# Interactive Text Generation for Information Retrieval

Luis Rodríguez[1], Alejandro Revuelta[1], Ismael García-Varea[1] and Enrique Vidal[2⋆]

[1] Departamento de Sistemas Informáticos, Universidad de Castilla La Mancha, Albacete, Spain
[2] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Valencia, Spain

**Abstract.** Interactive text generation is aimed at facilitating text generation in those situations where text typing is somehow constrained. This approach achieves a significant amount of typing effort reduction in most tasks. Natural language based interfaces for information retrieval constitute a good scenario to include this kind of assistance in order to improve the system usability and provide considerable help in constrained input-interfaces. An initial proposal is presented here along with an experimental framework to assess its appropriateness.

## 1 Introduction to Interactive Text Generation

Interactive Text Generation is an application focused on developing automatic assistance in generation of text documents. It is motivated by the fact that the time spent in typing (including, as well, thinking about what is going to be typed) can be quite long sometimes. A system able to predict, with some degree of accuracy, what someone is going to type will be utterly helpful and could save a considerable amount of effort.

In addition, in some situations, typing becomes a slow and uncomfortable task. For example, in some devices, such as mobile phones, no suitable input mechanisms are available. Moreover, some disabled people are not able to achieve a fast enough typing speed and, unfortunately, in some cases, this is the only way for them to communicate.

Previous approaches to this topic can be found in the literature. Most of them just attempt to predict the next single word and/or to complete the characters of each new word being typed by the user [9]. Other systems just focus on measuring the accuracy of off-line text predictions [1]. The proposal presented in this paper considers a more general scenario where a whole natural language query is predicted according to the text previously typed. The rationale behind this is to reduce the effort needed to obtain a satisfactory response from a database accepting natural language input queries.

## 2 Information Retrieval and ITG

In the last decades, Information Retrieval (IR) systems have gained a lot of importance in daily life. Querying databases is now a usual task and, therefore, appropriate

⋆ Work supported by the EC (FEDER, FSE), the Spanish Government (MICINN, MITyC, "Plan E", under grants MIPRCV "Consolider Ingenio 2010" CSD2007-00018, iTrans2 TIN20 and Junta de Comunidades de Castilla La Mancha under PIB08-0210-7127

RodrÃ■guez L., Revuelta A., GarcÃ■a-Varea I. and Vidal E.
Interactive Text Generation for Information Retrieval.
DOI: 10.5220/0003026900620071
In *Proceedings of the 10th International Workshop on Pattern Recognition in Information Systems (ICEIS 2010)*, page
ISBN: 978-989-8425-14-0

interfaces are crucial to guarantee a real benefit from the huge amount of data currently available. In this sense, interfaces based on natural language have drawn a lot of attention in the last years since they constitute the most natural way for a user to communicate with a system. In the case of accessing to a database this is even more important since the way in which the information is structured can be so complex that other communication alternatives are often useless. Nevertheless, these interfaces are far from being perfect and they present some drawbacks. On the one hand, in spite of allowing the use of natural language, these interfaces are usually constrained to a specific vocabulary or grammatical structure. When the input is not constructed following these constraints (and usually the typical user is not aware of them) the system response is useless and the user normally feels frustrated. On the other hand, typing text can be complicated in some situations (mobile devices, disabled people, etc.) and this fact can weaken the benefit that a natural language based communication can achieve. This issue is becoming more and more important since nowadays a lot of information is worldwide available and portable devices are becoming the main communication tool for a significant amount of people.

To overcome these two problems, ITG can be really useful. In the first place, regarding the second drawback, ITG can significantly reduce the effort in terms of interactions (that is key strokes) to generate text in very different tasks. In the second place, concerning the first question, ITG can boost the correct use of a natural language based interface since it can be seen as an interactive user guide to the IR system. This way, ITG can be used to lead the input text to the kind of constructions expected by the system and therefore to improve the final results obtained. The point here is that the ITG system could help the user, in the short term, by providing a quick way to obtain an acceptable result for the query currently carried out and, on the other hand, in the long term by showing the kind of queries that the system is more likely to accept.

