Chapter 2
Computer Assisted Transcription: General Framework

With Contribution Of: Verónica Romero and Luis Rodriguez.

Contents

This chapter described the common basics on which are grounded the computer assisted transcription approaches described in the three subsequent chapters: Chaps. 3, 4 and 5. Besides, a general overview is provided of the common features characterizing the up-to-date systems we have employed for handwritten text and speech recognition.

Specific mathematical formulation and modeling adequate for interactive transcription of handwritten text images and speech signals are derived from a particular instantiation of the interactive–predictive general framework already introduced in Sect. 1.3.3. Moreover, on this ground and by adopting the *passive left-to-right interaction protocol* described in Sect. 1.4.2, the two basic computer assisted handwriting and speech transcription approaches were developed (detailed in Chaps. 3 and 4, respectively), along with the evaluation measures used to assess their performance.

2.1 Introduction

The tasks of transcribing handwritten documents and speech signals are becoming an important research topic, specially because of the increasing number of online digital libraries publishing large quantities of digitized legacy manuscripts and

A.H. Toselli et al., *Multimodal Interactive Pattern Recognition and Applications*,
DOI 10.1007/978-0-85729-479-1_2,

audio-(visual) documents. The vast majority of this material, hundreds of terabytes worth of digital text-image and speech-signal data, remain waiting to be transcribed into a textual electronic format (such as ASCII or PDF) that would provide new ways of indexing, consulting and querying these documents.

Up-to-date systems for both handwritten text recognition (HTR) and automatic speech recognition (ASR) share the same consolidated technology based on Hidden Markov Models (HMMs), whose fundamentals will be given in the succeeding Sect. 2.2. Unfortunately, these systems are not accurate enough to substitute the users in these tasks. The open vocabulary issue added to the existing variety of handwriting styles and the intrinsic complexity of the acoustic signal, explain most of the difficulties encountered by these recognition systems. In order to produce adequately good transcriptions using such systems, once the full automatic recognition process of one document has finished, heavy human expert revision is required. But usually, given the high error rates involved, such a post-editing solution is quite inefficient and uncomfortable for the human correctors.

Depending on the specific requirements of the different transcription tasks, two kind of produced transcription qualities are distinguished: transcriptions containing some controlled amount of errors and totally correct transcriptions. Transcriptions of the first type can be used as metadata for indexing, consulting and querying (handwriting/audio) documents, while the second type corresponds to the conventional literal transcriptions, such as, for example, the so-called *paleographic transcription* in the case of old manuscript documents.

Transcription systems of the first type look for reducing error location effort, by marking (if needed) recognized words for the user to supervise and correct, for which the recognition system is not confident enough. While this partial supervision does not guarantee perfect transcriptions, it does help the system to adaptively train its statistical models for greater accuracy. Moreover, if (hopefully minor) recognition errors can be tolerated in non-supervised parts, it might result in considerable supervision effort reduction. This *semisupervised, active learning* approach has been developed within the GIDOC system [6] for handwriting transcription, which is extensively described in Chap. 5.

In the case of totally correct transcriptions are needed, the approaches of the so-called *Computer Assisted Transcription of Text Images* (CATTI) and *Computer Assisted Speech Transcription* (CATS) can be used as an effective alternative to post-editing. In these *interactive–predictive approaches*, the recognition system and the human transcriber tightly cooperate to generate the final transcription, thereby combining human accuracy with recognition system efficiency. The two implemented systems (CATTI and CATS) are presented in detail in Chaps. 3 and 4, while the remaining sections of this chapter are devoted to the common general framework on which these two approaches are based.

2.2 Common Statistical Framework for HTR and ASR

Following the classical pattern recognition paradigm exposed in Sect. 1.2, both handwriting and speech recognition tasks seek to decode their respective symbolic

and phonetic representation in terms of characters and/or words. In the already consolidated approach for both systems, the input pattern x is a sequence of feature vectors describing a *text image or speech signal* along its corresponding horizontal or time axis. On the other hand, system hypotheses are sequences of transcribed words, w, from a certain language. In this context, following a similar procedure as for Eqs. (1.1) and (1.2) (see Sect. 1.2):

$$\begin{aligned}\hat{w} = \arg\max_{w} \Pr(w \mid x) &= \arg\max_{w} \Pr(x \mid w) \cdot \Pr(w) \\ &\approx \arg\max_{w} P(x \mid w) \cdot P(w), \end{aligned} \tag{2.1}$$

where $P(x \mid w)$ is given by morphological word models and $P(w)$ by a language model.

