Chapter 3
Computer Assisted Transcription of Text Images

With Contribution Of: Verónica Romero and Moisés Pastor.

Contents

Grounded in the interactive–predictive transcription framework drawn in the previous chapter, an interactive approach for efficient transcription of handwritten text images, along with its more ergonomic and multimodal variants are presented. All these approaches, rather than full automation, aim at assisting the expert in the proper transcription process in an efficient way. In this sense, an interactive scenario is stated, where both automatic handwriting recognition system and human transcriber (user) cooperate to produce the final transcription of text-images.

Additionally, an explanation of both basic off- and on-line HTR systems used embedded in the CATTI approaches is given in some detail. This focusing mainly on the preprocessing, feature extraction and on specific aspects of the modeling and decoding-searching process, which complement the ones already introduced in Sect. 2.2.

Moreover, in this chapter, it will be shown how user-interaction feedback directly allows us to improve system accuracy, while multimodality increases system ergonomics and user acceptability. Multimodal interaction is approached in such a way that both the main and the feedback data streams help each-other to optimize overall performance and usability. All these are supported by experimental results obtained on three cursive handwritten tasks suggesting that, using these approaches,

A.H. Toselli et al., *Multimodal Interactive Pattern Recognition and Applications*,
DOI 10.1007/978-0-85729-479-1_3, © Springer-Verlag London Limited 2011

considerable amounts of user effort can be saved with respect to both pure manual work and non-interactive, post-editing processing.

3.1 Computer Assisted Transcription of Text Images: CATTI

So far, the interactive transcription framework, search approaches and assessment measures presented in Sects. 2.3, 2.5 and 2.6, respectively, can be straightforwardly applied to the transcription task of handwritten documents. The application performing this kind of task, resulting from applying all the before-mentioned concepts, shall be called from now on *Computer Assisted Transcription of Text Images* (CATTI) [28, 30].

The CATTI application involves an interactive scenario, where both automatic *handwritten text recognition* system (HTR) and human transcriber (which henceforth will be referred to as the user) cooperate together to produce the final transcription of text-images. During the CATTI process, the user is directly involved in the transcription process since he/she is responsible of validating and/or correcting the HTR output. As illustrated in Fig. 3.1, and following the *left-to-right* protocol for interactive transcription laid down in Sect. 2.3, the process starts when the HTR system proposes a full transcription $\hat{s}$ (or a set of n-best transcriptions) of a given feature vector sequence x, representing a handwritten text line image.[1] Then, the user reads this transcription until he or she finds a mistake; i.e, he or she validates a prefix p' of the transcription which is error-free. Now, the user can enter some keystrokes (letters or whole-words), κ, to correct the erroneous text that follows the validated prefix. This action produces a new prefix p (the previously validated prefix, p' followed by κ). Taking into account this new prefix p, the HTR system suggests a suitable continuation (or a set of the best possible continuations) to this prefix (i.e., a new $\hat{s}$), thereby starting a new cycle. This process is repeated until a correct, full transcription T of x is accepted by the user. A key point of this interactive process is that, at each user-system iteration, the system can take advantage of the prefix validated so far to attempt to improve prediction.

The example shown in Fig. 3.1 illustrates how a 67% *estimated effort reduction* (EFR) is achieved (cf. Sect. 2.6). It is worth nothing that in this example the non-interactive post-editing operation would have required the user to correct *six* errors from the original recognized hypothesis whereas with the interaction feedback, only *two* user-corrections (the red color text in the final transcription T) are necessary to get the final error-free transcription. In spite of Fig. 3.1, which shows an example of interaction-correction at character level, only whole-word correction interactions will be considered in this chapter for the reasons already commented in Sect. 2.3,

Next section is mostly devoted to explain the implementation details of the search techniques based on word-graphs, employed to solve the optimization problem set

[1] For simplicity henceforward, we will refer x directly as the input handwritten text image.

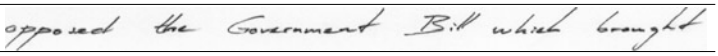
	x						
STEP-0	p						
STEP-1	$\hat{s} \equiv \hat{w}$	opposite	this	Comment	Bill	in that	thought
	p'	oppos					
	κ	ed					
	p	opposed					
STEP-2	$\hat{s}$		the	Government	Bill	in that	thought
	p'	opposed	the	Government	Bill		
	κ					which	
	p	opposed	the	Government	Bill	which	
FINAL	$\hat{s}$						brought
	p'	opposed	the	Government	Bill	which	brought
	κ						
	$p \equiv T$	opposed	the	Government	Bill	which	brought

Fig. 3.1 Example of CATTI interaction to transcribe an image sentence "*opposed the Government Bill which brought*". Initially the prefix p is empty, and the system proposes a complete transcription $\hat{s} \equiv \hat{w}$ (as happens in the normal non-interactive HTR) of the input x. In each interaction step the user reads this transcription, accepting a prefix p' of it. Then, he or she types in some keystrokes, κ, to correct some words of the transcription provided by the system, thereby generating a new prefix p (the accepted one p' plus the text κ added by the user). At this point, the system suggests a suitable continuation $\hat{s}$ of this prefix p and this process is repeated until a complete and correct transcription of the input image is reached. In the final transcription, T, the user-typed text is typeset in different font and *red* color. In this example the correct transcription has six words and the initial hypothesis, $\hat{w}$, has six errors. Therefore, the estimated post-editing effort (WER) is 100%, while in the corresponding interactive estimate (WSR) is 33%, since only two (word) corrections are needed. This results in an estimated effort reduction (EFR) of $100 - 33/100 = 67\%$ (see Sect. 2.6 for definitions of WER, WSR and EFR)

up by Eq. (2.6). Section 3.3 depicts an additional interaction issue, along with the involved theoretical background, which increases the performance of CATTI, mainly, in terms of ergonomics and usability. Section 3.4 introduces and describes the multimodal version of CATTI. Complementing the information already given in Sect. 2.2, a general description of the *off*- and *on-line* text processing systems is given in Sect. 3.5. Application tasks, experimental data and reported results are finally shown in Sect. 3.6.

3.2 CATTI Search Problem

As explained in Sect. 2.5.1, the optimal solution for the search problem set up by Eq. (2.6) is solved by using the Viterbi algorithm on the corresponding finite-state network restricted by a special language model built by the concatenation of a *linear* model (which accounts for the words of the prefix p) and a conventional n-gram model (which models all the possible words of the suffix s). However, given that the direct adaptation of the Viterbi algorithm leads to a computational cost that grows quadratically with the number of words of each sentence, more efficient techniques based on word-graphs (WG) can be used to obtain a linear cost search.

3.2.1 Word-Graph-Based Search Approach

As was stated in Sect. 2.5.2, a WG derived from a handwriting recognition process can be seen as a compact representation of the highest $P(w \mid x)$ transcriptions of a given text image x. Moreover, the probability of a given WG edge $p(e)$ is defined by Eq. (2.13). Here, in order to avoid the numeric underflow problem, mainly occurring during repeated multiplication of probabilities, we are going to use instead log-probabilities. With this in mind, Eq. (2.13) can be rewritten in these terms as follows:

$$\log p(e) = \log P\left(x_{t(i)+1}^{t(j)} \mid \omega(e)\right) + \log P\left(\omega(e)\right). \tag{3.1}$$

Similarly, as was pointed out at the end of Sect. 2.2, in order to balance the absolute values of these both (log-) probability terms, they are weighted by the so-called *grammar scale factor* (GSF) α, and the *word insertion penalty* (WIP) β (see [16]). Hence, the final resulting score of each edge is then computed as

$$\varphi(e) = \log P\left(x_{t(i)+1}^{t(j)} \mid \omega(e)\right) + \alpha \log P\left(\omega(e)\right) + \beta. \tag{3.2}$$

Note that Eqs. (3.1) and (3.2) become identical far $\alpha = 1$ and $\beta = 0$.

During the CATTI process, the two step search approach previously explained in Sect. 2.5.2 is performed on the above-defined WG in order to complete the prefixes accepted by the user. That is, the decoder first parses the validated prefix p, defining in this way a set of path end nodes Q_p (cf. Sect. 2.5.2), and then, the most likely transcription suffix departing from any of the nodes in Q_p is obtained.

3.2.2 Word Graph Error-Correcting Parsing

As already commented, a WG is a compact representation of a large *subset* of the highest possible likely transcriptions for a given input handwritten text image, whose number depends mainly of the WG density. Hence, it may happen that some prefixes given by the user cannot be exactly found in the WG. The solution is not to use p, but using the prefix $\tilde{p}$ from all the possible prefixes on the WG that best matches p. This prefix $\tilde{p}$ (that best matches the validated prefix p) can be considered as a hidden variable, so departing from Eq. (2.4), the problem of searching the most likely suffix $\hat{s}$ given p can be formulated as

$$\begin{aligned}
\hat{s} &= \arg\max_{s} \Pr(s \mid x, p) \\
&\approx \arg\max_{s} P(s \mid x, p) \\
&= \arg\max_{s} \sum_{\tilde{p}} P(s, \tilde{p} \mid x, p)
\end{aligned}$$

$$= \arg\max_{s} \sum_{\tilde{p}} P(x \mid p, \tilde{p}, s) \cdot P(\tilde{p}, s \mid p)$$

$$= \arg\max_{s} \sum_{\tilde{p}} P(x \mid p, \tilde{p}, s) \cdot P(s \mid p, \tilde{p}) \cdot P(\tilde{p} \mid p)$$

$$= \arg\max_{\tilde{p}, s} \sum_{\tilde{p}} \sum_{q \in Q_{\tilde{p}}} P(x, q \mid p, \tilde{p}, s) \cdot P(s \mid p, \tilde{p}) \cdot P(\tilde{p} \mid p). \tag{3.3}$$

We can make the naïve assumption that $P(x, q \mid p, \tilde{p}, s)$ and $P(s \mid p, \tilde{p})$ do not depend of p given $\tilde{p}$, to rewrite Eq. (3.3) as

$$\hat{s} \approx \arg\max_{s} \sum_{\tilde{p}} \sum_{q \in Q_{\tilde{p}}} P(x, q \mid \tilde{p}, s) \cdot P(s \mid \tilde{p}) \cdot P(\tilde{p} \mid p) \tag{3.4}$$

and following similar assumptions made on Eq. (2.6), previous equation can be rewritten as

$$\hat{s} \approx \arg\max_{s} \max_{\tilde{p}} \max_{q \in Q_{\tilde{p}}} P\left(x_1^{t(q)} \mid \tilde{p}\right) \cdot P\left(x_{t(q)+1}^{M} \mid s\right) \cdot P(s \mid \tilde{p}) \cdot P(\tilde{p} \mid p) \tag{3.5}$$

where $P(\tilde{p} \mid p)$ models the similarity distribution probability between $\tilde{p}$ and p. Moreover, $P(\tilde{p} \mid p)$ can be modeled in terms of probabilistic error correcting parsing. To do so, firstly we add to each original WG edge e, a set of extra edges representing different editing operations [1]. In Fig. 3.2, it is shown an example of all the added new edges between two adjacent nodes i and j. The probabilities of the added edges are considered to be proportional to $\exp^{-d(\omega(e), v)}$, where V is a task vocabulary (cf. Sect. 1.5.1), $v \in V \cup \{\lambda\}$ and $d(\cdot, \cdot)$ is the Levenshtein distance between $\omega(e)$ and v. As was seen in Sect. 1.5.1, an edge has been defined by its start and end nodes. However, this is not longer possible due to the fact that now there is more than one edge between two adjacent nodes. For this reason, each edge must now be defined by its start and end nodes, and a word label related with this edge: $e' = (i, j, v)$. Using log-probabilities, the score of the different edges can be

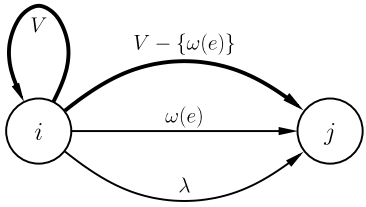


Fig. 3.2 Example of edges added between two adjacent nodes i and j of a WG for probabilistic error correcting parsing. The edge labeled with the word $\omega(e)$ is the original edge and corresponds to the operation of substitution of word $\omega(e)$ with itself. The group of edges labeled with $V - \{\omega(e)\}$ represents the substitution of $\omega(e)$ with any other word in the vocabulary excepting $\omega(e)$. The edge labeled with λ (empty symbol) models a deletion operation, whereas the last group V models insertion operations, involving an edge for each word in the vocabulary from the state i to itself

reformulated as

$$\varphi(i, j, v) = \begin{cases} \log P(x_{t(i)+1}^{t(j)} \mid \omega(e)) + \alpha \log P(\omega(e)) + \beta - \gamma d(\omega(e), v), & i \neq j, \\ \beta - \gamma d(\lambda, v), & i = j, \end{cases} \tag{3.6}$$

where e is the original edge between the nodes i and j, and γ is a penalization factor applied to control the number of different characters between $\omega(e)$ and v. The γ value should be greater than 0 because, otherwise, we would be encouraging WG paths, whose corresponding associated word-label sequences are more different from the given prefix's one. Note further that, if $\omega(e) = v$, then the number of different characters will be 0 and therefore Eqs. (3.6) and (3.2) become identical.

