Composition of Constraint, Hypothesis and Error Models to Improve Interaction in Human-Machine Interfaces

J.Ramon Navarro-Cerdan*, Rafael Llobet, Joaquim Arlandis, Juan-Carlos Perez-Cortes

Institut Tecnològic d'Informàtica, Universitat Politècnica de València, València, Spain

---

Abstract

Although there are many tasks where output strings are automatically generated from a set of evidence, they are not perfect and human intervention is often required to correct the result. In this paper we present a generic Symbol Input Interaction Method for Human-Machine Interfaces that combines multi-source information: an input Hypothesis Model an Error Model, a Constraint Model and a user interaction scheme.

We use Weighted Finite-State Transducers (WFSTs) to represent the different sources of information available: the initial hypotheses, the possible errors, the constraints imposed by the task (interaction language) and the user input. The fusion of these models to find the most probable output string can be performed efficiently by using carefully selected transducer operations. The proposed system initially suggests an output based on the set of hypotheses, possible errors and constraint models. Then, if human intervention is needed, a multimodal approach, where the user input is combined with the aforementioned models, is applied to produce, with a minimum user effort, the desired output. This approach offers the practical advantages of a decoupled model (e.g. input-system + parameterized rules + post-processor),

*Corresponding author at: Institut Tecnològic d'Informàtica, Ciutat Politècnica de la Innovació (Universitat Politècnica de València), Camí de Vera s/n, Building 8G - Access B; Tel.: +34 963877242; Fax.: +34 963877239

Email addresses: `jonacer@iti.upv.es` (J.Ramon Navarro-Cerdan), `rllobet@iti.upv.es` (Rafael Llobet), `arlandis@iti.upv.es` (Joaquim Arlandis), `jcperez@iti.upv.es` (Juan-Carlos Perez-Cortes)

keeping at the same time the error-recovery power of an integrated approach, where all the steps of the process are performed in the same formal machine (as in a typical HMM in speech recognition) to avoid that an error at a given step remains unrecoverable in the subsequent steps. After a presentation of the theoretical basis of the proposed multi-source information system, its application to two real world problems, as an example of the possibilities of this architecture, is addressed. The experimental results obtained demonstrate that a significant user effort can be saved when using the proposed procedure. A simple demonstration, to better understand and evaluate the proposed system, is available on the web.[1]

Keywords: multi-source information fusion, human-machine interaction, weighted finite-state transducer composition, interactive multimodal string correction.

1. Introduction

In a general architecture for a Human-Machine Interface, the entry subsystem can be any source of input data such as an OCR, a gesture or speech recognition system, a vehicle navigation or control system, a tactile input device, etc. These systems are often subject to variable amounts of error and uncertainty. Therefore, methods to detect and correct input errors and, if necessary, allowing user interaction to improve and/or validate the input data, are needed. Particularly, when the input modality has a significant inherent complexity but the language syntax and semantics of the input data are known in advance, a sophisticated symbol-input correction or post-processing method is likely to improve the efficiency of operation and the convenience for the user. Examples of complex entry subsystems are those character or gesture recognition modules in which the input data potentially shows a high level of uncertainty, or those which meet ergonomic trade-offs due to size, weight, form-factor or handling constraints, like mobile phones, touch-based keyboards, etc. All these entry methods have in common that they produce an input hypothesis which should be post-processed and validated.

In addition, the most recent systems are usually equipped with a combination of several input sources, such as a touch screen, a keyboard, a sound

[1]https://demos.iti.upv.es/hi/

input system, new sensors like accelerometers, GPS, etc. In these cases, a method capable of combining the evidence of multimodal inputs can also be useful [1, 2, 3, 4].

Formally, the role of an input interaction system is to maximize the likelihood that the strings received as input hypotheses from the different, possibly multimodal, input subsystems are correct, in the sense that they are compatible with the constraints imposed by the task (language). In this context, different methods to address the problem of error correcting and keystroke saving in input interaction systems have been proposed.

Error correcting has been traditionally handled by using a lexicon to validate the known words and asking the operator to verify or re-enter symbol by symbol the unknown ones. Alternatively, similar words to those potentially misspelled can be found in the dictionary using the Levenshtein edit distance and a dynamic programming technique at relatively high computational costs. Other specific techniques can be used to carry out approximate search in the lexicon. In [5] an excellent survey of string search methods is presented.

More sophisticated methods based on $n-$grams or on finite-state machines have also been extensively studied [6, 7, 8, 9]. In them, a candidate input string is parsed and an output string is generated from the set of transitions in the automaton with the lowest cost (highest probability). The classical technique, widely used in different fields, to find the maximum likelihood path on a finite-state machine and to perform error-correcting parsing on a regular grammar is the Viterbi Algorithm [10, 11]. In [12], OCR errors are corrected using context-dependent confusion rules and a language model, both represented by means of Finite-State Transducers (WFST). The confusion rules are extracted from a training corpus by aligning the misrecognized words of the OCR output with their corresponding ground truth. In [13] a text correction method, also based on WFST, is presented. The approach consists of three main independent phases: detecting misspelled words, generating candidate corrections and ranking corrections. Misspelled words are obtained as the difference between the set of input words and the language model, while candidate generation is done by generating a list of words that have edit distance less than k to each of the input words and selecting the subset that also exists in the dictionary. These processes are carried out using basic operations over Finite-State Automata.

All these approaches take a string provided by the symbol-input subsystem, apply a Language Model and often optimize a transformation cost using an Error Model, but, in general, they do not take into account multiple input sources or detailed probabilistic input evidence or the possibility of user-interaction in the loop to improve the resulting string.

One of the main practical advantages of our approach lies on the use of a de-coupled model, where the different parts of the system remain individual processes, but the optimality of the whole composition is guaranteed by fully propagating the uncertainty from each process to the next. In the recent work [14], a generic formulation of this idea is discussed and presented in an elegant way using relational algebra and a probabilistic scheme. The discussion describes basic de-coupled approaches as those where each stage merely passes a simple (canonical) output to the next stage. In that case, errors in the early stages can produce a cascade of unrecoverable errors in later stages, since each component makes a locally optimal choice without taking into account the rest of the components. However, if the uncertainty in the predictions at all stages of the system (pipeline) are accounted for, global optimality is attainable.

In previous conference papers, preliminary versions of the work presented here can be found: In [15] and [16], we propose the use of Weighted Finite-State Transducers (WFSTs) for stochastic error-correcting language modeling applied to OCR hypotheses. The combination of different models, including an error model, and the use of a vector of input hypotheses with their *a-posteriori* probabilities are introduced there, however no interaction nor multimodality is allowed in the proposed scheme. In [17], an application to License Plate Recognition, introducing the combination of several possibly incomplete and unaligned hypotheses is presented. In [18], a preliminary and simplified version of the final formulation proposed here, in which only symbol sequences are allowed as input data, is introduced. This new paper includes several novel contributions. The concept of multimodality is introduced and new example applications that use this scheme are presented. The input data can be any type of information (text, speech or, in general, any set of evidences) as long as it can be represented by a Finite State Machine. There is also an error model integrated into the user interaction subsystem, which allows the recovery of errors in the user input stream, while the user interacts with the system. The fusion of interactivity and multiple input modalities leads to a more general concept of *constraint model* employed here to replace the notion of *language model* used so far in the cited works. The scope of the tasks to which this new paradigm can be applied is wider and more diverse.

The rest of the paper is organized as follows: In the next section, the general interaction system is described. In Section 3, a brief description about the foundations of WFSTs is presented. In Section 4, the proposed method is explained in detail. In Section 5, two example applications of the proposed method are presented: an interactive multimodal OCR post-processing system and an efficient entry data in GPS devices. Experimental results and conclusions are presented in Sections 6 and 7.

2. The interaction system: a practical scenario

In this Section, a simple but practical scenario is presented to better understand the proposed method, even though the methodology is explained in more detail in Section 4.

We will consider a practical task involving a massive form processing task in which large amounts of handwritten documents must be converted into text. Examples of these applications are census data acquisition, surveys or invoices processing to name only a few.

Usually, this process consist of three main phases: a) forms scanning, b) text transcription of the form fields by means of an OCR and c) human revision, correction and validation. The most costly phase in terms of time and human resources is, by far, the last one, as OCR engines are far to be error-free, especially when working with handwritten text, which leads to long manual validation times.

However, it is possible to significantly reduce the cost of the final process if a) an OCR post processing and b) an efficient interaction system are incorporated into the process chain. In the first place, the OCR post processing phase can reduce the errors initially produced by the OCR engine, which will reduce the number of fields that will need to be corrected (or even validated if the confidence given by the classifier is high enough). This can be achieved if there is new evidence known beforehand that can be incorporated into the system, such as information related with the language associated to each form field (a dictionary, syntactic or grammatical restrictions, etc.) or the probabilities with which the OCR engine confuses the different symbols (i.e. the confusion matrix). Additionally, efficient interaction can reduce the time needed to correct the remaining errors after the previous phase, understanding efficiency as obtaining the correct string with the minimum interaction (e.g. keystrokes). Again, an efficient interaction can be achieved if new evidences, as the information introduced by the user or the features of the interaction device (keyboard type, keyboard layout, key distances, etc.), which provide an estimation of the common error patterns associated to the interaction subsystem, are combined with the previous available evidences instead of being considered separately.

We propose an interactive multimodal system that combines all those sources of information, including probabilistic evidence, to deal with the problem of processing an initial hypothesis string coming from any input sub-system (for example, an OCR engine) in order to obtain an improved output according to a Constraint Model.

In detail, the following sources of information are typically available: a) the input hypothesis, which can be as simple as a sequence of symbols or as complex as a graph representing a set of probabilistic phrases belonging to a grammar, that we call the Hypothesis Model (HM); b) a model of the expected errors in the input hypothesis and their probabilities, that we call the Error Model (EM); c) the language of the interaction task, which the expected string belongs to, called the Constraint Model (CM); and, when needed, d) the information from the user interaction, that we call the Interaction Model (IM).

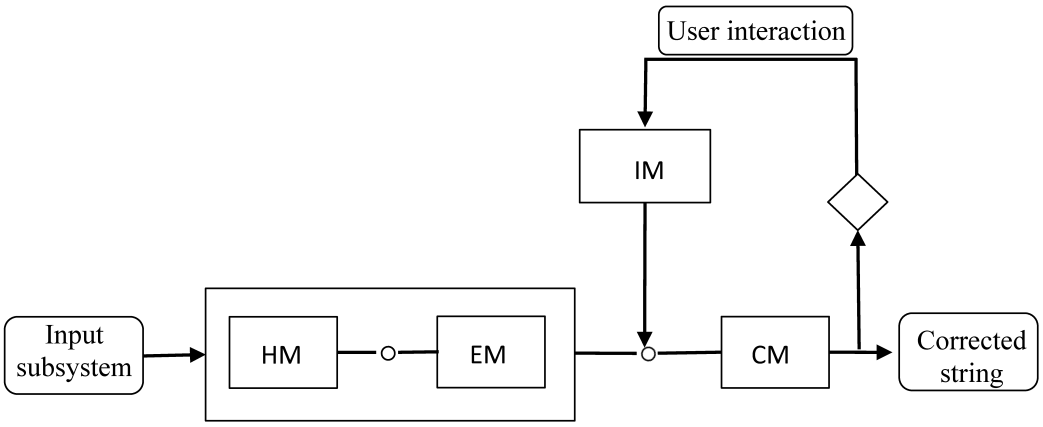

Figure 1: Interactive multimodal post-processing scheme.

We represent each of these models (HM, EM, CM, IM) by a Weighted Finite-State Transducer (WFST) and compose all of them to address the problem of finding the most probable string in the CM according to the current hypothesis (HM), the possible errors (EM) and the user entry (IM). This problem can be solved by properly composing all these models and finding the shortest (most probable) path in the composed transducer. We have used the OpenFST library [22] to create the transducers and to perform the operations mentioned above.