The aim of this paper is to roughly explore the possibilities that this proposal could offer on a simple system and to discuss an initial experimental framework for this kind of applications. To this end, a database (AMSABEL) containing information about train routes will be used in the experiments.

## 3 ITG Theoretical Background

The task proposed in this work naturally fits into the Interactive Pattern Recognition (IPR) framework presented in [10]. The basic process consists in, as was commented before, predicting some portion of text based on the one previously typed. In IPR, we have an input pattern and the system proposes a possible decoding result. The user, based on the input and this proposal sends some feedback to the system. The point here is to take advantage of this feedback to improve the following decoding proposals.

Using the terminology adopted in [10], the *feedback* is essentially a prefix, composed of the sequence of words typed, or predicted by the system so far and *validated by the user*, and the system has to find a suitable continuation, or suffix, for this prefix (the difference is that here, no input pattern is available and the system suggestion is based only on the user-validated prefix).

The usual strategy to predict a multi-word suffix $\hat{\mathbf{s}}$ would be to maximize the probability of the suffix $\mathbf{s}$ given a prefix $\mathbf{p}$ [7], that is:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmax}}\, Pr(\mathbf{s}|\mathbf{p}) \tag{1}$$

which is aimed at obtaining the best possible suffix. This is perfectly reasonable in a classical pattern recognition task where we are trying to minimize the number of classification errors and the system is actually a completely automatic system. Nevertheless, under a general interactive text generation approach, the system works as follows. Given the text typed so far, the system produces a prediction that the user reads until a mistake is found. Next, the mistake is corrected and the system provides a new continuation taking this correction into account (as can be seen in Fig. 1). Here, we are interested in minimizing the number of user interactions (i.e, to save user effort) and, under this criterion, the approach showed in Eq. (1), which is usually solved by using a dynamic programming approach, is not necessarily optimal. In fact, a simple greedy strategy consisting in suggesting the most probable word given the text typed (prefix) so far turns out to be the optimal strategy for this scenario (as it is proved in [6]). It is worth noticing that this can be easily extended to multi-word predictions by considering each predicted word as part of the current prefix and, repeating this process until reaching the desired amount of predicted words.

**Iteration 1**
*Prediction:* `which are the classes of train 1093.`
*Prefix:* **which**
*Amendment:* **which** *c*
**Iteration 2**
*Prediction:* **which c**`ities can you go from Madrid.`
*Prefix:* **which cities can you go from**
*Amendment:* **which cities can you go from** *V*
**Iteration 3**
*Prediction:* **which cities can you go from V**`alencia.`
RESULT: **which *c*ities can you go from *V*alencia.**

$$KSR = \frac{3}{35} = 0.09 \rightarrow 9\%$$

**Fig. 1.** Example of an editing session and the corresponding Key Stroke Ratio computation. The system generates an initial prediction. Then, the user validates a correct prefix (boldfaced) and introduces an amendment (shown in italics). The system, taking into account this information, generates a new prediction. The process is iterated until a correct, full sentence is achieved. In the final result, the user only had to type the two characters shown in italics plus a final confirmation stroke. The KSR measure is obtained by dividing the number of user amendments between the overall number of characters.

### 3.1 Language Modelling using $n$-grams

In order to predict suffixes, a (*language*) model for $Pr(\mathbf{s}|\mathbf{p})$ is needed. Nowadays, under the statistical framework, $n$-grams [3] are the most widely used language models in

natural language processing applications. In general, an $n$-gram model computes the probability of a whole sentence, or sequence of words, $\mathbf{w} = \mathbf{w}_1^l$ as:

$$Pr(\mathbf{w}) \approx \prod_{i=1}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \tag{2}$$

where $Pr(\cdot)$ denotes the real probability distribution and $P(\cdot)$ the estimated-model (approximate) distribution [3].

In the problem addressed here, this model has to be slightly modified since we are not dealing with whole sentences but with suffixes that complete a given prefix.