This heuristic can be implemented using dynamic programming and it can be further improved by visiting the WG nodes in topological order [1], and by incorporating beam search techniques [12] to discard those nodes whose scores are worse than the best score at the current stage of the parsing. Moreover, given the incremental nature of p, the error-correcting algorithm takes advantage of this peculiarity to parse only the new appended words of p provided by the user in the last interaction.

3.3 Increasing Interaction Ergonomics in CATTI: PA-CATTI

In CATTI application, user is repeatedly interacting with transcription process, thus trying to make this interaction process easier is really crucial for the success of such application.

Section 3.1 describes the CATTI process in which the user, before typing a new word in order to correct progressively a given hypothesis, needs first to position the cursor in the place where he or she wants to type such word. This is done by performing what, from now on, we will call *Pointer Action* (PA), which involves any kind of pointer-device like a typical mouse for example. By doing so, the user is already providing some very useful information to the system: he/she is validating a prefix up to the position where he placed the cursor, and, additionally, is signaling that the following word located after the cursor is incorrect. Hence, the system can already capture this fact and directly propose a new suitable suffix in which its first word is different from the first one in the previous suffix. This way, many explicit user corrections are avoided.

In Fig. 3.3 we can see an example of the CATTI process with the new interaction mode, which will be referred henceforth as PA-CATTI. As in the conventional CATTI, the process starts when the HTR system proposes a full transcription $\hat{s} \equiv \hat{w}$ of the input image x. Then, the user reads this prediction until a transcription error is found (denoted in this case by v) and makes a PA to position the cursor at this point. This way, the user validates an error-free transcription prefix p'. Now, before the user introduces a word to correct the erroneous one (as happens with the conventional CATTI), the HTR system, taking into account this validated prefix and the

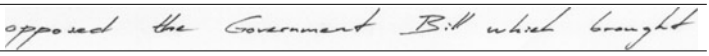
	x						
STEP-0	p						
STEP-1	$\hat{s} \equiv \hat{w}$	**opposite**	**this**	**Comment**	**Bill**	**in that**	**thought**
	PA	↑					
	p'						
	----	----------					
	$\hat{s}$	**opposition**	**this**	**Comment**	**Bill**	**in that**	**thought**
	κ	opposed					
	p	**opposed**					
STEP-2	$\hat{s}$		**the**	**Government**	**Bill**	**in that**	**thought**
	PA				↑		
	p'	**opposed**	**the**	**Government**	**Bill**		
FINAL	$\hat{s}$					**which**	**brought**
	PA						
	$p' \equiv T$	**opposed**	**the**	**Government**	**Bill**	**which**	**brought**

Fig. 3.3 Example of CATTI operation with pointer actions (PA). Starting with an initial recognized hypothesis $\hat{s} \equiv \hat{w}$, the user validates through a PA its longest well-recognized prefix p', which is used then by the system to suggest a new recognized hypothesis $\hat{s}$. In case the first word of $\hat{s}$ is also incorrect (see interaction 1), the user types the correct word κ (as in conventional CATTI process), generating a new consolidated prefix p (p' concatenated to κ), used later by the system to suggest a new hypothesis $\hat{s}$ starting again a new cycle. On the other hand, in case the word following to p' has been corrected in the new suggested hypothesis $\hat{s}$ (see interaction 2), no further corrective actions are required and the system start a new cycle. This whole process is repeated until the final error-free transcription T is obtained. In this final transcription, words in *red* color represent those which were corrected by user. Note that in the iteration 1, an unsuccessful PA was performed followed by the necessary typing of the correct word "`opposite`", whereas in the iteration 2, the performed PA was successful in predicting the correct word "`which`" and also the final full correct transcription is obtained

wrong word (v) that follows it, suggests a suitable continuation (i.e., a new $\hat{s}$). If the wrong word v appears corrected in this new $\hat{s}$, then a new cycle starts. Otherwise, as in the conventional CATTI, the user proceeds to correct it by directly typing the correct word, κ, producing a new consolidated prefix p (the previously validated prefix p' followed by κ) which is used by the HTR system to suggest a new suffix and a new cycle starts again. This process is repeated until a correct transcription of x is accepted by the user.

In the example shown in Fig. 3.3, without interaction, a user should have to correct about *six* errors from the original recognized hypothesis $\hat{w}$. If the conventional CATTI were used, *two* word corrections would have had to be performed. However in this new PA-based interaction, which somehow tries to anticipate the possible corrections that should be carried out by the user in the conventional CATTI context, just one user-correction is required to get the final error-free transcription. Note that in the iteration 1, the performed single PA is unsuccessful and the correct word is finally typed.

This new kind of interaction needs not be restricted to a single PA. Several scenarios arise, depending on the number of times the user performs a PA. In the simplest one, the user only makes one PA (i.e. single PA) when it is necessary to displace the

cursor. In this case, the PA does not involve any extra human effort, because it is also the same action that the user should make in the conventional CATTI to position the cursor before typing the correct word. Another interesting scenario to be considered consists in performing systematically a PA before writing, although the cursor is already in the correct position. In this case, however, there is a cost associated to this kind of PAs, since the user does need to perform additional actions, which may or may not be beneficial. Finally, this last scenario can be easily extended, allowing the user to make several PAs before deciding to write the correct word.

Since we have already dealt with the problem of finding a suitable suffix $\hat{s}$ for a given consolidated prefix p (p' plus κ), we focus now on the problem in which the user only performs a single PA. In this case, in order to search for the best transcription suffix $\hat{s}$, the decoder has to cope with the input image x, the validated prefix p' and its following wrong word v:

$$\hat{s} = \arg\max_{s} \Pr(s \mid x, p', v) \approx \arg\max_{s} P(x \mid p', s, v) \cdot P(s \mid p', v). \tag{3.7}$$

PA-CATTI interaction falls within what was described in Sect. 1.4.4: *Interaction with Weaker Feedback* where Eq. (1.34) is close related with Eq. (3.7), with h', d and h instantiated, respectively, by p', v and s. Concerning the first term of Eq. (3.7), $P(x \mid p', s, v)$, can be modeled following similar assumptions and developments made for Eq. (2.6) (cf. Sect. 2.3). On the other hand, $P(s \mid p', v)$ can be provided by a language model constrained by the validated prefix p' and by the erroneous word v that follows it.

With respect to the scenario which allows the user to perform several PAs before deciding to write the correct word, the successive corresponding values of v must be cached and $P(s \mid p', v)$ must be computed taking into account all the previously discarded values of v (not just the one from the previous step).

3.3.1 Language Model and Search

$P(s \mid p', v)$ can be approached by adapting an n-gram language model so as to cope with the validated prefix p' and with the erroneous word v that follows it. The language model described in Sect. 2.4 would provide a direct way to model the probability $P(s \mid p')$, but as in addition we have to take into account that the first word of s is conditioned by v, some extra considerations are needed to model adequately $P(s \mid p', v)$.

Let $p' = w_1^k$ be a validated prefix and $s = w_{k+1}^l$ be a possible suffix, considering that the wrong-recognized word v only affects the first word of the suffix w_{k+1}. Then, after following similar procedure to obtain Eq. (2.8), $P(s \mid p', v)$ can be computed as

$$P(s \mid p', v) \simeq P\left(w_{k+1} \mid w_{k+2-n}^k, v\right) \cdot \prod_{i=k+2}^{k+n-1} P\left(w_i \mid w_{i-n+1}^{i-1}\right) \cdot \prod_{i=k+n}^{l} P\left(w_i \mid w_{i-n+1}^{i-1}\right)$$

$$= P\left(s_1 \mid {p'}_{k-n+2}^{k}, v\right) \cdot \prod_{j=2}^{n-1} P\left(s_j \mid {p'}_{k-n+1+j}^{k}, s_1^{j-1}\right) \cdot \prod_{j=n}^{l-k} P\left(s_j \mid s_{j-n+1}^{j-1}\right) \tag{3.8}$$

where ${p'}_1^k = w_1^k = p'$ and $s_1^{l-k} = w_{k+1}^l = s$. Now, taking into account that the first word s_1 of the possible suffix has to be different from the erroneous word v, $P(s_1 \mid {p'}_{k-n+2}^{k}, v)$ can be formulated as follows:

$$P\left(s_1 \mid {p'}_{k-n+2}^{k}, v\right) = \begin{cases} 0, & s_1 = v, \\ \frac{P(s_1 \mid {p'}_{k-n+2}^{k})}{1 - P(v \mid {p'}_{k-n+2}^{k})}, & s_1 \neq v. \end{cases} \tag{3.9}$$

As in the conventional CATTI, the search problem involved by Eq. (3.7) can be solved by building a special language model, where the "suffix language model" of the Eq. (3.8) is modified in accordance with Eq. (3.9). Thanks to the finite-nature of this special language model, the search involved in Eq. (3.7) can be carried out using the Viterbi algorithm.

Due to the nature of PA-CATTI approach, where the system must react immediately by emitting a new suggested suffix after each pointer action performed by the user, the response speed becomes a very crucial factor to be taking into account. For this reason, search implementation based on WG technique results the more convenient solution. The restriction entailed by Eq. (3.9) can be easily implemented by directly deleting the WG edge labeled with v after the prefix has been matched. An example of this is shown in Fig. 3.4, where we have assumed that the user validated the prefix "`antiguos ciudadanos que en`" and the wrong-recognized word was "`el`". Hence, the WG has the edge labeled with "`el`" disabled.

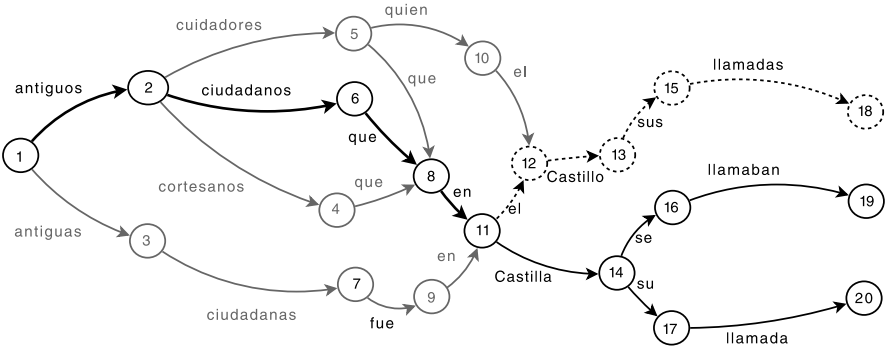


Fig. 3.4 Example of the WG generated after the user validates the prefix "`antiguos ciudadanos que en`" (represented by thicker-edges path). The edge corresponding to the wrong-recognized word "`el`" was disabled (*dashed-line path*)

Fig. 3.5 *Top*: illustrations of CATTI multimodal user-interaction using keyboard and electronic-pen, respectively, on a touch-screen device. *Bottom*: page fragment showing a line image being processed, with a partially corrected system suggestion (in *grey* and *black* roman font) and the (previous) corrections made by the user through pen strokes and handwriting input marked in *bold red*

3.4 Multimodal Computer Assisted Transcription of Text Images: MM-CATTI

As was described in the CATTI approach (see Sect. 3.1), the user is recurrently interacting with the system in order to produce the final required transcription. Hence, the quality and ergonomics of the interaction process is crucial for the success of the system. Traditional peripherals like keyboard and mouse can be used to unambiguously provide the feedback associated with the validations and/or corrections of the successive system predictions. In this sense, in the previous Sect. 3.3, it has been shown, based on the concept of what we have called PA, how the use of pointer-devices like (for example) a mouse can foster the CATTI interaction process to easily provide such a corrective feedback.