Figure 1 shows the proposed multimodal interaction system, where the operator $\circ$ denotes the composition of transducers. In a first step, HM, EM and CM are composed, which allows to transduce any string $x \in L_{HM}$ into any other string $y \in L_{CM}$ according to the error operations defined in the EM (assuming that the EM can transduce any string into any other). In the decoding phase, the most probable of these transductions can be found. If the proposed output is not the one the user expects, then interaction is allowed to improve the resulting string. In this case, an additional WFST called IM is created and composed with HM$\circ$EM, which imposes new constraints to the system. It is even possible to model the fact that the user can make mistakes while interacting with the system by including a new Error Model associated to the IM representing the most common user error patterns.

Although in this paper the technique is seen as a transformation from the HM to the CM, this method can also be interpreted as a noisy channel scheme, where the observed string is considered a noisy version of the intended string [19], i.e. the CM generates an error-free string with a given probability and the EM (noise model) decides whether or not to insert errors to produce the observed string (HM) [20].

To illustrate these ideas, let us consider a simple example based on the aforementioned practical scenario. If the goal is to process a form field that must contain, for instance, the name of an animal, and only {cat, cow, bat, goat} are possible options, then the WFST representing the CM would be that shown in Figure 2. On the other hand, if the OCR output is, say, the string *aat*, the HM representing this output would be that shown in Figure 3 (top). Furthermore, the fact that the OCR classifier gave, in some cases, several class hypotheses per symbol, together with their *a posteriori* probabilities, could be also modeled: for example [a:1] for the first symbol, [a:0.6, o:0.4] for the second and [t:0.8, d:0.2] for the last one, in which case the HM would be the one shown in Figure 3 (bottom). Finally, assuming for simplicity, that our alphabet is restricted to $\Sigma = \{a,b,c,g,o,t,w\}$, if it is known (according to some previous empirical tests) that the probabilities of confusion between symbols associated with the OCR classifier (confusion matrix) and the probabilities of insertion and deletion are those shown in Figure 4 (left), then the associated EM would be the transducer shown in Figure 4 (right) (where, for readability, only a few arcs have been represented). This model represents the edit operations (substitution, insertion and deletion) allowed to transform the initial hypothesis into a valid string. Although the confusion matrix shown in the example contains a number of zeros, it is usually smoothed in order to allow the substitution of any pair of symbols, since it is often considered possible even if it has not been observed in the matrix estimation process.

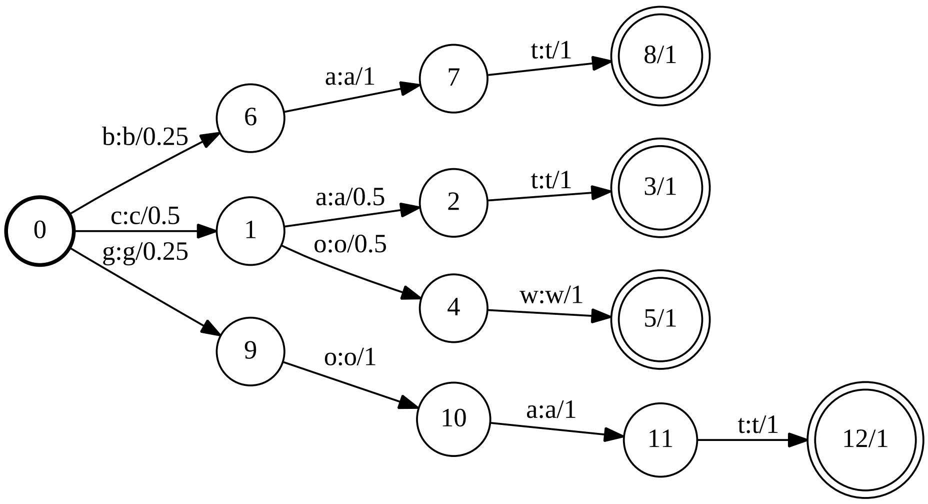

Figure 2: Constraint Model representig the dictionary {*cat, cow, bat, goat*}. Double-circled states are final states and include the probability to be final.

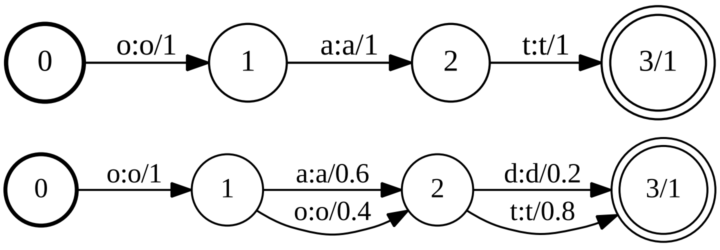

Figure 3: Hypothesis Model representig the OCR output string *aat* without *a posteriori* probabilities (top) and with more than one possible class per character together with its probabilities (down).

If all these models are properly composed, the shortest path of the resulting transducer gives us the most likely transformation of a hypothesis in the HM into a string in the CM through the error operations defined in the EM, where the input/output symbols along this shortest path represent

	a	b	c	g	o	t	w	ε
a	**0.7**	0	**0.1**	0	**0.2**	0	0	**0.1**
b	0	0.7	0.1	0	0.2	0	0	0.1
c	0.1	0	0.7	0	0.2	0	0	0.1
g	0	0	0.1	0.7	0.2	0	0	0.1
o	0	0	0.3	0	0.7	0	0	0.1
t	0	0.2	0.1	0	0	0.7	0	0.1
w	0.1	0	0.1	0	0	0	0.8	0.1
ε	**0.1**	0.1	0.1	0.1	0.1	0.1	0.1	0.1

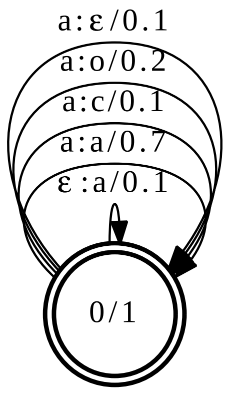

Figure 4: Example of OCR confusion matrix together with insertion and deletion probabilities (symbol ε), with rows representing input symbols and columns representing output (left) and its associated Error Model (right). For the sake of readability, only probabilities in bold have been showed in the model.

the sequence of operations needed to transform one into the other, and the product of the probabilities at each transition, the likelihood of the transformation. In our example, {a/c, a/a, t/t}, i.e., *aat* → *cat*, with probability $p = 0.05 \times 0.21 \times 0.56 = 0.00588$. Table 1 shows the operations in more detail.

Table 1: Most likely transduction to correct input *aat*

	HM produces	EM consumes/produces	CM consumes	Transformation
STEP 1	a:1	a/c:0.1	c:0.5	a/c:0.05
STEP 2	a:0.6	a/a:0.7	a:0.5	a/a:0.21
STEP 3	t:0.8	t/t:0.7	t:1	t/t:0.56

As mentioned before, if the resulting string is not correct, it can be efficiently re-estimated by incorporating the user interaction in the model. A convenient method of interaction for the user is often the introduction of a prefix of the expected string. For example, if the right word is *goat*, the user should sequentially introduce the characters of this word until the system produced the correct output. In this case, if the user introduces the character *g*, the IM representing this prefix would be that shown in Figure 5. This model only accepts (and produces) strings starting with *g*, therefore, if it is incorporated between EM and CM, it acts as a filter that blocks any string at the output of EM not starting with the representing prefix. In this case, the most likely transformation (actually the only possible) would be {ϵ/g, a/o, a/a, t/t}, i.e., *aat* $\rightarrow$ *goat*. Table 2 shows the operations in more detail.

Table 2: Most likely transduction to correct input *aat* with prefix *g*

	HM	EM	IM	CM	Transformation
	prod.	cons./prod.	cons./prod.	cons.	
STEP 1	-	ϵ/g:0.1	g/g:1	g:0.25	ϵ/g:0.025
STEP 2	a:1	a/o:0.2	o/o:1	o/o:1	o/o:0.2
STEP 3	a:0.6	a/a:0.7	a/a:1	a:1	a/a:0.42
STEP 4	t:0.8	t/t:0.7	t/t:1	t:1	t/t:0.56

In short, the proposed system has a dual purpose: to transform the observed evidence (HM) into a valid string based on a maximum likelihood approach and, in case the proposed string is not the correct one, to obtain the intended string with the minimum user interaction effort.

One of the main contributions of the approach proposed is that the HM does not necessarily encode a single string, but also more complex inputs (Section 4.2). In addition, it allows multimodal user interaction, as explained before. When the corrected string proposed by the system is not the one intended by the user, the IM can be dynamically combined (as the user interacts with the system) with the other models to get the intended output with the minimum user effort.

It is even possible to model potential mistakes of the user while interacting with the system (for example, typing an adjacent key instead of the desired one, double-typing a key, etc.). In this case, the User Interaction Model can have associated its own Error Model that allows recovering user-input errors (see Section 4.4). In fact, the IM can be seen as a second Hypothesis Model, which leads to a multimodal error interaction system where several inputs (hypotheses) are combined to propose an output. Even several interaction sources can conceivably coexist, thereby increasing the potential multimodality of the system.

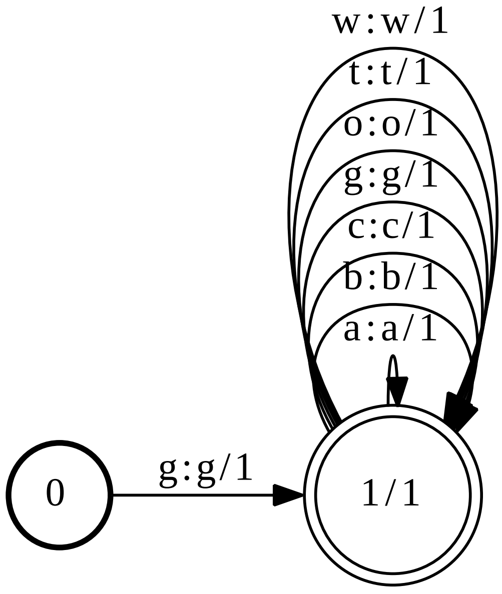

Figure 5: Interaction Model representig the prefix $\{g\}$ over an alphabet $\Sigma = \{a,b,c,g,o,t,w\}$.

In the proposed approach, no assumption is made on where the initial hypothesis comes from, or how the user interacts with the system. The initial hypothesis could be the output of a Human-Machine Interface like an OCR, tactile input or gesture recognition system, sensor inputs from a physical process, a biological sequence like DNA or protein strings, or even a combination of several of them. The user interaction can be typically done by means of a physical, "soft" (touchscreen-based) or reduced (as in a mobile phone) keyboard, but also by means of a speech or gesture-based recognizer an on-line OCR system or any sensor or input subsystem designed to read voluntary human actions.

In this section, a practical scenario has been presented, although the proposed method can be applied to a wide range of problems. In the following sections, the theoretical foundations of WFST are reviewed first, and then the different proposed models are explained in more detail and generality. Finally, two example applications that make use of the proposed methodology are presented and tested.

3. Weighted Finite-State Transducers

Weighted Finite-State Transducers (WFST) have proved to be flexible and useful formalisms with uses in many disciplines, mainly in Pattern Recognition and Machine Translation. We present in this work a novel use of WFSTs in combination with Stochastic Error-Correcting Models for Human Interaction tasks.

WFSTs are generalizations of Finite-State Automata (FSA) [21, 23, 24]. FSAs are often represented as finite directed graphs with nodes as states and arcs as transitions. Each transition is associated to a symbol from an alphabet Σ. Formally, an FSA is a five-tuple $(Q,\ q_0,\ F,\ \Sigma,\ \delta)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, Σ is a finite set of symbols and $\delta : Q \times \Sigma \to Q$ is the set of transitions between states. Each transition t is labeled with a symbol $s(t) \in \Sigma$. In a FSA, given a string $s \in \Sigma^*$, s is accepted when a path from the initial state to a final state of the graph exists and their transition labels concatenated are equal to the string s.

Finite State Transducers (FSTs) add ouput labels in the transitions to the previous formalism. Therefore, an FST is defined as a six-tuple $(Q,\ q_0,\ F,\ \Sigma,\ \Delta,\ \delta)$ where Σ is a set of input symbols, Δ is a set of output symbols and function δ is defined as $\delta : Q \times \Sigma \to Q \times \Delta$. FSTs are able to *transduce* a string of an input language over Σ into a string of an output language over Δ. A weighted version of an FST (WFST) can be defined by including weights in their transitions. These weights can have the meaning of the cost of taking a particular arc in a path through the transducer.