Basically, the idea, when adopting this kind of models, is to take advantage from the last $n-1$ words in the prefix to predict an appropriate continuation. Clearly, these models fail in taking advantage from the total information available in the whole prefix. However, no other types of (longer-dependence) language models are commonly acknowledged to overcome the capabilities of $n$-grams in natural language applications. Therefore, for the time being, the system is completely based on $n$-grams.

Let $\mathbf{p} = \mathbf{w}_1^k$ be the given user prefix and let $\mathbf{s} = \mathbf{w}_{k+1}^l$ be a possible suffix hypothesis. The system computes $Pr(\mathbf{s} \mid \mathbf{p})$ as is shown in Eq. (3).

$$\begin{aligned} Pr(\mathbf{s} \mid \mathbf{p}) &= Pr(\mathbf{p}, \mathbf{s}) / Pr(\mathbf{p}) \\ &\approx \frac{\prod_{i=1}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1})}{\prod_{i=1}^{k} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1})} \\ &= \prod_{i=k+1}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \end{aligned} \tag{3}$$

Here, the user feedback is reflected in the terms from $k+1$ to $k+n-1$ of this factorization, where we have additional information coming from the already known words $\mathbf{w}_{k-n+2}^k$. Hence, this expression can be further factorized as:

$$\begin{aligned} Pr(\mathbf{s} \mid \mathbf{p}) &\approx \prod_{i=k+1}^{k+n-1} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \cdot \prod_{i=k+n}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \\ &= \prod_{j=1}^{n-1} \Pr(\mathbf{s}_j | \mathbf{p}_{k-n+1}^{k}, \mathbf{s}_1^{j-1}) \cdot \prod_{j=n}^{l-k} \Pr(\mathbf{s}_j | \mathbf{s}_{j-n+1}^{j-1}) \end{aligned} \tag{4}$$

The first term 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.

### 3.2 Searching for a Suffix

The maximization in Eq. (1) is intended to minimize the number of classification errors (i.e, the number of whole suffixes correctly predicted). However, the goal here is to

[3] As in [11], we assume that for any string $\mathbf{z}$, the substring $\mathbf{z}_i^j$ denotes the empty string $\lambda$ if $j < i$. In addition, we assume that $Pr(\mathbf{z}|\lambda) \equiv P(\mathbf{z})$

minimize the user effort and, in this case, the optimal strategy to achieve this is based on a greedy algorithm [6]. On account of this, using $n$-gram models, at step $i$ and having the prefix $\mathbf{p} = \mathbf{w}_1^{i-1}$, the next predicted word will be:

$$v^* = \underset{v \in V}{\text{argmax}}\, P(v|\mathbf{w}_{i-n+1}^{i-1}) \tag{5}$$

where $V$ is the vocabulary considered. This leads to the simple Algorithm 1.

**Algorithm 1**: Greedy strategy to complete a user-validated prefix. Note that the greedy solutions shorter than *maxLen* are just the prefixes of the resulting $\mathbf{w}$. Given a prefix, the algorithm iterates over all the words in the vocabulary to obtain the one with the highest probability.

```
user validated prefix (p), vocabulary (V), maximum prediction length (maxLen),
n-gram size (n) whole sentence prediction (w) begin
    w = p; i = |p| + 1;
    while i < maxLen do
        v* = λ; g* = 0;
        forall v ∈ V do
            if g* < P(v|w_{i-n+1}^{i-1}) then
                g* = P(v|w_{i-n+1}^{i-1});
                v* = v;
        w = w · v*;
        i = i + 1;
    return w;
end
```

The previous description assumes that the system works at word-level and the user has to type a whole word before getting a prediction. Nevertheless, the system can also work at character-level (and this is, indeed, the normal operation mode) which means that as soon as the user introduces a new character the sytem will make a new prediction. To deal with this situation, when the final characters in the prefix do not form a word but a word-prefix, the system will apply Eq. (5) but considering only the words "compatible" with this word-prefix (for example, in the second iteration of the example shown in Fig. 1 only those words beginning with the letter 'f' are actually considered in the search).

