on the IAM datasets, for both the WER and CER. This is a direct effect of having more data for training the optical model which share a lot of character classes, such improvement would not be possible for languages that do not share so many character classes, for examples languages written using different scripts.

Test datasets	RIMES optical model		IAM optical model		Unified optical model	
	CER	WER	CER	WER	CER	WER
RIMES	12.32	38.88	–	–	10.02	36.3
IAM	–	–	18.91	52.13	15.26	44.41

TABLE 6.7: Specialized and unified optical model raw performance (without language model).

We now provide the detailed analysis of the performance of the two unified stages using the four setups depicted above and for the two test datasets.

6.4.1 Results on RIMES test dataset

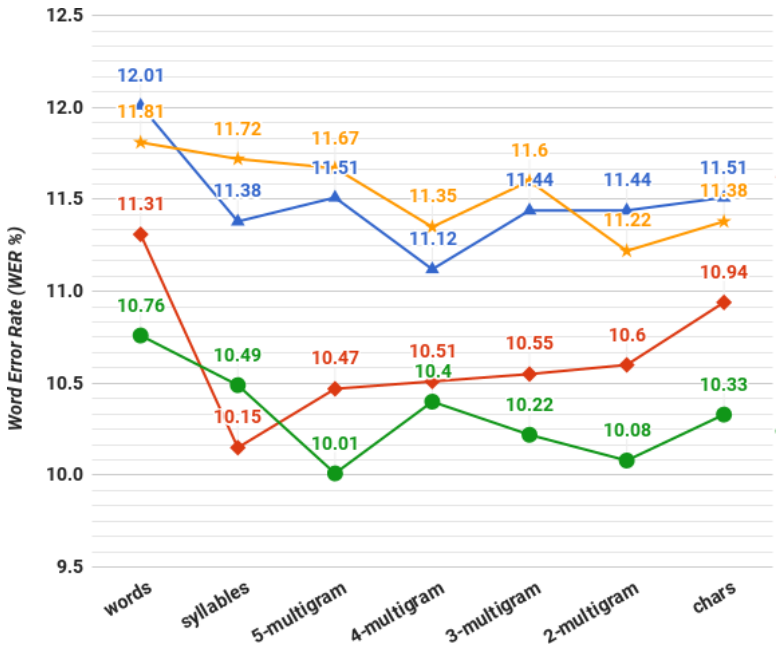


FIGURE 6.9: WER (%) on the RIMES dataset using the four experimental setups : blue = SS, Orange = SU, Red = US, Green = UU.

Figure 6.9 shows the performance obtained with the four configurations of the system, as a function of the lexical units considered (words, multigrams, and characters). A performance enhancement of around 1% of the WER is reported for every system that integrates the unified optical model (Red and Green curves). Improvement is observed for every language models, which perform nearly equally and better than the word language model, and the character language model. The contribution of the unified language model appears rather moderate compared to the improvement brought by the optical model. But most importantly, we can highlight the fact that unifying two languages in one single model does not affect the recognition performance. This is observed for both optical models, either specialized or unified. These interesting results must be analysed also by considering the lexicon size of the sub-lexical units based language models which is rather small when considering character and 2-multigrams.

Similar to the experiments with small vocabulary language models, we carried the experiments with the FR large vocabulary language models which gave the results illustrated in figure 6.10. These experiments results present different observations. Looking at the blue (SS) and orange (SU) curves we observe that unifying large vocabulary language models (FR+EN) does not affect brutally the recognition performance even if their unified FR and EN lexicons

are not of the same size; remember that the unified FR+EN lexicon of word ($116k$ word token) consists of $29k$ FR words and $87k$ EN words.

We observe for every configuration that the unified optical model operating with the specialized and unified 2-multigram model reaches the state of the art results 9.85% WER. This can be explained by the capacity of the 2-multigram model to model effectively with low OOV rate, the information held on the unified FR+EN corpora with a very small lexicon size of $3.1k$ tokens. By comparing the OOV rate reported by the FR+EN unified large vocabulary model (0.06% for lexicon of $3.1k$ token) and those reported by the IAM+RIMES unified small vocabulary models (0.11% for lexicon of $1.3k$) we can justify the enhanced performance of the FR+EN unified large vocabulary language model of 2-multigram (9.85% WER) over the IAM+RIMES small vocabulary 2-mulitgram language model (10.08% WER). On the other hand, the performance of the other unified models of words, multigrams and characters overcomes the specialized ones.

However, we see that the word language model performs poorly for any configuration. The word model trained on thes generic large resources do not match the RIMES vocabulary where it was limited to the $50k$ most frequent word of the French Wikipedia corpora resulting to high OOV rate.

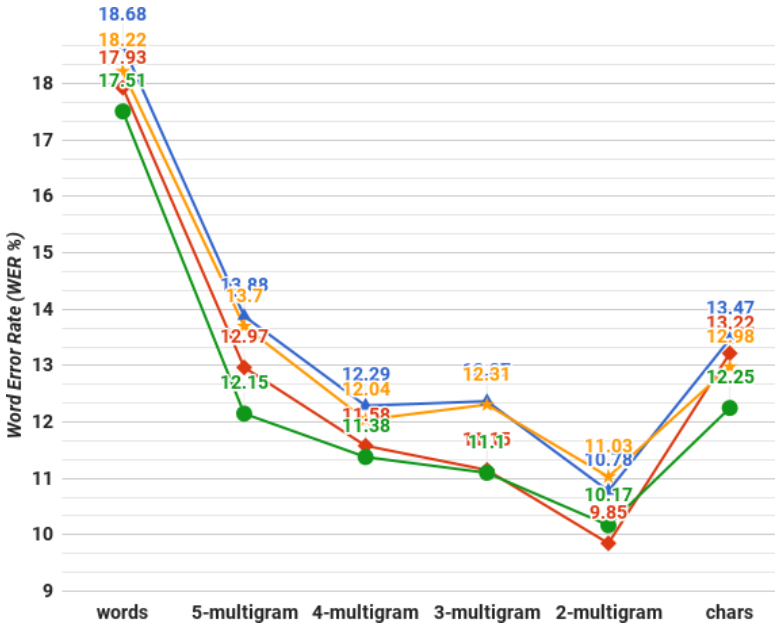


FIGURE 6.10: WER (%) on the RIMES dataset using the four experimental setups : blue = SS, Orange = SU, Red = US, Green = UU using very large FR+EN language models