Additionally, each state q has an initial weight $w_i(q)$ and a final weight $w_f(q)$. Any state q can be *initial* if $w_i(q) \neq \bar{0}$ and/or *final* if $w_f(q) \neq \bar{0}$.

Given a transducer whose weights represent probabilities, if all the outgoing arcs of a given state plus its final weight add up to 1, then the relations $(input, output)$ that this transducer produces are called joint relations and have probability $P(input, output)$. If it is each subset of outgoing arcs of a given state that are labeled with the same input symbol what totals

1, then the relations are called conditional relations and have probability $P(output|input)$. If a number of source WFST are to be composed with the restriction that the resulting WFST is markovian, then one of the information sources has to be modelled as joint and the rest as conditional probabilities, as explained in [25].

An FSA can be seen as an FST with same input and output symbols in each transition. This is referred to as an identity transducer. The same is true for a WFSA respect to a WFST.

FSTs (and WFSTs) are very flexible and powerful for some applications due to some of their properties and the operations they allow. The methodology presented in this paper makes strong use of the *composition* operation [26] that can be described as follows: for transducers T_1 and T_2, if T_1 transduces the string $x \in \Sigma$ to $y \in \Delta$ with weight w_1 and T_2 transduces $y \in \Delta$ to $z \in \Gamma$ with weight w_2, then their composition $T_3 = T_1 \circ T_2$ transduces x to z with weight $w_1 \otimes w_2$.

Different semirings $\{\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1}\}$ (where $\mathbb{K}$ is a set, $\bar{0}, \bar{1} \in \mathbb{K}$ and $\oplus, \otimes$ are the additive and multiplicative operations respectively) can be used to compute the weights, some of which are shown in Table 3.

Table 3: Several types of semirings

Semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Tropical	$\mathbb{R} \cup \{+\infty, -\infty\}$	$+$	min	$+\infty$	0

Although the examples showed in Section 2 are explained in terms of a probability semiring, we have actually used tropical semirings to avoid underflow problems when computing the total probability along the shortest path.

4. Description of the method

In this section we discuss the use of WFSTs to deal with the problem of efficient interaction. The method consists of building and composing WFSTs that encode different pieces of information and represent distinct models, extending the ideas of language modeling through Stochastic Error Correcting Parsing proposed in [9] and [15].

As mentioned above, several sources of information can be identified: the input hypothesis, the expected errors and the constraints that the strings of the task must satisfy. Each of these sources of information can be represented by a WFST that we call the Hypothesis Model (HM), the Error Model (EM) and the Constraint Model (CM) respectively. HM produces the set of initial hypotheses, then EM accepts and probably transforms these hypotheses and, finally, CM only accepts the strings compatible with the encoded set of constraints or language.

The initial goal is to find the most probable string accepted by CM according to the current hypothesis (encoded in HM) and the possible errors (encoded in EM). This is done by composing HM $\circ$ EM $\circ$ CM and finding the most probable path in the composed transducer.

Although the correct string can be found by properly combining HM, EM and CM, it could happen that the selected string in the CM is not the one the user expects. In this case, a user interaction is needed to get the expected output. To deal with this interaction process, a new WFST called the User Interaction Model (IM) is used. This WFST encodes the extra information given by the user (for example a prefix of the correct output). It is built dynamically as the user interacts with the system and composed with the other transducers, which imposes new constraints to the resulting transducer. The more information the user introduces, the more likely the composed transducer will converge to the intended output.

The details of how each of these models can be efficiently encoded by a WFST and how they are combined to deal with the problem of finding the most probable hypothesis are presented in the next sections.

4.1. The Constraint Model (CM)

A stochastic FSA that accepts the smallest k-Testable Language in the Strict Sense consistent with a task-representative constraint sample can be obtained using a grammatical inference algorithm [27]. The set of strings accepted by this automaton is equivalent to a classical language model obtained using n-grams, for $n = k$.

Using a stochastic grammatical inference algorithm is convenient because it can take advantage of very different constraint sets, from a list of sequences of symbols, with each valid sequence appearing only once (equivalent to a lexicon) to a list of symbol sequences extracted from a real instance of the task (with each one appearing as many times as in the sample) or a list of sequences with strings or string categories as the symbols of the grammar, etc. Only in the first case, when using a classical lexicon, the automaton is not required to be stochastic, since a lexicon is not a representative sample.

In the other examples, the model can take advantage of the probabilistic information present in the data.

The response of the model is also defined by constant k. If k is equal to or higher than the length of the longest sequence in the sample, only sequences that exist in the sample will be valid, giving rise to a deterministic model. Otherwise, when k is lower, the model is non-deterministic, the behaviour is similar to a classical $n-$gram and the output sequences may be or not in the reference sample.

Note that the stochastic or probabilistic nature of the underlying grammar (i.e. its ability to take into account the relative frequencies of the different symbol sequences) does not depend on the choice of k. Therefore, for instance, a deterministic model can be based on a stochastic grammar, when the constraints database is a real sample of sequences and k is large, and a non-deterministic model could make use of a non-probabilistic grammar, when the database is a simple lexicon and k is small.

In Figure 2, an example of a CM derived from a set of animal names was showed. In a more generic context, Figure 6 shows the probabilistic identity transducer derived from the sample S=$\{$a, bab, bac, ca, cab$\}$ and $k = 3$. For simplicity, in this description we use an identity transducer (a transducer with input and output symbols equal in each transition), which can be seen as an acceptor of the language $L(S)$. The transition weights (probabilities in this case) are shown in each arc. The probabilities of the final states (double circled) are shown after the state number.

Given a sequence s, $P(s)$ is computed as the product of the probabilities along the path (including the probability of the final state) that accepts (or produces) s. Because the proposed transducer reflects joint relations, $P(s)$ represents the probability that s is accepted (or produced) by the CM.

4.2. The Hypothesis Model (HM)

The HM encodes the initial input to the system before any correction or user interaction is performed. This input consists of a set of hypotheses together with their probabilities, which are represented as paths in a finite state automata. The goal of the system is to select and possibly transform one of these hypothesis to produce the correct output.

If the task, for instance, involves an OCR classifier for recognition of text fields, the HM can represent the output of the OCR classifier in Figure 3. In general terms, this model represents a sequence of n-dimensional vectors $\bar{v}_1 \ldots \bar{v}_m$, where each character is assigned n *a posteriori* probabilities, one for

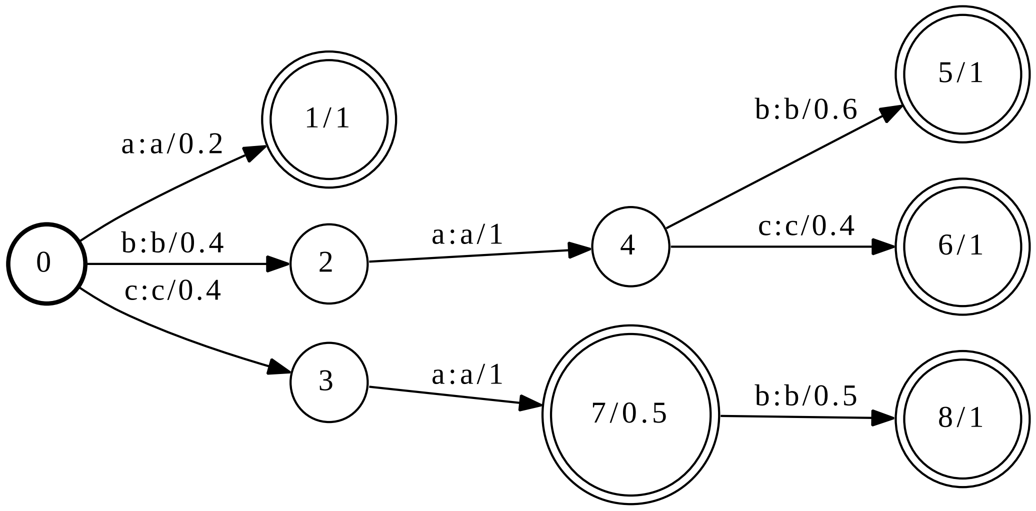

Figure 6: Example of an identity transducer representing a constraint model.

each possible class, m is the length of the string read and $v_{i,j}$ is the probability that the i^{th} character is of class j^{th}. This sequence can be represented as an identity WFST with $m+1$ states and n transitions between each pair of states. Figure 7 shows an example of a WFST with alphabet $[a, b, c]$ that represents the symbol-input $[0.6, 0.2, 0.2], [0.0, 0.7, 0.3], [0.1, 0.6, 0.3]$. This means that the first symbol is a with probability 0.6 or b with probability 0.2 or c with probability 0.2, the second symbol is b or c with probabilities 0.7 and 0.3 repectively, and so on. Transitions with zero-probability are not shown in the graph. This topology makes it possible to represent sequences in which the probability of each symbol does not depend on the context. If the probability of each symbol depended on the previous history, a topology like the example in Figure 8 could be used instead. For example, in this model the probability of a in the second position of the sequence is 0.4, 0.5 or 0.8, depending on whether the first symbol is a, b or c respectively.

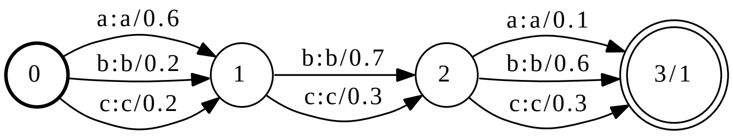

Figure 7: Example of a hypothesis model in which the probability of each symbol does not depend on the context.

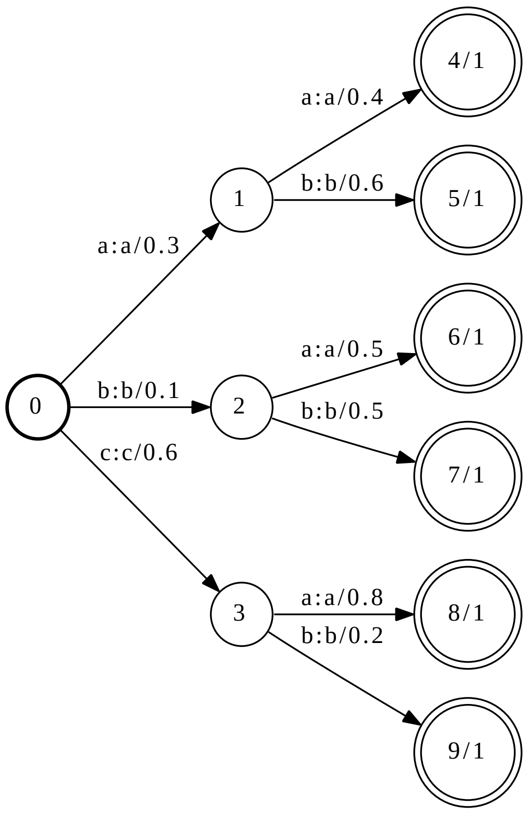

Figure 8: Example of a hypothesis model in which the probability of each symbol depends on the previous history.

These transducers model the uncertainty of the input to avoid the error induced by propagating only the most probable hypothesis (*abb* in the first example or *ca* in the second one) to the next stages.

Similarly to the CM, the HM is in general a probabilistic model. In this case, each path from an initial state to a final state of the model represents a hypothesis h.

Depending on the task, the structure of the HM can be very variable. The only restriction is that can be represented by a WFST encoding the input hypotheses. In Section 5.2 an HM aimed at a different task is described.

4.3. The Error Model (EM)

In some tasks, it is likely that at least one of the symbol sequences that the input hypothesis can generate is compatible with the constraint model. However, when this is not the case or when a compatible sequence similar to one in the set of hypotheses could also be acceptable according to some criterion, an additional model that allows generate distorted versions of the input sequences that can be considered in the search of the best final result is necessary.

In a classical $n-$gram model, these cases are dealt with by a *smoothing* method. In our case, an *Error Model* performs this task by allowing that "distorted" hypothesis sequences appear in the global search. In a sense, the Error Model, which can be as complex as needed, gives a finer control of the mentioned "smoothing effect". In the simplest case, the three classical edit operations are included: *insertions*, *substitutions*, and *deletions*. Given two symbols s_1, s_2 and the empty symbol ε, insertions, substitutions and deletions are transductions of type ε/s_2, s_1/s_2 and s_1/ε respectively.

