Character-level Interaction in Multimodal Computer-Assisted Transcription of Text Images

Author: Daniel Martín-Albo Simón
Advisor: Enrique Vidal Ruiz
Co-advisor: Verónica Romero Gómez

IARFID Master Thesis
Universidad Politécnica de Valencia

February 3, 2011

Abstract

At present, automatic handwriting text recognition (HTR) systems are very effective in restricted applications involving the use of a limited vocabulary or constrained handwriting. However, in the completely opposite cases, such as old manuscripts or spontaneous text, current HTR systems do not achieve acceptable results. Because of these unsatisfactory results, is often required heavy human intervention to validate and correct, if necessary, the results of such systems.

To resolve this inconvenience, it was proposed, in previous works, an interactive framework that allowed the human transcriber and the automatic HTR system to cooperate to generate iteratively a correct transcription of the text image. In this scenario the system uses the text image and a human validated part, called prefix, of its transcription to propose an appropriate continuation. After that, the user finds and corrects the next system error, providing a new consolidated prefix which the system uses to suggest a new better continuation. This process is performed until the full transcription is completely accepted by the user.

In this work, multimodal interaction at character-level is studied. Until now, multimodal interaction had been studied only at whole-word level. However, the interactive system presented before requires constant human intervention to enhance the transcription. Therefore is necessary to make this process as comfortable and ergonomic as possible. Character-level pen-stroke interactions may lead to the scenario we are looking for. Since it is faster and easier to introduce only a handwritten character instead of a whole word.

The on-line feedback HTR subsystem, responsible for interpreting the pen-stroke produced by the user, is based on Hidden Markov Models (HMMs) and n-grams. To train this subsystem an on-line handwriting corpus has been used.

Empirical results show that, despite losing the deterministic accuracy of the keyboard, this approach can save significant amounts of user effort with respect to both fully manual transcription and non-interactive post-editing correction.

Contents

List of Figures

List of Tables

Introduction

At present time, the use of automatic handwritten text recognition systems (HTR) for the transcription of manuscript document images is far from being useful, mainly because of the unrestricted vocabulary and/or handwriting styles involved in such documents. Typically, the automatic transcriptions obtained by these HTR systems need a heavy human post–editing process in order to obtain transcriptions of standard quality. In practice, such a *post–editing* solution becomes rather inefficient, expensive and hardly acceptable by professional transcribers.

In previous works [?, ?], a more effective, *interactive* on-line approach was presented. This approach, called "Computer Assisted Transcription of Handwritten Text Images" (CATTI), combines the accuracy ensured by the human transcriber with the efficiency of the HTR systems to obtain final perfect transcriptions. Empirical results show that the use of CATTI systems can save a substantial quantity of human effort with respect to both pure manual transcriptions and post-editing.

So far, human corrective feedback for CATTI has been studied at two different levels: a) whole–word interactions (both typed [?] and handwritten using an e–pen interface [?]) and b) (typed) character–level corrections [?]. According to the results of these works, character level corrections can save a significant quantity of human effort with respect to whole-word corrections, while multimodal, e-pen interaction seems more ergonomic for human transcribers, which is a key point in the design of friendly and usable user interfaces.

In this work, we focus on character level interaction using the more ergonomic *e–pen handwriting* modality, which is perhaps the most natural way to provide the required feedback in CATTI systems. It is important to note, however, that the use of this kind of non–deterministic feedback typically increases the overall interaction cost in order to correct the possible feedback decoding errors. Nevertheless, by using informations derived from the interaction process, we will show how the decoding accuracy can be significantly improved over using a plain e–pen handwriting recognizer which can not take advantage of the interaction context.

Finally, this document is organized as follows: In the first chapter, the on-line and off-line HTR systems are introduced. These systems follow the classical three-module architecture in Pattern Recognition systems. These modules will be fully explained in this chapter.

In Chapter 2, the theoretical concepts of the existing CATTI and multimodal CATTI systems are explained.

The multimodal version of the CATTI at level character is presented in Chapter 3. In order to improve the quality of the system different approaches are explained in this chapter.

In Chapter 4, first, the corpora used to test the system is presented. Then, after selecting the appropriate assessment measures which can help us to compare the results, we proceed to evaluate the different scenarios proposed in Chapter 3.

And finally, in Chapter 5 the future work and conclusions of this document are presented.

Chapter 1

Handwritten Text Recognition

Handwritten text recognition is the ability of a computer to interpret handwritten input from sources such as paper documents, photographs, touch-screens and other devices. This task is not simple since there are different types of problems to solve, such as:

- the variability of the handwriting.
- the complexity of segmenting text in lines or words.
- the different styles.
- the open vocabulary.

These constraints make that the HTR systems do not achieve acceptable results. However, humans solve this problem easily. This may be due to the inter-cooperation between different levels of knowledge. These levels are: morphological, lexical and syntactical.

This situation also occurs [?] in automatic speech recognition (ASR). Therefore, it is normal that both problems are dealt with techniques based on the cooperation of all the previously mentioned knowledge sources [?, ?, ?, ?]. The different techniques can be modelled using finite state models, such as HMMs, grammars or automata.

HTR systems can be classified into off-line and on-line depending on the mode of acquisition. The first recognizes the image of the written text with an optical scanning and the second one by receiving the movements of the pen by a touch-screen.

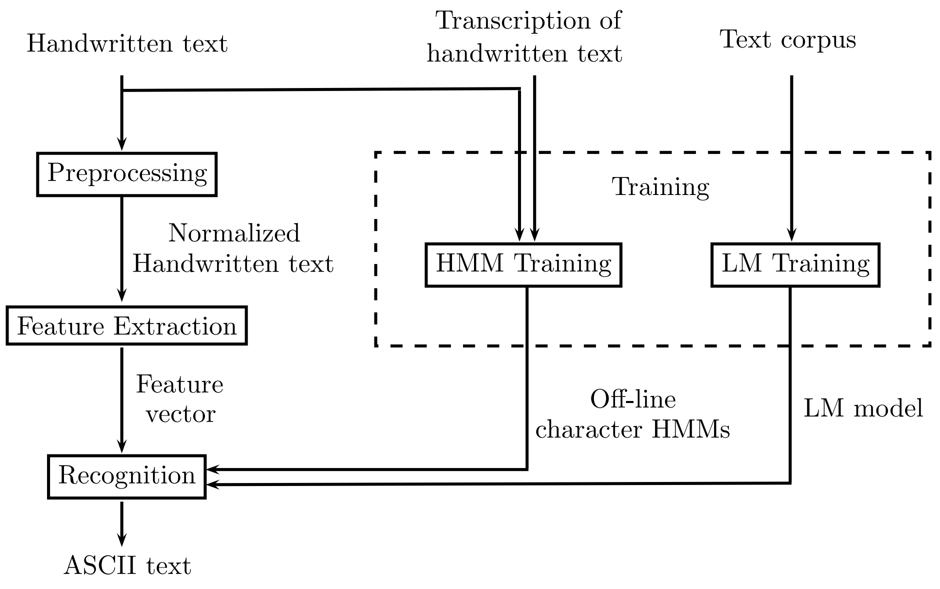


Fig. 1.1 — Overview of the HTR system.

Both systems follow the classic architecture (See Fig.**??**) of pattern recognition of four main blocks: preprocessing, feature extraction and recognition.

Preprocessing: The goal of preprocessing is, on the one hand, to discard irrelevant information that can negatively affect the recognition and on the other hand to reduce the variability of the information.

Feature extraction: Is a special form of dimensionality reduction. Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately.

Recognition: This module is in charge of assigning a label to a given input value.

Training: Finally, there is an additional module responsible for train the different models used on the previous step: Hidden Markov models, language models and lexical models.

In this work we focus on the on-line method, in this chapter, therefore, we will first study in detail the off-line system. Later we will see an overview of the off-line method for completeness. More information about the off-line recognition system can be found in [**?**].

1.1 On-line Handwritten Text Recognition

The on-line HTR subsystem presented in this section is in charge of decoding the feedback touchscreen data (formed by a set of 2d-points, sampled at a regular time instants) for multimodal text correction on the MM-CATTI at Character Level system, in other words, it recognizes the pen strokes (characters) written by the user to amend the errors during the off-line HTR process. Now, in the next subsections, we will see in detail the different modules that conforms the system.

1.1.1 Preprocessing

The preprocessing techniques used are similar to those shown in [?]. However, in this work, only two steps have been made: remove of duplicated points and noise reduction.

Remove of duplicated points: Duplicated points appear in a trajectory when the pen remains down and motionless for sometime. They can be removed by checking whether the coordinates of the previous point are the same. If they are in the same position, one of them is kept and the other is removed.

Noise reduction: The noise is mainly due to erratic hand motion and registration errors during the digitalization process. To reduce this noise, a simple smoothing technique is used, replacing every point by the mean value of its neighbours [?].

1.1.2 Feature Extraction

Each preprocessed trajectory is transformed into a new temporal sequence of seven-dimensional real-valued feature vectors [?, ?]. The seven features computed for each sample point are:

Normalized Position (x_t,y_t): The coordinates of each trajectory are linearly scaled and translated to obtain new values in the range [0,100], preserving the original aspect-ratio of the trajectory.