6.4.2 Results on IAM test dataset

Similarly, considering the IAM dataset, figure 6.11 below reports the performance of the four configurations of the system, as a function of the lexical units considered (words, multigrams, and characters). Once again, the same behavior on the IAM dataset than on the RIMES dataset is observed. The unified

optical model increases the WER by 4% and the CER by 2% CER, compared to the performance of the systems that integrate a specialized optical model. This is observed for any of the small vocabulary language models with the best improvement observed for the sub-lexical units based systems and especially for the 2-multigram language model. Once again the contribution of the language model appear rather limited to the recognition performance, but considering the two French and English languages there is no reason to believe that some performance improvement should be gain by combining the two language. In fact, we could expect that the language modelling task should become more complex. These results clearly show that the unification of both language models in one single language model has nearly no effect on the performance. This was also observed on the RIMES dataset. As a conclusion we can highlight that combining the two models allows improving the performance of the optical model by the benefit of having more training samples per character classes, while at the same time combining the sub-lexical units based language models does not degrade the performance while getting a very compact bi-lingual language model, with a small lexicon size.

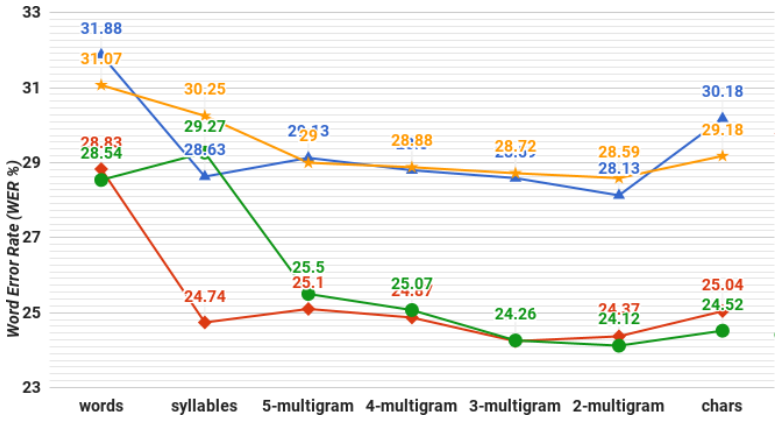


FIGURE 6.11: WER (%) on the IAM dataset using the four experimental setups : blue = SS, Orange = SU, Red = US, Green = UU.

The large vocabulary FR, EN and the unified FR+EN language model was evaluated for the four evaluation setups and the obtained results are illustrated in figure 6.12. The results shows the best recognition performances are obtained with the unified optical models with all language models. The specialized optical model that works with the unified FR+EN language models show a slight performance degradation (as shown by the orange curve) compared to the EN language models (blue curve) for all language models except the 4-multigram and the characters language models.

By recalling the lexicon size and the OOV rates reported for the large vocabulary FR+EN unified languages model and the small vocabulary IAM+RIMES language models listed in the tables 6.3, 6.4, 6.5 and 6.6, we observed that the unified large vocabulary language models reports small OOV rates which justify their state of the art recognition performance. For example: the unified FR+EN large vocabulary language model of word reported 1.15% OOV words while the specialized EN language model of word reported 1.4% OOV

words For this reason even the unified language model of words shows similar performance as the multigram ones with the advantage of the small unified FR+EN lexicon size of multigram.

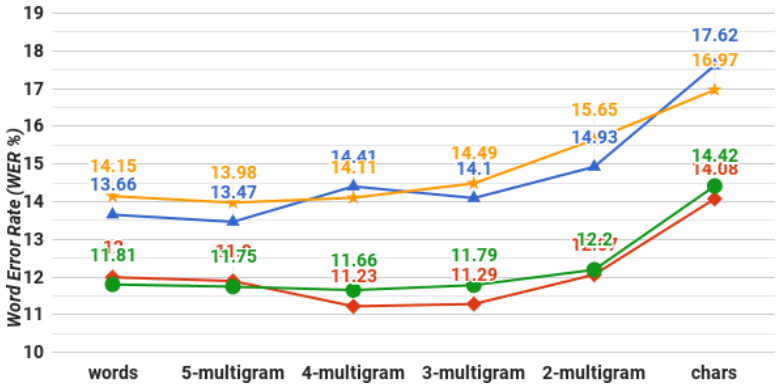


FIGURE 6.12: WER (%) on the IAM dataset using the four experimental setups : blue = SS, Orange = SU, Red = US, Green = UU using very large FR+EN language models

6.5 Conclusion

In this chapter we studied the unified French/English models of words, syllables, multigrams and characters for handwriting recognition. The sub-lexical units based systems offer many advantages over the character model which models badly the words, and over the word model which models only a limited number of words that are found in the vocabulary of the training corpus. The evaluation of the unified (optical and language) models in different operating setup with the specialized (optical and language) models offered us a clear vision about the behavior of the unified models. The enhancement of the unified optical model has the major contribution in enhancing the overall recognition system performances as shown by the results above. The unified language models have a slight enhancing contribution when they are operating with the unified optical models. Indeed, they show equivalent behavior to the pure specialized (optical and language) models behavior when operating with the specialized optical models. But the more important contribution of these results regarding the language models is to show that one single bilingual language model composed of sub-lexical units does not degrade the performance, thus allowing the design of lightly language models composed of sub-lexical units of both languages that can be introduced in one single recognition engine. As a general conclusion, these results show that sub-lexical units are good units for both French and English languages and that they support to be mixed into one single recognition system.

Conclusions and Perspectives

The main goal of this thesis was to develop improved language modelling approaches for performing efficient large vocabulary unconstrained and continuous handwriting recognition of French and English handwritten documents. In the literature, large vocabulary handwriting recognition systems are described based on two main components: an optical model and a language model. The optical model matches the handwriting script observed in an image as a numerical sequence of observations features (denoted by S) to a sequence of characters which may represent a sequence of words (denoted by W) by the computation of a likelihood function denoted by $P(S \mid W)$. The language model constrains the recognition hypotheses to be some admissible sequences of words by the language, by the computation of the a priori probability P(W) of each hypothesis. In between the sequence of characters produced by the optical model and the sequence of words examined by the language model, the system's vocabulary allows to constrain the character hypotheses to be some admissible words only. In practice, there is no vocabulary which can cover all language words, including named entities and alphanumeric strings such as reference numbers or bank identification numbers. As a consequence, the words which are not introduced in the system's vocabulary (Out-Of-Vocabulary words, OOV) can not be recognized by the system. This is what is called a closed vocabulary recognition system. To improve the recognition performance one may try to increase the size of the system's vocabulary, so as to reduce the OOV rate. But this is only possible with the expense of increasing the system's complexity which leads to the degradation of the system's performance. Therefore, it is required to find a compromise between the vocabulary size, the time complexity of the system, and the recognition performance.