Different probabilities can be assigned or estimated for these operations. The confusion matrix of the input process, which reflects the probabilitiy that each pair of symbols are confused, estimated using a representative corpus, can be used to compute the probability of substitutions. For instance, for an Optical Character Recognition input, the frequency with which the classifier takes one character for another is modeled. For keyboard input, we would estimate the probabilities that users press a key that is near the one intended. In devices with small keyboards where several characters share the same key (e.g. a mobile phone), the frequencies of the different mistakes observed are also used to estimate the confusion matrix.

The EM can also be seen as a *static* or *generic* model of the uncertainty measured in the input subsystem, in contrast with the *dynamic* or *particularized* estimation provided by the input hypothesis model.

The insertion and deletion probabilities are usually task-dependent and, in most cases, they should be estimated empirically. In Figure 4 an example with the probabilities of the edit operations and its associated EM was shown. Figure 9 contains examples of different Error Model transducers with symbols in {a,b} ($\Sigma = \Delta = \{a, b\}$). The transducer on the left converts any string x in Σ^* to any other string y in Δ^* with $p(y|x)$ by means of substitutions, insertions and deletions. In this case, the probability of leaving a symbol unchanged (substitution by itself) has been set to 0.8, the probability of substitution by a different symbol to 0.2 and the probability of insertion and deletion to 0.05. A transducer like this one can be right-composed with a HM, so any hypothesis h produced by the HM is converted into $\hat{h}$, where $\hat{h}$ is a noisy version of h. The transducer on the right represents the *trim* operation, i.e., deletion of blanks at the beginning and the end of the string. If this transducer is right-composed with a given transducer T, the state 0 will delete all the blanks coming from T at the beginning of the string, then the state 1 will keep all coming symbols a and b unchanged and, finally, the state 2 will allow the deletion of blanks at the end of the string. In this case, this transformation is made without cost, as all the transitions have probability 1.

These are only a couple examples showing how modeling the errors and their probabilities in an independent transducer allows for additional flexibility. For example, building an EM that allows insertions or deletions only at the beginning or at the end of the string, or establishing a bound on the number of error operations is straightforward. It is also possible to include other error sources such as transpositions of two adjacent symbols that often happen in keyboard typewriting tasks, special confusion matrices that take into account the distances among the keys, etc.

4.4. The User Interaction Model (IM)

When human intervention is necessary, the decision process must include a feedback option. The modular nature of the proposed method makes relatively easy the addition of new models representing complementary information. In this case, the goal is that the system can take advantage of the user feedback to provide new outputs compatible with it, in real time, as the user introduces new symbols (typically a prefix of the intended output). It is

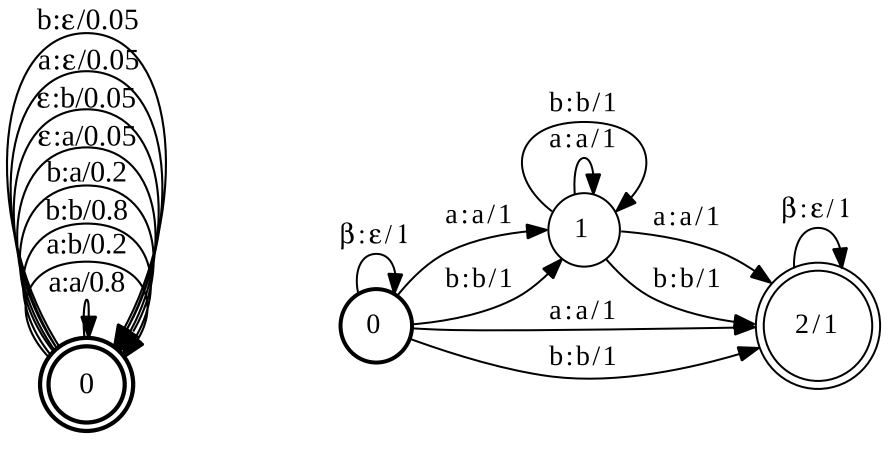

Figure 9: Examples of two Error Model transducers, with all possible insertions, deletions and substitutions (left) and with the posibility of deletions only at the beginnig and at the end of the string (right). β represents the white symbol.

expected that an interaction far shorter than the one necessary to introduce the complete output string may be enough.

In the simplest case, if the characters entered represent a prefix of the correct output, a straightforward method for implementing the interaction model is the following: given a prefix $P = p_1, p_2, \ldots, p_n$, an identity transducer with $n + 1$ states is built. A transition with input and output symbol p_i and probability 1 is added between states s_i and s_{i+1} with $i = 1, \ldots, n$. Also, transitions with input and output symbol $\sigma \in \Sigma$, where Σ represents the set of symbols of the task, are added as cycles in the last state s_{n+1}. This transducer accepts/produces any string with symbols in Σ starting with P. Figure 10 shows an example of a transducer representing the prefix $P = ab$.

The effect of right-composing this transducer with the model composed so far, HM $\circ$ EM, is "filtering" the strings produced and leaving only those starting with $P = ab$. In Section 4.5, further details about this composition operation are given.

Other models could easily be created to allow interaction when the entered characters represent a suffix or, in general, any substring of the desired output.

When the interaction subsystem is not accurate enough (as could happen in mobile devices where the small size of the keyboard makes typing error-prone), this model can be generalized to try to recover user-interaction errors. An error model can be associated to the IM in the same way as it was

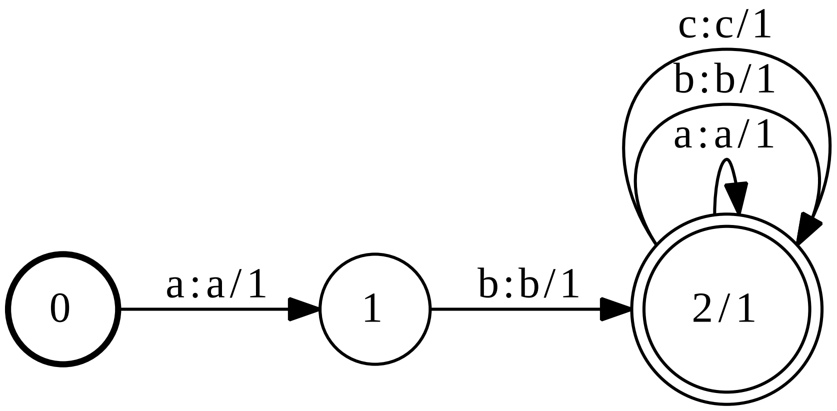

Figure 10: Example of an identity transducer representing the prefix $P = ab$ and alphabet $\Sigma = a, b, c$.

done with the HM. In this case, there would be an EM associated to the HM ($\mathrm{EM}_{\mathrm{hm}}$), representing the errors that can occur in the hypotheses, and an EM associated to the IM ($\mathrm{EM}_{\mathrm{im}}$), representing the errors due to the interaction process.

In a keyboard, typically, the user interaction errors consist of typing adjacent keys to the desired ones and, to a lesser extent, skipping keys or typing repeated or extra characters. The probability of accepting a symbol different than the typed one could be a function of the physical distance between the corresponding keys (typed character and accepted character) in the keyboard, while the probability of insertion and deletion transitions could be empirically estimated. Of course, other sources of interaction errors, such as spelling mistakes or ambiguities could also be taken into account.

For example, if the user types the prefix $P = ab$ and we assume that a, b and c are adjacent keys on the keyboard (with b between a and c), the User Interaction Model shown in Figure 11, which represents the user interaction composed with its error model, could be used. If this transducer is right-composed with HM $\circ$ EM, all the strings produced with symbols in a, b, c will be accepted, but those starting with ab will be more probable. This transducer is an extension to that shown in Figure 10 with extra transitions: arcs labeled with $\varepsilon : s$ (with s in a, b, c) allow extra symbols to be added to the prefix, arcs of type $s : \varepsilon$ allow to remove symbols from the prefix, and arcs of type $s : s$ allow to accept either the typed symbol, or any other related one (with lower probability).

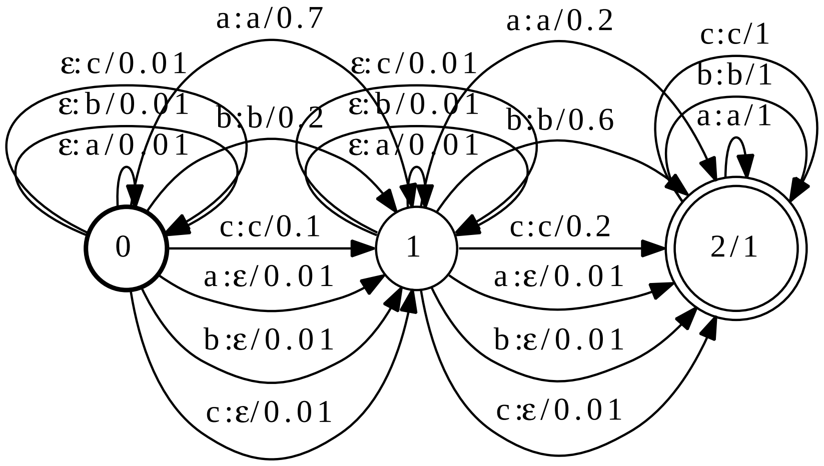

Figure 11: Example of an User Interaction Model representing the prefix $P = ab$, with error recovery transitions.

One of the advantages of this technique is that the proposed model allows different ways of interaction (typewritten or handwritten characters, speech, etc.) as long as the input can be represented by means of a probabilistic transducer.

The computational cost obtained is practical even for large Constraint Models, as can be seen in the experiments (Section 6). This means that the resulting string can be almost immediately available as the user interacts with the system.

4.5. Composing the HM, EM, IM and CM

With the information encoded in HM, EM, IM and CM, the most probable string according to the grammar and probabilities defined in each model can be found. This is achieved by composing all these models, searching the most probable path and concatenating its output symbols.

Given two transducers T_1 and T_2, the composition $T_1 \circ T_2$ selects the intersection between the set of strings produced by T_1 and the set of strings accepted by T_2 and then, transduces these strings from the input in T_1 to the output in T_2. If T_2 is an identity transducer (i.e., the input and output symbols are the same in each transition), it acts as a filter and leaves the accepted strings unchanged, although the probabilities can change.

Also, the probability of the resulting strings is computed according to the weights defined in each transition. If T_1 produces the string x with probability $p(x)$ and T_2 transduces x to y with probability $p(y)$, then $T_1 \circ T_2$ transduces x to y with probability $p(x)p(y)$.

We have performed the composition of the different transducers as shown in Figure 12. Let L(H) denote the set of weighted strings produced by the HM. The transducer HM $\circ$ EM expands L(H) with new strings, generally by producing perturbed or noisy versions of the original hypotheses in L(H) that we call L($\hat{H}$). Figure 13 shows the composition of the transducers HM and EM depicted in Figures 7 and 9 (left) respectively. This automaton transduces the strings produced by the HM into any string in Σ^*, with a probability related to the similarity between the original and the transduced string (for clarity, probabilities have been omitted in the figure).

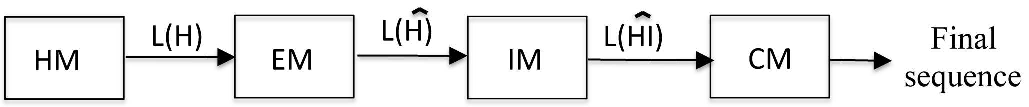

Figure 12: Composition of the different transducers used in the proposed system.

As the user interacts, the identity transducer IM is created. This transducer accepts (and produces) strings containing a substring "similar" (or equal in the case that the IM does not contain error transitions) to that entered by the user. Therefore, the effect of right composing the IM with HM $\circ$ EM is to select from L($\hat{H}$) those strings compatible with the grammar defined in the IM or, in the case that the IM includes error transitions, to decrease its likelihood as the differences with the original user entry increases.

Finally, let L($\hat{H}$I) denote the set of strings produced by HM $\circ$ EM $\circ$ IM. Then, the effect of right composing the CM with HM $\circ$ EM $\circ$ IM is to select from L($\hat{H}$I) those strings accepted by the constraint model and compute new weighs for them according to the probabilities defined in the CM.