Normalized First Derivates (x'_t,y'_t): These are calculated following the method given in [?].

Second derivates (x''_t,y''_t): The second derivates are calculated in the same way as the first derivates, but using x'_t and y'_t instead of the normalized position.

Curvature (k_t): This is the inverse of the local radius of the trajectory in each point. It is calculated as:

$$k_t = \frac{x'_t y''_t - x''_t y'_t}{(x'^2_t + y'^2_t)^{\frac{3}{2}}} \tag{1.1}$$

1.1.3 Recognition

After the feature extraction an engine is used to identify the corresponding characters. Several different recognition techniques are currently available. Now, we will see the approach we used in this work.

Probabilistic Framework

As we explained in the previous section, each handwritten character is represented by a sequence of variable size of seven-dimensional feature vectors, x=(x_1 x_2 x_3 x_4 x_5 x_6 x_7). Hence, the traditional recognition problem [?] can be formulated as the problem of finding the most likely character $\hat{c}$:

$$\hat{c} = \arg\max_c Pr(c \mid x) \tag{1.2}$$

Using the Bayes' rule:

$$\hat{c} = \arg\max_c \frac{Pr(x \mid c) \cdot Pr(c)}{Pr(x)} \tag{1.3}$$

Since $Pr(x)$ is constant for any likely character sequence, the probability can be simplified as shown in (**??**).

$$\hat{c} = \arg\max_c Pr(x \mid c) \cdot Pr(c) \tag{1.4}$$

The first term of (**??**), representing morphological-lexical knowledge, can be modelled using Hidden Markov Models [?, ?]. The second term, which represents syntactic knowledge, is approximated by a language model, usually n-grams [?]. In practice, the values of $Pr(x \mid c)$ and $Pr(c)$ are not always on the same scale, so we need to balance the absolute values of both probabilities. To achieve this, is normally used a language weight called Grammar Scale Factor (GSF), which weights the influence of the language model on the recognition result [?]. Moreover, as we need to compute the probability of a string of characters, we have to multiply a set of numbers, being each of the numbers quite small, so the product gets extremely small very fast, which can cause underflow problems. In order to avoid this we will use log-probabilities. Finally, the unknown *true* probability ($Pr(...)$) will be approximated by the model probability ($P(...)$).

So, (**??**) can be rewritten as:

$$\hat{c} \approx \arg\max_{c} \log P(x \mid c) + \alpha \cdot P(c) \text{ where } \alpha \text{ is GSF} \tag{1.5}$$

Character and Language Modelling

HMMs are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, etc. Basically each character HMM is a stochastic finite-state automaton that models the succession, along the horizontal axis, of feature vectors which are extracted from instances of this character [?, ?, ?]. Each HMM state generates a feature vectors following an adequate parametric probabilistic law; in this case, a Gaussian Mixture.

The required number of Gaussians in the mixture must be optimized during the tests. On the other hand, the number of states for the HMMs is variable for each character class. The number of states N_{S_c} chosen for each HMM character class M_c was computes as $N_{S_c} = l_c/f$ where l_c is the average length of the sequences of feature vectors used to train M_c, and f is a design parameter measuring the average number of feature vectors modelled per states. This rule of setting up N_{S_c} tries to balance modelling effort across states. Fig.**??** shows a sketch of how a HMM models an off-line character.

Once we have defined the topology of the HMMs, the model can be easily trained using labeled samples of handwritten characters. This training process is carried out using a well known instance of the EM algorithm called forward backward or Baum-Welch [?].

Finally, the concatenation of characters into words, $Pr(c)$ (second term of (**??**)), can be modelled using an n-gram language model, with Kneser-Ney back-off smoothing [?], which uses the previous n-1 characters to predict the next one.

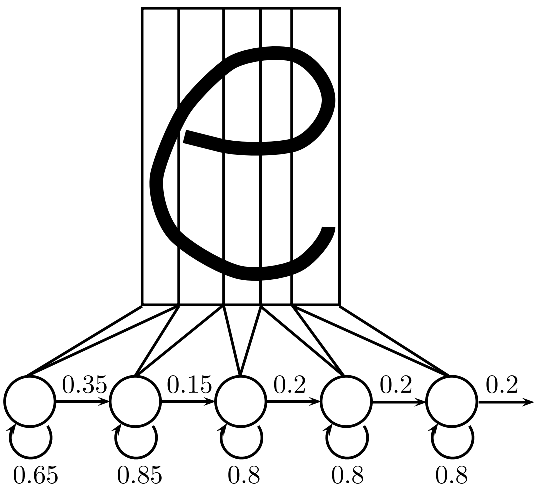


Fig. 1.2 — Example HMM of five states that models the character 'e'.

In (**??**) we can see the formulation.

$$Pr(c) \approx \prod_{i=1}^{N} P(c_i \mid c_{i-n+1}^{i-1}) \tag{1.6}$$

Once every model is available, recognition of new test characters can be performed. To perform this work; i.e. decode the input feature vectors sequence x into the output character $\hat{c}$, we use the Viterbi algorithm [**?**, **?**].

1.2 Off-line Handwritten Text Recognition

In this section the off-line system used in this work is presented. Just as we did in the previous section, now, we present the different modules that forms the off-line system.

1.2.1 Preprocessing

In this module we need to improve the visual characteristics of the handwritten documents to improve subsequent recognition. This is because these documents are written by different people with different font types and sizes in the words, in addittion to underlined or crossed-out words.

In the preprocessing module used here the following steps take place: background removal and noise reduction, slope and slant correction and size normalization.

Background removal and noise reduction

It is quite common in handwritten documents to suffer from degradation problems. This causes that the background color is not uniform and therefore hinders the readability of the document. To avoid this appropriate filtering methods should be developed in order to remove noise.

In this work, background removal and noise reduction is performed by applying a two-dimensional median filter [?] on the entire image and subtracting the result from the original image. Then, a grey-level normalization to increase the image contrast is applied.

Slope correction

The slope is the angle between the direction of the line on which the writer aligned the words on a text line and the horizontal direction. The slope correction processes an original image to put the text line into horizontal position by applying a rotation operation with the same slope angle, but in the opposite direction. Since it is likely that each word (or group of words) has a different slope angle, the original image is divided into segments surrounded by wide blank spaces. Then, slope correction is applied to each segment separately. As this process is based on more or less sophisticated heuristics, in this work, to define this minimum blank space, a vertical projection of the image is carried out, and the average size of all the spaces is computed.

After that, to obtain the angle we use a method based on horizontal projections [?].

Slant correction

Slant correction shears the isolated line image horizontally to bring the writing in an upright position. The slant angle is determined using a method based on vertical projection profile [?]. The method is based on the observation that the columns distribution of the vertical projection profile presents a maximum variance for the non slanted text.

Size normalization

The purpose of this operation is to make the system invariant to character size and to reduce the areas of background pixels which remain on the image because of the ascenders and descenders of some letters. Since each word of the main body can have a different height, the original image is divided into segments surround by wide blank spaces in the same way that it was previously explained on the slope correction. Then, upper and lower lines of each segment are computed. To achieve this, the Run-Length Smoothing Algorithm (RLSA) [?] is applied and upper and lower contours for each segment are detected. Then, eigenvector line fitting algorithm [?] is applied to the contour points to compute upper and lower baselines. The lines separate the whole main text body from the zones including ascenders and descenders.

In order to obtain an uniform text line, it is necessary to set the same normalization height for all the main body segments. To this end, we compute the average of the size of the main body of all the segments an linearly scale the main body heights of each segment to this value. Finally, the ascenders and descenders are linearly scaled in height to a size determined as an empirically chosen percentage of the new main vertical size (30% for ascenders and 15% for descenders). Since these zones are often huge blank areas, this scaling operation has the effect of filtering out most of the uninformative image background. It also accounts for the large height variability of the ascenders and descenders as compared with that of the main text body.

1.2.2 Feature Extraction

As with the on-line system, off-line system is also based on HMMs, so each preprocessed text line image has to be represented as a sequence of feature vectors. The approach used in this work follows the ideas described in [?].

In summary, a grid is applied to divide the text line image in squared cells. And for every cell the following features are computed:

- Normalized gray level

- Horizontal gray level derivative
- Vertical gray level derivative

The normalized gray level provides information about the trace density on the analyzed cell. On the other hand, the derivatives give information about how this density varies along different directions. To compute the normalized gray level, the analysis window is smoothed by convolution with a two-dimensional gaussian filter.

The horizontal derivative is calculated as the slope of the line which best fits the horizontal function of column-average gray level. The fitting criterion is the sum of squared errors weighted by a one-dimensional gaussian filter which enhances the role of central pixels of the window under analysis. The vertical derivative is computed in a similar way.

1.2.3 Recognition

This module follows almost the same schemes presented in on-line recognition section.

As in the on-line case, we use left-to-right continuous density character HMMs with a mixture of Gaussian densities assigned to each state. However, instead of using a variable number of states for all HMMs, in this case it is fixed. The best parameter values were empirically tuned.