In the literature, there is two types of language model: closed vocabulary language models and open vocabulary language models. The closed vocabulary language models are trained on a fixed size vocabulary and any sequence of words (W) that contains one or more OOVs are discarded from the estimation of the language model. On the contrary, the open vocabulary language models account for the OOV words by affecting them a certain amount of the total probability thanks to the unknown class (<unk>), which introduces the OOV words within the language model vocabulary and training corpus. Furthermore, the OOVs can be represented in the language model vocabulary and training corpus by their sequence of characters. Thus, the language model will represent a hybrid language model of words and characters. But such modelling approach is not convenient, because words and character classes are competing each other, which increases the language model confusion. A free lexicon recognition system can be used to tackle the OOV problem. This system makes use of character classes only, with a high order language model to

introduce sufficient context in the model. Doing this way, almost every possible character string is admissible, and there is almost no OOV. However, even high order character language models perform poorly when working alone because of their poor capacity to model the context of the language sentence. Another effective way to tackle the OOV problem is to train separately two language models: a word language model accounting for the most frequent words in the training corpus, and a character language model. During the decoding procedure the two models are synchronously working where the language model of character is introduced where the word language model has decoded an OOV word.

Alternatively, in this thesis we introduced sub-lexical units made of syllables or multigrams in order to tackle the OOV problem. A complete processing chain of handwriting processing and recognition was implemented from line detection in the document image to line recognition using the proposed open-vocabulary sub-lexical units based language model. An automatic iterative text line segmentation smearing method was proposed with which we enhanced the extraction of the text line images on different datasets, without the need to tune any parameter. Two types of optical models have been implemented. The first implementation consists of a classical HMM optical model working with Histogram of Gradient (HoG) features. The second implementation consists of a state of the art BLSTM-RNN optical model which deals with the pixel values directly. These systems have been trained and evaluated on the French RIMES and the English IAM datasets. The efficiency of the BLSTM-RNN optical model over the HMM one was noticeable by looking at the Character Error Rate (CER) rates for the optical models alone. The system vocabulary and language models were trained on the RIMES and IAM datasets as well as on some additional external resources such as the French and English Wikipedia pages and the English LOB, Brown and Wellington datasets. The results obtained with the BLSTM-RNN based recognition system are close to the state of the art performance with the advantage of the system's simple architecture with a reduced complexity thanks to the use of sub lexical units.

Sub-lexical based language models have been investigated using multigrams and syllables. Two approaches have been used to provide word decomposition into sub-lexical units: a supervised and an unsupervised word decomposition approach. Small and large vocabularies have been employed in the recognition systems in order to assess the actual potential of sub-lexical based models compared to full-word models. This approach has led to a significant increase in the overall lexical coverage, indicated by a considerable reduction of the out-of-vocabulary (OOV) rates measured on the test datasets. This has introduced a promising step towards the solution of dealing with lexicon with low coverage rates. As a result, significant improvements of the recognition performance have been achieved compared to the traditional full-word based language models.

Motivated by the interesting recognition performance obtained with sub-lexical units, we studied the possibilities of building a unified handwriting recognition system for the French and English languages. The unification of the individual (specialised) French and English recognition systems has been

examined considering both the unification of the optical model and the unification of the language model. By the introduction of sub-lexical units, the unified recognition system outperforms each individual specialized system.

As perspectives to the use of sub-lexical units in a handwriting recognition system we can mention the use of connectionist language models. Such models can account for variable length dependencies of language units in the language model. This would remove the constraint of using the fix length dependency modelling introduced by the n-gram modelling approach. Also, by introducing connectionist language models we may explore new system architectures composed solely of RNN for both the optical stage and the language stage. In this respect, we may take inspiration from the recent attention models that have been proposed to achieve caption generation from images, or translation tasks. Another perspective is to introduce sub-lexical units in the optical model, which would allow to introduce some multigrams optical model accounting for the most frequent co-occuring characters. This could be done at the expense of increasing the number of classes in the optical model. Finally, we may extend the unified modelling approache to include more languages, possibly with different scripts such as Hindi, or Arabic languages to explore more in depth the advantages of using sub-lexical units for unifying handwriting recognition systems.

Bibliography

Abadi, Martín et al. (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467*.

Abdel-Hamid, Ossama et al. (2012). "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, pp. 4277–4280.

AbdulKader, Ahmad (2008). "A two-tier Arabic offline handwriting recognition based on conditional joining rules". In: *Arabic and Chinese Handwriting Recognition*. Springer, pp. 70–81.

Adda-Decker, Martine (2003). "A corpus-based decompounding algorithm for German lexical modeling in LVCSR." In: *INTERSPEECH*.

Adda-Decker, Martine and Gilles Adda (2000). "Morphological decomposition for ASR in German". In: *Workshop on Phonetics and Phonology in ASR, Saarbrücken, Germany*, pp. 129–143.

Ahmad, Irfan and Gernot A Fink (2016). "Class-Based Contextual Modeling for Handwritten Arabic Text Recognition". In: *Frontiers in Handwriting Recognition, 2016. Proceedings. fifteenth International conference ICHFR*. IEEE, pp. 369–374.

Ait-Mohand, Kamel, Thierry Paquet, and Nicolas Ragot (2014). "Combining structure and parameter adaptation of HMMs for printed text recognition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36.9, pp. 1716–1732.

Allauzen, Cyril et al. (2007). "OpenFst: A general and efficient weighted finite-state transducer library". In: *International Conference on Implementation and Application of Automata*. Springer, pp. 11–23.

Arica, Nafiz and Fatos T Yarman-Vural (2001). "An overview of character recognition focused on off-line handwriting". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 31.2, pp. 216–233.

Bacchiani, Michiel et al. (1996). "Design of a speech recognition system based on acoustically derived segmental units". In: *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*. Vol. 1. IEEE, pp. 443–446.

Bahrampour, Soheil et al. (2015). "Comparative study of caffe, neon, theano, and torch for deep learning". In: *arXiv preprint arXiv:1511.06435*.

Bartlett, Susan, Grzegorz Kondrak, and Colin Cherry (2009). "On the syllabification of phonemes". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 308–316.