## 4 Integrating ITG into a Natural Language based Information Retrieval System

Different approaches can be taken when including an ITG system into an information retrieval application. Perhaps, the most simple one is based on a completely decoupled architecture where the ITG is used basically as a tool to construct a query hypothesis. In this case, the user interacts with the ITG system as it is shown in Fig. 1, that is, typing text that is simply auto-completed by the ITG engine. Once the user validates the whole sentence, it is used to query the database.

A second alternative can be considered based on a loosely-coupled approach. In this case, an ITG system is used as in the previous case but, instead of waiting for the user validation of the whole natural language query, the different text predictions will be used to build a new query in order to retrieve information from the database. As a consequence, what the user obtains in each interaction is not merely a text completion but an answer to the query being constructed.

Other (and more complex) alternatives can be also explored. For instance, a strongly coupled approach, where the ITG system could be fed with some feedback from the information retrieval process in order to decide which prediction is more convenient in the current iteration.

### 4.1 Brief Description of the AMSABEL System

In order to assess this initial proposal, a simple information retrieval system called AMSABEL [8] will be used. AMSABEL is based on the use of statistical machine translation (SMT) techniques to translate from a natural language into a structured query language (SQL). Currently, AMSABEL is able to accept queries both in Spanish and English. The resulting SQL sentences are used to access a railway database where information about train routes is stored (specifically, for each route, fields as departure and arrival cities, starting and ending dates, starting and arrival times, ticket prices, etc. can be queried).

The translation of the Spanish or English input into SQL is performed by using statistical phrase-based models [5]. These models perform the translation in three steps. Firstly, the input sentence is segmented into phrases (which are sequences of consecutive words). Then, each segment is translated into the corresponding segments in the target language and, finally, the target phrases are properly ordered to achieve the final translation. Formally, in statistical machine translation we are given a source sentence $\mathbf{f}$, and we try to find the optimal target sentence $\mathbf{e}$ as:

$$\underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{f}|\mathbf{e}) \cdot Pr(\mathbf{e}) \tag{6}$$

where $Pr(\mathbf{e})$ is the language model probability (usually $n$-grams, see section 3.1) and $Pr(f|e)$ is the translation model probability. In the case of phrase models, this probability is expressed as it is stated in Eq. (7):

$$\begin{aligned} Pr(\mathbf{f}|\mathbf{e}) &= Pr(\bar{\mathbf{f}}_1^I|\bar{\mathbf{e}}_1^I) \\ &= \prod_{i=1}^{I} \phi(\bar{\mathbf{f}}_i|\bar{\mathbf{e}}_i) d(\mathbf{a}_i - \mathbf{b}_{i-1}) \end{aligned} \tag{7}$$

where $\bar{\mathbf{f}}_i$ is the $i$-th phrase in $\mathbf{f}$, $\bar{\mathbf{e}}_i$ is the $i$-th phrase in $\mathbf{e}$, $\phi(\bar{\mathbf{f}}_i|\bar{\mathbf{e}}_i)$ is the probability of being $\bar{\mathbf{f}}_i$ a translation of $\bar{\mathbf{e}}_i$ and $d(\mathbf{a}_i - \mathbf{b}_{i-1})$ is the distortion model used for reordering target phrases. The order of a target phrase $\bar{\mathbf{f}}_i$ depends on a probability distribution based on its start position ($\mathbf{a}_i$) and the end position of $\bar{\mathbf{f}}_{i-1}$ ($\mathbf{b}_{i-1}$).

Phrase models are obtained from word to word alignments ([2]). The search for the most likely translation is performed by using the *Moses* beam-search decoder [4].