Morphological word models are built from respective *lexical entries (words)*, each of them modeled by a stochastic finite-state automaton which represents all possible concatenations of individual characters/phonemes that may compose the word. In turn, each character/phoneme is modeled by continuous density left-to-right HMM, with a Gaussian mixture per state. This mixture serves as a probabilistic law to the emission of feature vectors on each model state. By embedding the character/phoneme HMMs into the edges of the word automaton, a *lexical HMM* is obtained. These HMMs (morphological word models) estimate the word-conditional probabilities $P(x \mid w)$ of Eq. (2.1). The number of states as well as Gaussians of the HMMs define the total amount of parameters to be estimated. Therefore, these numbers need to be empirically tuned to optimize the overall performance for the given amount of training vectors available.

In general, the true probability $\Pr(w)$ of a specific concatenation of words w into text lines or sentences is given by

$$\Pr(w) = \Pr(w_1) \cdot \prod_{i=2}^{l} \Pr\left(w_i \mid w_1^{i-1}\right), \tag{2.2}$$

where $\Pr(w_i \mid w_1^{i-1})$ is the probability of the word w_i when we have already seen the sequence of words $w_1 \cdots w_{i-1}$. The sequence of words prior to w_i is the so-called history. In practice, estimating the probability $\Pr(w)$ for each possible w sequence can become difficult since sentences can be arbitrarily long and, in fact, many of them may be missing in the training set used for estimation process. Noting that for a vocabulary of $|V|$ different words, the number of distinct histories of length i is $|V|^{i-1}$. Therefore, the correct estimation of $\Pr(w)$ can be really unworkable, and for which reason this is often approximated using smoothed n-gram models, which use the previous $n-1$ words to predict the next one:

$$\Pr(w) \approx P(w) = P(w_1) \cdot \prod_{i=2}^{n-1} P\left(w_i \mid w_1^{i-1}\right) \cdot \prod_{i=n}^{l} P\left(w_i \mid w_{i-n+1}^{i-1}\right). \tag{2.3}$$

In this chapter, as well as in Chaps. 3, 4 and 5, we are going to use n-grams *language model* (bi-grams in most of the cases), with Kneser–Ney back-off smoothing [3, 4], estimated from the given transcriptions of the trained set. Bi-gram models estimate the probability $P(w)$ in Eq. (2.1), and are the basis for the "dynamic", prefix-conditioned language model $P(s \mid p)$ of Eq. (2.8), as will be explained in Sect. 2.4.

To train HMMs and n-gram language model, a corpus is required in which each training sample (handwritten text image or speech utterance) is accompanied by its correct transcription. These transcriptions must accurately describe all the elements appearing in each sample, such as phonemes in the case of speech or letters (lower-case and upper-case), symbols, abbreviations, etc., in the case of handwriting images. On one hand, HMMs are trained using a well-known instance of the expectation-maximization (EM) algorithm called forward–backward or Baum–Welch re-estimation [2]. And on the other hand, n-gram language model probabilities can be estimated from the raw text of the training transcriptions (or from external text corpora) based on the relative frequency of word sequences.

Once all the *character/phoneme*, *word* and *language* models are available, recognition of new test sentences can be performed. Thanks to the homogeneous finite-state (FS) nature of all these models, they can be easily *integrated* into a single *global* (huge) FS model, on which Eq. (2.1) is easily solved. Given an input sequence of feature vectors, the output word sequence hypothesis corresponds to a path in the integrated network that, with highest probability, produces the input sequence. This optimal path search is very efficiently carried out by the well-known (*beam-search*-accelerated) Viterbi algorithm [2]. This technique allows integration to be performed "on the fly" during the decoding process. In this way, only the memory strictly required for the search is actually allocated. The Viterbi algorithm can also be easily adapted to solve Eq. (2.6) required in the CATTI and CATS interactive framework, as will be seen in Sect. 2.3.

