

# Stochastic Error-Correcting Parsing for OCR Post-processing.

Juan C. Perez-Cortes

Juan C. Amengual

Joaquim Arlandis

Rafael Llobet

Instituto Tecnológico de Informática (ITI)

Universidad Politécnica, Camino de Vera s/n

46071, Valencia, Spain

jcperez@disca.upv.es

## Abstract

*In this paper, stochastic error-correcting parsing is proposed as a powerful and flexible method to post-process the results of an optical character recognizer (OCR). Deterministic and non-deterministic approaches are possible under the proposed setting. The basic units of the model can be words or complete sentences, and the lexicons or the language databases can be simple enumerations or may convey probabilistic information from the application domain.*

## 1 Introduction

The result of automatic optical recognition of printed or handwritten text is often affected by a considerable amount of error and uncertainty, and it is therefore essential the application of a correction algorithm. A significant portion of the ability of humans to read a handwritten text is due to their extraordinary error recovery power, thanks to the lexical, syntactic, semantic, pragmatic and discursive language constraints they apply.

Among the different levels at which language can be modeled [2], the lowest one is the word level, involving lexical constraints on the sequential relations of characters inside each word. The next one is the sentence level, which takes into account syntactic and semantic constraints on the sequence of words or word categories inside a sentence (or a field, for instance, in a forms-processing application). The higher levels consider a wider context and require specific *a priori* knowledge of the application domain.

Word and sentence level models typically apply dictionary search methods,  $n$ -grams, Edit Distance-based techniques, Hidden Markov Models, and other character or word category transition models.

A common goal of OCR post-processing methods is to guarantee (or, at least, maximize the probability) that the

words or sentences generated by corrections to the OCR output are correct in the sense that they belong to the language of the task. This language can be as simple as a small set of valid words (e.g. the possible values of the *country* field in a form) or as complex as an unconstrained sentence in a natural language. Correspondingly, several approaches exist to the problem, which can be categorized into two main groups: deterministic and non-deterministic.

The deterministic approach is often applied to uncomplicated languages and requires, in its simplest form, the compilation of a complete lexicon (a finite list of tokens, which can be words, sentences, etc.). In this case, a token  $t$ , output by an OCR algorithm and referred to as the *recognized string*, will be said to belong to the language with a certain likelihood when its difference, according to a suitable similarity measure, to the most similar token  $c$  in the lexicon is lower than a given threshold. Token  $c$  will be returned as the *corrected string* in response to  $t$ . When it is not feasible to build a list with all possible valid strings, a non-enumerative representation of the language can be used, like a formal grammar or a set of prefixes, word roots, suffixes, etc.

In a non-deterministic model, the corrected string is not forced to belong *necessarily* to an explicit or implicitly enumerated list of tokens. Instead, the recognized (input) string is analyzed, and a corrected string is built by maximizing the probability that every symbol or sub-sequence of symbols belongs to the language. In other words, the corrected string is a version of the recognized string that conforms more closely to the constraints of the language model.

In forms-processing applications, it is often useful a deterministic model for fields like *name*, *city*, *state*, etc., but a non-deterministic model is preferred for more complex inputs as *address*, *occupation*, etc. In the latter case, however, the sequential application of a deterministic model for each word, followed by a non-deterministic model, with symbols corresponding to words or syntactic word categories, can provide better results. Similar combinations, or

non-deterministic models alone, can be also useful in other, more general, document processing tasks.

## 2 Previous work

A large number of data structures have been proposed to efficiently perform approximate search in a lexicon, an essential part for deterministic correction models. The simplest one is an alphabetically sorted list of words. Unfortunately, specific techniques and indices are needed to carry out approximate search in them. The same can be said about other generic data structures like hash tables, search trees, etc. In [6], an excellent survey of approximate string search methods is presented.

Other more specific techniques exist, like the methods that try to benefit from the high performance of exact search algorithms, by generating a number of neighboring strings to the one sought for, and searching all of them [12]. Unfortunately, that neighborhood can grow exponentially with the number of symbols of the string, and thus heuristic and probabilistic constraints have to be imposed.

In other cases, the lexicon is subdivided according to different criteria, like the length of the words, the first symbols, etc., reducing the search time by only a constant factor. Another interesting group of techniques is based on fast metric-space nearest neighbor search techniques, considering the strings as points in a dissimilarity space and performing dimensionality reduction techniques, tree search, or fast search based on the properties of the metrics.