Bassil, Youssef and Mohammad Alwani (2012). "Ocr post-processing error correction algorithm using google online spelling suggestion". In: *arXiv preprint arXiv:1204.0191*.

Bastien, Frédéric et al. (2012). "Theano: new features and speed improvements". In: *arXiv preprint arXiv:1211.5590*.

Bauer, Laurie (1993). *Manual of information to accompany the Wellington corpus of written New Zealand English*. Department of Linguistics, Victoria University of Wellington Wellington.

Beeton, Barbara (2010). "Hyphenation exception log". In: *TUGboat-TeX Users Group* 31.3, p. 160.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Bengio, Yoshua et al. (1995). "LeRec: A NN/HMM hybrid for on-line handwriting recognition". In: *Neural Computation* 7.6, pp. 1289–1303.

Bengio, Yoshua et al. (2003). "A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Bengio, Yoshua et al. (2009). "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. ACM, pp. 41–48.

BenZeghiba, Mohamed Faouzi, Jerome Louradour, and Christopher Kermorvant (2015). "Hybrid word/Part-of-Arabic-Word Language Models for arabic text document recognition". In: *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, pp. 671–675.

Berger, Adam L, Vincent J Della Pietra, and Stephen A Della Pietra (1996). "A maximum entropy approach to natural language processing". In: *Computational linguistics* 22.1, pp. 39–71.

Bimbot, Frederic et al. (1994). "Modeles de séquencesa horizon variable: Multigrams". In: *Proc. XX-emes Journées d'Etude sur la Parole*, pp. 467–472.

Bisani, Maximilian and Hermann Ney (2005). "Open vocabulary speech recognition with flat hybrid models." In: *INTERSPEECH*, pp. 725–728.

— (2008). "Joint-sequence models for grapheme-to-phoneme conversion". In: *Speech Communication* 50.5, pp. 434–451.

Bledsoe, Woodrow Wilson and Iben Browning (1959). *Pattern recognition and reading by machine*. PGEC.

Bluche, Théodore (2015). "Deep Neural Networks for Large Vocabulary Handwritten Text Recognition". PhD thesis. Université Paris Sud-Paris XI.

Bluche, Théodore, Hermann Ney, and Christopher Kermorvant (2013a). "Feature extraction with convolutional neural networks for handwritten word recognition". In: *2013 12th International Conference on Document Analysis and Recognition*. IEEE, pp. 285–289.

Bluche, Theodore, Hermann Ney, and Christopher Kermorvant (2013b). "Tandem HMM with convolutional neural network for handwritten word recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 2390–2394.

Bluche, Théodore, Hermann Ney, and Christopher Kermorvant (2014). "A Comparison of Sequence-Trained Deep Neural Networks and Recurrent Neural Networks Optical Modeling for Handwriting Recognition". In: *International*

Conference on Statistical Language and Speech Processing. Springer, pp. 199–210.

Bluche, Théodore et al. (2014). "The a2ia arabic handwritten text recognition system at the open hart2013 evaluation". In: *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*. IEEE, pp. 161–165.

Bourlard, Hervé and Christian J Wellekens (1990). "Links between Markov models and multilayer perceptrons". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.12, pp. 1167–1178.

Brainstorm (2015). *Brainstorm Kernel Description*. URL: https://github.com/IDSIA/brainstorm (visited on 02/25/2017).

Brakensiek, Anja, Jörg Rottland, and Gerhard Rigoll (2002). "Handwritten address recognition with open vocabulary using character n-grams". In: *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*. IEEE, pp. 357–362.

Broder, Andrei Z et al. (1997). "Syntactic clustering of the web". In: *Computer Networks and ISDN Systems* 29.8-13, pp. 1157–1166.

Brown, Keith (2006). "Encyclopedia of language and linguistics". In:

Brown, Peter F et al. (1990). "A statistical approach to machine translation". In: *Computational linguistics* 16.2, pp. 79–85.

Bukhari, Syed Saqib, Faisal Shafait, and Thomas M Breuel (2009). "Script-independent handwritten textlines segmentation using active contours". In: *2009 10th International Conference on Document Analysis and Recognition*. IEEE, pp. 446–450.

Bullinaria, John A (2004). "Introduction to neural networks". In: *Lecture notes*.

Bunke, Horst, Samy Bengio, and Alessandro Vinciarelli (2004). "Offline recognition of unconstrained handwritten texts using HMMs and statistical language models". In: *IEEE transactions on Pattern analysis and Machine intelligence* 26.6, pp. 709–720.

Casey, Richard G and Eric Lecolinet (1996). "A survey of methods and strategies in character segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 18.7, pp. 690–706.

Chatelain, Clément, Laurent Heutte, and Thierry Paquet (2006). "Segmentation-driven recognition applied to numerical field extraction from handwritten incoming mail documents". In: *International Workshop on Document Analysis Systems*. Springer, pp. 564–575.

Chen, Stanley F and Joshua Goodman (1996). "An empirical study of smoothing techniques for language modeling". In: *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 310–318.

Chollet, François et al. (2015). "Keras: Deep learning library for theano and tensorflow". In: *URL: https://keras. io/k*.

Chomsky, Noam (2002). *Syntactic structures*. Walter de Gruyter.

Chou, Philip A and Tom Lookabaugh (1994). "Variable dimension vector quantization of linear predictive coefficients of speech". In: *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*. Vol. 1. IEEE, pp. I–505.

Choueiter, Ghinwa F (2009). "Linguistically-motivated sub-word modeling with applications to speech recognition". PhD thesis. Massachusetts Institute of Technology.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555*.

Collobert, Ronan, Samy Bengio, and Johnny Mariéthoz (2002). *Torch: a modular machine learning software library*. Tech. rep. Idiap.

Comrie, Bernard, Stephen Matthews, and Maria Polinsky (1997). *The atlas of languages: The origin and development of languages throughout the world*. Bloomsbury.

Creutz, Mathias et al. (2007). "Morph-based speech recognition and modeling of out-of-vocabulary words across languages". In: *ACM Transactions on Speech and Language Processing (TSLP)* 5.1, p. 3.

Deligne, Sabine and Frederic Bimbot (1995). "Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams". In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1. IEEE, pp. 169–172.

Deligne, Sabine and Frédéric Bimbot (1997). "Inference of variable-length linguistic and acoustic units by multigrams". In: *Speech Communication* 23.3, pp. 223–241.