It should be mentioned that, in practice, HMM and bi-grams (log-)probabilities are generally "balanced" before being used in Eqs. (2.1) or (2.6). This is carried out by using a "*Grammar Scale Factor*" (GSF), the value of which is tuned empirically.

2.3 Common Statistical Framework for CATTI and CATS

The IPR paradigm framework using human feedback outlined in Sect. 1.3.3 (for *interaction with deterministic feedback*) can be directly applied to the transcription of handwritten documents and speech signals. According to this IPR paradigm, in the interactive transcription framework the system should take into account the current state to improve the following predictions. To start the process, the system makes an initial prediction consisting in a whole transcription of the input image/speech signal. Then, the user reads this prediction until an error is found. At this point, the user corrects this error, generating a new, extended prefix (the previous validated prefix plus the amendments introduced by the user). This new prefix is used by the recognition system to attempt a new prediction, thereby starting a new cycle that is

repeated until a final correct transcription is achieved. This interactive process follows the so-called "*left-to-right interactive–predictive processing protocol*" stated in Sect. 1.4.2.

Ergonomics and user preferences dictate exactly when the system should start a new cycle. Typically, it can start after each new user keystroke or after each user-entered whole word. A timed system reaction scheme is also possible, where new predictions are computed only upon detecting a short period of user inactivity. Even though keystroke-level interaction is most preferable in practice, for the sake of clarity and unless stated otherwise, only whole-word interactions will be considered in the present study. This will allow us to properly estimate the user-effort reduction achieved by the interactive transcription with respect to conventional post-editing of automatic transcriptions (see Sect. 2.6).

Formally, the interactive transcription approach can be seen as an instantiation of the problem formulated in Eq. (1.14) where, in addition to the given sequence of feature vectors x, a user-validated *prefix* p of the transcription is available. This prefix, which corresponds to the pair (h', d) in Eq. (1.14), contains information from the previous system's prediction (h') plus user's actions, in the form of amendment keystrokes (d). In this way, the HTR system should try to complete this prefix by searching for the most likely *suffix* according to the already-given Eq. (1.32) (see Sect. 1.4), which, for convenience, is shown again below:

$$\begin{aligned}\hat{s} = \arg\max_{s} \Pr(s \mid x, p) &\approx \arg\max_{s} P(s \mid x, p) \\ &= \arg\max_{s} P(x \mid p, s) \cdot P(s \mid p). \end{aligned} \tag{2.4}$$

Equation (2.4) is very similar to Eq. (2.1), where w is the concatenation of p and s. The main difference is that here p is given. Therefore, the search must be performed over all possible suffixes s of p and the language model probability $P(s \mid p)$ must account for the words that can be written after the fixed prefix p.

In order to solve Eq. (2.4), the image x can be considered split into two fragments, x_1^b and x_{b+1}^M, where M is the length of x. By further considering the boundary point b as a hidden variable, we can write:

$$\hat{s} \approx \arg\max_{s} \sum_{0 \le b \le M} P(x, b \mid p, s) \cdot P(s \mid p). \tag{2.5}$$

We can now make the *naïve* (but realistic) assumption that the probability of x_1^b given p does not depend on the suffix and the probability of x_{b+1}^M given s does not depend on the prefix and, approximating the sum over all the possible segmentations by the dominating term, Eq. (2.5) can be rewritten as

$$\hat{s} \approx \arg\max_{s} \max_{0 \le b \le M} P\left(x_1^b \mid p\right) \cdot P\left(x_{b+1}^M \mid s\right) \cdot P(s \mid p). \tag{2.6}$$

This optimization problem entails finding an optimal boundary point, $\hat{b}$, associated with the optimal suffix decoding, $\hat{s}$. That is, the signal x is actually split into

two segments, $x_p = x_1^{\hat{b}}$ and $x_s = x_{\hat{b}+1}^{M}$, the first one corresponding to the *prefix* and the second to the *suffix*. Therefore, the search can be performed just over segments of the signal corresponding to the possible suffixes and, on the other hand, we can take advantage of the information coming from the prefix to implement the language model constraints modeled by $P(s \mid p)$.

2.4 Adapting the Language Model

Perhaps the simplest way to deal with $P(s \mid p)$ is to adapt an n-gram language model to cope with the consolidated prefix. Given that a conventional n-gram models the probability $P(w)$ (where w is the concatenation of p and s, i.e the whole sentence), it is necessary to modify this model to take into account the conditional probability $P(s \mid p)$.