On the other hand, the  $n$ -gram models constitute a simple but widely used family of non-deterministic techniques. Their high potential can be illustrated by some results reported in [2], where an  $n$ -gram model was applied to a 10,000 English word lexicon, finding only 68% of the 676 possible digrams, 20% of the 17576 trigrams and 2% of the 456976 4-grams.

The general concept used by the technique proposed in this work was first applied to text correction in [10]. The idea is to build a finite-state machine (or a Markov model) that accepts (or generates with a certain probability) the strings in the lexicon or language sample. When the model is applied to a candidate word, if it is accepted, no correction is performed, and if not, the smallest set of transitions that could not be traversed shows which is the most similar string in the model. When this concept is applied to a formal grammar, it is called Error-Correcting Parsing (ECP). The classical algorithm, widely used in different fields, to find the maximum likelihood path on a Markov model and to perform ECP on a regular grammar is the Viterbi Algorithm, based on the Dynamic Programming paradigm. Their applications were recognized through the work of Forney [4], and it was used for the first time in [11] for text correction. In [13] and [7], improvements and new con-

cepts were applied to the method, including the contribution of the confusion matrix to the error model. In [14], further refinements were introduced.

In [3], a related approach to the problem of finding the most similar strings is presented. In that method, no error-correcting parsing is needed, but the construction of the automaton allows an efficient conventional parsing, directly yielding the required results. A high spatial cost is, however, the main problem of the approach.

Many recent works on language modeling have been carried out in the field of continuous speech recognition [8]. Although the requirements are very different (for example, deterministic language models are seldom applied to speech), many basic techniques used in that discipline can be applied to OCR tasks with little modification.

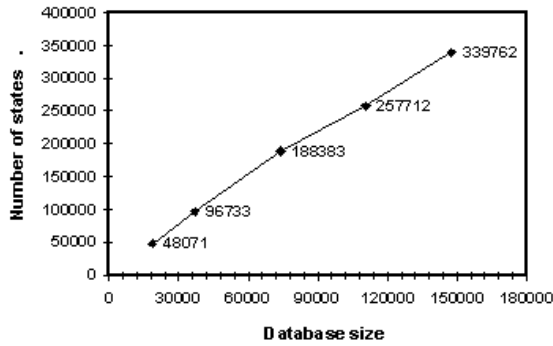
## 3 Stochastic Error Correcting Parsing for OCR Post-processing

In recent years, mainly due to important achievements in parsing efficiency, the Viterbi Algorithm has been used for stochastic error correcting parsing with excellent results in complex pattern recognition and speech-oriented language modeling tasks [1].

Here, we propose the application of a grammatical inference algorithm to build a stochastic finite-state machine that accepts the smallest  $k$ -Testable Language in the Strict Sense ( $k$ -TS language) [5] consistent with a task-representative language sample. The set of strings accepted by such an automaton is equivalent to the language model obtained using  $n$ -grams, for  $n=k$ . The stochastic extension of the basic  $k$ -TS language is performed through a maximum likelihood estimation of the probabilities associated to the grammar rules, evaluated according to their frequency of utilization by the input strings. This computation is carried out incrementally and simultaneously with the inference process.

A major advantage of the chosen setting resides in its flexibility. The language sample can be a simple lexicon (with each word appearing only once), a list of words extracted from a real instance of the task (with each word appearing as many times as in the sample), a list of sentences with the characters, words or word 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 language sample (as will be seen, this gives rise to clearly poorer results in the experiments presented in section 4). In the other cases, if the sample is representative of the task, the model will take advantage of the probabilistic information present in the data.