Nevertheless, 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. Different possibilities can be explored: gaze and gesture tracking, spoken commands, etc. Here we will focus on *touch-screen* communication, which is perhaps the most natural modality to provide the required feedback in CATTI systems. Figure 3.5 (top) shows a user interacting with a CATTI system using the keyboard and another one interacting by means of a touch-screen. Both the original image and the successive *off-line* HTR system's transcription hypotheses can be easily aligned and jointly displayed on the touchscreen, as shown in Fig. 3.5 (bottom).

More formally speaking, let x be the input image and s' a suffix suggested by the system as continuation of a consolidated prefix p in the previous interaction step (see Fig. 3.6). Hence, ps' constitutes a whole recognized hypothesis. Let t be the on-line *touchscreen pen strokes* provided by the user, which are sequences of real-valued vectors as described in Sect. 3.5.2. Let also p' be the longest error-free prefix validated by the user on the recognized hypothesis ps', thereby resulting

that $ps' \equiv p'\sigma$, where σ corresponds to the remaining word sequence whose first word(s) was/were incorrectly recognized. Actually, the validated prefix p' is implicitly stated when the user performs some pen strokes t aiming at correcting the first wrong word in ps' or, what is the same, the first word of σ. Moreover, the user may additionally type some keystrokes (κ) on the keyboard in order to correct (other) parts of this σ and/or to add more text. Using this information, the system has to suggest a new suffix s for the next interaction, as a continuation of the user-validated prefix p', conditioned by the on-line touchscreen pen-strokes t and the typed text κ. That is, the problem is to find s given x and a feedback information composed of $p'\sigma$, t and κ, considering all possible *decodings*, d, of the on-line data t (i.e., letting d be a hidden variable). After some mathematical formulation development, this general set-up scenario can be seen as an instantiation of the problem already formulated in Eq. (1.27), where the (p', σ) and (t, κ) correspond with h' and f, respectively:

$$\hat{s} \approx \arg\max_{s} \max_{d} P(t \mid d) \cdot P(d \mid p', \sigma) \cdot P(x \mid p', \sigma, d, \kappa, s) \cdot P(s \mid p', \sigma, d, \kappa). \tag{3.10}$$

According to this very general discussion, it might be assumed that the user can type with independence of the result of the on-line handwritten decoding process. However, it can be argued that this generality is not realistically useful in practical situations. Alternatively, it is much more natural that the user waits for a specific system outcome ($\hat{d}$) from the on-line touchscreen feedback data (t), prior to start typing amendments (κ) to the (remaining part of the previous) system hypothesis. Furthermore, this allows the user to fix possible on-line handwritten recognition errors in $\hat{d}$.

For this more pragmatic and simpler scenario, following a similar approach presented in Sect. 1.3.5, each interaction step can be formulated in two phases. In the first one, the user produces some (may be null) on-line touchscreen data t (to correct part of σ) and the system has to decode t into a word (or word sequence) $\hat{d}$ using the previous hypothesis $p'\sigma$:

$$\hat{d} = \arg\max_{d} P(t \mid d) \cdot P(d \mid p', \sigma). \tag{3.11}$$

Once $\hat{d}$ is available, the user can enter adequate amendment keystrokes κ, if necessary, and produce a new consolidated prefix p (based on the validated prefix p', the first incorrectly recognized words of σ, $\hat{d}$ and κ), which leads to the following expression, identical to Eq. (2.4):

$$\begin{aligned} \hat{s} &\approx \arg\max_{s} P(x \mid p', \sigma, \hat{d}, \kappa, s) \cdot P(s \mid p', \sigma, \hat{d}, \kappa) \\ &= \arg\max_{s} P(x \mid p, s) \cdot P(s \mid p). \end{aligned} \tag{3.12}$$

The process continues in this way until p is accepted by the user as a full correct transcription of x.

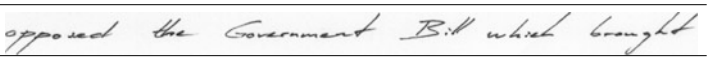
	x						
STEP-0	p						
STEP-1	$\hat{s} \equiv \hat{w}$	**opposite**	**this**	**Comment**	**Bill**	**in that**	**thought**
	p', t, σ	opposed	this	Comment	Bill	in that	thought
	$\hat{d}$	opponent					
	κ	sed					
	p	**opposed**					
STEP-2	$\hat{s} \equiv s'$		**the**	**Government**	**Bill**	**in that**	**thought**
	p', t, σ	**opposed**	**the**	**Government**	**Bill**	which	thought
	$\hat{d}$					which	
	κ						
	p	**opposed**	**the**	**Government**	**Bill**	**which**	
FINAL	$\hat{s} \equiv s'$						**brought**
	p', t, σ	**opposed**	**the**	**Government**	**Bill**	**which**	**brought**
	κ						
	$p \equiv T$	opposed	**the**	**Government**	**Bill**	which	**brought**

Fig. 3.6 Example of MM-CATTI interaction with a CATTI system, to transcribe an image sentence "*opposed the Government Bill which brought*". Each interaction step starts with a transcription prefix p that has been consolidated in the previous step. First, the system suggests a suffix $\hat{s}$ and the user handwrites some touchscreen text, t, to amend $\hat{s}$. This action also validates a correct prefix p' (and a remaining word sequence σ starting with the first wrong recognized word of $\hat{s}$), which can be used by the on-line HTR subsystem to obtain a decoding of t. After observing this decoding, $\hat{d}$, the user may type additional keystrokes, κ, to correct possible errors in $\hat{d}$ (and perhaps to amend other parts of $\hat{s}$). A new consolidated prefix, p, is built from the previous correct prefix p', the decoded on-line handwritten text, $\hat{d}$, and the typed text κ. System suggestions are printed in *boldface* and typed text in typewriter font. User corrections are shown in *different font* and *red* color. In the final transcription, T, typed text is additionally *underlined*. Assuming all interactions as whole-word corrections, the post-editing WER would be 100% (5 substitutions plus one insertion out of 6 correct words), while the MM-CATTI WSR is 50%; i.e., 2 touch-screen + 1 keyboard word corrections (see definitions of WER and WSR in Sect. 2.6)

An example of this kind of inter-leaved off-line image recognition and on-line touchscreen interaction is shown in Fig. 3.6. In this example, we are assuming that on-line handwriting is the modality preferred by the user to make corrections, relaying on the keyboard mainly (or only) to correct eventual on-line text decoding errors. Note that the potential increase in comfort of this setting comes at expense of a hopefully small number of additional interaction steps using the keyboard. In this example the user would need *three* interactions using MM-CATTI, compared with the *two* interactive corrections needed by CATTI (in Fig. 3.1) and *six* post-editing corrections required by the original, off-line recognized hypothesis.

Although Fig. 3.6 may suggest otherwise, we should remind that, as mentioned in Sect. 3.1, only whole-word interactions are considered in the present chapter. Furthering this assumption, but without loose of generality, we consider here that, in each interaction, the user only attempts to correct the single word σ_1 (first word of the word sequence σ); that is, $\hat{d}$ consists in a single, whole word.

Since we have already dealt with Eq. (3.12) in Sect. 2.3 (Eqs. (2.4)–(2.6)), we focus now on Eq. (3.11). As in Sect. 3.1, $P(t \mid d)$ is provided by (HMM) mor-

phological models of the word(s) in d (see Sect. 3.5.2). On the other hand, here, $P(d \mid p', \sigma)$ can be provided by a language model constrained by information derived from the validated error-free prefix p' and by the remaining words sequence σ produced at the previous iteration. Equation (3.11) may lead to several scenarios depending on the assumptions and constraints adopted for $P(d \mid p', \sigma)$. We examine some of them hereafter.

The simplest one corresponds to a conventional, non-interactive on-line HTR setting, where all the available conditions are ignored; i.e., $P(d \mid p', \sigma) \equiv P(d)$. This scenario is considered here as a *baseline*.

A more informative setting arises by taking into account part of the information derived from the previous off-line HTR prediction σ. The user introduces the touchscreen data t in order to correct the first m wrong words σ_1^m that follows the validated prefix p'. Therefore, we can assume an *error-conditioned* model such as $P(d \mid p', \sigma) \equiv P(d \mid \sigma_1^m)$; clearly, knowing the word(s) that the user has already deemed incorrect should prevent the on-line decoder making the same error(s).

If, in addition to σ_1^m, the information derived by the accepted prefix p' is also taken into account, a particularly useful scenario arises. In this case the decodings of t are further constrained to be suitable continuations of the prefix accepted; that is: $P(d \mid p', \sigma) \equiv P(d \mid p', \sigma_1^m)$ and Eq. (3.11) becomes

$$\hat{d} \approx \arg\max_{d} P(t \mid d) \cdot P\left(d \mid p', \sigma_1^m\right). \tag{3.13}$$

This *multimodal* model, referred to as MM-CATTI [29, 30], is the one studied in more detail in this chapter.

3.4.1 Language Model and Search for MM-CATTI

Language modeling and search techniques needed for the on-line HTR feedback subsystem in MM-CATTI are essentially similar to those described in Sect. 2.4 for the main, off-line HTR system. Language model constraints are implemented on the base of n-grams, depending on each multimodal scenario considered.

The simplest *baseline* scenario does not take into account any interaction-derived information and $P(d)$ could be provided by the same n-gram used for the off-line decoder. However, if only single whole-word touchscreen corrections are assumed, as discussed in the previous subsection, only *uni*-grams actually make sense.

The whole-word assumption also simplifies the *error-conditioned* model, $P(d \mid \sigma_1^m)$, because only the first (wrong) word of σ is to be taken into account. Let $v = \sigma_1$ be this wrong word. Therefore the *error-conditioned* language model probability can be written as

$$P(d \mid v) = \begin{cases} 0, & d = v, \\ \frac{P(d)}{1 - P(v)}, & d \neq v. \end{cases} \tag{3.14}$$

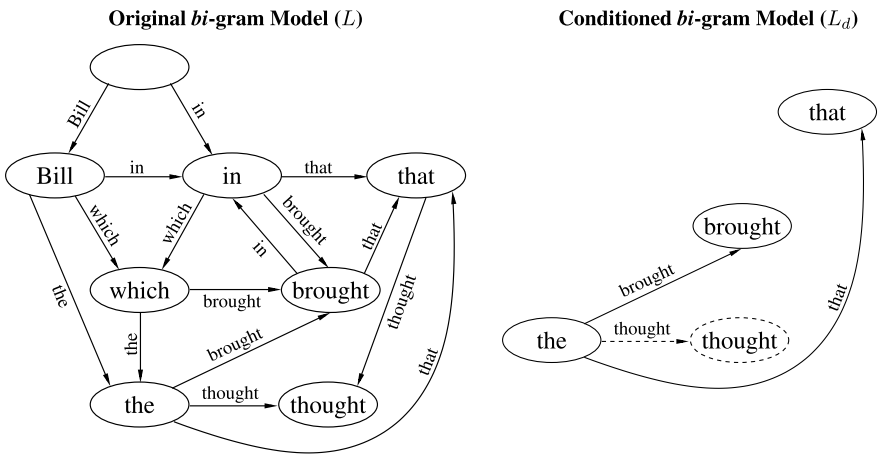


Fig. 3.7 Example of MM-CATTI dynamic *bi*-gram language model generation. L is the original *bi*-gram model used by off-line HTR system, whereas L_d is the *bi*-gram sub-model, derived from L, which takes as initial state that corresponding to the prefix "`the`". This simplified language model is used by the on-line HTR subsystem to recognize the touchscreen handwritten word "`brought`", intended to replace the wrong off-line recognized word "`thought`", which was disabled in L_d (displayed in *dashed line*)

Finally, in MM-CATTI the language model probability is approximated by $P(d \mid p', v)$. That is, the on-line HTR subsystem should produce a hypothesis $\hat{d}$ for the touchscreen strokes t, taking into account a user-validated prefix, p', and the first wrong word, $v = \sigma_1$, in the off-line HTR suggestion. In this case, arguments similar to those in Sect. 3.3.1 apply and, under the same single whole-word assumption, we can use Eq. (3.9) changing s_1 with d, leading to

$$P(d \mid p', v) = \begin{cases} 0, & d = v, \\ \frac{P(d|p'^{k}_{k-n+2})}{1-P(v|p'^{k}_{k-n+2})}, & d \neq v \end{cases} \tag{3.15}$$

where k is the length of p'.