Let $p = w_1^k$ be a consolidated prefix and $s = w_{k+1}^l$ be a possible suffix. We can compute $P(s \mid p)$, as is shown in Eq. (2.7):

$$\begin{aligned} P(s \mid p) &= P(p, s)/P(p) \\ &= \frac{\prod_{i=1}^{l} P(w_i \mid w_{i-n+1}^{i-1})}{\prod_{i=1}^{k} P(w_i \mid w_{i-n+1}^{i-1})} = \prod_{i=k+1}^{l} P\big(w_i \mid w_{i-n+1}^{i-1}\big). \end{aligned} \tag{2.7}$$

Moreover, for the terms from $k+1$ to $k+n-1$ of this factorization, we have additional information coming from the already known words w_{k-n+2}^k, leading to:

$$\begin{aligned} P(s \mid p) &= \prod_{i=k+1}^{k+n-1} P\big(w_i \mid w_{i-n+1}^{i-1}\big) \cdot \prod_{i=k+n}^{l} P\big(w_i \mid w_{i-n+1}^{i-1}\big) \\ &= \prod_{j=1}^{n-1} P\big(s_j \mid p_{k-n+1+j}^{k}, s_1^{j-1}\big) \cdot \prod_{j=n}^{l-k} P\big(s_j \mid s_{j-n+1}^{j-1}\big), \end{aligned} \tag{2.8}$$

where $p_1^k = w_1^k = p$ and $s_1^{l-k} = w_{k+1}^l = s$. The first term of Eq. (2.8) accounts for the probability of the $n-1$ words of the suffix, whose probability is conditioned by words from the validated prefix, and the second one is the usual n-gram probability for the rest of the words in the suffix.

2.5 Search and Decoding Methods

In this section, we are going to proceed to discuss briefly some possible implementations of interactive transcription decoders based on Eq. (2.6). To begin with, a simple possibility would be to perform the decoding in two steps: first, the validated prefix p could be used to segment the signal x into x_p and x_s and, then, x_s

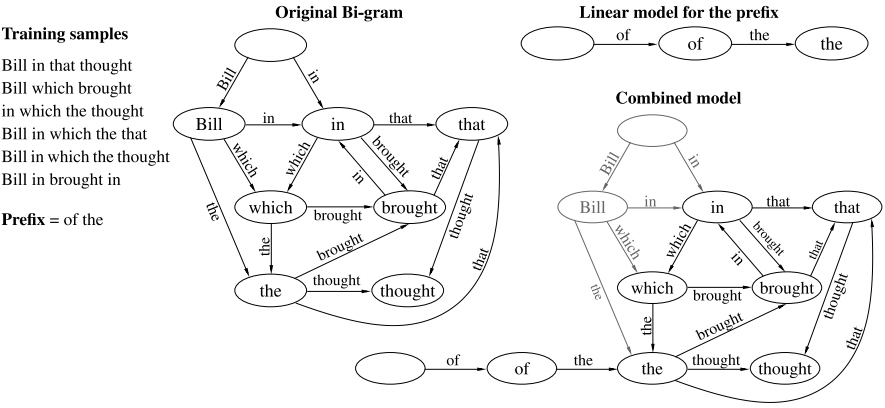

Fig. 2.1 Example of interactive transcription dynamic language model building. First, a bi-gram for the training set of the figure is obtained. Then, a linear model which accounts for the prefix "*of the*" is constructed. Finally, these two models are combined into a single *prefix-constrained model*

could be decoded by using a "suffix language model" (SLM) as in Eq. (2.8). The problem here is that the signal cannot be optimally segmented into x_p and x_s if only the information of the prefix p is considered.

A better approach is to explicitly rely on Eq. (2.6) to implement a decoding process in one step, as in classical handwriting/speech recognition. The decoder should be forced to match the previously validated prefix p and then continue searching for a suffix $\hat{s}$ according to the constraints in Eq. (2.8). Two different possible implementations for this last approach have been considered. The first one is based on the well-known Viterbi algorithm [2], while the second one is based on word-graph techniques similar to those described in [1, 5] for Computer Assisted Translation and for multimodal speech post-editing. Both implementations are described in the following sections.