A set of semi-automatically generated input queries was used to assess the system performance (752 English and 748 Spanish sentences were employed) along with the information that the user expects to obtain for each query (see Table 1) . We think that using this synthetic corpus can be justified in this first evaluation stage, considering all the implications of experiments involving real users. On the other hand, the produced sentences can be, in our opinion, considered as plausible queries for a user interacting with a natural language based IR system. In Fig. 2 some examples of these test queries are shown.

please , I would like to know which destinations are there from Guadalajara.
which are the classes of train 1047 ?
what times can you go from Toledo to Alicante the 2011-06-03 ?
which days of the week can you go to Guadalajara from Ciudad Real ?

**Fig. 2.** Examples of test sentences.

The evaluation procedure was aimed at directly testing the system usefulness from a potential user point of view. To this end, the results of the input natural language queries were classified based on the outcome obtained (the point here is to measure the system accuracy according to the correctness of the information retrieved). The following query categories were established:

**Q1** *Exact Information*: The system provides the user with the exact information required.
**Q2** *More Fields*: The system returns the information required but more fields are also provided.
**Q3** *More Rows*: The system provides the information required but more rows from the database tables are also shown.
**Q4** *Incorrect*: The query does not include the expected information.

Categories 1 and 2 are completely useful since they provide the user with the information that he or she requested (adding in the case of Q2 type more fields). Q3 type can be also considered useful since the information requested is provided although some useless additional information is also included in the result (causing, maybe, some annoyance to the user). In the case of the Spanish test corpus, about 81% of system results were classified as type Q1, less than 0.3% as type 2, about 1.3% as type 3 and, finally, about 17% as type 4 (that is, quite useless).

### 4.2 ITG Experiments

Once the information retrieval system used has been described, a framework to estimate the possible usefulness of the ITG proposal when included into a IRS will be discussed.

Firstly, regarding the metric employed to estimate the effort that the system can save, the Key Stroke Ratio (KSR) measure was adopted. This measure is simply computed

by dividing the number of key strokes performed by the user by the overall characters in the test set (see Fig. 1).

Initially, the KSR of a pure ITG approach on AMSABEL corpus was computed and it is shown in the first row of Table 2. The idea is to measure the effort required to generate the exact text queries (without involving database accessing) in the test set. This can be seen, to some extent, as the effort of using a completely decoupled approach in which the system waits for a completely validated text construction and, based on this, a final single access to the database is performed for each query.

As was described in section 4, ITG can be incorporated into an IR system in different ways. In this work, we will focus on a loosely-coupled architecture. The idea is that, each time the ITG system makes a text prediction, this prediction will become into a query to the database. Once the query has been performed the user will directly validate the database response in the case it is correct or, otherwise, the part of the text prediction that he or she considers error-free (as it is shown in Fig. 3). Then, a simple correction will be made on this text prediction and the process will be repeated until the user considers the information retrieved as satisfactory. In the experiments performed here, the user was simulated by the test sets shown in Table 1. The effort will be measured in KSR terms.

**Table 1.** Test sets used in ITG for information retrieval.

| | Spanish | English |
|---|---|---|
| Number of sentences | 748 | 752 |
| Running words | 9413 | 9763 |
| Vocabulary size | 447 | 484 |
| Perplexity (3-grams) | 5.2 | 4.9 |



In the first row of Table 2 the results of using ITG on the whole AMSABEL system are shown. These experiments are aimed at measuring the KSR required to obtain one of the three useful query types described in section 4.1. This way, the KSR needed to obtain all the queries classified as type Q1 was computed and the same procedure was used for Q2 and Q3 queries (Q4 queries were not considered since they are useless from the information retrieved point of view). As can be observed, the Q1 numbers shows that it is not necessary to generate the exact input natural language query to get a perfect result and that some effort can be saved with respect to replicate the input sentence with ITG (as numbers in Table 2 shown) and, what is more important with a small effort in terms of key strokes, useful Q3 queries can be achieved.