A simple implementation of Eq. (3.15) is shown in Fig. 3.7, based on the same language model example of Fig. 2.1. In this case, $p' =$"`of the`" and the user wants to correct the wrong off-line recognized word "`thought`", by handwriting the word "`brought`" (for example) on the touchscreen. The on-line HTR subsystem uses a *bi*-gram model, conditioned by the context word "`the`" (which is now the initial state) and the word transition edge "`thought`" is disabled.

As shown in the example, and unlike it happened in CATTI (cf. Fig 2.1), the linear language model of the prefix p' is no longer required, because the corresponding on-line touchscreen data of the prefix p' do no exist in this case. Moreover, as we are assuming only single whole-word corrections, only the direct transitions from the starting node (the "`the`" node in the example) need be considered.

As in CATTI searching (Sect. 3.2), owing to the finite-state nature of the n-gram language model, the search involved in Eqs. (3.13) and (3.15) can be efficiently carried out using the Viterbi algorithm [10]. Note that under the assumption of just one whole-word correction per interaction, Viterbi search implementation is the only choice that makes sense for the on-line HTR feedback decoding. Moreover, and as in CATTI, decoding search in MM-CATTI (specially decoding related to Eq. (3.12)) can be implemented using one of the two different approximations presented in Sect. 2.5. The on-line decoding phase (Eq. (3.13)) may also be implemented using WGs, particularly for the case we decide that it is possible to write/correct more than one word with the e-pen touchscreen.

3.5 Non-interactive HTR Systems

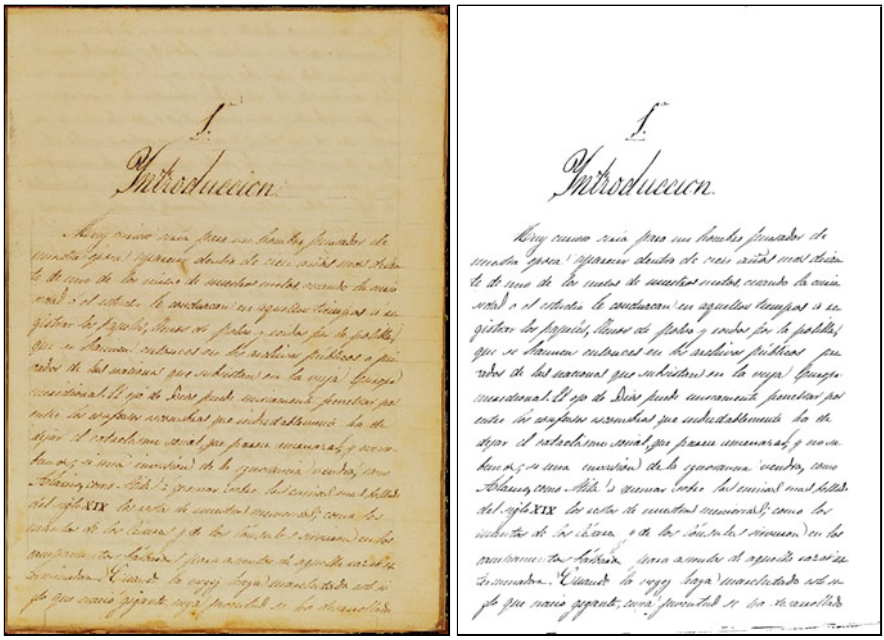
This section is devoted to describe in more detail the HTR systems employed for both the off-line and the on-line versions. In particular, it will be shed more light on the preprocessing and feature extraction phases carried out for each HTR version, along with additional specific information related to the modelling topic itself used in each case.

3.5.1 Main Off-Line HTR System Overview

The off-line HTR system used here follows a classical architecture composed of three modules: (a) *preprocessing*, aimed at correcting image degradations and geometry distortions, and dedicated to decompose page images into their constituent line images; (b) *feature extraction*, where a real-value vector sequence representation of each line image is obtained; and (c) *recognition*, which obtains a most likely word sequence for the given input sequence of feature vectors. The following subsections describe the three modules in some detail.

Off-Line HTR Preprocessing

Image degradation is a quite common problem in many text images and more so in ancient documents [6]. Typical degradations include the presence of smear and skew, backgrounds with big variations and uneven illumination, spots due to the humidity or marks resulting from the ink that goes through the paper (commonly called bleed-through). In addition, other kinds of difficulties appear in these images, such as different character styles and sizes, underlined and/or crossed-out words, etc. The combination of these problems contributes to make the recognition process difficult. Therefore, preprocessing becomes essential to reduce the impact of these problems, as well as to extract the actual (line) images of the text to be recognized. A survey of preprocessing techniques proposed for text images can be seen in [17, 21]. In this



(a) (b)

Fig. 3.8 Preprocessing example: (**a**) original page image; (**b**) result after page skew correction, background removal, noise reduction and increase of contrast

work, the following preprocessing steps take place in order: background removal and noise reduction, skew correction, line extraction, slope-slant correction and size normalization.

Background removal and noise reduction are performed by applying a 2-dimensional median filter [5, p. 540] on the whole page image and subtracting the result from the original image. This is often followed by a grey-level normalization to increase the foreground/background image contrast (see Figs. 3.8a and 3.8b).

Skew is one of the distortions introduced during document scanning process. It is understood as the angle of the document paper image with respect to the horizontal x-axis. *Skew correction* is carried out globally on each page image by searching for the angle which maximizes the variance of the horizontal projection profile. It is assumed here that this maximal variance value should correspond with the horizontal projection profile of the de-skewed text lines [19, 26] (see again Figs. 3.8a and 3.8b).

Line detection is based again on the horizontal projection profile of the optimally de-skewed input image. Local minima in this curve are potential cut-points between consecutive text lines (see Fig. 3.9a). Obviously, clear separation is not always possible and cut-points detection needs to be adequately combined with connected components techniques [14]. Figure 3.9b shows some line images obtained with this method.

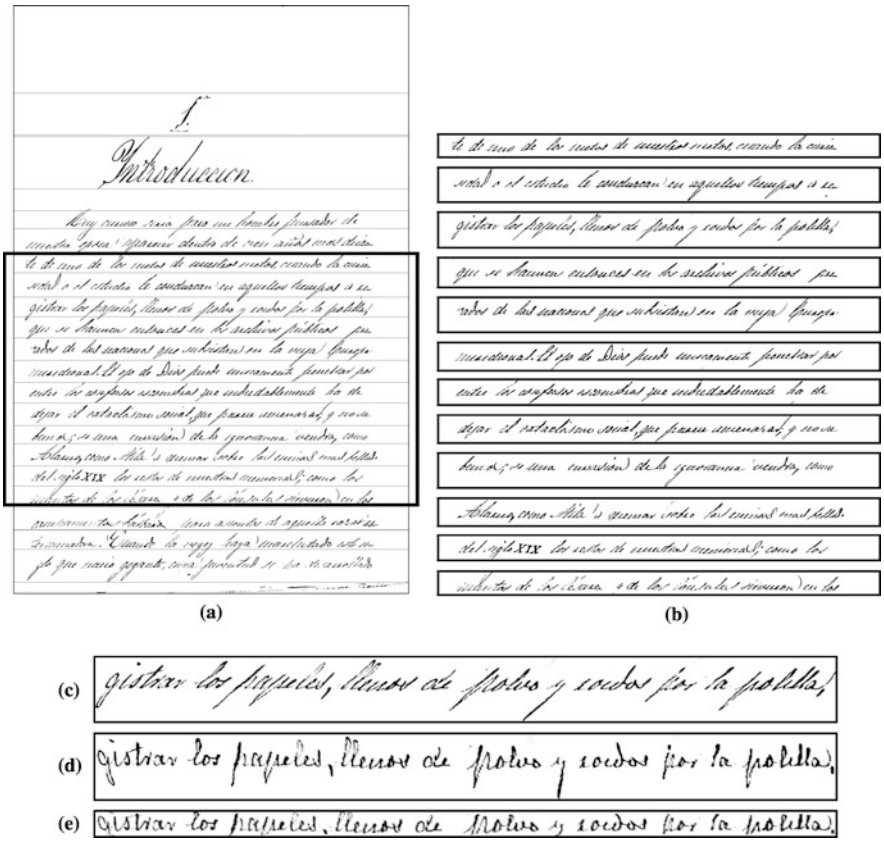


Fig. 3.9 Preprocessing example: (**a**) image with cutting lines computed from the horizontal projection profile; (**b**) separated line images from the highlighted region; (**c**) a separated line image; (**d**) slant correction; (**e**) size normalization

The slant is the angle between the vertical and the dominant direction of the written vertical strokes. *Slant correction* is applied to each previously separated line image. Much in the same way as in the case of line detection, the slant is computed by searching for the angle which maximizes the variance of the vertical projection profile of the de-slanted text [19]. This tends to render the written text strokes in an upright position (see Fig. 3.9d) and significantly improves the accuracy of the HMM recognition techniques.

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. To obtain the angle we use a method based on horizontal projections, very similar to the method used on the skew correction operation.

Finally, (non-linear) *size normalization* aims at making the optimally de-slanted line images invariant to character size and attempts to reduce large areas of background pixels which remain on the image because of the presence of long ascenders and descenders of some letters [23] (see Fig. 3.9e).

Off-Line HTR Feature Extraction

As our HTR system is based on HMMs, each preprocessed text line image has to be represented as a sequence of feature vectors. Several approaches have been proposed to obtain this kind of sequences [2, 3, 14]. The approach used in this chapter follows the ideas described in [2].

First, a grid is applied to divide the text line image into $N \times M$ rectangular cells. N is chosen empirically, whereas M is such that M/N is proportional to the original text line image aspect ratio, with a proportionality coefficient tuned empirically. Each cell of the grid is characterized by three features: *average gray level*, *horizontal gray level derivative* and *vertical gray level derivative* [26], which are computed from a $n \times m$ pixels analysis window, S, centered in that cell. The size of the analysis window (centered in a cell) is also empirically adjusted and its area typically overlaps partially (or completely) the neighbor cell areas.

The *average gray level*, $\overline{g}$, is computed through convolution with two 1-d Gaussian filters, w_i and w_j:

$$\overline{g} = \frac{1}{nm} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} S(i,j) \cdot w_i \cdot w_j,$$

$$w_i = \exp\left(-\frac{1}{2}\frac{(i-n/2)^2}{(n/4)^2}\right), \; w_j = \exp\left(-\frac{1}{2}\frac{(j-m/2)^2}{(m/4)^2}\right). \tag{3.16}$$

The *horizontal gray level derivative*, d_h, is calculated as the slope of the line which best fits the horizontal function of column-average gray level in the analysis window. The fitting criterion is the sum of squared errors weighted by a 1-d Gaussian filter:

$$d_h = \frac{(\sum_{j=0}^{m-1} w_j g_j)(\sum_{j=0}^{m-1} w_j j) - (\sum_{j=0}^{m-1} w_j)(\sum_{j=0}^{m-1} w_j g_j j)}{(\sum_{j=0}^{m-1} w_j j)^2 - (\sum_{j=0}^{m-1} w_j)(\sum_{j=0}^{m-1} w_j j^2)} \tag{3.17}$$

where g_j is, in this case, the column-average gray level at column j, defined by

$$g_j = \frac{\sum_{i=0}^{n-1} S(i,j)}{n}.$$

The *vertical gray level derivative*, d_v, is computed in a similar way.

Columns of cells (also called *frames*) are processed from left to right and a feature vector is constructed for each *frame* by stacking the three features computed in their constituent cells. Hence, at the end of this process, a sequence of M $(3N)$-dimensional feature vectors is obtained. Figure 3.10 shows a graphical representation of the feature vectors sequence extracted for the word image *sometimes*.