Chapter 2

Computer Assisted Transcription of Handwritten Text Images Overview

In this chapter the existing interactive pattern recognition (IPR) framework presented in [?] is applied to HTR. As discussed above, rather than full automation, we opted for a system in which the user works with the system to obtain a valid transcription. This framework, combines the efficiency of the automatic handwriting recognition system with the accuracy of the human transcriptor, integrating the human activity into the recognition process and taking advantage of the user's feedback.

The CATTI (Computer Assisted Transcription of Handwritten Text Images) system follows the same architecture used on the previous chapter, composed of four modules: preprocessing, feature extraction, training and recognition. The main difference is that the recognition module used must be adapted to cope with the user feedback.

Depending on the mode in which the user interacts with the system, two interaction modes can be defined:

Keyboard and mouse interaction: In this scenario, the user interacts with the system using keystrokes and mouse actions. Two different levels of interaction, whole-word and character level, has been studied using this traditional peripherals in previous works [?, ?].

Multimodal interaction: In this scenario, a more comfortable way of interaction is proposed. Here the user can make corrections using a touch-screen or an e–pen. However, this method of interaction results in indeterminism, since it is necessary to decode the user input through an additional on-line subsystem. For this multimodal approach only whole-word interactions have been studios until now.

In this work we are going to study the multimodal interaction scenario at character level. Now, we describe in detail the first three scenarios (CATTI at whole-word level, CATTI at character level and MM-CATTI at whole-word level) to understand the context of this work.

2.1 CATTI

In the original CATTI framework, the human transcriber is directly involved in the transcription process since he or she is responsible of validating and/or correcting the HTR outputs. The process starts when the HTR system proposes a full transcription of a feature vector sequence x, extracted from a handwritten text line image. The user validates an initial part of this transcription, p', which is error-free and introduces a correct word, v, thereby producing correct transcription *prefix*, $p = p'v$. Then, the HTR system takes into account the available information to suggest a new suitable continuation *suffix*. This process is repeated until a full correct transcription of x is accepted by the user [?] (see Tab. ??).

At each step of this process, both the image representation, x, and a correct transcription prefix p are available and the HTR system should try to complete this prefix by searching for the most likely suffix $\hat{s}$ as:

$$\hat{s} = \arg\max_{s} P(s \mid x, p) = \arg\max_{s} P(x \mid p, s) \cdot P(s \mid p) \tag{2.1}$$

Since the concatenation of p and s constitutes a full transcription hypothesis, $P(x \mid p, s)$ can be approximated by concatenated character Hidden Markov Models (HMMs) [?, ?] as in conventional HTR.

Perhaps the simplest way to deal with $P(s \mid p)$ is to adapt a n-gram language model to cope with the increasingly consolidated prefixes [?]. If k and l are, respectively, the lengths of p and s and $p = p_1^k$ and $s = s_1^l$:

$$P(s \mid p) \simeq \prod_{j=1}^{n-1} P(s_j \mid p_{k-n+1+j}^{k}, s_1^{j-1}) \cdot \prod_{j=n}^{l} P(s_j \mid s_{j-n+1}^{j-1}) \tag{2.2}$$

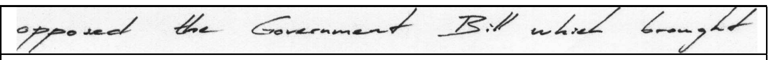
		x	
INTER-0	Step 1	v	
	Step 2	$p'v$ $\hat{s} \equiv \hat{w}$	opposite the Comment Bill in that thought
INTER-1	Step 1	v	opposed
	Step 2	$p'v$ $\hat{s} \equiv s'$	opposed the Government Bill in that thought
INTER-2	Step 1	v	which
	Step 2	$p'v$ $\hat{s} \equiv s'$	opposite the Government Bill which brought
Final		p ≡ T	opposed the Government Bill which brought

Table 2.1 — Example of CATTI whole-word interaction to transcribe an image of the sentence *"opposed the Government Bill which brought"*. The process starts when the HTR system proposes a full transcription of the handwritten text image x. Then, each interaction consists in two steps. In the first step the user type a word to amend the suffix proposed by the system in the previous step. This defines a correct prefix $p'v$. On the second step, using the consolidated prefix, the system proposes a new potential suffix. The process ends when the user enters the special character '#'.

The first term of (**??**) accounts for the probability of the $n-1$ words of the suffix, whose probability is conditioned by words from the validated prefix, and the second one is the usual n-gram probability for the rest of the words int the suffix.

Two searching approaches were presented to generate a suitable continuation in each step of the interaction process. The first one is base on the well known Viterbi algorithm [**?**] and the other is based on word-graph techniques similar to those described in [**?**] for Computer Assisted Translation and for multimodal speech post-editing.

The results show that the computational cost of the second one is much lower than using the naive Viterbi adaptation, at the expense of a moderate accuracy degradation. Therefore, using word-graph techniques the system is able to interact with the human transcriptor in a time-efficient way.

2.2 CATTI at Character Level

In order to make the system more ergonomic and friendly to the user, interaction based on characters (rather than full words) has been also studied [?] with encouraging results. Now, as soon as the user types a new keystroke (character), the system proposes a suitable continuation following the same process described in the previous section.

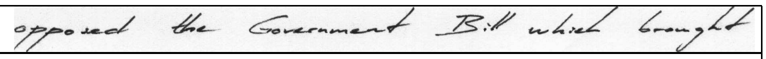
		x	
INTER-0	Step 1	c	
	Step 2	$p = p'c$ $\hat{s} \equiv \hat{w}$	opposite the Comment Bill in that thought
INTER-1	Step 1	v	e
	Step 2	$p = p'c$ $\hat{s} \equiv s'$	oppose d the Government Bill in that thought
INTER-2	Step 1	v	w
	Step 2	$p = p'c$ $\hat{s} \equiv s'$	opposed the Government Bill w hich brought
Final		p $\equiv$ T	opposed the Government Bill which brought

Table 2.2 — Example of CATTI at character level interaction to transcribe an image of the sentence *"opposed the Government Bill which brought"*. The process starts when the HTR system proposes a full transcription of the handwritten text image x. Then, each interaction consists in two steps. In the first step the user type a character (c) in order to amend the suffix proposed by the system in the previous step. This defines a correct prefix $p = p'c$. On the second step, using the consolidated prefix, the system proposes a new potential suffix. The process ends when the user enters the special character '#'.

As the user operates now at character level, the last word of the prefix may be incomplete. In order to *autocomplete* this last word, it is assumed that the prefix p is divided into two parts: the fragment of the prefix formed by complete words (p'') and the last incomplete word of the prefix (v_p). In Tab.?? we can see an example of this interaction. For example, in INTER-1 p'' is empty and $v_p =$"oppose".

In this case the HTR decoder has to take into account x, p'' and v_p, in order to search for a transcription suffix $\hat{s}$, whose first part is the continuation of v_p:

$$\hat{s} = \arg\max_s P(s \mid x, p'', v_p) = \arg\max_s P(x \mid p'', v_p, s) \cdot P(s \mid p'', v_p) \tag{2.3}$$

Again, the concatenation of p'', v_p and s constitutes a full transcription hypothesis and $P(x \mid p'', v_p, s)$, can be modelled with HMMs. On the other hand, to model

$P(s \mid p'', v_p)$ we assume that the suffix s is divided into two fragments: v_s and s''. v_s is the first part of the suffix that corresponds with the final part of the incomplete word of the prefix, i.e, $v_p v_s = v$ where v is an existing word in the task dictionary (Σ), and s'' is the rest of the suffix. So, the search must be performed over all possible suffixes s of p, and the language model probability $P(v_s, s'' \mid p'', v_p)$ must ensure that the concatenation of the last part of the prefix v_p, and the first part of the suffix, v_s, form an existing word (v) in the task dictionary. This probability can be decomposed into two terms:

$$P(v_s, s'' \mid p'', v_p) = P(s'' \mid p'', v_p, v_s) \cdot P(v_s \mid p'', v_p) \tag{2.4}$$

The first term accounts for the probability of all the whole-words in the suffix, and can be modelled directly by (**??**). The second term should ensure that the first part of the suffix (usually a word-ending-part) v_s, will be a possible suffix of the incomplete word v_p, and can be stated as:

$$P(v_s \mid p'', v_p) = \begin{cases} \frac{P(v_p, v_s \mid p'')}{\sum_{v'_s} P(v_p, v'_s \mid p'')} & \text{if } v_p v_s \in \Sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

2.3 Multimodal CATTI

The goal of making the interaction friendlier to the user led the develop of multimodal CATTI (MM-CATTI) [**?**]. As discussed previously, using more ergonomic multimodal interfaces should result in an easier and more comfortable human-machine interaction, at the expense of the feedback being less deterministic to the system.

This is the idea underlying MM-CATTI, which focus on touch-screens, perhaps the most natural modality to provide feedback in CATTI systems. It is worth noting that the use of this kind of more ergonomic feedback comes at the cost of additional interactions steps needed to correct possible feedback decoding errors. Therefore, solving the multimodal interaction problem amounts to achieving a modality synergy where both main and feedback data streams help each other to optimize the system's performance.

