Efficient OCR Post-Processing Combining Language, Hypothesis and Error Models[*]

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

Instituto Tecnologico de Informatica
Universidad Politecnica de Valencia
Camino de Vera s/n, 46071 Valencia, Spain
{rllobet,jonacer,jcperez,arlandis}@iti.upv.es

Abstract. In this paper, an OCR post-processing method that combines a language model, OCR hypothesis information and an error model is proposed. The approach can be seen as a flexible and efficient way to perform Stochastic Error-Correcting Language Modeling. We use Weighted Finite-State Transducers (WFSTs) to represent the language model, the complete set of OCR hypotheses interpreted as a sequence of vectors of *a posteriori* class probabilities, and an error model with symbol substitutions, insertions and deletions. This approach combines the practical advantages of a de-coupled (OCR + post-processor) model with the error-recovery power of a integrated model.

1 Introduction

Any method of optical recognition of printed or handwritten text is subject to variable amounts of error and uncertainty in the output. The application of a correction algorithm is therefore very important. The excelent performance shown by humans when we read a handwritten text is mostly due to our extraordinary error-recovery ability, thanks to the lexical, syntactic, semantic, pragmatic and discursive language constraints we routinely apply.

The goal of an OCR post-processing method is to optimize the likelihood that the strings generated as OCR hypotheses are correct, in the sense that they are compatible with the constraints imposed by the task. These constraints conform the Language Model and 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.

In practice, the simplest method to handle OCR output correction is to use a lexicon to validate the known words and ask the operator to verify or input manually the unknown words. Specific techniques can be used to carry out

[*] Work partially supported by the Spanish MICINN grants TIN2009-14205-C04-02 and Consolider Ingenio 2010: MIPRCV (CSD2007-00018) and by IMPIVA and the E.U. by means of the ERDF in the context of the R+D Program for Technological Institutes of IMPIVA network for 2010 (IMIDIC_2009/204).

approximate search in the lexicon. In [10] an excellent survey of string search methods is presented.

Other methods are based on n-grams or on finite-state machines [9,12,3], where a candidate string is parsed and the set of transitions with the lowest cost (highest probability) defines the output string. The classical algorithm, 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 [7,8].

All these approaches use a string provided by the OCR as input, apply a Language Model and often optimize a transformation cost using an Error Model, but, in general, they do not take into account another valuable knowledge source that we call the *Hypothesis Model*. Depending on the OCR classifier used, this can include the *a posteriori* class probabilities of the output hypothesis or another reliability index for the most likely classes. Another element to take into account is the classifier's confusion matrix., that should be efficiently and adequately included into the Error Model.

2 Weighted Finite-State Transducers

Weighted Finite-State Transducers (WFST) have been widely used in speech recognition, machine translation and pattern recognition, among other disciplines. In this paper we propose the use of WFSTs in Stochastic Error-Correcting Language Modeling for OCR post-processing.

A WFST can be seen as a generalization of a Finite-State Automata (FSA) [1,4]. An FSA can be seen as a finite directed graph with nodes representing states and arcs representing transitions. Each transition is labeled with a symbol from an alphabet Σ. Formally, an FSA is defined as a five-tuple (Q, q_0, F, Σ, δ) where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the subset of final states, Σ is a finite set of symbols and $\delta : Q \times \Sigma \rightarrow Q$ is the set of transitions between states. Each transition t is labeled with a symbol $s(t) \in \Sigma$. FSAs are used to *accept* or *reject* sets of strings over Σ: given a string $w \in \Sigma^*$, w is accepted if there is a path from the initial state to a final state of the graph whose transition labels form the string w when concatenated.

However, in Finite State Transducers (FSTs) each transition is labeled with an input symbol $\in \Sigma$ and an output symbol $\in \Delta$. Therefore, the function δ is defined as $\delta : Q \times \Sigma \rightarrow Q \times \Delta$. FSTs are used to *transduce* strings of an input language over Σ into strings of an output language over Δ. The weighted version of an FST (WFST) include a weight in their transitions, used to represent the cost of taking a particular path. Furthermore, each state q has an initial weight $w_i(q)$ and a final weight $w_f(q)$. A state q is *initial* if $w_i(q) \neq \bar{0}$ and *final* if $w_f(q) \neq \bar{0}$.

An FSA and its weighted counterpart WFSA can be seen as an FST or WFST respectively, with same input and output symbols in each transition. This is called the identity transducer.

The FSTs (and WFSTs) are considered specially flexible and powerful due to some fundamental properties. In particular, the approach presented in this paper relies on the *composition* operation [6]. Given two 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 $T_3 = T_1 \odot T_2$ transduces x to z with weight $w_1 \otimes w_2$.