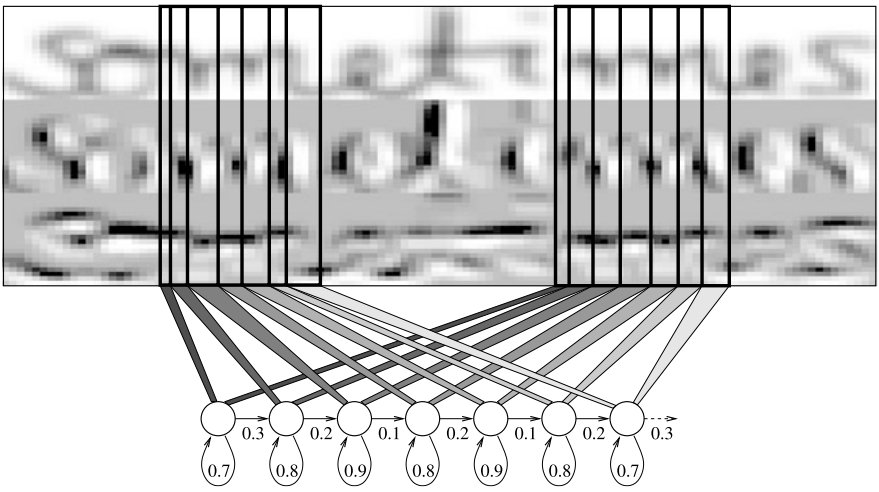


Fig. 3.10 Example of feature-vector sequence and HMM modeling of instances of the character "m" within the word "sometimes". The model is shared among all instances of characters of the same class. The zones modeled by each state show graphically subsequences of feature vectors compounded by stacking the normalized grey level and its both derivatives features

Modeling and Search

As already explained in Sect. 2.2, *characters* (considered in this case as the basic recognition units) are modeled by continuous density left-to-right HMMs, with state emission probabilities given by mixtures of Gaussian densities. The number of Gaussian densities of the mixtures as well as the number of states were empirically chosen after tuning the system. It is also important to mention here that all the experimental results reported in Sect. 3.6.2 have been obtained using HMM topologies with the same number of states for all the character classes. Figure 3.10 shows an example of how a HMM models two feature vector subsequences corresponding to the character "m".

Concerning to the modeling of lexical entries (words) and syntactic constraints derived from each specific task, as well as the way they are used to perform the search decoding, have been already described in Sect. 2.2.

3.5.2 On-Line HTR Subsystem Overview

The on-line HTR subsystem is intended to decode the feedback touchscreen data for multimodal text correction; i.e. to recognize the pen strokes (words) written by the user in successive CATTI interactions in order to correct or replace the errors produced by the main, off-line HTR decoder. In general, touchscreen data consist of a series of pen-positions (x_t, y_t), sampled at regular time instants $t = 1, 2, \ldots$.

Each sample of this *trajectory* can be accompanied by information about the pen pressure, or at least by one bit indicating whether the pen is actually touching the screen or it is "up". In this work no pressure information is used.

The conceptual architecture adopted for the on-line HTR subsystem is analogous to that used in the main off-line HTR system, with exception of the preprocessing and feature extraction modules, which are explained hereafter.

On-Line HTR Preprocessing

An overview of preprocessing techniques for on-line HTR can be seen in [8]. In this chapter, the preprocessing of each trajectory involves only two simple steps: repeated points elimination and noise reduction. *Repeated points* appear in a trajectory when the pen remains down and motionless for some time. These uninformative data are trivially removed, along with the points marked as "*pen-up*". *Noise* in pen strokes is due to erratic hand motion and inaccuracy of the digitalization process. To reduce this kind of noise, a simple smoothing technique is used which replaces every point (x_t, y_t) in the trajectory by the mean value of its neighbors [9]. Note that the temporal order of the data points is preserved throughout these preprocessing steps.

On-Line HTR Feature Extraction

Each preprocessed trajectory is transformed into a new temporal sequence of 6-dimensional real-valued feature vectors [27]. These time-domain features are point locations (although in this case only y coordinate is considered), first and second time derivatives and curvature.

Normalized Vertical Position: first, the coordinate pairs of each trajectory point are linearly scaled and translated to obtain new pairs of values (x_t, y_t), so that y_t is in the range $[0, 100]$ and the original aspect-ratio of the trajectory is preserved.

Normalized First Derivatives: x'_t and y'_t are calculated using the method given in [32]:

$$x'_t = \frac{\Delta x_t}{\|\nabla\|}, \qquad y'_t = \frac{\Delta y_t}{\|\nabla\|}, \tag{3.18}$$

where

$$\Delta x_t = \sum_{i=1}^{r} i \cdot (x_{t+i} - x_{t-i}), \qquad \Delta y_t = \sum_{i=1}^{r} i \cdot (y_{t+i} - y_{t-i}),$$

$$\|\nabla\| = \sqrt{{\Delta x_t}^2 + {\Delta y_t}^2},$$

and r defines a window of size $2r + 1$ which determines the neighbor points involved in the computation. Setting $r = 2$ has provided satisfactory results in this case.

It is worth noting that the normalization of derivatives by $\|\nabla\|$ implicitly entails an effective *writing speed normalization*. In our experiments, this has proved to lead to better results than using explicit speed normalization preprocessing techniques such as *trace segmentation*, based on re-sampling the trajectory at equal-length (rather than equal time) intervals [20, 31].

Second derivatives: x''_t and y''_t, are computed in the same way as the first derivatives, but using x'_t and y'_t instead of x_t and y_t.

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

$$k_t = \frac{x'_t \cdot y''_t - x''_t \cdot y'_t}{(x'^2_t + y'^2_t)^{3/2}}. \tag{3.19}$$

Although this feature is an explicit combination of the previous features, it has lead to slightly but consistently improved results in our experiments.

Character, Word and Language Modeling and Search

Modeling and search for on-line recognition follow almost the same schemes used in off-line recognition, described in Sect. 3.5.1.

As in the off-line case, we use continuous density left-to-right character HMMs with Gaussian densities assigned to each state mixture. However, instead of using a fixed number of states for all HMMs, it is variable for each character class. The number of states s_c chosen for each HMM character class M_c was computed as $s_c = l_c/f$, 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 modeled per state (*state load factor*). This rule of setting up s_c tries to balance modeling effort across states and, for our task, has significantly improved the recognition accuracy. On the other hand, lexical modeling is carried out in exactly the same way as in the off-line HTR case.

Language modeling and search are simpler in this case because, as discussed in Sect. 3.4.1, we have restricted our present MM-CATTI study to single whole-word touchscreen corrections. That is, the language models used in the MM-CATTI search only allow one word per user-interaction. As mentioned at the end of Sect. 2.2, a GSF is also used here in practice to balance the HMM and language model probabilities of Eq. (3.11).

3.6 Tasks, Experiments and Results

The experimental framework adopted to assess the effectiveness of the basic HTR systems (off-line and on-line) and for the three approaches proposed in this chapter: CATTI, PA-CATTI and MM-CATTI, is described in the following subsections. This includes information about the different corpora and performance measures employed in the experiments as well as the obtained results.

3.6.1 HTR Corpora

Three off-line corpora were employed in the experiments. Two of them, ODEC-M3 [25] and IAMDB [13, 15], contain handwritten text in modern Spanish and English, respectively. IAMDB is publicly available, thereby serving as a reference to compare the obtained results. The third corpus, CS [24], consists of cursive handwritten page images in old Spanish, which allow us to report results on the kind of legacy documents.

Sentence-segmented images are used both in ODEC-M3 and IAMDB, while only *line-segmented* images are available in CS. Each sentence or line image is accompanied by its ground truth 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. These transcriptions are used to train the *bi*-gram language models for ODEC-M3 and CS. IAMDB, on the other hand, consists of hand-copied sentences from the much larger electronic text LOB corpus [11] which contains about 1 000 000 running words. Therefore, in this case, the whole LOB corpus (after removing all the test sentences) was used for *bi*-gram training. Finally, the lexicon of each task is defined as the set of words found in training or in test transcriptions. Such a "closed vocabulary" scheme is commonly used in Automatic Speech Recognition [4, 10] to ease results reproducibility.

On the other hand, to train the on-line HTR feedback subsystem and test the MM-CATTI approach, the on-line handwriting UNIPEN corpus, which also is publicly available, was chosen.

In the next subsections, detailed descriptions of all off-line corpora as well the on-line corpus are given.

ODEC-M3

This corpus consists of images of casual handwritten Spanish paragraphs. It was compiled from spontaneous answers extracted from survey forms made for a telecommunication company.[2] These answers were written by a heterogeneous group of people, without any explicit or formal restriction. In addition, since no guidelines were given as to the kind of pen or the writing style to be used, paragraphs are very variable and noisy. Many of them were written using different case and font types, variable sizes and include words which are underlined, crossed-out or contain orthographic mistakes, unusual abbreviations, symbols, etc. Examples of these difficulties are shown in Fig. 3.11.

Because of some of these difficulties, line extraction was carried out in a semi-automatic way, based on a conventional line-extraction method mentioned in Sect. 3.5.1. Most of the phrases were processed automatically, but manual supervision was applied to difficult line-overlapping cases such as that shown in Fig. 3.11

[2] Data kindly provided by ODEC, S.A. (www.odec.es).

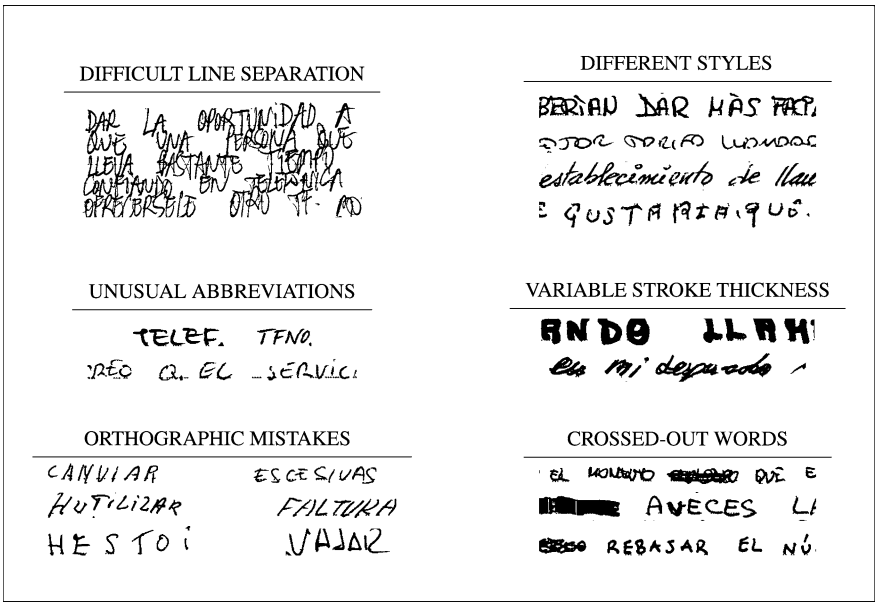


Fig. 3.11 Examples of difficulties found in several paragraphs of the ODEC-M3 Corpus

Table 3.1 Basic statistics of the database ODEC, where OOV stands for *out-of-vocabulary* words

Number of	Training	Test	Total	Lexicon	OOV	Tr. Ratio
Writers/phrases	676	237	913	–	–	–
Words	12 287	4 084	16 371	2 790	518	4.4
Characters	64 666	21 533	86 199	80	0	808

(top-left). By adequately pasting the lines extracted from each paragraph, a single-line (long) image which encompasses the whole paragraph was obtained. This resulted in 913 binary images, which were partitioned into a training set of 676 images and a test set of 237 images. The transcriptions of all the images are also available, containing 16 371 words with a vocabulary of 2 790 different words. It is important to remark that we do not distinguish between words written in lowercase characters or uppercase. Therefore, to train the n-gram models, the transcription of the 676 training images were converted to uppercase and the punctuation signs $\{-/;:+*()|,\ !?\}$ were eliminated. The average ratio for n-gram training is 4.4 running word instances per vocabulary word. Nevertheless, to train the character HMMs we use the transcription that describes with detail and accuracy all the elements appearing in each handwritten text images, such as lowercase or uppercase letters, symbols, abbreviations, spacing between words and characters, crossed-words, etc. All this information is summarized in Table 3.1. More information on this corpus can be found in [25].