Deligne, Sabine and Yoshinori Sagisaka (2000). "Statistical language modeling with a class-basedn-multigram model". In: *Computer Speech & Language* 14.3, pp. 261–279.

Deligne, Sabine, François Yvon, and Frédéric Bimbot (1996). "Introducing statistical dependencies and structural constraints in variable-length sequence models". In: *International Colloquium on Grammatical Inference*. Springer, pp. 156–167.

Della Pietra, Stephen et al. (1992). "Adaptive language modeling using minimum discriminant estimation". In: *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, pp. 103–106.

Diringer, David (1951). *The alphabet: a key to the history of mankind*.

Doetsch, Patrick, Michal Kozielski, and Hermann Ney (2014). "Fast and robust training of recurrent neural networks for offline handwriting recognition". In: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, pp. 279–284.

Doetsch, Patrick et al. (2016). "RETURNN: The RWTH extensible training framework for universal recurrent neural networks". In: *arXiv preprint arXiv:1608.00895*.

Dreuw, Philippe et al. (2012). "RWTH OCR: A large vocabulary optical character recognition system for Arabic scripts". In: *Guide to OCR for Arabic Scripts*. Springer, pp. 215–254.

Eck, Douglas and Juergen Schmidhuber (2002). "A first look at music composition using lstm recurrent neural networks". In: *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* 103.

El-Desoky, Amr et al. (2009). "Investigating the use of morphological decomposition and diacritization for improving Arabic LVCSR." In: *INTERSPEECH*, pp. 2679–2682.

El-Sheikh, Talaat S and Ramez M Guindi (1988). "Computer recognition of Arabic cursive scripts". In: *Pattern Recognition* 21.4, pp. 293–302.

El-Yacoubi, Mounim A, Michel Gilloux, and J-M Bertille (2002). "A statistical approach for phrase location and recognition within a text line: an application to street name recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.2, pp. 172–188.

Farabet, Clement et al. (2013). "Learning hierarchical features for scene labeling". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1915–1929.

Feild, Jacqueline L, Erik G Learned-Miller, and David A Smith (2013). "Using a probabilistic syllable model to improve scene text recognition". In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, pp. 897–901.

Fink, Gernot A (2014). *Markov models for pattern recognition: from theory to applications*. Springer Science & Business Media.

Fiscus, J (2015). *Sclite scoring package, version 1.5. US National Institute of Standard Technology (NIST)*.

Fiscus, Jon (1998). "Sclite scoring package version 1.5". In: *US National Institute of Standard Technology (NIST), URL http://www. itl. nist. gov/iaui/894.01/tools*.

Flipo, Daniel, Bernard Gaulle, and Karine Vancauwenberghe (1994). "Motifs françis de césure typographique". In: *Cahiers gutenberg n*.

Francis, Winthrop Nelson and Henry Kučera (1979). *Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers*. Brown University, Department of Lingustics.

Freund, Yoav and Robert E Schapire (1999). "Large margin classification using the perceptron algorithm". In: *Machine learning* 37.3, pp. 277–296.

Ganapathiraju, Aravind et al. (2001). "Syllable-based large vocabulary continuous speech recognition". In: *IEEE Transactions on speech and audio processing* 9.4, pp. 358–366.

Gatos, Basilios, Nikolaos Stamatopoulos, and Georgios Louloudis (2011). "ICDAR2009 handwriting segmentation contest". In: *International Journal on Document Analysis and Recognition (IJDAR)* 14.1, pp. 25–33.

Gatos, Basilis, Nikolaos Stamatopoulos, and Georgios Louloudis (2010). "Icfhr 2010 handwriting segmentation contest". In: *Frontiers in handwriting recognition (icfhr), 2010 international conference on*. IEEE, pp. 737–742.

Gers, Felix A and E Schmidhuber (2001). "LSTM recurrent networks learn simple context-free and context-sensitive languages". In: *IEEE Transactions on Neural Networks* 12.6, pp. 1333–1340.

Ghosh, Debashis, Tulika Dube, and Adamane Shivaprasad (2010). "Script recognition—a review". In: *IEEE Transactions on pattern analysis and machine intelligence* 32.12, pp. 2142–2161.

Ghoshal, Arnab, Pawel Swietojanski, and Steve Renals (2013). "Multilingual training of deep neural networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 7319–7323.

Good, Irving J (1953). "The population frequencies of species and the estimation of population parameters". In: *Biometrika*, pp. 237–264.

Goodman, Joshua T (2001). "A bit of progress in language modeling". In: *Computer Speech & Language* 15.4, pp. 403–434.

Gorski, Nikolai et al. (1999). "A2ia check reader: A family of bank check recognition systems". In: *Document Analysis and Recognition, 1999. ICDAR'99. Proceedings of the Fifth International Conference on*. IEEE, pp. 523–526.

Govindan, VK and AP Shivaprasad (1990). "Character recognition—a review". In: *Pattern recognition* 23.7, pp. 671–683.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pp. 6645–6649.

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5, pp. 602–610.

— (2009). "Offline handwriting recognition with multidimensional recurrent neural networks". In: *Advances in neural information processing systems*, pp. 545–552.

Graves, Alex et al. (2006). "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 369–376.

Grosicki, Emmanuele and Haikal El-Abed (2011). "Icdar 2011-french handwriting recognition competition". In: *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, pp. 1459–1463.

Grosicki, Emmanuèle et al. (2009). "Results of the rimes evaluation campaign for handwritten mail processing". In: *2009 10th International Conference on Document Analysis and Recognition*. IEEE, pp. 941–945.

Guillevic, Didier and Ching Y Suen (1998). "Recognition of legal amounts on bank cheques". In: *Pattern Analysis and Applications* 1.1, pp. 28–41.

Hacioglu, Kadri and Wayne Ward (2001). "Dialog-context dependent language modeling combining n-grams and stochastic context-free grammars". In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. Vol. 1. IEEE, pp. 537–540.

Hamdani, Mahdi, Amr El-Desoky Mousa, and Hermann Ney (2013). "Open vocabulary Arabic handwriting recognition using morphological decomposition". In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, pp. 280–284.

Hindson, Matthew (2016). *Free English language hyphenation dictionary*. http://hindson.com.au/info/free/free-english-language-hyphenation-dictionary/. [Online; accessed 21-Dec-2015].

Hinton, GE, DE Rumelhart, and RJ Williams (1985). "Learning internal representations by back-propagating errors". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* 1.