In summary, the goal of the proposed method is to find the most likely transduction of an input string $h = (h_1, h_2, \ldots, h_n)$ produced by HM, into an output string $\hat{s} = (s_1, s_2, \ldots, s_m)$ accepted by the CM by means of the intermediate transductors EM and IM. Formally, the string $\hat{s}$ is computed as:

$$\hat{s} = \arg\max_{h,s} P(h, s)$$

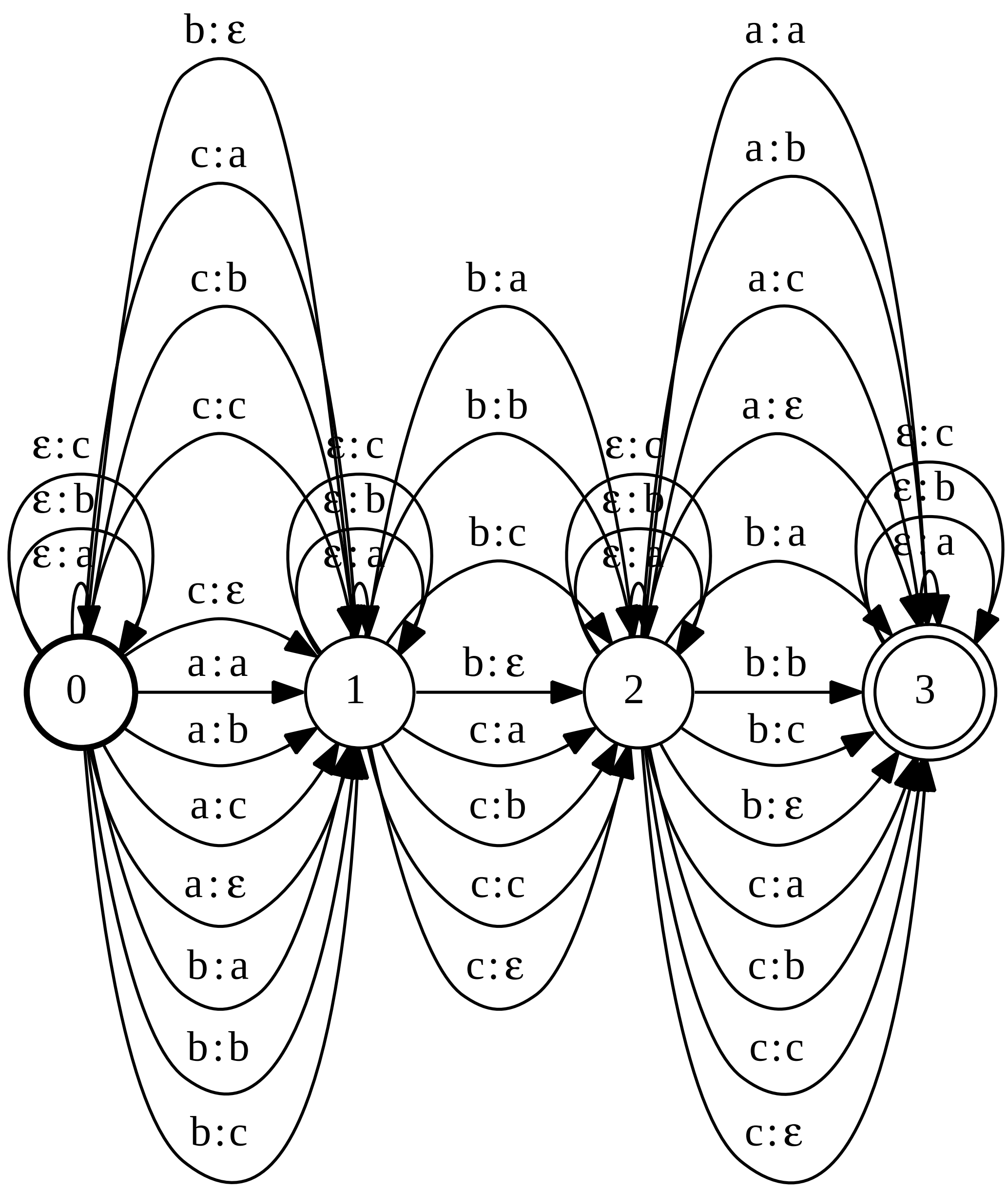

Figure 13: Composition of the HM shown in Fig. 7 with an EM with all possible substitutions, insertions and deletions using an error model similar to Fig. 9 (left) but adding also *c* symbol.

where h is the set of strings produced by HM, s the set of strings accepted by CM, and $P(h, s) = P(\text{HM,EM,IM,CM})$.

Assuming independence among the HM,EM,IM and CM, and allowing different weights for the models:

$$P(h, s) = P(\text{HM})^{\lambda_{\text{HM}}} P(\text{EM})^{\lambda_{\text{EM}}} P(\text{IM})^{\lambda_{\text{IM}}} P(\text{CM})$$

where λ_i defines the contribution of model i to the final $P(h, s)$ value, taking as a reference the CM that has a fixed $\lambda_{\text{CM}} = 1$. These factors are estimated by the maximization of the first part of a ROC curve area using the algorithm described in [28].

This process is equivalent to search the most probable path in the transducer HM $\circ$ EM $\circ$ IM $\circ$ CM, which produces the string $\hat{s}$.
Therefore, P(h,s) is the probability of the most probable path, t, resulting from the concatenation of transtions, $(t_1, t_2, \ldots, t_n)$, in the transducer, where each transition t_i, as a result of the composition operation, contributes to the final score value with

$$P(h_i, s_i \mid t_i) = P(\text{HM} \mid t_i)^{\lambda_{\text{HM}}} P(\text{EM} \mid t_i)^{\lambda_{\text{EM}}} P(\text{IM} \mid t_i)^{\lambda_{\text{IM}}} P(\text{CM} \mid t_i)$$

and the final score is computed as

$$P(h, s) = \prod_{i=1}^{n} P(h_i, s_i \mid t_i)$$

Note that in this process, the only transducer that actually modifies the symbols of the original hypotheses is EM, being the other ones identity transducers that leave the input sequences unchanged, but playing the roles of filtering the acceptable strings and recomputing their probabilities.

Our final WFST is not markovian, since the individual automata represent joint probabilities [25]. However, this is not a problem if the goal is to find the best path taking into account the different evidences provided by each WFST, considered not as probabilities but as scores, and given that all the score values are positive and there is an order relation in the scores finally obtained.

4.6. Decoding

In the decoding phase, the most probable path in the composed transducer is found using a variant of the Dijkstra algorithm [29]. The concatenation of the output symbols along this path forms the output string, with a score that can be computed as the product of the probabilities of each transition along the path. If several alternatives are needed, the n-best paths can also easily be obtained.

To avoid underflow problems, instead of working with probabilities we have used tropical semiring WFSTs $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ where $\mathbb{K}$ are negative log probabilities, $\oplus$ is the `min` operation, $\otimes$ is $+$, $\bar{0}$ is $+\infty$ and $\bar{1}$ is 0.

4.7. Computational cost and lazy composition

WFST composition of very large transducers can incur in large computational costs. For a CM of 64000 states and 140000 transitions (like the one used in some of our experiments), a standard EM with all possible insertions, deletions and substitutions and an average-sized HM with 8 states and 5 transitions (hypotheses) per state, the resulting composed transducer can have up to 450000 states and more than two million transitions.

To avoid this problem, *lazy* composition have been used. Lazy operations delay the computation of the result until it is required by another operation. This is useful when a large intermediate transducer must be constructed but only a small part of it needs to be visited [21]. In our approach, given that the operation of shortest (lowest cost) path search does not need visiting all the states of the composed transducer, the composition is delayed until that operation is performed.

On the other hand, when more than two transducers need to be composed, *n-way composition* can be used [30]. Instead of composing the different transducers in pairs, as done in standard composition, this method directly constructs the resulting transducer to avoid creating unnecessary transitions, which leads to very much faster computational costs.

5. Applications

One of the advantages of the proposed framework is its flexibility, which allows to adapt it to a wide variety of applications and requirements. In this work, two applications are presented and analyzed as an example of the possibilities of the proposed scheme: a) an OCR post-processing system and b) an efficient data entry system for GPS navigation devices. In both cases, the OpenFST library [22] was used for the experiments.

5.1. Interactive multimodal OCR post-processing

In this section, an interactive multimodal system aimed at OCR post-processing is presented. A character recognition task dealing with known fields of scanned handwritten forms has been chosen as the input.

In this task, the Constraint Model encodes the list of valid strings of a particular field in the form. If the *a priori* probability of each valid string is known, the model can take advantage of this probabilistic information.

The Hypothesis Model encodes the output of the OCR classifier as explained in Section 4.2. Figure 7 shows an example of a WFST encoding a particular hypothesis model.

The Error Model defines the three usual edit operations: substitutions, insertions and deletions, as explained in Section 4.3 and shown in Figure 9 (left).

User interaction may be necessary when the system fails in finding the correct transcription. For this purpose, a keyboard that can be full-size, reduced, physical or virtual, or any other symbol input device (each input symbol will be regarded in any case as a *keystroke*) is used, and the goal is to allow the user to obtain a correct output string with the minimum effort. To address this process, a User Interaction Model that encodes a prefix is used as described in Section 4.4. In this task, no error model associated to IM (EM_{ui}) has been used, assuming that user entry is free from errors.

Experimental results are presented in Section 6.1.

5.2. Efficient entry data in GPS devices

It is difficult to interact efficiently with GPS devices (and with mobile devices in general), due to the size of the screen, the inherent limitations of a soft keyboard (small size and a limited set of characters) and the limited accuracy of the finger or pointing device detection.

However, when a priori knowledge of the input is available, efficient techniques to improve the efficiency of data entry can be applied. For example, when introducing the destination city in a GPS device, a priori knowledge like the complete list of city names, the city importance or population, the recent destinations, the current coordinates and the prefix introduced so far can be used to speed up the entry process. Additionally, a priori knowledge of known common errors like typing an adjacent key or of unavoidable errors produced by the abscense of some local characters in the keyboard, can also be used to recover these errors.

We used the method proposed in Section 4 to drastically reduce the number of keystrokes needed to enter the destination city. A similar technique could be used for entering the destination address, country, etc. The idea is based on representing each source of information by means of a WFST and composing them to suggest the most probable city.

In this application, the models CM, HM, EM and IM explained in Section 4 have been used as follows:

The CM encodes the list of valid cities. The population of each city can be used to provide probabilistic information to the model, assuming that more populated cities are more likely to be selected as destination. In the experiments presented in Section 6 a probability proportional to the population has been used. The value of k has been made equal to the length of the longest city name, therefore only cities that exist in the sample are suggested as possible destinations.

The HM encodes the most likely cities to be chosen as destination according to a number of factors such as the closeness to the current position or to any other position in a map that can be set by the user with a simple touch on the screen, being a touristic or recently specified destination, etc.

In our case, higher probabilities have been assigned to closer cities, assuming that they are more likely to be chosen as destination. For efficiency, in the experiments presented in Section 6, only cities within a radius of 50 km from the current or a predetermined position have been included in HM. A parameter has been used to weight the influence of the distance to the set or current position.

It is possible that the HM does not contain the desired destination, therefore an error model EM that allows to reach cities not included in the HM is necessary. For this purpose, we have used a transducer like the one in Figure 14 (assuming an alphabet $\Sigma = a, b, c$). If the upper arc of the initial state is first selected, then the transducer leaves the strings encoded by the HM unchanged, which allows the CM to accept any of these strings, whereas if the lower arc is selected, the transducer can generate any string, allowing the CM to accept other strings not initially included in HM. For this to work, it is necessary to include the empty string in the HM, i.e., the initial state must also be final.