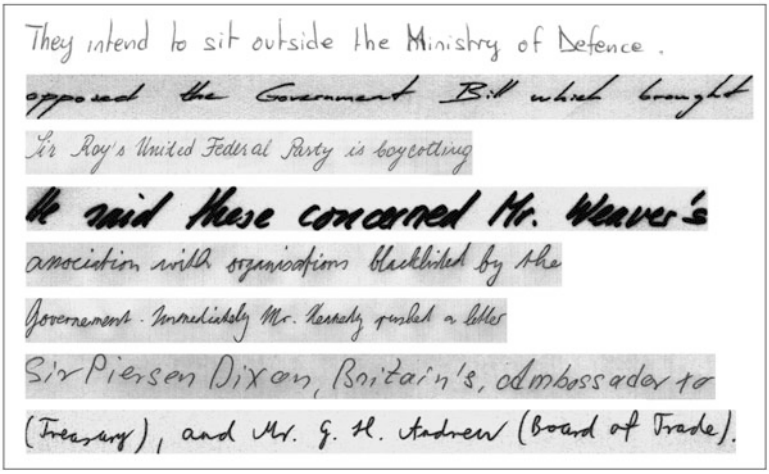


Fig. 3.12 Examples of handwritten lines from the IAMDB corpus

IAMDB

This corpus was compiled by the Research Group on Computer Vision and Artificial Intelligence (FKI) at Institute of Computer Science an Applied Mathematics (IAM) in Bern (Switzerland). The IAM Handwriting Database [13, 15] (IAMDB) consists of grey-level images of unconstrained handwritten English text forms. It is publicly accessible and freely available upon request for non-commercial research purposes.[3] The IAMDB images correspond to handwritten texts copied from the Lancaster-Oslo/Bergen Corpus [11] (LOB), which encompasses around 500 printed English texts of about 2 000 words each and about 1 000 000 total running words.

The IAMDB version 3.0 (the latest at this moment) is composed of 1 539 scanned text pages, handwritten by 657 different writers. No restriction was imposed on the writing style or the type of pen to be used. This dataset is also provided at sentence level. Line detection and extraction, as well as (manually) detecting sentence boundaries, was carried out by the IAM institute [14]. Using this information, lines could be easily merged into whole sentence line-images. Figure 3.12 shows examples of handwritten lines images from this corpus. This corpus was partitioned into a training set composed of 2 124 sentences, handwritten by 448 different writers, and a writer independent test set composed of 200 sentences written by 100 writers. Table 3.2 summarizes all this information.

Note that the amount of data available for training the (n-gram) language models for this task (the whole LOB corpus) is very much larger than the amount of data contained in the transcriptions of the available text images. Following [33], we take advantage of this opportunity by using the whole LOB corpus (except the

[3] http://iamwww.unibe.ch/~fki/iamDB

Table 3.2 Basic statistics of the database IAM, where OOV stands for *out-of-vocabulary* words

Number of	Training	Test	Total	Lexicon	OOV	Tr. Ratio
Writers	448	100	548	–	–	–
Sentences	2 124	200	2 324	–	–	–
Words	42 832	3 957	46 789	8 017	921	81
Characters	216 774	20 726	237 500	78	0	2 779

Fig. 3.13 Examples of pages images from CS corpus

200 sentences of the image test set) for n-gram training, while setting a reduced vocabulary which encompasses only the 8 017 different words found in the IAMDB text images. Only 651 462 running words of the LOB corpus belong to the IAMDB vocabulary proper. Therefore, for n-gram training we have a quite good effective average ratio of 81 word instances per IAMDB vocabulary word. As in the ODEC-M3 corpus, here we do not distinguish between words written in lowercase characters or uppercase.

CS MANUSCRIPT

This corpus was compiled from a XIX century Spanish manuscript identified as "Cristo-Salvador" (CS), which was kindly provided by the *Biblioteca Valenciana Digital* (BiVaLDi).[4] This is a rather small document composed of 50 color images of text pages, written by a single writer. Some examples are shown in Fig. 3.13.

The page images were preprocessed and divided into lines, as described in Sect. 3.5.1. The results were visually inspected and the few detection errors

[4] http://bv2.gva.es

Table 3.3 Basic statistics of the partition *page* of the database Cristo-Salvador. OOV stands for *out-of-vocabulary* words

Number of	Training	Test	Total	Lexicon	OOV	Tr. Ratio
Pages	53	53	53	–	–	–
Text lines	681	491	1 172	–	–	–
Words	6 435	4 483	10 918	2 277	1 010	2.8
Characters	36 729	25 487	62 216	78	0	470

(around 4%) were manually corrected, resulting in a dataset of 1 172 text line images. It is worth mentioning that, unlike the two previous corpora, in this case the extracted lines are not merged into sentence or paragraph images. The transcriptions of these line images are also available, containing 10 918 running words with a vocabulary of 2 277 different words. Note that as in the other two corpora, we do not distinguish between words written in lowercase characters or uppercase.

Two different partitions, *page* (or "*soft*") and *book* (or "*hard*") are defined for this dataset [24]. Here we only consider the (easier) *page* partition. Its test set contains 491 samples corresponding to the last ten lines of each document page, whereas the training set is composed of the 681 remaining lines. Table 3.3 summarizes this information.

For n-gram training, the average ratio of running word instances per vocabulary word is 2.8. It is important to remark that such a small ratio will certainly result in under-trained language models, which clearly increase the difficulty of the recognition task and prevent CATTI, PA-CATTI or MM-CATTI to take much advantage of prefix-derived constraints.

UNIPEN Corpus

The UNIPEN `Train-R01/V07` dataset[5] comes organized into several categories [7] such as lower- and uppercase letters, digits, symbols, isolated words and full sentences. Unfortunately, the UNIPEN isolated words category does not contain all (or almost none of) the required word instances to be handwritten by the user in the MM-CATTI interaction process with the ODEC, IAMDB, or CS text images. Therefore, they were generated by concatenating random character instances from three UNIPEN categories: `1a` (digits), `1c` (lowercase letters) and `1d` (symbols). Table 3.4 shows the basic statistics of these words, needed to test the HTR feedback subsystem for each off-line HTR task. We have just taken into account here, for each task (corpus), all the words that the user must introduce in a standard CATTI iteration process when the Viterbi-search implementation is used (cf. Sect. 2.5.1). Note that in the case of WG-search implementation had been used, probably slightly different words and/or number of their instances would have been obtained. Anyway,

[5] For a detailed description of this dataset, see http://www.unipen.org.

Table 3.4 For each off-line HTR task: number of *on-line* unique words and word instances needed as feedback to correct the word errors made by the plain off-line HTR system

Task	Unique words	Word instances
ODEC-M3	378	753
IAMDB	510	755
CS	648	1 196

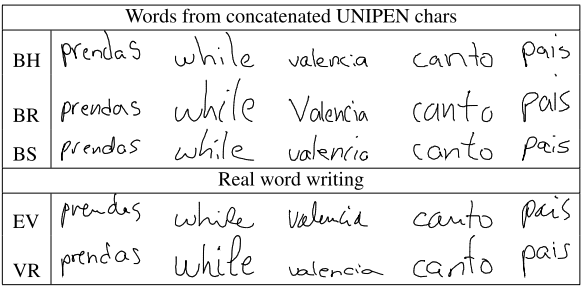


Fig. 3.14 Examples of words generated using characters from the three selected UNIPEN test writers (BH, BR, BS), along with samples of the same words written by two other writers in our labs

since we are interested in evaluating the feedback decoding subsystem (i.e. the on-line HTR subsystem), only Viterbi-search implementation is going to be considered. Even so, as we will see in Sect. 12.2, the demonstrator of MM-CATTI (MM-IHT) is implemented using an hybrid search-decoding scheme; that is, the off-line decoding phase is based on WG, whereas the feedback decoding phase relies directly on Viterbi.

To increase realism, the generation of each of these test words was carried out employing characters belonging to a same writer. Three different writers were randomly chosen, taking care that sufficient samples of all the characters needed for the generation of the required word instances were available from each writer. Each character needed to generate a given word was plainly aligned along a common word baseline, except if it had a descender, in which case the character baseline was raised 1/3 of its height. The horizontal separation between characters was randomly selected from *one* to *three* trajectory points. The selected writers are identified by their name initials as BS, BH and BR. Figure 3.14 shows some examples of words generated in this way, along with real samples of the same words written by two writers (EV and VR).

Training data were produced in a similar way using 17 different UNIPEN writers. For each of these writers, a sample of each of the 42 symbols and digits needed was randomly selected and one sample of each of the 1 000 most frequent Spanish and English words was generated, resulting in 34 714 training tokens (714 isolated characters plus 34 000 generated words). To generate these tokens, 186 881 UNIPEN character instances were used, using as many repetitions as required out of the 17 177 *unique* character samples available. Table 3.5 summarizes the amount of UNIPEN training and test data used in the experiments.

Table 3.5 Basic statistics of the UNIPEN training and test data used in the experiments

Number of different	Train	Test	Lexicon
Writers	17	3	–
Digits (1a)	1 301	234	10
Letters (1c)	12 298	2 771	26
Symbols (1d)	3 578	3 317	32
Total characters	17 177	6 322	68

3.6.2 Results

The three measures (WER, WSR and EFR) adopted to assess interactive transcription systems in Sect. 2.6 have been used to evaluate CATTI performance. In addition, to assess the new interaction mode of PA-CATTI approach, it has been introduced the so-called *Pointer Action Rate* (PAR). This can be defined as the number of additional PAs per word that the user has to do using the new user interaction mode. Note that the human effort needed for the verification of the transcription and positioning the cursor in the appropriate place in the conventional CATTI is the same as in the new CATTI system using single-PA interactions. In both cases the user should read the transcription proposed by the system until he or she finds an error and then positions the cursor in the place where the new word has to be typed. Moreover, since only single-word corrections have been considered, the *feedback decoding error rate* (FER) (that is, the conventional classification error rate) 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.

Different experiments have been carried out to assess the feasibility and potential of CATTI, PA-CATTI and MM-CATTI. In addition, non-interactive (off- and on-line) handwritten text recognition experiments have been performed to establish baseline performance figures.

Baseline Off-Line HTR Results

Conventional, non-interactive off-line HTR experiments were performed on the three off-line corpora described in Sect. 3.6.1, ODEC-M3, IAMDB and CS, using the basic system explained in Sect. 3.5.1. All the morphological (HMMs) and language (*bi*-gram) models were trained from the respective training images and transcriptions of each corpus. Although, as was noted earlier, in the training of IAMDB *bi*-gram language model not only the own IAMDB transcription corpus, but also the whole LOB corpus was used to. The HTR WER percentages obtained for the test images of each corpus were 22.9, 25.3 and 28.5, for ODEC-M3, IAMBD

and CS, respectively. All these results have been obtained after optimizing the parameters values corresponding to the preprocessing and the feature extraction processes explained in Sect. 3.5.1 for each of the tasks. The WER obtained for IAMDB (25.3%) is comparable with non-interactive, state-of-the-art results published for this dataset [33].

Baseline On-Line HTR Results

These experiments were carried out using the basic on-line HTR subsystem explained in Sect. 3.5.2. As discussed in Sect. 3.6.1, UNIPEN data were used to assess the performance of the on-line HTR feedback subsystem.

All the samples were preprocessed using the preprocessing and feature extraction methods outlined in Sect. 3.5.2. In order to tune the parameters of the 68 on-line character HMMs needed, isolated character recognition experiments were carried out on each of the `1a`, `1c` and `1d` UNIPEN categories. The *classification error rates* (ER) obtained for test digits, letters and symbols were 1.7%, 5.9% and 21.8%, respectively. These results are comparable with state-of-the-art results obtained for this dataset [18, 22].

In order to establish a word decoding baseline accuracy for the on-line HTR feedback subsystem, a simple word recognition experiment was carried out. The words needed to train and test the feedback subsystem for each task were generated by concatenating adequate UNIPEN characters. Therefore, new character HMMs were trained from these training words, using the parameters previously tuned through the isolated character recognition experiments. On the other hand, since only single words are to be recognized, a *uni*-gram language model was trained (from the training transcriptions of each off-line task) to estimate the corresponding prior word probabilities. The following word recognition error percentages (FER) were observed for ODEC-M3, IAMDB and CS, respectively: 5.1, 4.6 and 6.4.