3 Description of the Method

The proposed approach entails building and composing WFSTs that encode different informations and represent distinct models, extending the idea of OCR language modeling through Stochastic Error Correcting Parsing proposed in [3].

We identify three sources of information in the OCR post-processing task: a) the OCR output (including all the hypotheses for each character and their class probabilities), b) a model of the expected errors and their probabilities, and c) the language the strings of the task belong to. Each of these sources of information can be represented by a Stochastic Finite-State Machine that we call the Hypothesis Model (HM), the Error Model (EM) and the Language Model (LM) respectively.

3.1 The Language Model (LM)

We propose the use 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 $n = k$.

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 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. In the other cases, 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 sample, a post-processing method is obtained where only words that exist in the sample are valid, but if k is set to a lower value, a classical n-gram model will result, where the corrected words may not be in the reference sample.

Figure 1 shows the probabilistic identity transducer associated with the sample S={aba, abb, ba, bac} and $k = 3$. In this description, for convenience, we have used a transducer with input and output symbols equal in each transition, i.e., the identity transducer, which can be seen as an acceptor of the language $L(S)$.

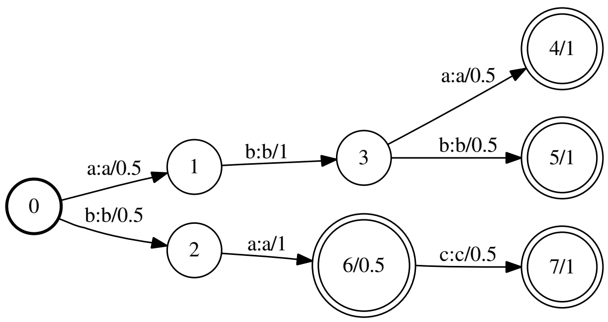

Fig. 1. Example of an identity transducer representing a language model

3.2 The Hypothesis Model (HM)

The output of a recognizer, in our case, an OCR classifier can be seen, in its most general form, as a sequence of n-dimensional vectors $\bar{v}_1 \ldots \bar{v_m}$, where n is the number of possible hypotheses for each character, m the length of the output string and $v_{i,j}$ the *a posteriori* probability of the j^{th} hypothesis of the i^{th} character. We propose to represent this sequence using a WFSA (or an identity WFST) with $m+1$ states and n transitions between each pair of states. Figure 2 shows an example of a WFST with alphabet $[a, b, c]$ that represents the OCR output $[0.8, 0.2, 0.0], [0.1, 0.7, 0.2], [0.0, 0.6, 0.4]$. This means that the first symbol of the OCR output is a with probability 0.8 or b with probability 0.2, the second symbol is a, b or c with probabilities 0.1, 0.7 and 0.2 respectively, and so on. Transitions with zero-probability are not shown in the graph.

Instead of working exclusively with the most probable output (*abb* in the example) this transducer models the uncertainty of the OCR classifier.

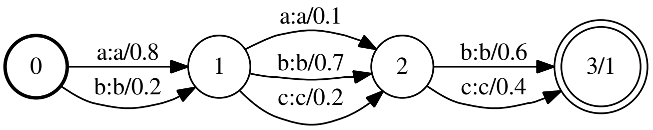

Fig. 2. Example of an identity transducer representing a hypothesis model. The *a posteriori* probabilities from the OCR classifier are shown as the arc weights.

3.3 The Error Model (EM)

In some cases, none of the character sequences included in the OCR hypothesis is compatible with the language model or a similar variant is more probable than any of the original alternatives. In a classical n-gram model, this effect is accounted for by a *smoothing* procedure. In our case, the possible variations allowed and their probabilities are represented by an *Error Model.*

Typically, the three usual edit operations will be defined: *substitutions* (including the substitution of a symbol by itself), *insertions* and *deletions.* Given two

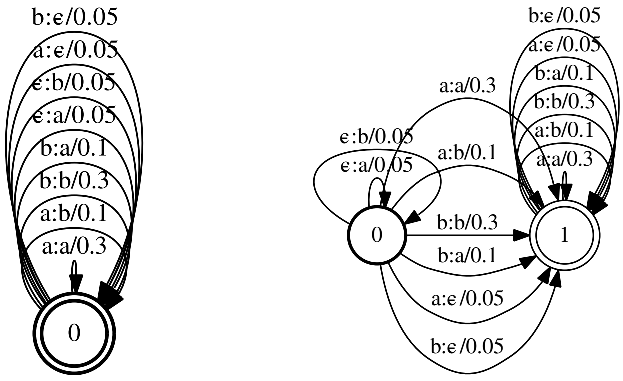

Fig. 3. Examples of two error model transducers, with all possible insertions, deletions and substitutions (left) and with insertions only at the begining of the string (right)

symbols s_1, s_2 and the empty symbol ϵ, substitutions, insertions and deletions are transductions of type s_1/s_2, ϵ/s_2 and s_1/ϵ respectively.