2.5.1 Viterbi-Based Implementation

In this case, the search problem corresponding to Eqs. (2.6) and (2.8) can be straightforwardly solved by building a special language model which can be seen as the "concatenation" of a *linear* model which strictly accounts for the successive words in p and a "suffix language model" of Eq. (2.8). First, an n-gram with the available training set is built. Then, a linear model which accounts for the validated prefix is constructed. Finally, these two models are combined into a single model as shown in Fig. 2.1. Owing to the finite-state nature of this special language model, the search involved in Eq. (2.6) can be efficiently carried out using the Viterbi algorithm [2]. Apart from the optimal suffix decoding, $\hat{s}$, a correspondingly optimal segmentation of the x is then obtained as a by-product.

It should be noted that a direct adaptation of the Viterbi algorithm to implement these techniques leads to a computational cost that grows quadratically with the number of words of each sentence. As for each user interaction a new derived language model is dynamically built for performing the decoding search, this can be problematic (very high time-consuming process) for large sentences and/or for fine-grained (character-level) interaction schemes. Nevertheless, using word-graph techniques similar to those described in [1, 5], very efficient, linear cost search can be easily achieved.

2.5.2 Word-Graph Based Implementation

As previously explained in Sect. 1.5.1, a word graph (WG) is a data structure that represents a set of strings in a very efficient way. In handwritten text as well as speech recognition, a WG represents the transcriptions with the highest $P(w \mid x)$ (see Eq. (2.1)) of the given text image or speech utterance. In this case, the word graph is just (a pruned version of) the Viterbi search trellis obtained when transcribing the given input x.

According to Eqs. (1.38) and (1.39) (Sect. 1.5.1), the probability of a word sequence, w, in a word-graph is computed as the sum of the probabilities of all the paths that generate w, $\gamma(w)$:

$$P(w) = \sum_{\phi_w \in \gamma(w)} \prod_{i=1}^{l} p(e_i), \tag{2.9}$$

where ϕ_w is a sequence of edges $e_1, e_2, \ldots, e_l$ such that $w = \omega(e_1), \omega(e_2), \ldots, \omega(e_l)$. Given a WG, a word sequence with the greatest probability can be written as

$$\hat{w} = \arg\max_{w} \sum_{\phi_w \in \gamma(w)} \prod_{i=1}^{l} p(e_i). \tag{2.10}$$

However, as this maximization problem is NP-hard, we approximate it by means of the efficient Viterbi search algorithm:

$$P(w) \approx \max_{\phi_w \in \gamma(w)} \prod_{i=1}^{l} p(e_i), \tag{2.11}$$

$$\hat{w} \approx \arg\max_{w} \max_{\phi_w \in \gamma(w)} \prod_{i=1}^{l} p(e_i). \tag{2.12}$$

The edge probability function, $p(e)$, where $e = (i, j)$, is equal to the product of the morphological/acoustic word probability $P(x_{t(i)+1}^{t(j)} \mid \omega(e))$ of the feature vector

subsequence $x_{t(i)+1}^{t(j)}$ between i and j WG nodes, and the language model probability $P(\omega(e))$ of the given word at the edge e. That is,

$$p(e) = P\left(x_{t(i)+1}^{t(j)} \mid \omega(e)\right) \cdot P\left(\omega(e)\right). \tag{2.13}$$

During the interactive transcription process the system has to make use of this word graph in order to complete the prefixes accepted by the human transcriptor. In other words, the search problem consists in finding the target suffix s that maximizes the posterior probability $P(s \mid x, p)$ given a prefix p as described in Eq. (2.4).

The search on the WG involves two phases. The first one deals with the parsing of the previously validated prefix p over the WG, looking for a set of end nodes Q_p of the paths, whose associated word sequence is p. In the second phase, the decoder continues searching for the suffix s, from any of the nodes in Q_p, that maximizes the posterior probability given by Eq. (2.4). In terms of WG, an analogous expression of Eq. (2.6) can be obtained:

$$\hat{s} = \arg\max_{s} \max_{q \in Q_p} P\left(x_1^{t(q)} \mid p\right) \cdot P\left(x_{t(q)+1}^{m} \mid s\right) \cdot P(s \mid p). \tag{2.14}$$

The boundary point b in Eq. (2.6) is now restricted to values $t(q)\ \forall q \in Q_p$.