The value of  $k$  can also be used to define the behavior of the model. In a lexical model, if  $k$  is made equal to the length of the longest word in the database, a deterministic



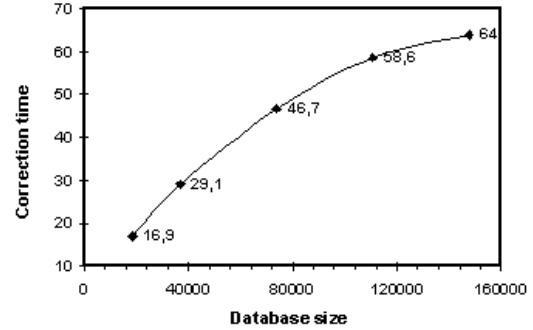
**Figure 1. Number of states of the model, plotted against the number of words in the lexicon.**

post-processing method is obtained (only words that exist in the database can be generated), but if  $k$  is set to a lower value, a non-deterministic model will result, where the corrected words can belong, or not, to the reference database. Note that the stochastic 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  $k$ , and that, therefore, for instance, a deterministic model can be based on a stochastic grammar, when the database is a language sample and  $k$  is large, and a non-deterministic model could make use of a regular grammar, when the database is a simple lexicon and  $k$  is small.

The worst case spatial cost of a straightforward implementation of the model is linear with the size of the database but, in practice, it is usually slightly sub-linear. In figure 1, the number of states of a model composed by words derived from a lexicon of 18450 Spanish names, artificially enlarged by randomly exchanging some of the letters, is shown. The average length of the strings was 6.66.

Once the language model has been built, a number of issues regarding the parsing procedure remain to be discussed. The most important one is the error model and dissimilarity measures that will allow the system to generate a correct word or sentence from an incorrect input string (in the non-deterministic case, substrings instead of the string as a whole are considered). The efficiency of the procedure is another important aspect to address.

We have used an Stochastic Error-Correcting Parsing algorithm based on an extension of the Viterbi Algorithm [1]. Formally, our problem is to find a minimal cost path through a multistage directed graph (trellis) associated to the finite automaton (model) and to the input string  $a$ . In this graph, each node  $q_{jk}$  corresponds to a state  $q_j$  of the automaton in the  $k^{th}$  stage of the parsing process, corresponding to the



**Figure 2. Times (in ms) in a 450Mhz Pentium.**

$k^{th}$  symbol of  $a$ , and each arc  $t_k = (q_{ik}, q_{j, k+1})$  represents a transition from state  $q_i$  in stage  $k$  and state  $q_j$  (which can be the same) in stage  $k + 1$  (next symbol of  $a$ ). All the details, diagrams and algorithms can be found in [1].

The extension for error correction implies the addition of new arcs to the graph, according to the error rules of the dissimilarity function chosen [1]. The most widely used error model is based on the insertion-deletion-substitution editing operations. We can use a different cost for each of the three operations. The cost of substitution is usually the lowest one, since it is often the only directly imputable to an OCR error. In many applications, insertions and deletions are rare, being caused by writer's mistakes or document segmentation problems. If the cost of these operations are high, they will be included only in otherwise high probability paths.

As for the cost of substitutions, it can be estimated from two sources of information: the confusion matrix of the classifier and the *a posteriori* class-conditional probabilities provided by the classifier output. The first one provides a very consistent estimation (since it can be derived from a very large sample) of the *a priori* probability of a given character being mixed up with another one by the classifier. The second one can be considered less reliable, but it takes into account the features of the character under analysis, and therefore conveys a more dynamic information, compared to the static nature of the confusion matrix. In the results presented here, only the confusion matrix has been used. Ongoing work is being carried out on the integration of the *a posteriori* class-conditional probabilities into the parsing method, and not only in the cost of the substitution operations.

The computational cost of the parsing procedure can be very low if techniques like beam-search are used to improve the efficiency of the algorithm [1]. In figure 2, the time to correct an input string with an average length of 5.95 symbols is plotted against the size of the lexicon. A beam width of  $\alpha = 30$  was used.

**Table 1. Percentages of incorrect names and characters at 0% rejection rate.**

	Word errors	Char. errors
OCR	6842 ( <b>32.54%</b> )	9294 ( <b>6.36%</b> )
Non-stochastic model	1006 ( <b>4.78%</b> )	1888 ( <b>1.29%</b> )
Stochastic model	303 ( <b>1.74%</b> )	659 ( <b>0.45%</b> )

## 4 Experiments

A deterministic correction model for the *name* field in a handwritten form processing application has been chosen to perform some tests. A commercial neural network-based OCR system in use by a document processing company was applied to a real handwritten form-processing task, taking the OCR output as the input to our method. The data provided by the OCR did not include a reliability index for each character, but the special symbol “?” was output when the quality of the recognition was below a certain threshold, which was fixed by the company operators using standard criteria. The confusion matrix was computed using a validation set of 777080 characters from a different set of forms.