To model the likelihood that the HM actually contains the desired destination, probabilities of α and $(1 - \alpha)$ with $\alpha \in [0 - 1]$ are set in the upper and lower arcs of the EM respectively. The larger the value of α, the smaller the probability of selecting a destination not initially included in the HM and

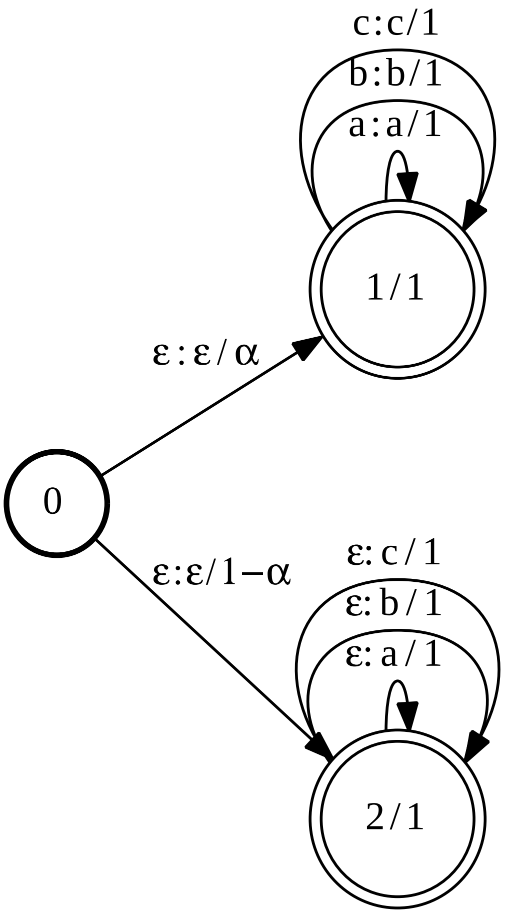

Figure 14: Error model used in the GPS task assuming an alphabet $\Sigma = a, b, c$.

viceversa.

Finally, the IM encodes a prefix of the destination city name. A first destination proposal can be done by combining only HM, EM and CM, however, user interaction will be probably required to select the desired destination. In this case, an IM model is dynamically created and composed with the previous models as the user inserts the prefix, as explained in Section 4.4. The IM can have its own error model associated (e.g. with the probabilites of pressing adjacent keys) if entry mistakes are likely.

6. Experiments

6.1. Interactive multimodal OCR post-processing

The constraint model defined for the multimodal interactive OCR post-processing experiments is built (as explained in Section 4.1) using a sample of $4.5M$ Spanish surnames, $76,119$ of which were unique. The ouput was restricted to be only exact known surnames (deterministic model), so a k value equal to the largest surname was selected.

The hypotheses used to test the approach were obtained through an OCR process on a sample of $25,262$ forms scanned in a real industrial task where a handwritten field with the surname of the writer was present. $14,364$ of the $25,262$ (56.9%) surnames were correctly transcribed by the proposed system with no human intervention while the $10,898$ remaining surnames showed wrong transcriptions. Two test sets were used to compute the performance of the proposed system: one consisting of the $25,262$ scanned surnames that we call the *total* test set and another one consisting of the $10,898$ incorrect transcriptions that we call the *wrongly corrected* test set. The *total* test represents a real scenario in which a given batch of forms must be processed, while the *wrongly corrected* test represents a scenario under the assumption that it would be possible to set a cost transformation threshold able to appropriately separate correct and wrongly transcripted inputs, which would allow to manually review only the second ones. Due to this design, most of the hypotheses into the second set are very far from the correct transcription, otherwise the automatic system would probably have recovered them correctly in the first place. In either case, the HMs were built including the a posteriori probabilities provided by the OCR, as explained in Section 4.2.

The topology used to build the error model was like the one presented in Figures 4 and 9 (left). The probabilities of the confusion matrix were estimated previously from a set of approximately $1,180,000$ characters not used in the test set. This matrix gives us information about the OCR classifier confusion distribution. The confusion matrix estimation has been obtained by means of the OCR engine and the characters used during classifier training by means of a leaving-one-out design with a smoothing factor added to model the confusion between characters that did not appear during the matrix estimation process.

To estimate the effort required to obtain a correct transcription using the system proposed, the *keystroke ratio* (KSR) is defined as the number of keystrokes (or equivalent single-symbol input operations) needed by the user divided by the number of keystrokes or operations needed to type the complete string.

To compute the KSR, a realistic assumption is that the user types the shortest prefix needed to obtain the reference transcription, which implies that the operator starts typing the correct string from the first symbol even if it is already correct. Using the mouse to move the cursor to the first unmatched symbol would typically increase the interaction time and effort compared to using exclusively the keyboard. However, using i.e. the right arrow or the tab key to accept a single symbol and move to the next one (symbol acceptation keystrokes) is usually acceptable and faster than retyping the symbol. We can therefore split the KSR into two parts that we call *character stroke ratio* (CSR) and *accept stroke ratio* (ASR). The first estimates the effort to type new symbols and the second is intended to evaluate the effort associated to inspect and validate the proposed characters, and the sum of both measurements gives rise to the KSR ($KSR = CSR + ASR$).

An interactive transcription, where the user types the shortest prefix required to obtain the correct word, a surname in these experiments, was simulated. In Table 4 it is shown an example on how an input OCR hypothesis is used by the system to provide new surnames as the user appends a new corrected symbol to the prefix. Bold characters in the prefix contribute to the CSR and regular characters add to the ASR.

According to these prefixes, a User Interaction Model (IM) like the one described in Section 4.4 and similar to the one showed in Figure 10 is dynamically built and composed with the rest of the models, until the resulting string converges to the expected one. In this task, the IM does not include an associated error model, assuming that the user does not introduce errors while interacting with the system (although in the application tested in the following section an IM with error model is used).

Table 4: Proposed transcriptions for an input OCR hypothesis (only the most probable class for each symbol is shown) and a prefix sequence entered by the operator

OCR hypothesis	Prefix	Proposed transc.	Correct transc.
ONAVES	[none]	NAVES	CHAVES
	C	CHAVES	
HICICA	[none]	MICLEA	MUGICA
	M	MICLEA	
	M**U**	MUGICA	
TEZCTAL	[none]	TESTAL	MOZOTA
	M	MERCHAN	
	MO	MORATAL	
	MOZ	MOZOTA	
IAIIIV	[none]	ZAIDIN	PADIN
	P	PALLIN	
	PA	PALLIN	
	PA**D**	PADIN	

A first set of experiments was conducted in order to test the improvement achieved when the user interaction is dynamically added to other sources of information. To this end, four different approaches with different combinations of information sources (models) were considered: i) *Fully-Manual*, in which there are no models and the user must manually introduce the whole string; ii) *Predictive Text* in which only IM and CM are considered, assuming that there is not a previous post-processing nor automatic correction stage; iii) *Error Correction*, which represents an automatic error correction system without integrated user interaction, i.e. HM + EM + CM; and iv) *Multi-modal Interaction* in which IM is also included, i.e. HM + EM + CM + IM.

The KSR, CSR and ASR were computed in all of the above mentioned approaches and for the two defined tests (*total* and *wrongly corrected*). Obviously, the approach *Fully-Manual* is included only as a reference and, in this case, the KSR is just 1.

Table 5 shows the KSR, CSR and ASR for the first two approaches. These results show that the predictive text scheme proposed achieves a reduction of 0.47 in the KSR with regard to a fully-manual digitization in the first test, and 0.42 in the second.

Table 6 shows the KSR, CSR and ASR for *Error Correction* and *Multi-*

Table 5: KSR, CSR and ASR results for *Fully-Manual* and *Predictive Text* approaches.

	Total set test			*Wrongly corrected*		
Approach	KSR	CSR	ASR	KSR	CSR	ASR
Fully-Manual	1	1	0	1	1	0
Predictive Text	0.53	0.40	0.13	0.58	0.44	0.15

modal Interaction approaches and for *total* and *wrongly corrected* test sets. In this case the obtained results show a net gain, in terms of KSR, of 0.23 in the *total* test, while in the harder *wrongly corrected* test, the net gain raises up to 0.32. These gains can be seen as the saving, in terms of human requirements, achieved when the IM is composed with the rest of information sources.

Table 6: The KSR, CSR and ASR results for *Error Correction* and *Multimodal Interaction* approaches over the *total* and *wrongly corrected* test sets.

	Total			*Wrongly corrected*		
Approach	KSR	CSR	ASR	KSR	CSR	ASR
Error Correction	0.55	0.24	0.31	0.83	0.51	0.32
Multimodal Interaction	0.32	0.23	0.09	0.51	0.33	0.18

On the other hand, the comparison between *Predictive Text* and *Multimodal Interaction* shows us the gain achieved when an extra source of information (the OCR output in this case) together with an error correction method, are incorporated into the user interaction scheme. In this case, the results show a reduction of 0.21 and 0.07 in the KSR for *total* and *wrongly corrected* test sets respectively.

A second set of experiments were conducted in order to test the influence of the number of valid strings, i.e., the complexity of the CM, using the *wrongly corrected* test set To this end, subsets of increasing size were randomly extracted from the sample of surnames to build different constraint models.

In Figure 15, KSR, CSR and ASR results for a range of constraint model sizes is shown. These results can be compared to a *Predictive Text* approach (Figure 16) where only the user input and a language model are used to predict the text, with no OCR information available (no HM).

Even if the OCR hypotheses that reach this stage are specially uncertain and difficult to recover, as mentioned, since they are those that failed on the automatic recognition phase, the results show that they contain useful information that can be retrieved by the interactive multimodal method, reducing significantly the effort needed for the transcription.

If "accept strokes" are assumed to involve a lower effort than "character strokes", a subjective but reasonable weighted KSR (WKSR) can be defined in order to compare the required user effort in a more realistic way than just using the KSR. Figure 17 shows a comparison of a WKSR=0.67·CSR + 0.33·ASR for the Interactive Multimodal system against a non-multimodal Predictive Text approach.

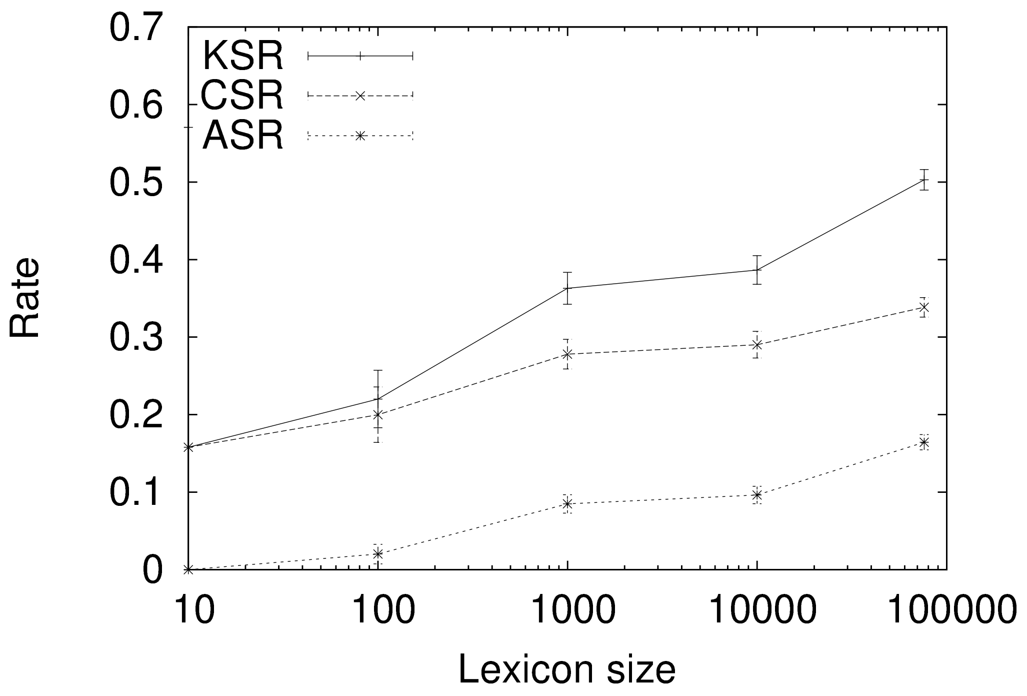

Figure 15: KSR, CSR and ASR (95% confidence interval) for varying lexicon size, for a multimodal approach.

The computational cost is another important issue for this task, where the size of the models can be very large in practice, and the typical operations involve large batches of documents to recognize.