This search problem can be efficiently carried out using dynamic programming. In order to make the process faster, first, we apply a dynamic-programming Viterbi-like algorithm backwards from the final node to the initial one. In this way, we compute the best path and its probability from any node to the final node. Then, we look for the set of boundary nodes Q_p. Then we should only have to multiply the probability computed from the initial node to any node $q \in Q_p$ by the probability from q to the final node (previously computed), and finally, choose the node with the maximum probability score.

As the WG is a representation of a *subset* of the possible transcriptions for a source handwritten text image or the speech signal, it may happen that some prefixes given by the user cannot be exactly found in the WG. To circumvent this problem some error-correcting parsing algorithms have been implemented as we will see throughout this book for the different systems studied.

Example: Word Graph of Handwritten Text

Although the following example focuses on a specific WG obtained from the recognition of a handwritten text image, this is completely analogous to one obtained from the recognition of a speech signal. In Fig. 2.2 an example of a WG that represents a set of the possible transcriptions of the handwritten sentence "antiguos ciudadanos que en Castilla se llamaban" is shown. In this case, the function t associates each node with a horizontal position of the handwritten image. Following the WG notation given in Sect. 1.5.1, the function $\omega(e)$ relates each edge with a word hypothesis between horizontal image positions $t(i) + 1$ and $t(j)$, and the function $p(e)$ returns the probability of the hypothesis that $\omega(e)$ appears between $t(i) + 1$ and $t(j)$.

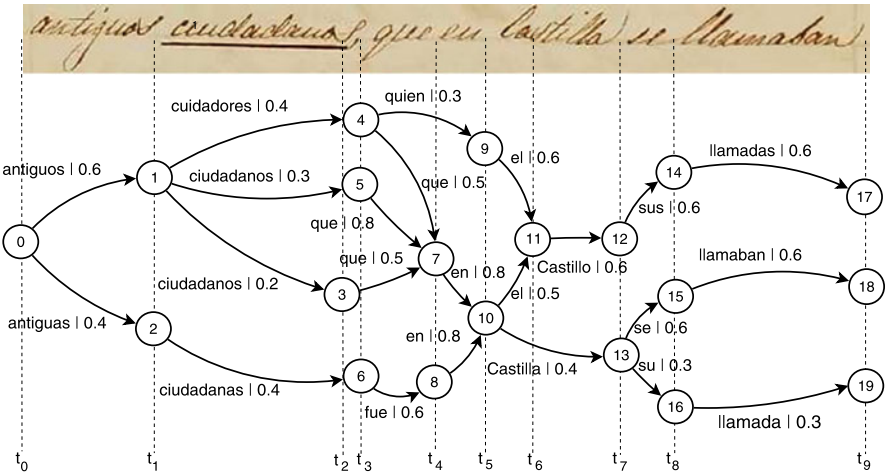

Fig. 2.2 Image positions t_i are associates with the nodes $t(0) = t_0 = 0, t(1) = t(2) = t_1, t(3) = t_2$, $t(4) = t(5) = t(6) = t_3$, $t(7) = t(8) = t_4$, $t(9) = t(10) = t_5$, $t(11) = t_6$, $t(12) = t(13) = t_7$, $t(14) = t(15) = t(16) = t_8, t(17) = t(18) = t(19) = t_9$

Given a path on the WG represented on Fig. 2.2, for example the path ϕ_1:

$$\phi_1 = \big\{(0,1),(1,5),(5,7),(7,10),(10,11),(11,12),(12,14),(14,17)\big\} \quad (2.15)$$

whose probability is computed as

$$\begin{aligned} P(\phi_1) &= p(0,1)p(1,5)p(5,7)p(7,10)p(10,11)p(11,12)p(12,14)p(14,17) \\ &= 0.6 \cdot 0.3 \cdot 0.8 \cdot 0.8 \cdot 0.5 \cdot 0.6 \cdot 0.6 \cdot 0.6 = 0.012. \end{aligned}$$

And the word sequence associated with it is $w^{(1)} =$ "*antiguos ciudadanos que en el Castillo sus llamadas*". However, ϕ_1 is not the unique path that generates $w^{(1)}$. We also have

$$\phi_2 = \big\{(0,1),(1,3),(3,7),(7,10),(10,11),(11,12),(12,14),(14,17)\big\}.$$