Note that these FER values are obtained without taking advantage of any interaction-derived contextual information (i.e., just using plain *uni*-grams). Therefore these figures represent the highest accuracies that could be expected if, e.g., an off-the-shelf on-line HTR system were adopted to implement the MM-CATTI feedback decoder.

CATTI Results

The CATTI approach presented in Sect. 3.1 was applied to the three off-line HTR tasks before-described, using the same parameter values used for the baseline, non-interactive off-line HTR results presented earlier. Table 3.6 shows the estimated interactive human effort (WSR) required for each task using the Viterbi-based implementation presented in Sect. 2.5.1, in comparison with the corresponding estimated post-editing effort (WER) before reported in the subsection of baseline off-line HTR results. Besides, it shows the *estimated effort reduction* (EFR), computed as the relative difference between WER and WSR (see Sect. 2.6).

Table 3.6 Performance of non-interactive off-line HTR (baseline WER) and CATTI (WSR), along with their relative difference (Estimated Effort Reduction—EFR) using the Viterbi-based search. All results are percentages

Corpus	WER	WSR	EFR
ODEC-M3	22.9	18.9	17.5
IAMDB	25.3	21.1	16.6
CS	28.5	26.9	5.7

According to these results, to produce 100 words of a correct transcription in the ODEC-M3 task, for example, a CATTI user should have to type only less than 20 words; the remaining 80 are automatically predicted by CATTI. That is to say, the CATTI user would save about 80% of the (typing and, in part thinking) effort needed to produce all the text manually. On the other hand, when interactive transcription is compared with post-editing, from every 100 (non-interactive) word errors, the CATTI user should have to interactively correct only less than 78. The remaining 17 errors would be automatically corrected by CATTI, thanks to the feedback information derived from other interactive corrections.

The different performance figures achieved in the different tasks can be explained by quality differences in the original images and also by the relative lexicon sizes and *bi*-gram estimation robustness. The later is particularly problematic in the case of CS which, in addition, suffers from a segmentation into relatively short, syntactically meaningless lines, which further hinders the ability of the *bi*-gram language model to capture relevant contextual information.

It is interesting to realize that CATTI is more effective for lines or sentences that have several errors; clearly, if a sentence has just one (word) error, it *must* be interactively corrected by the user and the best CATTI can do is to keep the remaining text unchanged. Obviously, this is not guaranteed by Eq. (2.4) and, in the worst case, a single word change made by the user may lead to more errors; that is, WSR might be greater than WER. To analyze this behavior, Fig. 3.15 presents WER, WSR and EFR values for increasing initial numbers of errors per sentence, for ODEC-M3 and IAMDB (similar tendencies are observed for CS).

As expected, the estimated effort reduction increases with the number of errors per sentence, which clearly assess the ability of CATTI to correct more than one error per interaction step in sentences with several wrong-recognized words. Also, for sentences with a single error, CATTI does not help at all or is even worse than post-editing. Therefore, in practice, a good implementation of a CATTI user interface should allow the user to disable CATTI predictions when doing some (single-word) corrections. Taking this into account, the results of Table 3.6 can be recomputed after excluding all the sentences with zero or one errors, leading to better EFR. Namely, the EFR becomes 17.9%, 18.4% and 6.9% for ODEC-M3, IAMDB and CS, respectively.

On Table 3.7 we can see the WSR and the EFR obtained for each task using WG search (see Sect. 3.2.1) in comparison with the corresponding WER. The WGs used in the experiments were generated with the same GSF and WIP values used

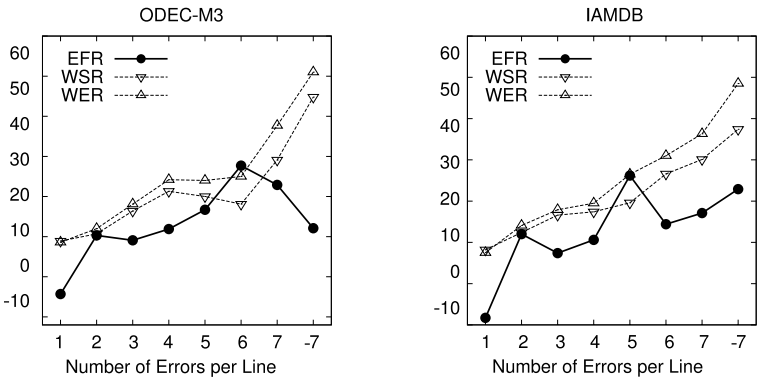


Fig. 3.15 WER, WSR and EFR (all in %) for varying number of errors per sentence, for ODEC-M3 (*left*) and IAMDB (*right*) corpora

Table 3.7 Performance of non-interactive off-line HTR (WER) and CATTI (WSR), along with the relative difference between them (Estimated Effort-Reduction—EFR) using the WG-based search approach. All results are percentages

Corpus	WER	WSR	EFR
ODEC-M3	22.9	21.5	6.1
IAMDB	25.3	22.5	11.1
CS	28.5	27.7	2.8

for the baseline results. As we have expected, the results obtained using the Viterbi-based search are better than those obtained with WGs. This is owing to the fact that the WG is just a pruned version of the Viterbi search trellis. Therefore, not all the possible transcriptions for the input handwritten text image are available, leading to some loss of system accuracy. However, the computational cost of using WGs is much lower than that using Viterbi adaptation, allowing in the former case, the user to interact in real-time with the system.

Nevertheless, from the reported results of Tables 3.6 and 3.7, it is clear that the estimated saved human effort (EFR) to produce error-free transcriptions with this CATTI approach is reduced in all the tasks. Furthermore, as previously explained, CATTI has the change to be more effective for lines/sentences with several errors. The EFR recomputed after excluding all the sentences with zero or one errors using the WG search approach are 6.8%, 12.9% and 3.8% for ODEC-M3, IAMDB and CS, respectively.

PA-CATTI Results

As commented at the end of Sect. 3.3.1, in order to be effective and fully useful, PA-CATTI approach requires short response times to emit a new suffix each time a

Table 3.8 Performance of the PA-CATTI with the single-PA interaction mode scenario (WSR single PA), along with the Estimated Effort-Reduction computed for WSR single PA with respect to conventional CATTI WSR (EFR_{CATTI}) and WSR single PA with respect to the non-interactive HTR WER (EFR_{PEDIT}). All results are percentages

Corpus	WSR single PA	EFR_{CATTI}	EFR_{PEDIT}
ODEC-M3	18.2	15.3	20.5
IAMDB	18.6	17.3	26.5
CS	23.7	14.4	16.8

PA is performed. A search implementation based on WG techniques is the solution which best fits this requirement. Table 3.8 reports the results obtained with the new single-PA interaction mode, which is the simplest one of the PA-related scenarios explained in Sect. 3.3. The first column shows the WSR obtained using the single-PA interaction mode, whereas the second and third columns show respectively the relative differences between the WSR single PA with respect to the conventional CATTI WSR (see second column of Table 3.7), and with respect to the WER of a conventional HTR system followed by human post-editing (see first column of Table 3.7).

According to Table 3.8, the estimated human effort to produce error-free transcription using PA-CATTI is significantly reduced with respect to using the conventional CATTI approach, and of course, with respect to the non-interactive HTR followed by manual post-editing. For example, in the IAMDB task, the new interaction mode can save about 26% of the overall effort, whereas the conventional CATTI would only save 11.1% using the WG-based search approach, or 16.6% using the Viterbi-based search approach (reported in Table 3.6).

Figure 3.16 plots the WSR, the EFR and the *Pointer-Action Rate* (PAR) as a function of the maximum number of allowed PAs before the user decides to write the correct word. These results are reported for both ODEC-M3 and IAMDB corpora (similar tendency has been observed for CS corpus as well). The EFR values have been computed between corresponding WSR and WER. From the both plots, it is revealed that a good trade-off between EFR and PAR can be obtained, for example, by setting the maximum number of PAs to 3, for which a significant amount of expected user effort is saved with a fairly low number of extra PAs per word.

MM-CATTI Results

The aim of these experiments is to assess the effectiveness of MM-CATTI in the scenarios described in Sect. 3.4.1. Multimodal operation offers ergonomics and increased usability at the expense of the system having to deal with non-deterministic feedback signals. Therefore, the main concern here is the accuracy of the on-line HTR feedback decoding and the experiments aim to determine how much this accuracy can be boosted by taking into account information derived from the proper

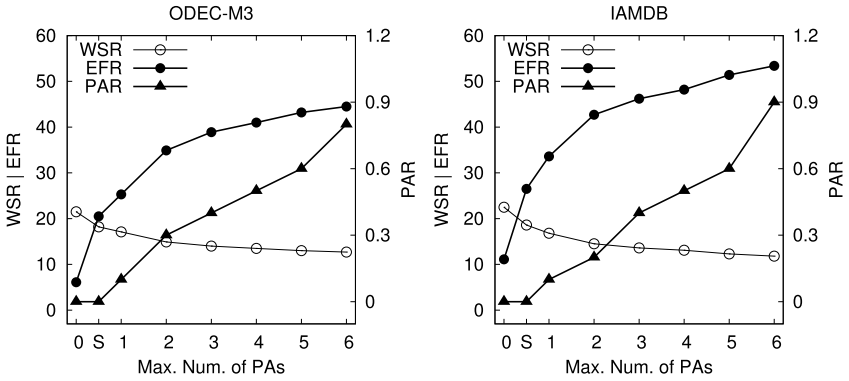


Fig. 3.16 *Word stroke ratio* (WSR), *estimated effort reduction* (EFR) and *pointer-action rate* (PAR) as a function of the maximum number of PAs allowed before the user decides to write the correct word. The first point "0" corresponds (without performing any PA) to the conventional CATTI, whereas the point "S" corresponds to the single-PA interaction scenario already discussed in Sect. 3.3

Table 3.9 Writer average MM-CATTI *feedback decoding error rates* (FER) for the different corpora and three language models: plain *uni*-gram (U, *baseline*), error-conditioned *uni*-gram (U_v) and prefix-and-error-conditioned *bi*-gram (B_v). The relative accuracy improvements for U_v and B_v with respect to U are shown in the last two columns

Corpus	FER (%)			Relative Improv. (%)	
	U	U_v	B_v	U_v	B_v
ODEC-M3	5.1	5.0	3.1	1.9	39.2
IAMDB	4.6	4.3	3.5	6.5	23.9
CS	6.4	6.2	5.8	3.1	9.3

interaction process. Ultimately, experiments aim at assessing which degree of synergy can actually be expected by taking into account *both* interactivity and multimodality.

Table 3.9 presents the writer average *feedback decoding error rates* (FER) for the different corpora considered and three language models which embody increasingly strong interaction-derived constraints (see Sect. 3.4.1). The first one corresponds to a plain *uni*-gram estimate of $P(d)$, already reported in "Baseline On-line HTR Results" in Sect. 3.6.2 as a *baseline*. The second corresponds to an error-conditioned *uni*-gram estimate of $P(d \mid v)$ (Eq. (3.14)). The third model is a prefix-and-error-conditioned *bi*-gram estimate of $P(d \mid p', v)$ (Eq. (3.15)). These models are derived from the original language models employed for the main, off-line HTR system, as explained in Sect. 3.4.1. As observed in Table 3.9, feedback decoding accuracy increases significantly as more interaction-derived constraints are taken into account.

As a final overview, Table 3.10 summarizes all the CATTI and MM-CATTI results obtained in this chapter. The forth and fifth columns show respectively the

Table 3.10 From left-to-right: post-editing corrections (WER), interactive corrections needed (WSR), e-pen *feedback decoding error rate* for baseline case (FER_{BL}) and multimodal decoding (FER_{MM}), overall multimodal interactive corrections (WSR_{MM}), and overall estimated effort reduction (EFR) achieved by the proposed approaches. All results are percentages

Corpus	Post-edit	CATTI	MM-CATTI			Overall EFR	
	WER	WSR	FER_{BL}	FER_{MM}	WSR_{MM}	CATTI	MM-CATTI
ODEC-M3	22.9	18.9	5.1	3.1	19.5	17.5	14.8
IAMDB	25.3	21.1	4.6	3.5	21.8	16.6	13.8
CS	28.5	26.9	6.4	5.8	28.4	5.6	0.4

e-pen FER baseline (BL) and FER multimodal decoding (MM), while the sixth column reports the total MM-CATTI WSR achieved. These figures correspond to the three-writers averaged decoding errors reported in Table 3.9. The last two columns show the overall estimated effort reductions (EFR) in both the conventional CATTI and MM-CATTI approaches.