Hochreiter, Sepp (1991). "Untersuchungen zu dynamischen neuronalen Netzen". PhD thesis. diploma thesis, institut für informatik, lehrstuhl prof. brauer, technische universität münchen.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hsu, Bo-June Paul (2009). "Language modeling for limited-data domains". PhD thesis. Massachusetts Institute of Technology.

Huang, Xuedong et al. (1993). "The SPHINX-II speech recognition system: an overview". In: *Computer Speech & Language* 7.2, pp. 137–148.

Huang, Xuedong et al. (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR.

Indurkhya, Nitin and Fred J Damerau (2010). *Handbook of natural language processing*. Vol. 2. CRC Press.

International-Phonetic-Association (1999). *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press.

Jelinek, Fred (1990). "Self-organized language modeling for speech recognition". In: *Readings in speech recognition*, pp. 450–506.

Jelinek, Frederick et al. (1991). "A Dynamic Language Model for Speech Recognition." In: *HLT*. Vol. 91, pp. 293–295.

Jia, Yangqing et al. (2014). "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, pp. 675–678.

Johansson, Stig (1980). "The LOB corpus of British English texts: presentation and comments". In: *ALLC journal* 1.1, pp. 25–36.

Jonas, S (2009). "Improved modeling in handwriting recognition". In: *Master's thesis, Human Language Technology and Pattern Recognition Group, RWTH Aachen University, Aachen, Germany*.

Jones, Rhys James, Simon Downey, and John SD Mason (1997). "Continuous speech recognition using syllables." In: *Eurospeech*.

Joshua T, Goodman JT (2001). "A bit of progress in language modeling extended version". In: *Machine Learning and Applied Statistics Group Microsoft Research. Technical Report, MSR-TR-2001-72*.

Jurafsky, Daniel et al. (1995). "Using a stochastic context-free grammar as a language model for speech recognition". In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1. IEEE, pp. 189–192.

Kaltenmeier, Alfred et al. (1993). "Sophisticated topology of hidden Markov models for cursive script recognition". In: *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*. IEEE, pp. 139–142.

Katsouros, Vassilis and Vassilis Papavassiliou (2011). *Segmentation of handwritten document images into text lines*. Citeseer.

Kessentini, Yousri, Thierry Paquet, and Abdelmajid Benhamadou (2008). "A multi-stream HMM-based approach for off-line multi-script handwritten word recognition". In: *a a* 1, p. 1.

Kimura, Fumitaka, Malayappan Shridhar, and Z Chen (1993). "Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words". In: *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*. IEEE, pp. 18–22.

Klakow, Dietrich and Jochen Peters (2002). "Testing the correlation of word error rate and perplexity". In: *Speech Communication* 38.1, pp. 19–28.

Kneser, Reinhard and Hermann Ney (1995). "Improved backing-off for m-gram language modeling". In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1. IEEE, pp. 181–184.

Koerich, Alessandro L, Robert Sabourin, and Ching Y Suen (2003). "Large vocabulary off-line handwriting recognition: A survey". In: *Pattern Analysis & Applications* 6.2, pp. 97–121.

Kozielski, Michal, Patrick Doetsch, and Hermann Ney (2013). "Improvements in rwth's system for off-line handwriting recognition". In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, pp. 935–939.

Kozielski, Michal et al. (2014a). "Multilingual Off-line Handwriting Recognition in Real-world Images". In: *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*. IEEE, pp. 121–125.

Kozielski, Michal et al. (2014b). "Open-lexicon Language Modeling Combining Word and Character Levels". In: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, pp. 343–348.

Kuhn, Thomas, Heinrich Niemann, and Ernst Günter Schukat-Talamazzini (1994). "Ergodic hidden Markov models and polygrams for language modeling". In: *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*. Vol. 1. IEEE, pp. I–357.

Kukich, Karen (1992). "Techniques for automatically correcting words in text". In: *ACM Computing Surveys (CSUR)* 24.4, pp. 377–439.

Lasagne (2017). *Lasagne, lightweight library to build and train neural networks in Theano, Kernel Description*. URL: http://lasagne.readthedocs.org/ (visited on 02/25/2017).

Le Cun, Yann, Leon Bottou, and Yoshua Bengio (1997). "Reading checks with multilayer graph transformer networks". In: *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*. Vol. 1. IEEE, pp. 151–154.

Lecocq, Pierre (1991). *Apprentissage de la lecture et dyslexie*. Vol. 190. Editions Mardaga.

LeCun, Yann et al. (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4, pp. 541–551.

Lee, Akinobu, Tatsuya Kawahara, and Kiyohiro Shikano (2001). "Julius—an open source real-time large vocabulary recognition engine". In:

Lee, Jay J and Jin H Kim (1997). "A unified network-based approach for online recognition of multi-lingual cursive handwritings". In: IWFHR.

Lee, Seong-Whan and Young-Joon Kim (1995). "A new type of recurrent neural network for handwritten character recognition". In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 1. IEEE, pp. 38–41.

Levenshtein, Vladimir I (1966). "Binary codes capable of correcting deletions, insertions and reversals". In: *Soviet physics doklady*. Vol. 10, p. 707.

Li, Yi et al. (2008). "Script-independent text line segmentation in freestyle handwritten documents". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.8, pp. 1313–1329.

Madhvanath, Sriganesh and Venu Govindaraju (1996). "Holistic lexicon reduction for handwritten word recognition". In: *Electronic Imaging: Science & Technology*. International Society for Optics and Photonics, pp. 224–234.

Madhvanath, Sriganesh and Venu Krpasundar (1997). "Pruning large lexicons using generalized word shape descriptors". In: *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*. Vol. 2. IEEE, pp. 552–555.

Majewski, Piotr (2008). "Syllable based language model for large vocabulary continuous speech recognition of polish". In: *International Conference on Text, Speech and Dialogue*. Springer, pp. 397–401.

Malaviya, Ashutosh, Christoph Leja, and Liliane Peters (1996). "Multi-script handwriting recognition with FOHDEL". In: *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*. IEEE, pp. 147–151.

Manning, Christopher D, Hinrich Schütze, et al. (1999). *Foundations of statistical natural language processing*. Vol. 999. MIT Press.

Mantas, J (1986). "An overview of character recognition methodologies". In: *Pattern recognition* 19.6, pp. 425–430.

Mao, Song, Azriel Rosenfeld, and Tapas Kanungo (2003). "Document structure analysis algorithms: a literature survey". In: *Electronic Imaging 2003*. International Society for Optics and Photonics, pp. 197–207.