Therefore, the exact probability of $w^{(1)}$ is

$$P\big(w^{(1)}\big) = P(\phi_1) + P(\phi_2) = 0.012 + 0.005 = 0.017$$

and approximating that by the Viterbi algorithm, $P(w^{(1)})$ will be the probability of the path with the maximum probability, i.e:

$$P\big(w^{(1)}\big) \approx P\big(\tilde{w}^{(1)}\big) = P(\phi_1) = 0.012.$$

Now, to obtain the word sequence with the greatest probability, all the word sequences on the WG must be taken into account. Figure 2.2 shows all the word sequences on the WG and their corresponding paths:

- $w^{(1)}$ = "*antiguos ciudadanos que en el Castillo sus llamadas*",
  $\phi_1 = \{(0,1),(1,5),(5,7),(7,10),(10,11),(11,12),(12,14),(14,17)\}$,
  $\phi_2 = \{(0,1),(1,3),(3,7),(7,10),(10,11),(11,12),(12,14),(14,17)\}$,
- $w^{(2)}$ = "*antiguos ciudadanos que en Castilla se llamaban*",
  $\phi_3 = \{(0,1),(1,5),(5,7),(7,10),(10,13),(13,15),(15,18)\}$,
  $\phi_4 = \{(0,1),(1,3),(3,7),(7,10),(10,13),(13,15),(15,18)\}$,
- $w^{(3)}$ = "*antiguos ciudadanos que en Castilla su llamada*",
  $\phi_5 = \{(0,1),(1,5),(5,7),(7,10),(10,13),(13,16),(16,19)\}$,
  $\phi_6 = \{(0,1),(1,3),(3,7),(7,10),(10,13),(13,16),(16,19)\}$,
- $w^{(4)}$ = "*antiguos cuidadores quien el Castillo sus llamadas*",
  $\phi_7 = \{(0,1),(1,4),(4,9),(9,11),(11,12),(12,14),(14,17)\}$,
- $w^{(5)}$ = "*antiguos cuidadores que en el Castillo sus llamadas*",
  $\phi_8 = \{(0,1),(1,4),(4,7),(7,10),(10,11),(11,12),(12,14),(14,17)\}$,
- $w^{(6)}$ = "*antiguos cuidadores que en Castilla se llamaban*",
  $\phi_9 = \{(0,1),(1,4),(4,7),(7,10),(10,13),(13,15),(15,18)\}$,
- $w^{(7)}$ = "*antiguos cuidadores que en Castilla su llamada*",
  $\phi_{10} = \{(0,1),(1,4),(4,7),(7,10),(10,13),(13,16),(16,19)\}$,
- $w^{(8)}$ = "*antiguas ciudadanas fue en el Castillo sus llamadas*",
  $\phi_{11} = \{(0,2),(2,6),(6,8),(8,10),(10,11),(11,12),(12,14),(14,17)\}$,
- $w^{(9)}$ = "*antiguas ciudadanas fue en Castilla se llamaban*",
  $\phi_{12} = \{(0,2),(2,6),(6,8),(8,10),(10,13),(13,15),(15,18)\}$,
- $w^{(10)}$ = "*antiguas ciudadanas fue en Castilla su llamada*",
  $\phi_{13} = \{(0,2),(2,6),(6,8),(8,10),(10,13),(13,16),(16,19)\}$,

and their probabilities are

$$P\left(w^{(1)}\right) = P(\phi_1) + P(\phi_2) = 0.012 + 0.005 = 0.017, \quad P\left(w^{(6)}\right) = P(\phi_9) = 0.014,$$

$$P\left(w^{(2)}\right) = P(\phi_3) + P(\phi_4) = 0.016 + 0.007 = 0.023, \quad P\left(w^{(7)}\right) = P(\phi_{10}) = 0.003,$$

$$P\left(w^{(3)}\right) = P(\phi_5) + P(\phi_6) = 0.004 + 0.002 = 0.006, \quad P\left(w^{(8)}\right) = P(\phi_{11}) = 0.008,$$

$$P\left(w^{(4)}\right) = P(\phi_7) = 0.009, \quad P\left(w^{(9)}\right) = P(\phi_{12}) = 0.011,$$

$$P\left(w^{(5)}\right) = P(\phi_8) = 0.010, \quad P\left(w^{(10)}\right) = P(\phi_{13}) = 0.003.$$

Finally, the word sequence with the greatest probability is $w^{(2)}$ for both the exact approach ($P(w^{(2)}) = 0.023$) and the approximated by the Viterbi algorithm ($P(\tilde{w}^{(2)}) = 0.016$).