The MM-CATTI EFR is calculated under the simplifying (but reasonable) assumption that the cost of keyboard-correcting a feedback on-line decoding error is similar to that of another on-line touchscreen interaction step.[6] That is, each correction using keyboard is counted twice: one for the failed touch-screen attempt and another for the keyboard correction itself. According to these results, the expected user effort for the more ergonomic and user-preferred touch-screen-based MM-CATTI is only moderately higher than that of CATTI in the ODEC-M3 and on the IAMDB corpora. On the CS corpora the results shown that the expected user effort is very similar to the expected effort on a post-editing system. However, this extra human effort entails an human-machine interaction more easier and comfortable.

3.7 Conclusions

In this chapter, the approaches CATTI, PA-CATTI and MM-CATTI presented in Sects. 3.1, 3.3 and 3.4 have been tested in three different tasks: ODEC, IAMDB and CS. These tasks involve the transcription of handwritten answers from survey forms, handwritten full English sentences of different categories (editorial, religion, fiction, love, humor, ...) and an ancient handwritten document from 1853, respectively.

At deeper depths it has been proposed a new interactive, on-line framework, which combines the efficiency of automatic HTR system with the accuracy of the user in the transcription of handwritten documents. We have called this approach

[6]This is most probably a pessimistic assumption since, in this application, interaction through touch-screen is clearly more ergonomic than through keyboard. Moreover, in practice, it seems often preferable to try again a failed touch-screen correction, rather than typing a definitive fix on the keyboard.

"Computer Assisted Transcription of Text Images" (CATTI). Here, the words corrected by user become part of increasingly longer prefixes of the final target transcription. These prefixes are used by the CATTI system to suggest new suffixes that user can iteratively accept or modify until a satisfactory, correct target transcription is finally produced. The experimental results obtained in the three above-mentioned tasks are encouraging and show that the CATTI approach speed up the human error-correction process.

Moreover, two different search-decoding implementations have been tested. The first one is based directly on the Viterbi algorithm, whereas the second one on word-graph techniques. From the obtained results, it can be concluded that, although the results obtained using the Viterbi-based approach are better than those using word-graph, the last one is preferable because the accuracy loss is not too high and the computational cost is much lower. In fact, this last issue allows the human transcriber to interact with the system in real time.

In order to foster the usability and ergonomics of CATTI, a new interaction way was proposed by considering what we called "Pointer Action" (PA-CATTI). This consists in that the system takes advantage of the positioning made by the user prior to correct the following word error, by proposing quickly (from that position) a new, hopefully more correct prediction. Therefore, PA-based user-feedback goes some way to anticipating upcoming user corrections. Implementation of PA-CATTI was carried out also on the base of word-graphs, which are the best solution to the fast-reaction-time required by PA-CATTI to have an acceptable usability. From experimental results it can be seen that this new kind of user-feedback can produce significant benefits, in terms of word stroke reductions, and this is specially noticeable for single-PA interaction scenario where the new prediction is obtained practically without extra human effort.

We have also studied the use of on-line touch-screen handwritten pen strokes as a complementary means to input the required CATTI correction feedback. We have called this multimodal approach "MM-CATTI". From the results, we observe that the use of this more ergonomic feedback modality comes at the cost of only 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 MM-CATTI ability to use interaction-derived constraints to considerably improve the on-line HTR feedback decoding accuracy. Clearly, this would have not been possible if just a conventional, off-the-shelf on-line HTR decoder were trivially used for the correction steps.

The advantage of CATTI, PA-CATTI and MM-CATTI over traditional HTR followed by post-editing goes beyond the good estimates of human effort reductions achieved. When difficult transcription tasks with high WER are considered, expert users generally refuse to post-edit conventional HTR output. In contrast, the proposed interactive approaches constitute a much more natural way of producing correct text. With an adequate user interface, CATTI, PA-CATTI or MM-CATTI let the users be dynamically in command: if predictions are not good enough, then the user simply keeps typing at his/her own pace; otherwise, he/she can accept (partial) predictions and thereby save both thinking and typing effort.

It should be mentioned here that, in addition to the laboratory experiments reported in previous section, a complete CATTI prototype (which includes PA and multimodality) has been implemented (see Sect. 12.2) and already submitted to preliminary, informal tests with real users. According to these tests, the system does meet the expectations derived from the laboratory experiments; both in terms of usability and performance. This is particularly true for the on-line HTR feedback decoding accuracy: even though the on-line HTR HMMs were trained from artificially built words using UNIPEN character samples, the accuracy in real operation with real users is observed to be similar to that shown in the laboratory results here reported. Of course even higher accuracy can be easily achieved by retraining the models with the text handwritten by the actual users.

References

1. Amengual, J. C., & Vidal, E. (1998). Efficient error-correcting Viterbi parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(10), 1109–1116.
2. Bazzi, I., Schwartz, R., & Makhoul, J. (1999). An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(6), 495–504.
3. Brakensiek, A., Rottland, J., Kosmala, A., & Rigoll, G. (2000). Off-line handwriting recognition using various hybrid modeling techniques and character n-grams. In *Proceedings of the international workshop on frontiers in handwriting recognition (IWFHR'00)* (pp. 343–352), Amsterdam, The Netherlands.
4. Chelba, C., & Jelinek, F. (1999). Recognition performance of a structured language model. In *Proceedings of European conference on speech communication and technology (Eurospeech)* (Vol. 4, pp. 1567–1570).
5. Chen, C. H. (Ed.) (2003). *Frontiers of remote sensing information processing*. Singapore: World Scientific.
6. Drira, F. (2006). Towards restoring historic documents degraded over time. In *Proceedings of the international conference on document image analysis for libraries (DIAL'06)* (pp. 350–357), Washington, DC, USA. Los Alamitos: IEEE Computer Society.
7. Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., & Janet, S. (1994). UNIPEN project of on-line data exchange and recognizer benchmarks. In *Proceedings of the international conference on pattern recognition (ICPR'94)* (pp. 29–33), Jerusalem, Israel.
8. Huang, B. Q., Zhang, Y. B., & Kechadi, M. T. (2007). Preprocessing techniques for online handwriting recognition. In *Proceedings of the international conference on intelligent systems design and applications (ISDA'07)* (pp. 793–800), Washington, DC, USA. Los Alamitos: IEEE Computer Society.
9. Jaeger, S., Manke, S., Reichert, J., & Waibel, A. (2001). On-line handwriting recognition: the NPen++ recognizer. *International Journal on Document Analysis and Recognition*, *3*(3), 169–181.
10. Jelinek, F. (1998). *Statistical methods for speech recognition*. Cambridge: MIT Press.
11. Johansson, S., Atwell, E., Garside, R., & Leech, G. (1996). *The tagged LOB corpus, user's manual*. Norwegian Computing Center for the Humanities, Bergen, Norway.
12. Lowerre, B. T. (1976). *The harpy speech recognition system*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.
13. Marti, U.-V., & Bunke, H. (1999). A full English sentence database for off-line handwriting recognition. In *Proceedings of the international conference on document analysis and recognition (ICDAR'99)* (pp. 705–708), Washington, DC, USA. Los Alamitos: IEEE Computer Society.

14. Marti, U.-V., & Bunke, H. (2001). Using a statistical language model to improve the preformance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, *15*(1), 65–90.
15. Marti, U.-V., & Bunke, H. (2002). The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, *5*(1), 39–46.
16. Ogawa, A., Takeda, K., & Itakura, F. (1998). Balancing acoustic and linguistic probabilities. In *Proceedings of the IEEE conference acoustics, speech and signal processing (ICASSP'98)* (Vol. 1, pp. 181–184), Seattle, WA, USA.
17. O'Gorman, L., & Kasturi, R. (Eds.) (1995). *Document image analysis*. Los Alamitos: IEEE Computer Society.
18. Parizeau, M., Lemieux, A., & Gagné, C. (2001). Character recognition experiments using UNIPEN data. In *Proceedings of the international conference on document analysis and recognition (ICDAR'01)* (pp. 481–485).
19. Pastor, M., Toselli, A. H., & Vidal, E. (2004). Projection profile based algorithm for slant removal. In *Lecture notes in computer science: Vol. 3212. Proceedings of the international conference on image analysis and recognition (ICIAR'04)* (pp. 183–190), Porto, Portugal. Berlin: Springer.
20. Pastor, M., Toselli, A. H., & Vidal, E. (2005). Writing speed normalization for on-line handwritten text recognition. In *Proceedings of the international conference on document analysis and recognition (ICDAR'05)* (pp. 1131–1135), Seoul, Korea.
21. Plamondon, R., & Srihari, S. N. (2000). On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 63–84.
22. Ratzlaff, E. H. (2003). Methods, report and survey for the comparison of diverse isolated character recognition results on the UNIPEN database. In *Proceedings of the international conference on document analysis and recognition (ICDAR'03)* (Vol. 1, pp. 623–628), Edinburgh, Scotland.
23. Romero, V., Pastor, M., Toselli, A. H., & Vidal, E. (2006). Criteria for handwritten off-line text size normalization. In *Proceedings of the IASTED international conference on visualization, imaging, and image processing (VIIP'06)*, Palma de Mallorca, Spain.
24. Romero, V., Toselli, A. H., Rodríguez, L., & Vidal, E. (2007). Computer assisted transcription for ancient text images. In *Lecture notes in computer science: Vol. 4633. Proceedings of the international conference on image analysis and recognition (ICIAR'07)* (pp. 1182–1193). Berlin: Springer.
25. Toselli, A., Juan, A., & Vidal, E. (2004). Spontaneous handwriting recognition and classification. In *Proceedings of the international conference on pattern recognition (ICPR'04)* (pp. 433–436), Cambridge, UK.
26. Toselli, A. H., Juan, A., Keysers, D., González, J., Salvador, I., Ney, H., Vidal, E. & Casacuberta, F. (2004). Integrated handwriting recognition and interpretation using finite-state models. *International Journal of Pattern Recognition and Artificial Intelligence*, *18*(4), 519–539.
27. Toselli, A. H., Pastor, M., & Vidal, E. (2007). On-line handwriting recognition system for Tamil handwritten characters. In *Lecture notes in computer science: Vol. 4477. Proceedings of the Iberian conference on pattern recognition and image analysis (IbPRIA'07)* (pp. 370–377), Girona, Spain. Berlin: Springer.
28. Toselli, A. H., Romero, V., Rodríguez, L., & Vidal, E. (2007). Computer assisted transcription of handwritten text. In *Proceedings of the international conference on document analysis and recognition (ICDAR'07)* (pp. 944–948), Curitiba, Paraná, Brazil. Los Alamitos: IEEE Computer Society.
29. Toselli, A. H., Romero, V., & Vidal, E. (2008). Computer assisted transcription of text images and multimodal interaction. In *Lecture notes in computer science: Vol. 5237. Proceedings of the joint workshop on multimodal interaction and related machine learning algorithms* (pp. 296–308), Utrecht, The Netherlands.
30. Toselli, A. H., Romero, V., Pastor, M., & Vidal, E. (2009). Multimodal interactive transcription of text images. *Pattern Recognition*, *43*(5), 1814–1825.

31. Vuori, V., Laaksonen, J., Oja, E., & Kangas, J. (2001). Speeding up on-line recognition of handwritten characters by pruning the prototype set. In *Proceedings of the international conference on document analysis and recognition (ICDAR'01)* (pp. 0501–0507), Seattle, Washington.
32. Young, S., Odell, J., Ollason, D., Valtchev, V., & Woodland, P. (1997). *The HTK book: hidden Markov models toolkit V2.1*. Cambridge Research Laboratory Ltd.
33. Zimmermann, M., Chappelier, J.-C., & Bunke, H. (2006). Off-line grammar-based recognition of handwritten sentences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(5), 818–821.