**Table 2.** ITG and information retrieval results on the AMSABEL corpus.

| | Spanish | | English | |
|---|---|---|---|---|
| **Query type** | KSR | improvement | KSR | improvement |
| **Pure ITG** | 15.6 | – | 14.7 | – |
| **Perfect queries (Q1)** | 14.2 | 16 % | 13.0 | 11% |
| **More columns (Q2)** | 14.2 | 16 % | 12.9 | 12% |
| **More rows (Q3)** | 10.7 | 36 % | 9.5 | 35% |

**Iteration 1**
*Prediction:* which days can you go to Madrid from Toledo
*SQL:* *SELECT DISTINCT Dias FROM Tren JOIN Viaje ON Viaje.Tren=Tren.Id_tren WHERE Destino='madrid' AND Origen='toledo'*
*Data:* Train 1012 → Monday, Tuesday, Friday
*Prefix:* **which**
*Amendment:* **which** *c*

**Iteration 2**
*Prediction:* **which c**lasses can you buy for train number 1040 ?
*SQL:* *SELECT DISTINCT Clase FROM Tren JOIN Billetes ON Billetes.Tren = Tren.Id_tren WHERE N_tren = '1040'*
*Data:* Train 1040 → Tourist, First class
*Prefix:* **which classes can you buy for train number 10**
*Amendment:* **which classes can you buy for train number 10***2*

**Iteration 3**
*Prediction:* **which classes can you buy for train number 102**8
*SQL:* *SELECT DISTINCT Clase FROM Tren JOIN Billetes ON Billetes.Tren = Tren.Id_tren WHERE N_tren = '1028'*
*Data:* Train 1028 → Tourist
*Prefix:* **which classes can you buy for train number 102**
*Amendment:* **which classes can you buy for train number 102***9*

**Iteration 4**
*Prediction:* **which classes can you buy for train number 1029 ?**
*SQL:* *SELECT DISTINCT Clase FROM Tren JOIN Billetes ON Billetes.Tren = Tren.Id_tren WHERE N_tren = '1029'*
*Data:* Train 1029 → Business, First class

$$KSR = \frac{3}{49} = 0.061 \rightarrow 6.1\%$$

**Fig. 3.** Example of information retrieval using ITG. In this example, the user tries to obtain information about the different ticket classes for train number 1029. Each time the user makes a key stroke, the system provides a text completion (*Prediction*) which is translated into *SQL* and the corresponding information retrieved is shown (*Data*). In the first three iterations, the information retrieved is not correct and, therefore, the user interacts with the ITG engine correcting the text prediction mistakes. Finally, at iteration 4, the information provided is fully correct and the process ends successfully.

## 5 Conclusions and Future Work

In this paper a completely new approach based on combining an interactive text generation system with a natural language based interface for information retrieval has been presented. Different approaches along with a proposal of an experimental framework have been also discussed. The results achieved are very promising since they suggest a significant effort reduction in this scenario.

Finally, as a future work, experiments involving real users should be considered, especially to evaluate the capabilities of this kind of systems to provide a more complete assistance apart from reducing typing effort.

## References


1. S. Bickel and P. Haider T. Scheffer. Predicting sentences using n-gram language models. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 193–200, Morristown, NJ, USA, October 2005. Association for Computational Linguistics.
2. P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roosin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.
3. F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, USA, 1998.
4. P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*, June 2007.
5. P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, pages 48–54, Edmonton, Canada, May 2003.
6. J. Oncina. Optimum algorithm to minimize human interactions in sequential computer assisted pattern recognition. *Pattern Recognition Letters*, 30(5):558–563, April 2009.
7. Duda R, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, New York, NY, 2nd edition, 2000.
8. A. Revuelta-Martínez, L. Rodríguez, and I. García-Varea. Multilingual access to online help systems and databases. *Procesamiento del Lenguaje Natural*, 44, 2010.
9. H. Trost, J. Matiasek, and M. Baroni. The language component of the fasty text prediction system. *Applied Artificial Intelligence*, 19(8):743–781, September 2005.
10. E. Vidal, L. Rodríguez, F. Casacuberta, and I. García-Varea. Interactive pattern recognition. In *Proceedings of the 4th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms, Volume 4892 of LNCS*, pages 60–71, Brno, Czech Republic, 28–30 June 2007.
11. E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines - part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1025–1039, 2005.