Each of these operations can be assigned a probability. The probability of substitutions is derived from the confusion matrix of the OCR classifier. This matrix is a table containing the confusion probability of each pair of symbols estimated using a representative corpus. It can be interpreted as a "static" model of the uncertainty of the OCR classifier, complementing the "dynamic" estimation provided by the *a posteriori* probabilities. The likelihoods of insertions and deletions are task-dependent and can be empirically estimated. Figure 3 shows an example of a WFST representing an error model with symbols in {a,b}.

3.4 Composing LM, EM and HM

The combination of the different models is performed through the composition operation between transducers:

Let L_1 be the set of strings that a given HM can produce, and L_2 the set of strings accepted by a given LM. Our goal is to find the most likely transduction of a string in L_1 into a string in L_2 by means of the intermediate transduction defined in an EM. This process is equivalent to finding the most probable path in the transducer HM $\odot$ EM $\odot$ LM.

The transducer T_1 = HM $\odot$ EM transduces any string from L_1 by applying the operations of the error model EM. Figure 4 shows the composition of the transducers HM and EM shown in Figures 2 and 3 respectively. This automaton transduces the strings accepted by HM to any string in Σ^*.

Therefore, the transducer T_2 = T_1 $\odot$ LM accepts only strings belonging to L_2, and the result of the transduction with the most probable path is the final corrected string. If several alternatives are needed, the n-best paths can also easily be obtained.

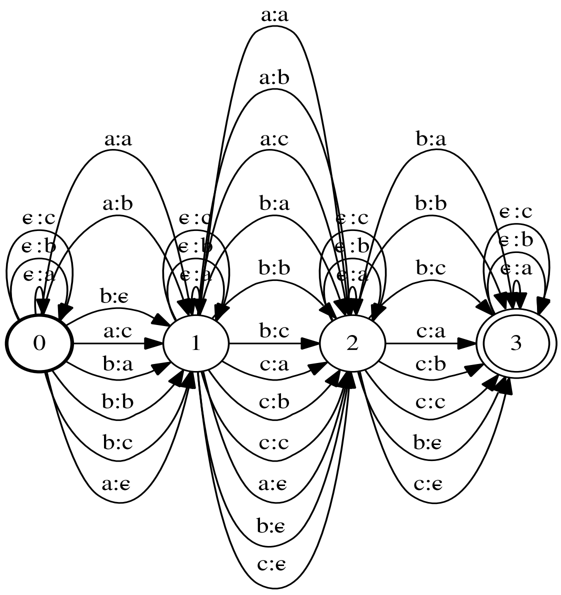

Fig. 4. Composition of the HM shown in Fig. 2 with an EM with all possible substitutions, insertions and deletions

3.5 Cost Definition and Parameter Optimization

The computation of the best path is obviously a key element in the process. A path is a sequence of transitions in the composed transducer and each transition t has an associated probability, which is computed as the product of the probabilities of the corresponding transitions in HM, LM and EM. Assuming independence and an equal influence from all models, we can define the probablity of a transition as:

$$P(t) = P(\mathrm{LM}, \mathrm{EM}, \mathrm{HM}|t) = P(\mathrm{LM}|t)P(\mathrm{EM}|t)P(\mathrm{HM}|t)$$

The probability of the output string can therefore be computed as the product of the probabilities of the transitions along the most probable path in the composed transducer. Given $x \in L_1$ and $y \in L_2$, the probability of the transduction x, y is $P(x, y) = \prod_{i=1}^{n} P(t_i)$, where $t_1 \ldots t_n$ is the sequence of transitions that transduces x into y.

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. Therefore, the most probable path will be found using a lowest cost path search.

Since the optimum influence of each model is generally not known, two parameters λ_e and λ_h are defined to obtain a log-linear parametric combination of the models with different weights:

$$P(t) = P(\mathrm{LM}|t)P(\mathrm{EM}|t)^{\lambda_e}P(\mathrm{HM}|t)^{\lambda_h}$$

We consider a fixed weight 1 for the LM, therefore its influence is controlled by the absolute values of the other parameters. The values of λ_e and λ_h, along with the cost of insertions and deletions, mentioned in Section 3.3, can be empirically estimated using a supervised training set.

In a typical form-processing task in the data entry industry, it is very important to obtain a consistent confidence value (in our case, the probability associated to the shortest path in the combined transducer) allowing the user to define a threshold and a reliable reject strategy. Consequently, we have optimized the aforementioned parameters using a criterion function that maximizes the recognition rate, defined as the percentage (with respect to the total test set) of strings that were accepted and successfully corrected, for a given error rate (percentage, also in the total test set, of the strings that were accepted and generated wrong corrections). With this strategy, only rejected strings have to be reviewed by human operators, meaning that –for a commercially acceptable error rate– the economic savings yielded by the system are roughly equivalent to the number of accepted strings.