Märgner, Volker and Haikal El Abed (2012). *Guide to OCR for Arabic scripts*. Springer.

Marti, U-V and Horst Bunke (2001). "Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system". In: *International journal of Pattern Recognition and Artificial intelligence* 15.01, pp. 65–90.

— (2002). "The IAM-database: an English sentence database for offline handwriting recognition". In: *International Journal on Document Analysis and Recognition* 5.1, pp. 39–46.

McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Miao, Yajie, Mohammad Gowayyed, and Florian Metze (2015). "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding". In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pp. 167–174.

Mikolov, Tomáš (2012). "Statistical language models based on neural networks". In: *Presentation at Google, Mountain View, 2nd April*.

Mikolov, Tomas et al. (2010). "Recurrent neural network based language model." In: *Interspeech*. Vol. 2, p. 3.

Mikolov, Tomáš et al. (2011a). "Extensions of recurrent neural network language model". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, pp. 5528–5531.

Mikolov, Tomas et al. (2011b). "Rnnlm-recurrent neural network language modeling toolkit". In: *Proc. of the 2011 ASRU Workshop*, pp. 196–201.

Mikolov, Tomáš et al. (2012). "Subword language modeling with neural networks". In: *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*.

Mohri, Mehryar (1997). "Finite-state transducers in language and speech processing". In: *Computational linguistics* 23.2, pp. 269–311.

Mohri, Mehryar, Fernando Pereira, and Michael Riley (2002). "Weighted finite-state transducers in speech recognition". In: *Computer Speech & Language* 16.1, pp. 69–88.

— (2008). "Speech recognition with weighted finite-state transducers". In: *Springer Handbook of Speech Processing*. Springer, pp. 559–584.

Mori, Shunji, Kazuhiko Yamamoto, and Michio Yasuda (1984). "Research on machine recognition of handprinted characters". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 4, pp. 386–405.

Mousa, Amr El-Desoky et al. (2010). "Sub-lexical language models for German LVCSR". In: *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, pp. 171–176.

Mousa, Amr Ibrahim El-Desoky and Hermann Ney (2014). *Sub-word based language modeling of morphologically rich languages for LVCSR*. Tech. rep. Fachgruppe Informatik.

Moysset, Bastien et al. (2014). "The A2iA multi-lingual text recognition system at the second Maurdor evaluation". In: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, pp. 297–302.

Neon (2017). *Neon: Fast, scalable, easy-to-use Python based deep learning framework by Nervana Kernel Description*. URL: https://github.com/nervanasystems/neon (visited on 02/25/2017).

New, Boris et al. (2004). "Lexique 3: A new French lexical database". In: *Behavior Research Methods, Instruments, & Computers* 36.3, pp. 516–524.

Pantke, Werner et al. (2013). "HADARA–A software system for semi-automatic processing of historical handwritten Arabic documents". In: *Archiving Conference*. Vol. 2013. 1. Society for Imaging Science and Technology, pp. 161–166.

Papavassiliou, Vassilis et al. (2010). "Handwritten document image segmentation into text lines and words". In: *Pattern Recognition* 43.1, pp. 369–377.

Paquet, Thierry and Yves Lecourtier (1993). "Automatic reading of the literal amount of bank checks". In: *Machine Vision and Applications* 6.2-3, pp. 151–162.

Pham, Vu et al. (2014). "Dropout improves recurrent neural networks for handwriting recognition". In: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, pp. 285–290.

Plötz, Thomas and Gernot A Fink (2009). "Markov models for offline handwriting recognition: a survey". In: *International Journal on Document Analysis and Recognition (IJDAR)* 12.4, pp. 269–298.

Povey, Daniel et al. (2011). "The Kaldi speech recognition toolkit". In: *IEEE 2011 workshop on automatic speech recognition and understanding*. EPFL-CONF-192584. IEEE Signal Processing Society.

Rabiner, Lawrence R (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286.

Ramisch, Carlos (2008). *N-gram models for language detection*.

Ridouane, Rachid, Yohann Meynadier, and Cécile Fougeron (2011). "La syllabe: objet théorique et réalité physique". In: *Faits de langue 37*, pp. 225–246.

Ries, Klaus, Finn Dag Buo, and Ye-Yi Wang (1995). "Improved language modelling by unsupervised acquisition of structure". In: *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. Vol. 1. IEEE, pp. 193–196.

Ripley, Brian D (1993). "Statistical aspects of neural networks". In: *Networks and chaos—statistical and probabilistic aspects* 50, pp. 40–123.

Robinson, Anthony J (1994). "An application of recurrent nets to phone probability estimation". In: *IEEE transactions on Neural Networks* 5.2, pp. 298–305.

Roekhaut, Sophie, Sandrine Brognaux, and Richard Beaufort (2012). "Syllabation graphémique automatique à l'aide d'un dictionnaire phonétique aligné". In:

Romero, Verónica, Joan Andreu, Enrique Vidal, et al. (2011). "Handwritten text recognition for marriage register books". In: *2011 International Conference on Document Analysis and Recognition*. IEEE, pp. 533–537.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Rosenfeld, Roni (2000). "Two decades of statistical language modeling: Where do we go from here?" In:

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1988). "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3, p. 1.

Rybach, David et al. (2009). "The RWTH aachen university open source speech recognition system." In: *Interspeech*, pp. 2111–2114.

Ryst, Elise (2014). "Syllabation en anglais et en français: considérations formelles et expérimentales". PhD thesis. Paris 8.

Sagisaka, Yoshinori and Naoto Iwahashi (1995). "Objective optimization in algorithms for text-to-speech synthesis". In: *Speech Coding and Synthesis. Elsevier Science BV*, pp. 686–706.

Sahu, Vijay Laxmi and Babita Kubde (2013). "Offline Handwritten Character Recognition Techniques using Neural Network: A Review". In: *International Journal of Science and Research (IJSR), India Online ISSN*, pp. 2319–7064.

Sánchez, Joan Andreu et al. (2013). "tranScriptorium: a european project on handwritten text recognition". In: *Proceedings of the 2013 ACM symposium on Document engineering*. ACM, pp. 227–228.

Schambach, Marc-Peter, Jörg Rottland, and Théophile Alary (2008). "How to convert a Latin handwriting recognition system to Arabic". In: *Proceedings of the international conference on frontiers in handwriting recognition, Montréal, Canada*.