Due to the similarity in the mathematical formulation and being beyond the scope of this work, we will not explain the formal framework underlying the multimodal interaction at whole-word.

Chapter 3

Multimodal CATTI at Character Level

Until now, human multimodal feedback has been assumed to come in the form of whole-word interactions. As we have previously concluded, it is necessary make the interaction with the system as easier and ergonomic as possible.

One way to increase the ergonomy and the usability in the system can be achieved through the use of character-level pen-strokes interactions. In this section this new level of interaction is presented. Now, as soon as the user introduces a new pen-stroke, the system proposes a full transcription. However, as we mentioned earlier, the use of this more ergonomic interfaces comes at the cost of new steps needed to correct possible feedback decoding errors.

In the new interaction method presented in [**?**], the main system is the same presented before, however it is necessary to introduce an on-line HTR subsystem in charge of processing and decoding user feedback provided in form of pen-strokes. This feedback recognition module must be adapted to take advantage of the interaction-derived information to boost its accuracy. On Fig.**??** we can see a schematic view of MM-CATTI system presented in this chapter.

Next, the formal framework of MM-CATTI at character level will be explained.

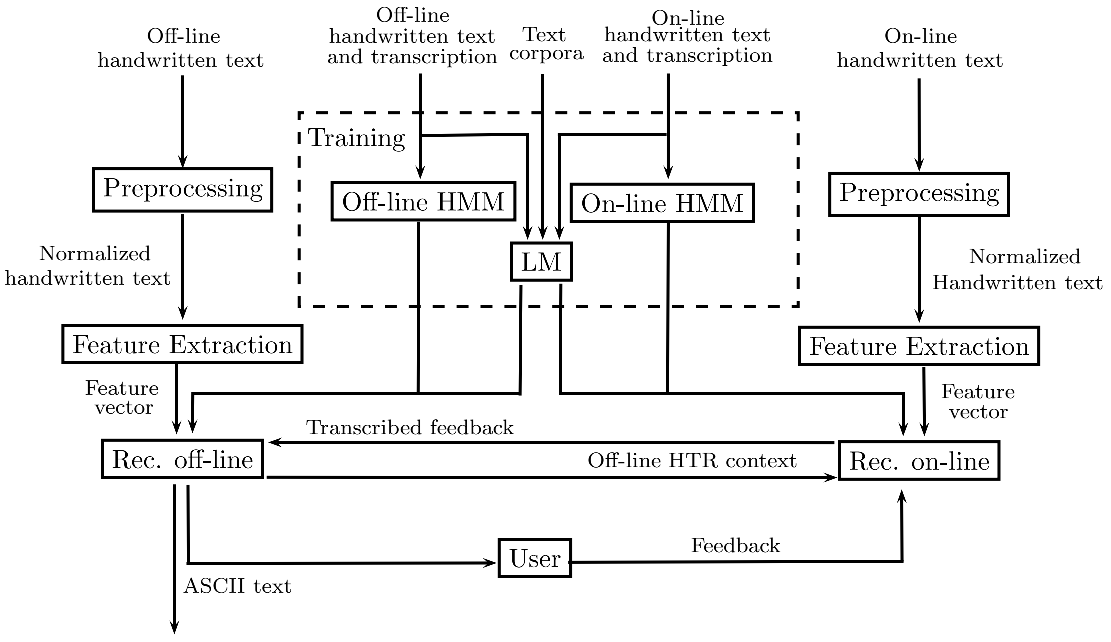


Fig. 3.1 — Overview of the MM-CATTI at Character Level system.

3.1 Formal Framework

Let x be the representation of the input image and p' an user-validated error-free prefix of the transcription. Let t be the on-line touchscreen pen-strokes provided by the user. These data are related to the suffix suggested by the system in the previous interaction step, s', and are typically aimed at accepting or correcting parts of this suffix. Using this information, the system has to find a new suffix, $\hat{s}$, as a continuation of the previous prefix p', considering all possible decodings, d, of the on-line data t and some information from the previous suffix s'. That is, the problem is to find $\hat{s}$ given x and a feedback information composed of p' and t, considering all possible decodings of the on-line data.

$$\begin{aligned}\hat{s} &= \arg\max_{s} P(s \mid x, s', p', t) = \arg\max_{s} \sum_{d} P(s, d \mid x, p', s', t) \\ &\approx \arg\max_{s} \max_{d} P(t \mid d) \cdot P(d \mid p', s') \cdot P(x \mid s, p', d) \cdot P(s \mid p', d) \qquad (3.1)\end{aligned}$$

This equation can be solved with an approximation with two steps. In the first one (formed by the first two terms of (**??**)) the system computes an *"optimal"* decoding ($\hat{d}$) of the on-line pen-strokes t provided by the user. If the online system does not properly recognize the pen-strokes the user can type the character with the keyboard.

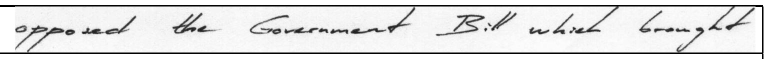
		x						
INTER-0	Step 1	t						
		d						
		κ						
	Step 2	$p'\hat{d}$						
		$\hat{s} \equiv \hat{w}$	opposite	this	Comment	Bill	in that	thought
INTER-1	Step 1	t	e					
		d	a					
		κ	`e`					
	Step 2	$p'\hat{d}$	oppose					
		$\hat{s} \equiv s'$	d	the	Government	Bill	in that	thought
INTER-2	Step 1	t					ω	
		d					w	
		κ						
	Step 2	$p'\hat{d}$	opposed	the	Government	Bill	w	
		$\hat{s} \equiv s'$					hich	brought
Final		κ						`#`
		p $\equiv$ T	opposed	the	Government	Bill	which	brought

Table 3.1 — Example of multimodal CATTI at character level interaction to transcribe an image of the sentence *"opposed the Government Bill which brought"*. The process starts when the HTR system proposes a full transcription of the handwritten text image x. Then, each interaction consists in two steps. In the first step the user handwrites some touchscreen to amend the suffix proposed by the system in the previous step. This defines a correct prefix p', which can be used by the on-line HTR subsystem to obtain a decoding of t. After observing this decoding, d, the user may type additional keystrokes, κ, to correct possible errors in d. On the second step, a new prefix is built from the previous correct prefix p' concatenated with $\hat{d}$ (the decoded on-line handwritten text d or the typed text κ). Using this information, the system proposes a new potential suffix. The process ends when the user enters the special character "#'. Typed text is printed in typewriter font.

In the second step, once the user feedback is available and correctly transcribed, d is replaced with $\hat{d}$ in the last two terms. This way, a new consolidated prefix $p = p'\hat{d}$ is obtained, which leads to a formulation identical to (**??**). After that, the system proposes a new suffix s given this new prefix p and the image x. These two steps are repeated until p is accepted by the user as a full correct transcription of x.

An example of this two-step process is shown in Tab **??**. In this example we assume that the user always prefers handwriting interaction, leaving the keyboard to correct only eventual on-line text decoding errors.

Since the last two terms of (**??**) are explained extensively (and besides they are not a fundamental part in this work) in [**?**], we focus now on the first two terms. As in Chapter 2, the prefix p' is divided into two parts: p'' (fragment of p' consisting of whole words) and v'_p (the last incomplete word of p'). Therefore the first step of the optimization (**??**) can be rewritten as:

$$\hat{d} = \arg\max_{d} P(t \mid d) \cdot P(d \mid p'', v'_p, s') \tag{3.2}$$

where, $P(t \mid d)$ is provided by a morphological (HMM) model of the character d and $P(d \mid p'', v'_p, s')$ can be approached by a language model dynamically constrained by information derived from the input image x and the validated prefix (both p'' and v'_p).

Equation (**??**) may lead to several scenarios depending on the assumptions and constraints adopted for $P(d \mid p'', v'_p, s')$. We examine some of them bellow.

The first and simplest scenario corresponds to a naive approach where any kind of interaction-derived information is ignored; that is, $P(d \mid p'', v'_p, s') \equiv P(d)$. This scenario will be used as *baseline*.

In a slighty more restricted scenario, we take into account just the information from the previous off-line HTR prediction s'. The user interacts providing t in order to correct the wrong character of s', e, that follows the validated prefix p'. Clearly, the erroneous character e should be prevented to be a decoding on-line HTR result. This *error-conditioned model* can be written as $P(d \mid p'', v'_p, s') \equiv P(d \mid e)$.

Another, more restrictive scenario, using the information derived from the validated prefix p', arises when we regard the portion of word already validated (v'_p), i.e. $P(d \mid p'', v'_p, s') \equiv P(d \mid v'_p, e)$. In this case the decoding should be *easier* as we know beforehand what should be a suitable continuation of the part of word accepted so far.

Finally, the most restrictive scenario arises when considering the whole information in p' (the set of complete words p'' as well as the last incomplete word v'_p) and e. This scenario can be written as $P(d \mid p'', v'_p, s') \equiv P(d \mid p'', v'_p, e)$.

All these models will be studied in the next section.