The number of strings tested was 21028, totalizing 146060 characters, and the training language sample consisted of 27125 (mostly Spanish) first names, 8088 of which were unique. Several tests were conducted: the first without correction of the OCR output, the second using the 8088-word conventional lexicon to build a non-stochastic *k*-TS automaton, and in the last one, the full list of 27125 names was used to build a stochastic model. Since a fairly complete list of Spanish names was available, all the models tested were deterministic (the value of *k* was set to the length of the longest name). The results are shown in table 1. The ECP postprocessing does improve the results of single character recognition as it could be expected [1], but it is remarkable the decrease achieved in word error rate. This is explained by the (combined) effect of both language and error models, which also appears in Speech Recognition when independently trained stochastic acoustic and language models are tightly integrated [9]. Some examples of both corrections and miscorrections are shown in table 2

## 5 Conclusions

A post-processing method for the correction of OCR results has been proposed. A stochastic finite-state automaton accepting a *k*-Testable Language in the Strict Sense is learned from a language sample extracted from the application domain. Then, a modified version of the Viterbi Algorithm is used to perform a stochastic error correcting parsing of the strings provided by the OCR module, and the result-

**Table 2. Examples of processed strings.**

Corrected names		Incorrectly “corrected” names		
OCR output	Corrected	OCR output	“Corrected”	Correct
HDRTD	MARTA	SANA	JANA	SARA
ROSDN?	ROSANA	NHRJH	NURIA	MARIA
HONICB	MONICA	IUAN	JUAN	IVAN
EOGENLO	EUGENIO	HARTA	MARTA	MARIA
?DRID	MARIA	ANTONI?	ANTONIO	ANTONIA
JE?OMIYO	JERONIMO	JIZID	JULIA	MARTA
OOLORES	DOLORES	SOAUA	JOANA	SONIA

ing string is given as the corrected output. Excellent time and space efficiency as well as very good correction results have been obtained. Moreover, the flexibility of the method allows the designer to define very precisely the conditions of the correction procedure for different tasks with widely variable requirements.

## References

- [1] J. Amengual and E. Vidal. Efficient error-correcting viterbi parsing. *IEEE Trans. on PAMI*, 20(10):1109, Oct. 1998.
- [2] H. L. Berghel. A logical framework for the correction of spelling errors in electronic documents. *Information Processing and Management*, 23(5):477–494, Sept. 1987.
- [3] H. Bunke. Fast approximate matching of words against a dictionary. *Computing*, 55(1):75–89, 1995.
- [4] G. Forney. The viterbi algorithm. *Proceedings of IEEE*, 61(3):268–277, March 1973.
- [5] P. Garcia and E. Vidal. Inference of *k*-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Trans. on PAMI*, 12(9):920–925, sep 1990.
- [6] P. Hall and G. Dowling. Approximate string matching. *ACM Surveys*, 12(4):381–402, December 1980.
- [7] J. Hull and S. Srihari. Experiments in text recognition with binary *n*-gram and viterbi algorithms. *IEEE Trans. on PAMI*, 4(5):520–530, sep 1982.
- [8] F. Jelinek. Up from trigrams, the struggle for improved language models. In *European Conference on Speech Communication and Technology, Berlin*, pages 1037–1040, 1993.
- [9] F. Jelinek. *Statistical Methods for Speech Recognition*, in *Language, Speech and Communication*. MIT Press, 1997.
- [10] H. Morgan. Spelling correction in system programs. *Communications of the ACM*, 13:90–94, 1970.
- [11] D. Neuhoff. The viterbi algorithm as an aid in text recognition. *IEEE Trans. Information Theory*, 21:222–226, 1975.
- [12] E. Riseman and A. Hanson. A contextual postprocessing system for error correction using binary *n*-grams. *IEEE Trans. Computer*, 23:480–493, 1974.
- [13] R. Shinghal and G. Toussaint. Experiments in text recognition with the modified viterbi algorithm. *IEEE Trans. on PAMI*, 1(2):184–192, April 1979.
- [14] R. C. S.N. Srihari, J.J. Hull. Integrating diverse knowledge sources in text recognition. *ACM Trans. Off. Inform. Sys.*, 1(1):68–87, 1983.