Throughout the interactive transcription process, when the user validates a prefix, the system uses this WG in order to complete this validated prefix following the already above-explained WG search phases. For example, given the prefix "*antiguos ciudadanos*", on the first phase, the decoder parses this prefix over the WG finding the set of nodes $Q_p = \{3, 5\}$, which correspond to paths from the initial node whose associated word sequence is "*antiguos ciudadanos*". Then, the decoder continues searching for the suffix s that maximizes the posterior probability from any of the nodes in Q_p. In this example, the suffix that maximizes the posterior probability is $\hat{s}$ ="*que en Castilla se llamaban*".

2.6 Assessment Measures

As was commented in Sect. 1.4.6, the corpus-based assessment paradigm could be still applicable to IPR tasks. Although recognition errors are meaningless in IPR (as the user will ensure that no errors will be produced), the correct labeling for each object can be used to determine how many interaction steps are needed to produce a correct hypothesis. In the case of interactive transcription systems, the effort needed by a human transcriber to produce correct transcriptions is estimated by the *Word Stroke Ratio* (WSR), which can be computed using the reference transcriptions. After each recognized hypothesis, the longest common prefix between the hypothesis and the reference is obtained and the first unmatching word from the hypothesis is replaced by the corresponding reference word. This process is iterated until a full match with the reference is achieved. Therefore, the WSR can be defined as the number of (word level) user interactions that are necessary to achieve the reference transcription of the text image considered, divided by the total number of reference words.

On the other hand, the quality of non-interactive transcriptions can be properly assessed with the well-known *Word Error Rate* (WER). It is defined as the minimum number of words that need to be substituted, deleted or inserted to convert a sentence recognized by the system into the corresponding reference transcription, divided by the total number of reference words. The WER is a good estimate of post-editing user effort.

These definitions make WER and WSR comparable. Moreover, the relative difference between them gives us a good estimate of the reduction in human effort that can be achieved by using an interactive transcription system with respect to using a conventional transcription system followed by human post-editing. This *estimated effort reduction* will be denoted hereafter as "EFR".

Another measure for assessing non-interactive transcriptions is the well-known *Sentence Error Rate* (SER), also called string error, defined as the number of sentences that have at least one misrecognized word. Because of SER is really stricter than WER, it is not used at all in transcription evaluation, mainly because it does not reflect the effort needed to correct all the misrecognized words within sentences. However, it is rather widely used in speech recognition (Chap. 4), machine translation (Chaps. 4 and 7) and interactive text generation (Chap. 10).

References

1. Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Ney, A. L. H., Tomás, J., & Vidal, E. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, *35*(1), 3–28.
2. Jelinek, F. (1998). *Statistical methods for speech recognition*. Cambridge: MIT Press.
3. Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *I.E.E.E. Transactions on Acoustics, Speech, and Signal Processing*, *ASSP-35*, 400–401.

4. Kneser, R., & Ney, H. (1995). Improved backing-off for n-gram language modeling. In *Proceedings of the international conference on acoustics, speech and signal processing (ICASSP)* (Vol. 1, pp. 181–184).
5. Liu, P., & Soong, F. K. (2006). Word graph based speech recognition error correction by handwriting input. In *Proceedings of the international conference on multimodal interfaces (ICMI'06)* (pp. 339–346), New York, NY, USA. New York: ACM.
6. Serrano, N., Sanchis, A., & Juan, A. (2010). Balancing error and supervision effort in interactive–predictive handwritten text recognition. In *Proceedings of the international conference on intelligent user interfaces (IUI'10)* (pp. 373–376), Hong Kong, China.

http://www.springer.com/978-0-85729-478-4

Multimodal Interactive Pattern Recognition and Applications

Toselli, A.H.; Vidal, E.; Casacuberta, F.

2011, XVI, 274 p., Hardcover
ISBN: 978-0-85729-478-4