3.2 Dynamic Language Modelling

Language model restrictions are implemented on the base of n-grams. As we mentioned earlier, the *baseline* scenario does not take into account any information derived from the interaction. In this case, since only one character is recognized $P(d)$ can be modelled directly using uni-grams.

The second case, $P(d \mid e)$, only considers the first wrong character. The language model probability can be seen in (**??**). As we can see, it uses a uni-gram model like the previous one. But in this case, it avoids to recognize the wrong character modifying the probability for that character.

$$P(d \mid e) = \begin{cases} 0 & \text{if } d = e \\ \frac{P(d)}{1-P(e)} & \text{if } d \neq e \end{cases} \tag{3.3}$$

In the next scenario, given by $P(d \mid v'_p, e)$, the on-line HTR subsystem counts not only on the first wrong character but also on the last incomplete word of the validated prefix v'_p. This scenario can be approached in two different ways: using a character language model or a word language model.

The first one uses a modified character n-gram model conditioned by the chunk of word (v'_p) and the erroneous character (e).

$$P(d \mid v'_p, e) = \begin{cases} 0 & \text{if } d = e \\ \frac{P(d|{v'_p}^k_{k-n+2})}{1-P(e|{v'_p}^k_{k-n+2})} & \text{if } d \neq e \end{cases} \tag{3.4}$$

In the second approach (**??**), we use a word language model to generate a more refined character language model. This can be written as:

$$P(d \mid v'_p, e) = \begin{cases} 0 & \text{if } d = e \\ \frac{P(d|v'_p)}{1-P(e|v'_p)} & \text{if } d \neq e \end{cases}$$

where:

$$P(d \mid v'_p) = \frac{P(v'_p, d)}{\sum\limits_{d'} P(d', v'_p)} = \frac{\sum\limits_{v_s} P(v'_p, d, v_s)}{\sum\limits_{v_s} \sum\limits_{d'} P(v'_p, d', v_s)} \tag{3.5}$$

being $v'_p d v_s$ an existing word of Σ.

Finally, the last scenario uses all available information; i.e., the erroneous character e, the last incomplete word of the prefix v'_p and the rest of the prefix p''. This can be written as

$$P(d|p'', v'_p, e) = \begin{cases} 0 & \text{if } d = e \\ \frac{P(d|v'_p, p''^{k}_{k-n+2})}{1-P(e|v'_p, p^{k}_{k-n+2})} & \text{if } d \neq e \end{cases} \tag{3.6}$$

Using a word n-gram model we can generate a more refined character language model as in the previous case, so:

$$P(d|v'_p, p''^{k}_{k-n+2}) = \frac{P(d, v'_p, p''^{k}_{k-n+2})}{\sum\limits_{d'} P(d', v'_p, p''^{k}_{k-n+2})} = \frac{\sum\limits_{v_s} P(v'_p, d, v_s \mid p''^{k}_{k-n+2})}{\sum\limits_{v_s}\sum\limits_{d'} P(v'_p, d', v_s \mid p''^{k}_{k-n+2})} \tag{3.7}$$

Now we will explain how implement the restrictions of each scenario. The first one uses a regular uni-gram character model since it does not impose any restriction on the model.

The second scenario, $P(d \mid e)$ can also be implemented using an uni-gram character model, but in this case, it is necessary to disable the character transition edge to e, since we know that this character should not be recognized.

The first approach of $P(d \mid v'_p, e)$ has been implemented using bi-gram character model conditioned with the chunk of word v'_p, also disabling the transition edge to e as has been done before.

For ease explanation, the second approach is shown in Figure **??**. In this example, $v'_p =$"in" and the user wants to correct the wrong off-line recognized character 'g', by handwriting the character 's'. In this case, the online HTR subsystem uses an uni-gram character model created from another word uni-gram model. This can be done taking into account the probability of the words that begin as the prefix "in". In addition, the character transition edge 'g' is disable, to avoid recognizing the character we already know that can not be.

The last scenario, $P(d|p'', v'', e)$, is similar to the previous one. In this case, we used a bi-gram word model conditioned by the prefix words p''. After that, we followed a process similar to the third scenario deriving a character model from the previous one.

As in CATTI searching [?], owing to the finite-state nature of the n-gram language model, the search in all scenarios can be performed efficiently using the Viterbi algorithm.

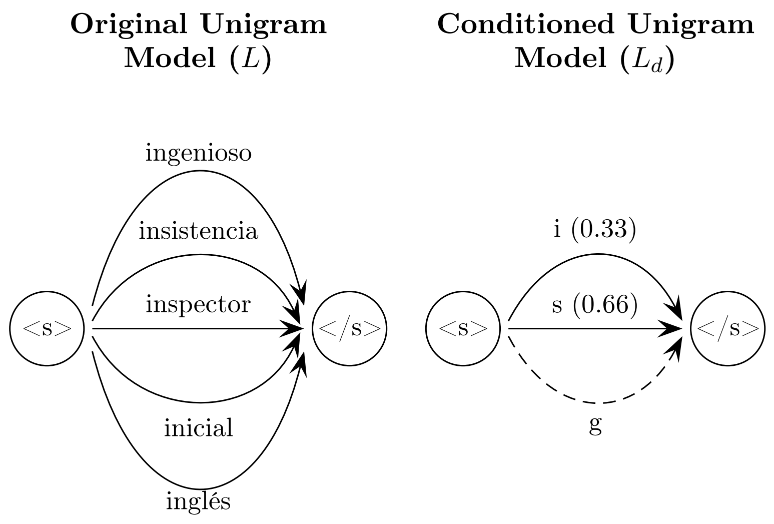


Fig. 3.2 — Example of the dynamic unigram language model generation of $P(d \mid v'_p, e)$ using the second approach. L is an equiprobable unigram model example, whereas L_d is the unigram sub-model derived from L. This simplified language model is used by the on-line HTR sub-system to recognize the touchscreen handwritten character '`s`', intended to replace the wrong off-line recognized character '`g`', which has been disabled in L_d.

Chapter 4

Experimental Framework

The experimental framework developed for test the effectiveness of the MM-CATTI at Character Level is described in the following sections. First of all, we will define the corpora used in the experiments. Then, the assessment measures and the parameters to be adjusted will be defined.

4.1 Corpora

Performance evaluation of computer assisted transcription often involves providing test data and comparing the process output with the expected output. To evaluate our work we used two publicly available corpora (an off-line and an on-line).

The first one, IAMDB corpus, was used to test the off-line HTR subsystem in charge of decoding pictures of handwritten sentences. The second, called UNIPEN corpus, was employed to simulate the user touchscreen feedback.

4.1.1 IAMDB Corpus

The first of the corpora, IAMDB [?], was compiled by the Research Group of Vision and Artificial Intelligence (FKI) at Institute of Computer Science an Applied Mathematics (IAM) in Bern (Switzerland). It is publicly accessible and freely available upon request for non-commercial research purposes. The IAMDB images corresponds to handwritten texts copied from the Lancaster-Oslo/Bergen (LOB) corpus [?][?], which consists of about 500 printed electronic English texts of about 2,000 words each. The text in the LOB corpus consists of many categories (e.g. editorial, religion, nature, science fiction).

The IAMDB database, currently at version 3.0, is composed of 1,539 scanned text pages, handwritten by 657 different writers. No restriction was imposed related to the writing style or with respect to the pen used. The database is provided at different segmentation levels: characters, words, lines, sentences and page images. For this test, sentence-segmented images were used. Each sentence or line image is accompanied by its ground through transcription as the corresponding sequence of words. To better focus on the essential issues of the considered problems, no punctuation marks, diacritics, or different word capitalizations are included in the transcriptions. From 2,324 sentences that forms the corpus, 200 were used as test, leaving the rest as training partition.

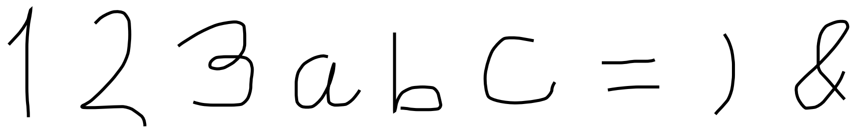


Fig. 4.1 — Examples from the categories 1a, 1c and 1d in the UNIPEN corpus.

4.1.2 UNIPEN corpus

The UNIPEN corpus [?] is an English publicly available on-line HTR database. It comes organized in several categories: lower and upper-case letters, digits, symbols, isolated words and full sentences. For our experiment, were used three UNIPEN categories: *1a* (digits), *1c* (lowercase letters) and *1d* (symbols). Some character examples from these categories are shown in Fig. **??**. To train the system 17 different writers were used. Moreover, test data were produced using three writers [?]. The distribution of data between train and test partitions ca be seen in Table **??**.

	Train	Test	Lexicon
writers	17	3	-
Lowercase Letters	12,298	2,771	26
Symbols	2,431	541	21
Digits	1,301	234	10
Total	16,030	3,546	57

Table 4.1 — Statistics of the UNIPEN training and test dataset used in the experiments.