Interaction average time when a large lexicon size is present has been obtained for multimodal and not multimodal systems. We found that there is a significant increase in this response time of multimodal versus non-multimodal systems, but in the worst case the interaction average time is approximately 3.0 ms, which is much shorter than the fastest human interaction time. The time measurement has been obtained on an Intel Xeon 2.5 GHz computer with 2 GB of memory, gnu/linux OS and gcc 4.4. The largest constraint model was built from 76, 119 unique words, but the use of substantially larger constraint models remains practical, since the computational cost typically grows sub-linearly.

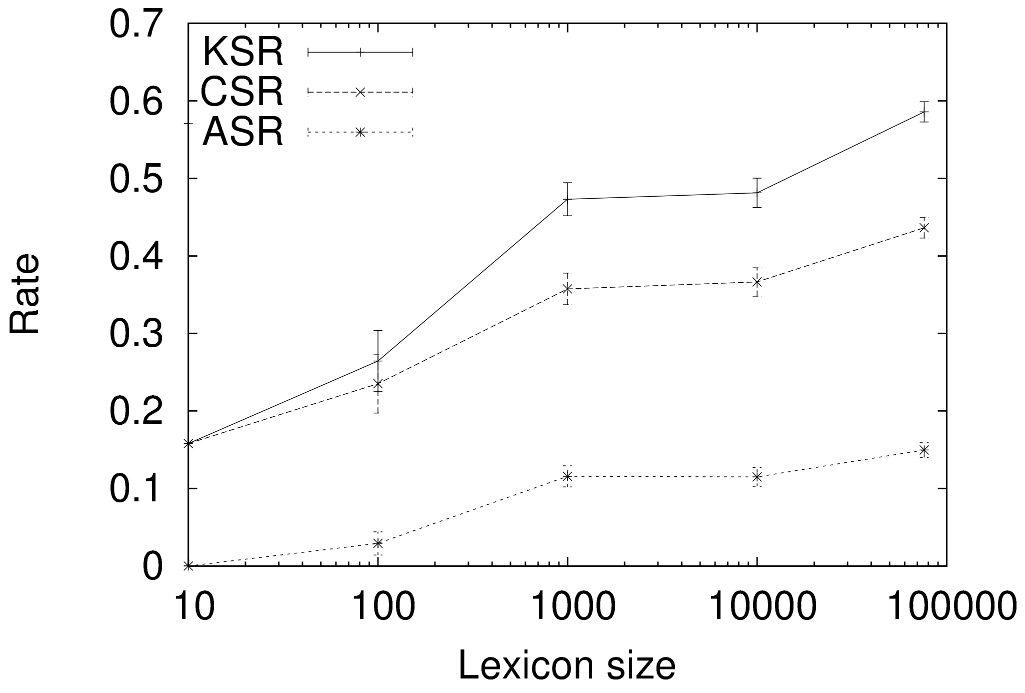

Figure 16: KSR, CSR and ASR (95% confidence interval) for varying lexicon size, for a non-multimodal approach.

6.2. Efficient data entry in GPS navigation devices

In this task, the names of 7660 Spanish municipalities were used to build the constraint model. As in previous experiments, a k equals to the longest municipality name was used, so only existing municipalities were possible outputs (deterministic model).

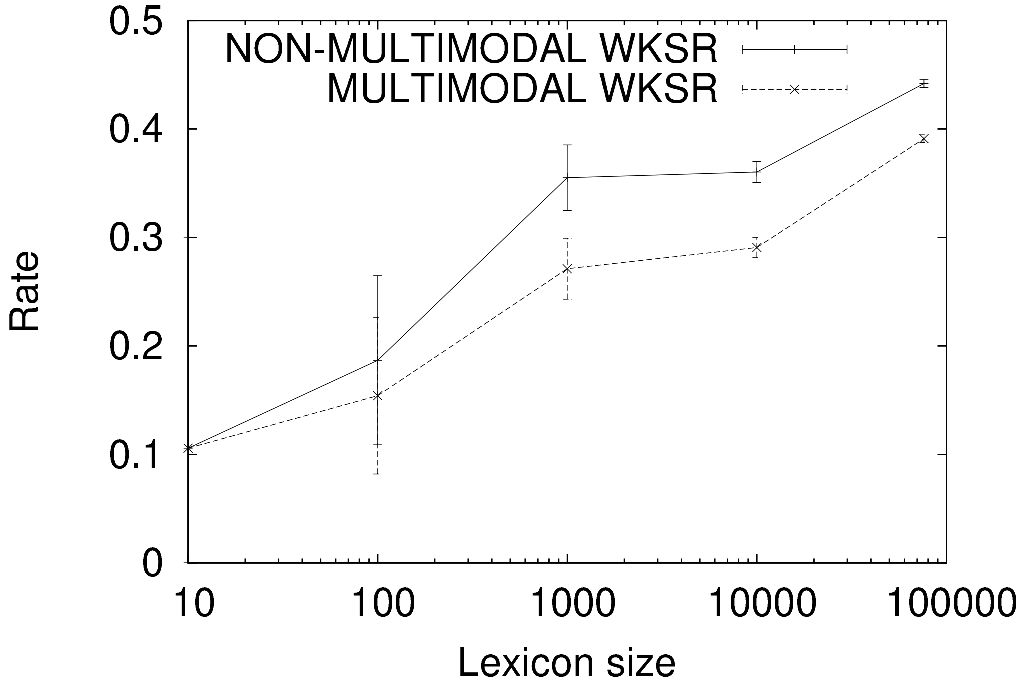

Figure 17: Weighted KSR comparison for multimodal and non-multimodal approaches (95% confidence interval).

The proposed application has not been embedded in a GPS navigation device. Instead, smart phones with *qwerty* touch-keyboards were used to simulate the data entry. A number of 84 test sets (bags), each of them consisting of 30 destinations (municipalities) randomly selected, which makes a total of 2520 destinations, were introduced by 21 users (120 destinations per user) and subsequently stored.

Each bag was built as follows: a coordinate pair representing a position within the Spanish territory was randomly generated. Then, 30 municipalities were selected using a random process that considered the proximity of the municipalities to the reference position and their population. The smaller the distance to the reference position and the larger the population, the higher the probability of the municipality to be selected.

Users were asked to use only the forefinger for data entry, to simulate the process commonly used in navigation devices. Also, they were asked not to correct mistyped characters, to assess the ability of the system to recover from user interaction errors. From the 2520 entered destinations, 1920 (76.2%) were free of errors, while the rest (23.8%) had at least one mistyped or missing character.

In order to test to what extent the addition of new evidences can improve the performance of the proposed method, three experiments with a growing number of models were carried out: a) only with an IM and a CM (EXP1), b) with EM_{IM} as well, i.e., the IM has its own error model that allows to recover user interaction errors (EXP2) and c) with the previous components plus a HM and an EM_{HM} (Figure 14) (EXP3).

EXP1 assumes a scenario where only the user input and the constraint model are used to predict the destination. This experiment can be considered a baseline. In EXP2, knowledge about common input errors related with the keyboard layout is added, allowing for input error recovery. Finally, in EXP3 a multimodal approach, in which a position and a radius enclosing the most likely destinations around this position are assumed.

KSR, CSR, ASR and WKSR were computed for all the experiments, as explained in Section 6.1, with the aim of assessing the required user effort to get the desired destination. Computational costs were also computed.

Table 7 shows the results obtained in each of the proposed experiments. KSR, CSR ASR and WKSR, along with the final error rates and response times (average delay between each user interaction and the generation of a new proposal) are shown. An error is computed when the proposed destination is not the intended one after the user has completely introduced all the characters (which occurs when the user entry contains errors and the system is unable to correct them). In EXP3 the destinations included in the test sets were within a radius of 30 km, so results shown in this table are for the same radius.

Table 7: User effort (KSR, CSR, ASR and WKSR), error rate and response time obtained in each of the proposed experiments.

	KSR	CSR	ASR	WKSR	Error(%)	Time (ms)
EXP1	0.58	0.45	0.13	0.34	15.56	0.05
EXP2	0.54	0.40	0.14	0.31	4.25	0.38
EXP3	0.18	0.15	0.03	0.11	0.40	1.62

It must be noted that, although 23.8% of the entries used in the test set had at least one mistyped or missing character, only 15.56% are erroneously corrected in EXP1, where no error model is applied. The reason is that the correct string can be found in the constraint model before the mistyped or missing character is reached in the input string. The data input process would be finished at that point in a real user interface. Therefore, even without error model, a small fraction of the strings are corrected by the system.

If the results obtained in EXP1 are taken as a baseline, it can be observed that the addition of new models encoding other available knowledge sources considerably improves the performance of the system in terms of user effort savings. It can be also concluded that the inclusion of an error model associated with user interaction (EXP2) drastically decreases the error rate. When an hypothesis model built from a given set of geographic coordinates, which can be manually set or taken from the current GPS position sensor, is also included (EXP3), then both significant user effort gains and error rate reductions are achieved.

Regarding response time, in all the cases the system is able to suggest new destinations much faster than a trained user would type new characters, which means that new suggestions will immediately appear as the user types new characters. In EXP2 and EXP3, the response time is higher due to the inclusion of new error and hypothesis models that require the shortest-path search algorithm to explore a larger number of paths. However, the interaction time is still in the range of a few milliseconds in a standard implementation.

Figure 18 shows KSR, CSR and ASR values for the different radius used to build the HM in EXP3. Destinations included in the test sets were within a radius of 30 km from a position previously set, which justifies the minimum of KSR at this point. The shorter the radius, the lower the probability that the desired destination is within the HM and, therefore, the higher the KSR. When the radius becomes zero, no HM is effectively used, which would be equivalent to EXP2. Conversely, the results indicate that the performance decreases more gradually when the radius modeled gets larger. So, the radius used to build the HM has an effect in the system performance. A radius too small can lead to a HM that does not contain the desired destination, resulting in an increase of the KSR, as the system will tend to propose destinations in the HM. On the other hand, a radius too large will give rise to an HM with a larger number of destinations, which can also produce a small increase of the KSR. The radius can be a user parameter, since the HM is easily built on-demand when interaction is required.

Figure 19 shows the relationship between the radius used to build HM and the response time achieved in EXP3. For the reasons discussed before,

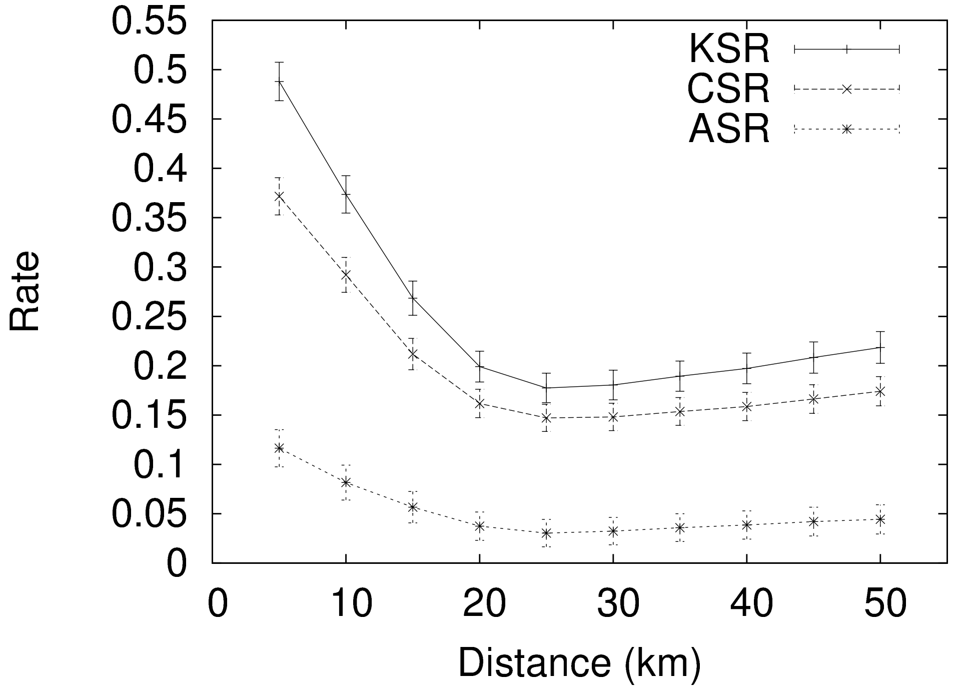

Figure 18: KSR, CSR and ASR related to radius used in building HM (95% confidence interval).

the response time increases as the radius is smaller or larger than the area in which desired destinations are most likely to be found (30 km in our experiments). In any case, the response time remains shorter than the typical period between keystrokes.