Schrumpf, Christian, Martha Larson, and Stefan Eickeler (2005). "Syllable-based language models in speech recognition for English spoken document retrieval". In: *Proc. of the 7th International Workshop of the EU Network of Excellence DELOS on AVIVDiLib, Cortona, Italy*, pp. 196–205.

Schukat-Talamazzini, Ernst Günter et al. (1995). "Permugram language models." In: *EUROSPEECH*.

Schuster, Mike and Kuldip K Paliwal (1997). "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681.

Senior, Andrew W and Anthony J Robinson (1998). "An off-line cursive handwriting recognition system". In: *IEEE transactions on pattern analysis and machine intelligence* 20.3, pp. 309–321.

Senior, Andrew William (1994). "Off-line cursive handwriting recognition using recurrent neural networks". In:

Sethy, Abhinav, Bhuvana Ramabhadran, and Shrikanth Narayanan (2003). "Improvements in English ASR for the MALACH project using syllable-centric models". In: *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*. IEEE, pp. 129–134.

Shaik, M Ali Basha et al. (2011). "Hybrid Language Models Using Mixed Types of Sub-Lexical Units for Open Vocabulary German LVCSR." In: *INTERSPEECH*, pp. 1441–1444.

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *The Bell System Technical Journa* 5.1, pp. 3–55.

Shi, Zhixin, Srirangaraj Setlur, and Venu Govindaraju (2009). "A steerable directional local profile technique for extraction of handwritten arabic text lines". In: *2009 10th International Conference on Document Analysis and Recognition*. IEEE, pp. 176–180.

site, IAM official (2016). *IAM Handwriting Database*. URL: http://www.fki.inf.unibe.ch/databases/iam-handwriting-database/iam-handwriting-database#icdar02 (visited on 03/06/2017).

Siu, Manhung and Mari Ostendorf (2000). "Variable n-grams and extensions for conversational speech language modeling". In: *IEEE Transactions on Speech and Audio Processing* 8.1, pp. 63–75.

Srihari, Sargur N (2000). "Handwritten address interpretation: a task of many pattern recognition problems". In: *International journal of pattern recognition and artificial intelligence* 14.05, pp. 663–674.

Srihari, Sargur N and Edward J Kuebert (1997). "Integration of hand-written address interpretation technology into the united states postal service remote computer reader system". In: *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*. Vol. 2. IEEE, pp. 892–896.

Stamatopoulos, Nikolaos et al. (2013). "Icdar 2013 handwriting segmentation contest". In: *2013 12th International Conference on Document Analysis and Recognition*. IEEE, pp. 1402–1406.

Stolcke, Andreas et al. (2002). "SRILM-an extensible language modeling toolkit." In: *Interspeech*. Vol. 2002, p. 2002.

Suen, Ching Y, Marc Berthod, and Shunji Mori (1980). "Automatic recognition of handprinted characters—the state of the art". In: *Proceedings of the IEEE* 68.4, pp. 469–487.

Sutton, Charles and Andrew McCallum (2006). "An introduction to conditional random fields for relational learning". In: *Introduction to statistical relational learning*, pp. 93–128.

Swaileh Wassim, Julien lerouge and Thierry Paquet (2016a). "Unified French / English syllabic model for handwriting recognition". In: *Handwriting Recognition (ICFHR), 2016 15th International Conference on Frontiers in Handwriting Recognition*. under submission.

Swaileh, Wassim, Julien Lerouge, and Thierry Paquet (2016). "A Unified French/English syllabic model for handwriting recognition". In: *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*. IEEE, pp. 536–541.

Swaileh, Wassim and Thierry Paquet (2016b). "A syllable based model for handwriting recognition". In: *Pattern Recognition (ICPR), 2016 23rd International Conference on Pattern Recognition*. under submission.

— (2016c). "Un modèle syllabique du français et de l'anglais pour la reconnaissance de l'écriture". In: *Document numérique* 19.2, pp. 117–134.

Swaileh, Wassim et al. (2017). "handwriting recognition with multigrams". In: *International conference in document analyses and Handwriting Recognition (ICDAR), 2017 14th International Conference on*.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9.

Tang, Yuan Y, Seong-Whan Lee, and Ching Y Suen (1996). "Automatic document processing: a survey". In: *Pattern recognition* 29.12, pp. 1931–1952.

Toselli, Alejandro Hector et al. (2004). "Integrated handwriting recognition and interpretation using finite-state models". In: *International Journal of Pattern Recognition and Artificial Intelligence* 18.04, pp. 519–539.

Van Merriënboer, Bart et al. (2015). "Blocks and fuel: Frameworks for deep learning". In: *arXiv preprint arXiv:1506.00619*.

Voigtlaender, Paul, Patrick Doetsch, and Hermann Ney (2016). "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks". In: *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*. IEEE, pp. 228–233.

Walker, Willie et al. (2004). "Sphinx-4: A flexible open source framework for speech recognition". In:

Wang, Stanley Xinlei (2009). "Using graphone models in automatic speech recognition". PhD thesis. Citeseer.

Wiesler, Simon et al. (2014). "RASR/NN: The RWTH neural network toolkit for speech recognition". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pp. 3281–3285.

Xu, Bo et al. (1996). "Speaker-independent dictation of Chinese speech with 32K vocabulary". In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 4. IEEE, pp. 2320–2323.

Young, Steve J and Sj Young (1993). *The HTK hidden Markov model toolkit: Design and philosophy*. University of Cambridge, Department of Engineering.

Yousfi, Sonia, Sid-Ahmed Berrani, and Christophe Garcia (2017). "Contribution of recurrent connectionist language models in improving LSTM-based Arabic text recognition in videos". In: *Pattern Recognition* 64, pp. 245–254.

Yu, Shun-Zheng (2010). "Hidden semi-Markov models". In: *Artificial Intelligence* 174.2, pp. 215–243.

Zamora-Martinez, Francisco et al. (2014). "Neural network language models for off-line handwriting recognition". In: *Pattern Recognition* 47.4, pp. 1642–1652.

Zimmermann, Matthias and Horst Bunke (2002). "Hidden Markov model length optimization for handwriting recognition systems". In: *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*. IEEE, pp. 369–374.

Zimmermann, Matthias, J Chappelier, and Horst Bunke (2006). "Offline grammar-based recognition of handwritten sentences". In: *IEEE transactions on pattern analysis and machine intelligence* 28.5, pp. 818–821.