4.2 Assessment Measures

Some types of measures have been adopted to assess the performance of character-level transcription. In order to make the post-editing process more accurately comparable to CATTI at character level, we introduce a *post-editing autocompleting* approach. Here, when the user enters a character to correct some incorrect word, the system automatically completes the word with the most probable word on the task vocabulary. Hence we define the *Post-editing Key Stroke Ratio* (PKSR), as the number of keystrokes that the user must enter to achieve the reference transcription, divided by the total number of reference characters. On the other hand, the effort needed by a human transcriber to produce correct transcriptions using CATTI at character level is estimated by the *Key Stroke Ratio* (KSR), which can be also computed using the reference transcriptions. The KSR can be defined as the number of (character level) user interactions that are necessary to achieve the reference transcription of the text image considered, divided by the total number of reference characters.

These definitions make PKSR and KSR comparable and the relative difference between them gives us a good estimate of the reduction in human effort that can be achieved by using CATTI at character level with respect to using a conventional HTR system followed by human autocompleting postediting. This *estimated effort reduction* will be denoted as “EFR”.

Finally, since only single-character corrections are considered, the conventional classification error rate (ER) will be used to assess the accuracy of the on-line HTR feedback subsystem under the different constraints entailed by the MM-CATTI interaction process.

4.3 Parameters

As in all classification systems some parameters need to be adjusted. In particular in this system the parameters to tune are the grammar scale factor (GSF) and the number of Gaussian densities in each state of the HMM.

4.4 Results

The aim of these experiments is to measure the effectiveness of MM-CATTI at Character level in the different approaches we have outlined in Sec. ??. As we mentioned earlier, the introduction of multimodal interactivity leads, on the one hand, to an ergonomic and easier way of working, but on the other hand, to a situation where the system has to deal with non-deterministic feedback signals. Therefore, two of the most important concerns here is the accuracy of the on-line HTR subsystem in charge of decoding the human feedback and the determination of how much this accuracy can be boosted by taking into account informations derived from the interaction process.

Class	Number of Characters
Lowercase Letters	4,743
Numbers	123
Symbols	15
Total	4,881

Table 4.2 — Statistics of the test set.

Table ?? shows the basic statistics of the data used in the experiment. The different characters used in this test were chosen as follows: for every mistranscribed character during the off-line classification process of IAMDB database, we chose three random samples (one sample for each test writer) of the mistaken character from the UNIPEN corpus. Thus, we can simulate the interaction made by the user to correct the wrong recognized character.

Once the test corpus was available, we carried out some experiments . For each scenario, we have tested values of GSF=(5,10,15,20,25) and N_G=(16,32,64,128,256).

On the Fig. ?? we can see the Error Rate obtained as a function of the parameter GSF and for different number of gaussian densities in the plain character uni-gram scenario $P(d)$, (CU). The best result is 7% and is obtained at GSF=15 and N_G=32.

Then, on the Fig. ?? we can see the evolution in error rate as we change GSF and N_G in the second scenario $P(d \mid e)$, (CU_e). The best result is 6.9% and is obtained at GSF=15 and N_G=32.

The Fig. ?? & ?? show the different values of the error rate as we modify GSF and N_G in the two approaches of the third scenario $P(d \mid v'_p, e)$ (CB_e & WU_e). The best result is equal to 5.0% and is achieved in the second approach with GSF=15

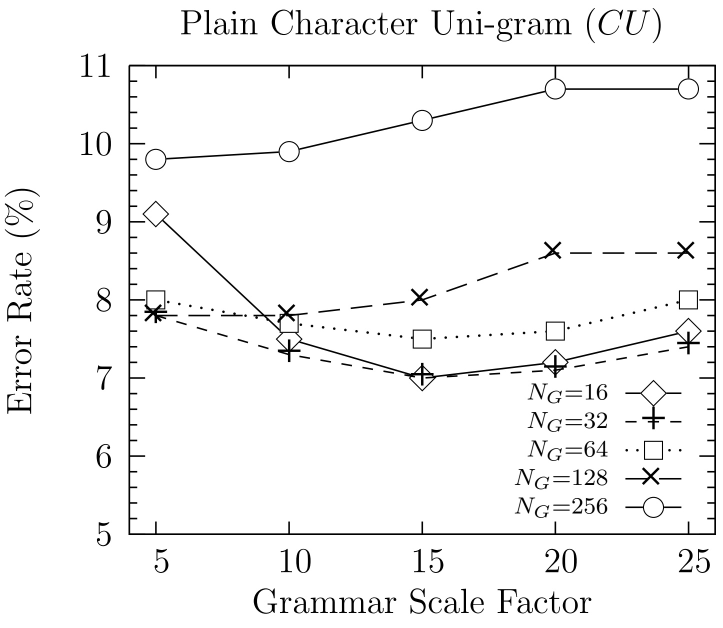


Fig. 4.2 — Error rate for different values of the parameter GSF and varying number of gaussian densities used in each state of the HMM on the plain character uni-gram scenario (CU).

and N$_G$=32.

And then, in Fig. **??** we can see the error rate (%) obtained as a function of GSF and N$_G$ for the last scenario (WB$_e$). The best result achieved is 4.3% with GSF=15 and N$_G$=64.

Table **??** shows the summary of the results obtained in the different scenarios. In addition, Tab. **??** also shows the relative accuracy improvements obtained of each scenario respect to the baseline case. Two types of results are reported for CATTI at character level: the PKSR (first column) and the KSR (second column). The 12.5% of KSR corresponds to a total of 1,627 characters that the user has to correct. In the MM-CATTI at character level these characters would have to be handwritten by the user on the touchscreen. It is simulated here using character samples belonging to a same writer from the UNIPEN corpus.

As a final overview, Table **??** compares the results of CATTI and MM-CATTI. The first column shows the PKSR achieved with the post-editing autocompleting approach. The second one, shows the KSR of CATTI at character level. The third and fourth columns show the MM-CATTI WSR for the baseline as well as

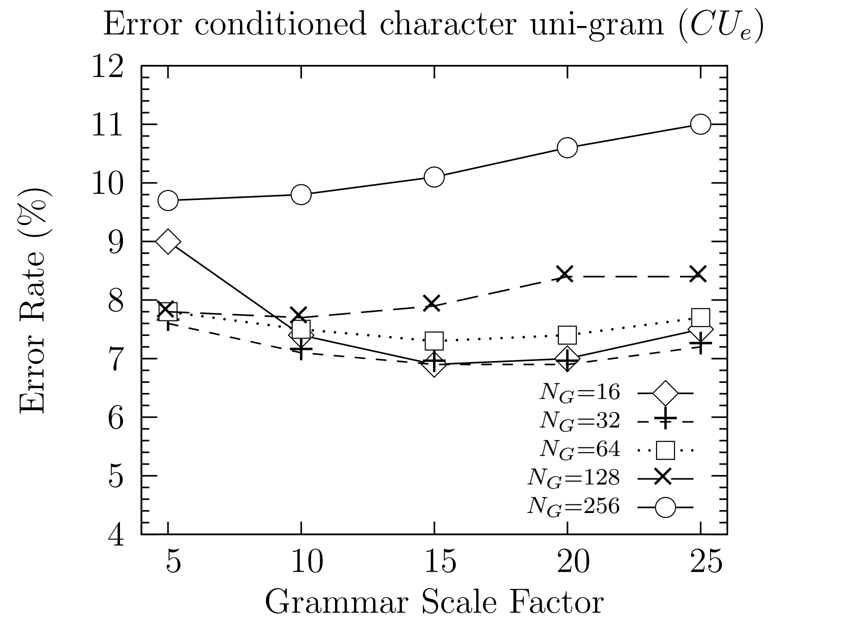


Fig. 4.3 — Error rate for different values of the parameter GSF and varying number of gaussian densities used in each state of the HMM on the plain character uni-gram scenario (CU$_e$).

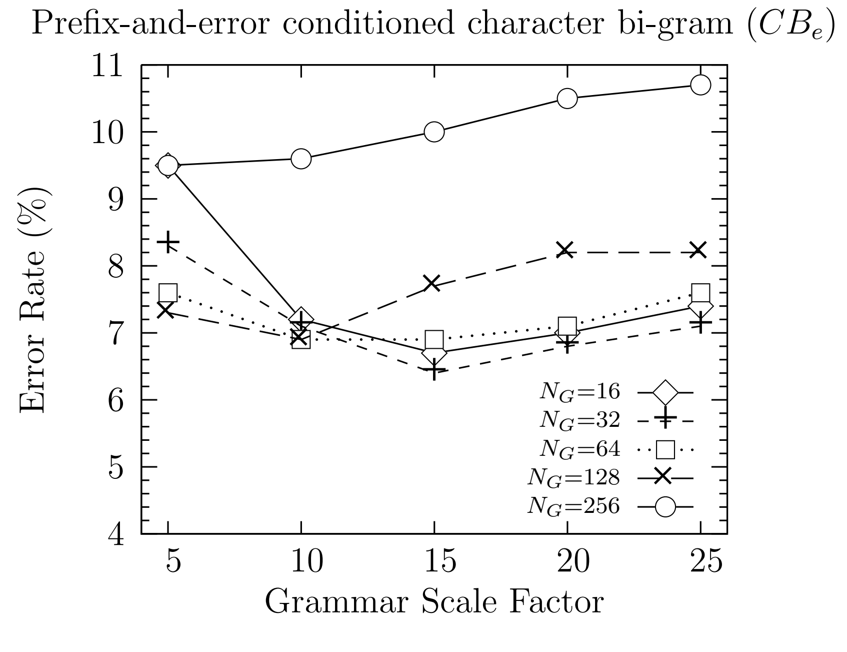


Fig. 4.4 — Error rate for different values of the parameter GSF and varying number of gaussian densities used in each state of the HMM on prefix-and-error conditioned character bi-gram (CB$_e$).