Finally, Figure 20 shows the error rate against the radius. The error rate approaches the one of EXP2 as the radius shrinks from its optimal value to zero, while remains very stable as the radius grows.

7. Conclusions

A flexible and generic Symbol Input Interactive Post-processing method has been presented. The use of WFSTs to encode and combine the set of Input Hypotheses, Error Models, a Constraint Model and a User Interaction Model has been proposed. In this method, no explicit parsing of an input string is performed, which is a clear difference with other systems. Instead,

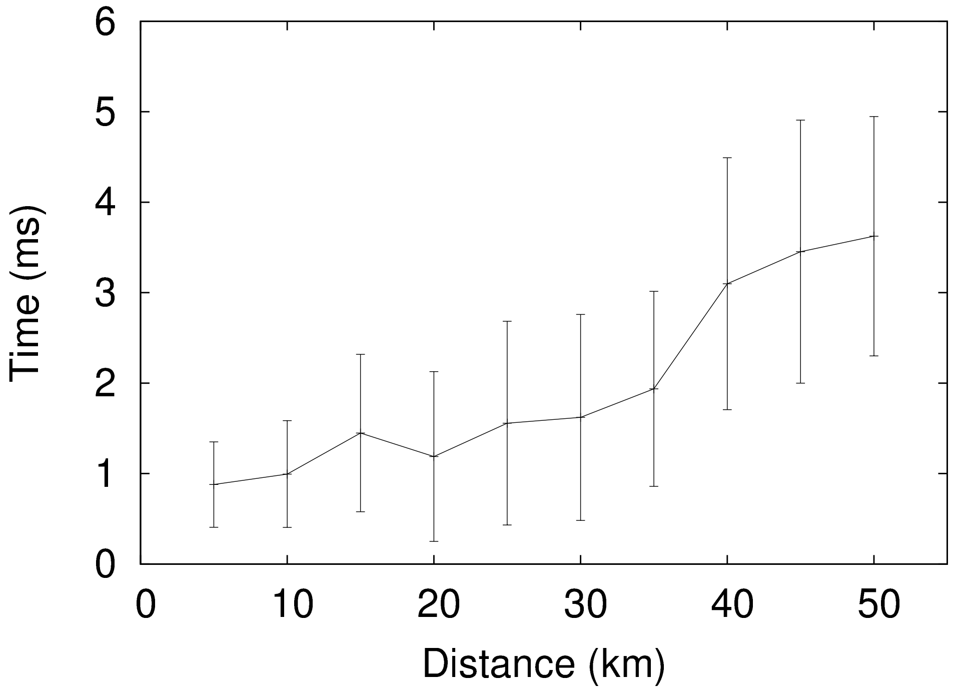

Figure 19: Average interaction time related to radius used in building HM (95% confidence intervals).

the *combination* of the different models is performed through the composition operation between transducers, and then the best path in the resulting transducer is found.

Independently encoding the different sources of information available allows for a decoupled architecture, which brings several advantages: ease the integration of different probabilistic models representing the available knowledge sources, use of multimodal approaches where inputs coming from different sources need to be combined, and design flexibility.

The different transducers are composed to perform the lowest cost path search, which gives the most probable string compatible with the Constraint, the Hypothesis and the Error Models. If the resulting string is wrong, the User Interaction Model can be used, along with a multimodal interactive technique, to get the correct string with a low effort level. No intermediate or irreversible decisions are taken from isolated models.

From a practical point of view, using independent Error, Constraint, In-

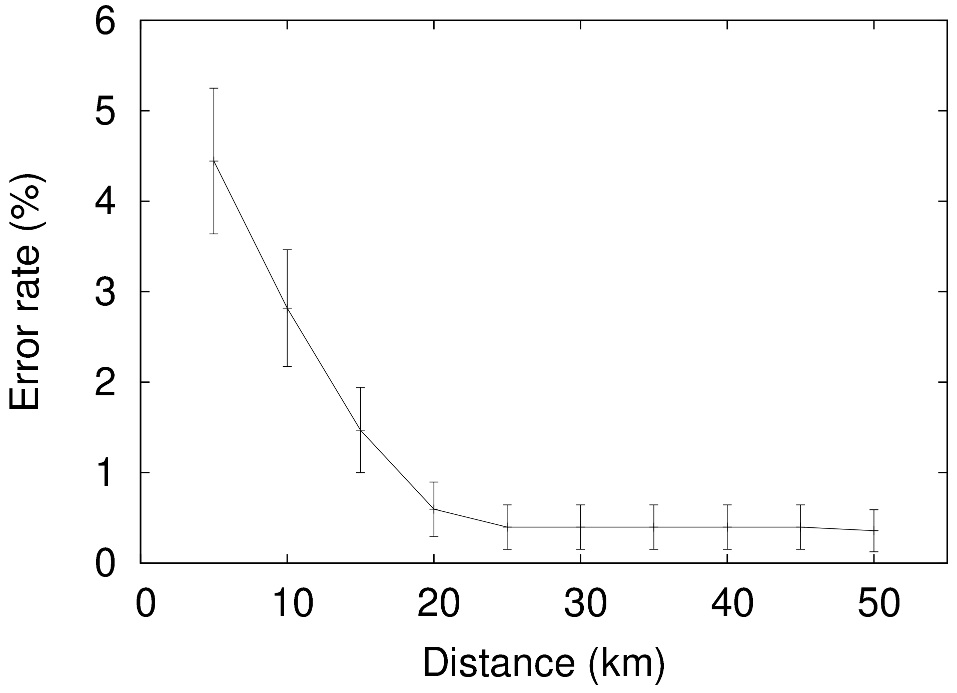

Figure 20: Error rate against radius (95% confidence intervals).

put Hypothesis and User Interaction Models gives a higher degree of design flexibility than other approaches where a global closely coupled model is used, such as Stochastic Parsing or Hidden Markov Models.

Although the method can be directly applied to many different tasks, two experiments of very different nature have been conducted to exemplify and test the proposed method. In both cases, significant improvements have been achieved. In the case of the OCR post-processing task, the approach presented takes advantage of all the information the OCR classifier can provide, such as the *a posteriori* class-conditional probabilities or the confusion matrix. Similarly, in the GPS data-entry task, significant improvements are achieved when an error model associated with user interaction or a hypothesis model encoding the most likely destinations are added. Also, the computational costs are very reasonable, allowing a smooth interaction with the system.

References

[1] R. Bastide, D. Navarre, P. Palanque, A. Schyn, P. Dragicevic, A model-based approach for real-time embedded multimodal systems in military aircrafts, in: Proceedings of the 6th international conference on Multimodal interfaces, ICMI '04, ACM, New York, NY, USA, 2004, pp. 243–250.

[2] C. Müller, G. Weinberg, Multimodal input in the car, today and tomorrow, IEEE MultiMedia 18 (1) (2011) 98–103.

[3] J. B. M. Georg F Meyer, S. M. Wuerger, Continuous audiovisual digit recognition using n-best decision fusion, Information Fusion 5 (2) (2004) 91–101.

[4] F. O. K. S. N. R. Bahador Khaleghi, Alaa Khamis, Multisensor data fusion: A review of the state-of-the-art, Information Fusion 14 (13) (2013) 28–44.

[5] P. A. V. Hall, G. R. Dowling, Approximate string matching, ACM Comput. Surv. 12 (4) (1980) 381–402.

[6] H. Berghel, A logical framework for the correction of spelling errors in electronic documents., Inf. Process. Manage. 23 (5) (1987) 477–494.

[7] T. Breuel, Language modeling for a real-world handwriting recognition task, in: AISB Workshop on Computational Linguistics for Speech and Handwriting Recognition, 1994.

[8] F. Farooq, D. Jose, V. Govindaraju, Phrase-based correction model for improving handwriting recognition accuracies, Pattern Recognition 42 (12) (2009) 3271–3277.

[9] J. C. Pérez-Cortes, J.-C. Amengual, J. Arlandis, R. Llobet, Stochastic error-correcting parsing for ocr post-processing, in: ICPR, 2000, pp. 4405–4408.

[10] J.-C. Amengual, E. Vidal, Efficient error-correcting viterbi parsing, IEEE Trans. Pattern Anal. Mach. Intell. 20 (10) (1998) 1109–1116.

[11] D. Neuhoff, The viterbi algorithm as an aid in text recognition., IEEE Trans. Inf. Theor. 21 (1975) 222–226.

[12] B. T. Al Azawi, M., Context-dependent confusions rules for building error model using weighted finite state transducers for ocr post-processing, in: Proc. of the IAPR International Workshop on Document Analysis Systems (DAS), 2014, pp. 116–120.

[13] H. H. Ahmed Hassan, Sara Noeman, Language independent text correction using finite state automata, in: Proc. of the International Joint Conference on Natural Language Processing, 2008, pp. 913–918.

[14] K. Raman, A. Swaminathan, J. Gehrke, T. Joachims, Beyond myopic inference in big data pipelines, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, ACM, New York, NY, USA, 2013, pp. 86–94.

[15] R. Llobet, J.-R. Cerdan-Navarro, J. Perez-Cortes, J. Arlandis, Ocr post-processing using weighted finite-state transducers, in: Pattern Recognition (ICPR), 2010 20th International Conference on, 2010, pp. 2021–2024.

[16] R. Llobet, J. R. Navarro-Cerdan, J. C. Pérez-Cortes, J. Arlandis, Efficient ocr post-processing combining language, hypothesis and error models, in: SSPR/SPR, 2010, pp. 728–737.

[17] J. Grande, J.-L. Guardiola, A. Perez-Jimenez, J.-C. Perez-Cortes, License plate recognition using weighted finite-state transducers and different evidence combination methods, in: J. Sanches, L. Mic, J. Cardoso (Eds.), Pattern Recognition and Image Analysis, Vol. 7887 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 229–236.

[18] J.-C. Perez-Cortes, R. Llobet, J. R. Navarro-Cerdan, J. Arlandis, Improvement of embedded human-machine interfaces combining language, hypothesis and error models, 2012 23rd International Workshop on Database and Expert Systems Applications 0 (2011) 359–363.

[19] P. F. Brown, S. D. Pietra, V. J. D. Pietra, R. L. Mercer, The mathematics of statistical machine translation: Parameter estimation, Computational Linguistics 19 (2) (1993) 263–311.

[20] Y. A. Park, R. Levy, Automated whole sentence grammar correction using a noisy channel model, in: ACL, 2011, pp. 934–944.

[21] M. Mohri, F. C. N. Pereira, M. Riley, The design principles of a weighted finite-state transducer library, Theor. Comput. Sci. 231 (1) (2000) 17–32.

[22] Allauzen C., Riley M., Schalkwyk J., Skut W., Mohri M.: OpenFst: A General and Efficient Weighted Finite-State Transducer LIbrary. Proceedings of CIAA, LNCS 4783,11–23 (2007)

[23] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R. C. Carrasco, Probabilistic finite-state machines–part i., IEEE transactions on pattern analysis and machine intelligence 27 (7) (2005) 1013–1025.

[24] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, R. C. Carrasco, Probabilistic finite-state machines-part ii, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (7) (2005) 1026–1039.

[25] J. Eisner, Parameter estimation for probabilistic finite-state transducers, in: ACL, 2002, pp. 1–8.

[26] M. Riley, F. Pereira, M. Mohri, Transducer composition for context-dependent network expansion, in: EUROSPEECH, 1997.

[27] P. Garcia, E. Vidal, Inference of k-testable languages in the strict sense and application to syntactic pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. 12 (9) (1990) 920–925.

[28] J. A. Nelder, R. Mead, A simplex method for function minimization, Computer Journal 7 (1965) 308–313.

[29] M. Mohri, Semiring frameworks and algorithms for shortest-distance problems, J. Autom. Lang. Comb. 7 (3) (2002) 321–350.

[30] C. Allauzen and M. Mohri, N-Way Composition of Weighted Finite-State Transducers, Int. J. Found. Comput. Sci. 20, 613–627 (2009).