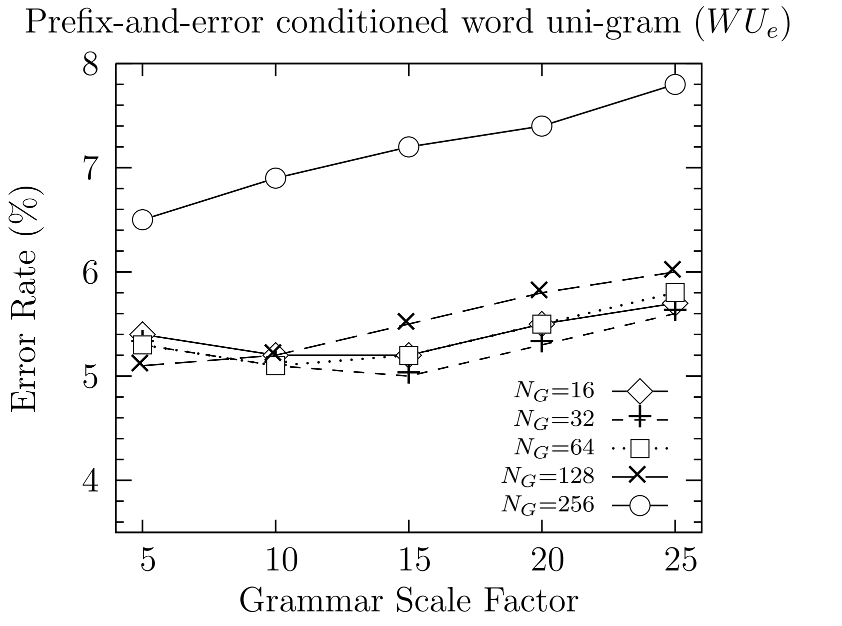


Fig. 4.5 — Error ratio for prefix-and-error word uni-gram model (WU$_e$).

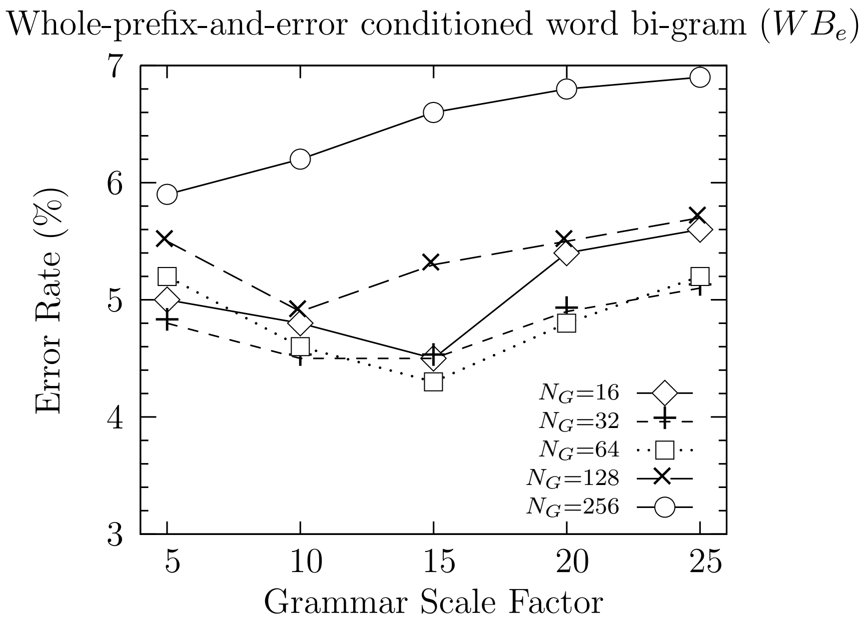


Fig. 4.6 — Error rate for the most informed scenario ($P(d \mid p'', v'', e)$).

Error Rate					Relative Improvement			
CU	CU$_e$	CB$_e$	WU$_e$	WB$_e$	CU$_e$	CB$_e$	WU$_e$	WB$_e$
7.0	6.9	6.7	5.0	4.3	1.4	4.3	28.6	38.6

Table 4.3 — Summary of results for the five different scenarios proposed for the on-line HTR subsystem: plain character uni-gram (CU, *baseline*), error conditioned character uni-gram (CU$_e$), prefix-and-error conditioned character bi-gram (CB$_e$), prefix-and-error conditioned word uni-gram (WU$_e$) and whole-prefix-and-error conditioned word bi-gram (WB$_e$). For each scenario, a total of 4,881 tests were made. These test samples were obtained from the mistakes of the off-line system. The relative accuracy improvements for CU$_e$, CB$_e$, WU$_e$ WB$_e$ are shown in the last four columns. All values are in percentages.

PKSR	CATTI	MM-CATTI		EFR		
		KSR			MM-CATTI	
	KSR	CU	WB$_e$	CATTI	CU	WB$_e$
15.8	12.5	13.4	13.0	20.9	15.2	17.7

Table 4.4 — From left to right: PKSR obtained with the post-editing autocompleting approach on the HTR system, KSR achieved with CATTI at character level and KSR obtained with the *baseline* (CU) and best (WB$_e$) scenarios for MM-CATTI approach. EFR for KSR of CATTI with respect to PKSR and for KSR for the two scenarios of MM-CATTI with respect to PKSR. All results are in percentages.

the best scenarios. This values are calculated under the simplifying assumption that the cost of keyboard-correcting a feedback on-line decoding error is similar to that of another on-line touchscreen interaction step. That is, each correction is counted twice: one for the failed touchscreen attempt and another for the keyboard correction itself. According to these results, the expected user effort for the best MM-CATTI approach is only barely higher than that of CATTI.

According to these data, the expected user effort for the MM-CATTI at Character Level is greater than using CATTI. However, the comparison is not entirely realistic since the human effort in MM-CATTI is much more friendlier and comfortable that than in CATTI.

Finally, the best classification error rate obtained for isolated digits, letters and symbols are 47.9%, 3.1% and 20.0%, respectively, with an average of 4.3%. This results are comparable with those of the state-of-the-art obtained for this dataset. For example, in [?] classification error rate of 1.5% and 6% are report for isolated digits and letters, respectively, by using Support Vector Machines. Moreover in [?], using neural networks, ER of 3% and 14% for digits and letters are presented. Finally in [?], an online scanning n-tuple classifier system is used for classifying isolated digits, letters and symbols, obtaining in this case ER of 1.1%, 7.6% and 20.4% respectively.

Although at first glance it may appear that our results are worse than those shown in the cited articles that is not correct. Since, they classify each category (digits, letters and symbols) separately, when we do it together. Thus, they get low classification error rates in symbols and numbers, as they can use a very constrained models unlike us, that we have a general model for the three classes (although specialized in characters, which is the most represented class during the training of the language models). Therefore, the correct way would be compare just the results of the character class, where we get much better results than them.

Chapter 5

Future work and conclusions

Although a considerable work has been carried out, there are still many options to explore. These have not been met because of lack of time, but it would be interesting to investigate them. Now, we can see a few proposals:

Explore other options for on-line correction: Currently, the amendments when the online system does not recognize correctly the user's feedback are generated using the keyboard. This assumption was made to compare the results between the CATTI and MM-CATTI systems. As we mentioned at the beginning of this work, one of the most important objectives is to enhance the ergonomics of the system. Therefore, we must explore other options, such as make the corrections with the e-pen. In this case, the user could indicate with a gesture that the character has been badly recognized. Thus, the system would have additional information, not just the erroneous character e but a list of all known bad recognized characters, to facilitate the recognition. On the other hand, since this task is not deterministic, as opposed to κ keystrokes, is necessary to study how many gestures the user needs to enter in order to correct the original feedback.

Conduct more experiments on other corpora: The work has only been tested with the the IAMDB corpus. It would be interesting to user others to verify that the data obtained is reliable. Moreover, it is necessary to carry out formal field tests to assess the correction of the system under real working conditions.

Using Adaptive Learning to improve the recognition of user's feedback: The correct transcriptions that are being generated during the interactive work can be used to dinamically improve the underlying system. That is, at present the system is trained with a characters writers corpus. Once the users starts entering real pen-strokes the system can use it to retrain its models in order to improve the decoding accuracy. In speech recognition, well known Adaptive Learning techniques exist for adapting the acoustic HMM models to the speaker [?, ?].

In conclusion, in this work, we have studied the character level interaction in the MM-CATTI system presented in previous works using pen strokes handwritten on a touchscreen as a complementary means to introduce the required CATTI correction feedback. Here, this feedback is used as a part of a prefix to improve the transcription given by the computer. Thus, the system proposes a new suffix that the user can accept as a final transcription or modify it in an iterative way until a full and correct transcription is finally produced.

Obtained results show that a significant benefit can be obtained. As we have seen, the use of interactivity and multimodality makes the results as good as state-of-the-art results on isolated character recognition. In addition, we observe that the use of this more ergonomic feedback modality comes at the cost of a reasonably small number of additional interaction steps needed to correct the few feedback decoding errors. The number of these extra steps is kept very small thanks to the ability to use interaction-derived constraints to considerably improve the on-line HTR feedback decoding accuracy. Moreover, the advantage of MM-CATTI over traditional HTR followed by post-editing goes beyond the reduction of the human effort. When difficult transcription tasks with high PKSR are considered, experts usually refuse to use post-edit conventional HTR output. In contrast, in our approach, if predictions are not good enough the user can ignore them and continue on their own. However, some predictions may be correct, so the user will save some typing effort. i.e; the system may not always give correct transcriptions, but in contrast never bothers.

Although the character-level results may be slightly worse than at word level that results are not directly comparable. A word-level correction encapsulates fairly well all the cognitive and physical human efforts needed to locate an error and type the correction. However, it is also quite clear that word-level corrections are not comfortable to users, in general. In contrast, character-level corrections are preferred by users, but it is unclear whether only the number of keystrokes can be fairly used for assessment purposes. A corrective pen-stroke generally needs no significant cognitive effort since, in most cases, it is part of the correction of an already detected word error.

Finally, the work presented in this paper has been submitted and accepted [?] in the fifth edition of the Iberian Conference on Pattern Recognition and Image Analysis, IbPRIA 2011. To be held in Las Palmas de Gran Canaria (Spain), June 8-10, 2011. IbPRIA is an international conference co-organised by the Spanish AERFAI and Portuguese APRP chapters of the IAPR International Association for Pattern Recognition.

Bibliography

[Abd04] *Online handwriting recognition using support vector machine*, volume A, 2004.

[AOI98] K. Takeda A. Ogawa and F. Itakura. Balancing acoustic and linguistic probabilites. *Proc. IEEE Conf. Acoustics, Speech, and Signal Processing*, pages 181–184, 1998.

[BPSW70] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Pattern Recognition*, 41(8):164–171, 1970.

[BS89] R. Bozinovic and S. Srihara. Off-line cursive script word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:68–83, 1989.

[BSM99] I. Bazzi, R. Schwartz, and J. Makhoul. An Omnifont Open-Vocabulary OCR System for English and Arabic. *IEEE Trans. on PAMI*, 21(6):495–504, 1999.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Willey & Sons, 1973.

[EYGSS99] A. El-Yacoubi, M. Guilloux, R. Sabourin, and C. Y. Suem. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans. on PAMI*, 21(8):752–760, 1999.

[GLV95] R. Garsid, G. Leech, and T. Váradi. Manual of information of accompany for the lancaster parsed corpus. 1995.

[GSP$^+$94] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. UNIPEN Project of On-Line Data Exchange and Recognizer Benchmarks. In *Proc. of the 14th International Conference on Pattern Recognition*, pages 29–33, Jerusalem (Israel), 1994.

[Gó10] Veronica Romero Gómez. *Multimodal Interactive Transcription of Handwritten Text Images.* PhD thesis, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia (Spain), July 2010. Advisor(s): Prof. E. Vidal and Dr. A.H. Toselli.

[HWW$^+$00] M. P. Harper, C. M. White, W. Wang, M. T. Johnson, and R. A. Helzerman. The effectiveness of corpus-induced dependency grammars for post-processing speech. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 102–109, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[HZK07] B. Q. Huang, Y. B. Zhang, and M. T. Kechadi. Preprocessing techniques for online handwriting recognition. In *ISDA '07: Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, pages 793–800, Washington, DC, USA, 2007. IEEE Computer Society.

[Jel98] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, 1998.

[JLG78] S. Johansson, G. Leech, and H. Goodluck. Manual of information of accompany the lancaster-oslo/bergen corpus of british english, for use with digital. 1978.

[JMW01] S. Jaeger, S. Manke, and A. Waibell. On-line Handwriting Recognition: The NPen++ Recognizer. *International Journal on Document Analysis and Recognition*, 3(3):169–181, 2001.

[KN95] R. Kneser and H. Ney. Improved backing-off for n-gram language modeling. *Proc. of the ICASSP 1995*, 1:181–184, 1995.

[KS06] Ergina Kavallieratou and Efstathios Stamatatos. Improving the quality of degraded document images. In *DIAL '06: Proceedings of the Second International Conference on Document Image Analysis for Libraries (DIAL'06)*, pages 340–349, Washington, DC, USA, 2006. IEEE Computer Society.

[Lee89] K.-F. Lee. *The development of the sphinx system.* Kluwer Academic Publishers, 1989.

[LS06] P. Liu and F. K. Soong. Word graph based speech recognition error correction by handwriting input. In *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces*, pages 339–346, New York, NY, USA, 2006. ACM.

[MARTV11] D. Martín-Albo, V. Romero, A. H. Toselli, and E. Vidal. Character-level interaction in multimodal computer-assisted transcription of text images. In *Pattern Recognition and Image Analysis. 5th Iberian Conference, IbPRIA*, Las Palmas de Gran Canaria, España, 2011. AERFAI and APRP.

[MB99] U.-V. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. In *Proc. of the ICDAR'99*, pages 705–708, Bangalore (India), 1999.

[MB01] U.-V. Marti and H. Bunke. Using a Statistical Language Model to improve the preformance of an HMM-Based Cursive Handwriting Recognition System. *Int. Journal of Pattern Recognition and Artificial In telligence*, 15(1):65–90, 2001.

[PLLG01] Marc Parizeau, Alexandre Lemieux, Re Lemieux, and Christian Gagne. Character recognition experiments using unipen data. In *In Proc. of the 6th ICDAR*, pages 481–485, 2001.

[PMSN01] Michael Pitz, Sirko Molau, Ralf Schlüter, and Hermann Ney. Vocal tract normalization equals linear transformation in cepstral space. In *IN PROC. OF THE EUROSPEECH'01*, pages 2653–2656, 2001.

[PS00] R. Plamondon and S. N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. on PAMI*, 22(1):63–84, 2000.

[PTV04] Moisés Pastor, Alejandro Toselli, and Enrique Vidal. Projection profile based algorithm for slant removal. In *International Conference on Image Analysis and Recognition (ICIAR'04)*, Lecture Notes in Computer Science, pages 183–190, Porto, Portugal, September 2004. Springer-Verlag.

[PTV05] Moises Pastor, Alejandro Toselli, and Enrique Vidal. Writing speed normalization for on-line handwritten text recognition. *Document Analysis and Recognition, International Conference on*, 0:1131–1135, 2005.

[Rab89] L. Rabiner. A Tutorial of Hidden Markov Models and Selected Application in Speech Recognition. *Proc. IEEE*, 77:257–286, 1989.

[Rat03] Eugene H. Ratzlaff. Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database. *Document Analysis and Recognition, International Conference on*, 1:623, 2003.

[RTV10] V. Romero, A. H. Toselli, and E. Vidal. Character-level interaction in computer-assisted transcription of text images. In *International Conference on Frontiers in Handwritten Recognition (ICFHR 2010)*, pages 539–544. Kolkata, India, November 2010.

[T$^+$04] A. H. Toselli et al. Integrated Handwriting Recognition and Interpretation using Finite-State Models. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 18(4):519–539, June 2004.

[Tos04] Alejandro Héctor Toselli. *Reconocimiento de Texto Manuscrito Continuo*. PhD thesis, Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia (Spain), March 2004. Advisor(s): Dr. E. Vidal and Dr. A. Juan (in Spanish).

[TPV07] A. H. Toselli, M. Pastor, and E. Vidal. On-Line Handwriting Recognition System for Tamil Handwritten Characters. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 370–377. Springer-Verlag, June 2007.

[TRPV10] A. H. Toselli, V. Romero, M. Pastor, and E. Vidal. Multimodal interactive transcription of text images. *Pattern Recognition*, 43(5):1814–1825, 2010.

[TRRV07] A. H. Toselli, V. Romero, L. Rodríguez, and E. Vidal. Computer Assisted Transcription of Handwritten Text. In *Proc. 9th ICDAR 2007*, pages 944–948. Curitiba, Paraná (Brazil), 2007.

[TRV08] A. H. Toselli, V. Romero, and E. Vidal. Computer assisted transcription of text images and multimodal interaction. In *Proceedings of the 5th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, volume 4892 of *LNCS*, pages 60–71. Utrech, The Netherlands, June 2008.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260 – 269, April 1967.

[WCW82] K.Y. Wong, R.G. Casey, and F.M. Wahl. Document Analysis System. *IBM J.Res.Devel*, 26(6):647–656, 1982.

[Woo01] Phil C. Woodland. Speaker adaptation for continuous density HMMs: A review. In *ITRW on Adaptation Methods for Speech Recognition*, pages 11–19, August 2001.