3.6 Pruning

WFST composition of very large transducers can incur in large computational costs. For a LM of 64000 states and 140000 transitions (like the one used in 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 together with a pruning scheme 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 [1]. In our approach, the composition is delayed until the search of the shortest path (the lowest cost path) in the resulting transducer is performed. In principle, it is necessary to completely compose the transducers to compute the shortest path, but we have used a simple pruning search optimization to provide an approximate solution that allows not to explore (and therefore compose) the whole transducer.

To deal with the shortest path search, a best-first algorithm which explores the automaton by expanding the lowest cost path at each state is used. A vector with the minimum cost found at each stage (path length) is maintained. During the search, a pruning is performed based on a parameter δ. If the cost of a partial solution of length n exceeds δ times the cost of the best path of length n found so far ($v[n]$), then the path of the partial solution is pruned.

Obviously, this heuristic leads to an approximate search, since the lowest cost path could be pruned. This can happen when δ is too low or when the best path contains high-cost transitions in its first stages. To avoid pruning a partial solution that could lead to the best path too early, a parameter ρ is used, so that the pruning scheme is not applied to partial solutions shorter than ρ states.

4 Experiments

The following experiments compare the system working with and without multiple hypotheses and *a posteriori* probabilities in HM. We used a sample of 14000 handwritten surnames from forms scanned in a real industrial task, with a reference language model of 4.5 million Spanish surnames (99157 of which were unique). A k equal to the largest surname was used in the LM, so only known surnames were accepted as corrected output. The OpenFST library was used for the experiments [1,2].

The corpus was divided into a training (15%) and a test (85%). The training set was used to estimate the parameters of the error model (insertion and deletion probabilities) and of the WFSTs composition (λ_h and λ_e) using the criterion function defined in Section 3.5. Since the influence of each individual model can vary depending on the selected approach –using multiple hypotheses and *a posteriori* probabilities (WFST-PP) or using only the most probable OCR output (WFST)– independent optimizations were performed for each approach.

Table 1 shows the best parameters found for WFST and WFST-PP. It can be noted that the optimal working point in the WFST approach is achieved when all the models have similar weights (note that LM has a fixed weight of 1), whereas the WFST-PP approach achieves better performance when the HM has a higher weight than the other two models. Also the insertion and deletion probabilities are lower in the WFST-PP approach, since more strings can be corrected with a lower cost by choosing one of the symbols proposed by the HM rather than by deletion and insertion operations.

Table 1. Optimal parameters found with and without *a posteriori* probabilities

	λ_e	λ_h	p_i	p_d
WFST-PP	1.17	2.38	0.005	0.004
WFST	1.05	1.04	0.007	0.008

Figure 5 shows the recognition and error rates of the proposed method using *a)* multiple hypotheses and *a posteriori* probabilities in HM (WFST-PP), *b)* the same approach using only the OCR strings (WFST), and *c)* the original, uncorrected OCR output.

The computational cost is another important issue in this task, where the size of the models can be very large in practice, and the typical operations involve large batchs of documents to recognize. A set of experiments were carried out to test the influence of the pruning method presented in Section 3.6. Figure 6 shows the average correction time (ms.) obtained in an Intel Xeon 2.5 GHz with 2 GB of memory, Linux OS and gcc 4.4, and the accuracy (percentage of well corrected words) achieved for different values of δ and ρ. These results were obtained for a

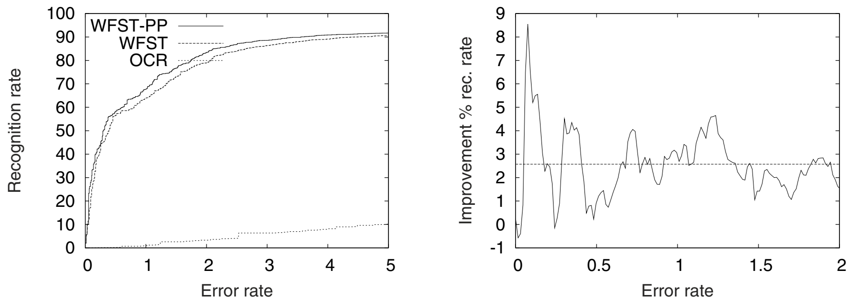

Fig. 5. Recognition rate against error rate comparison for different approaches in the range of 0 to 5% error rate and detail of the improvement obtained using an Hypothesis Model with posterior probabilities, in the range of 0 to 2% error rate

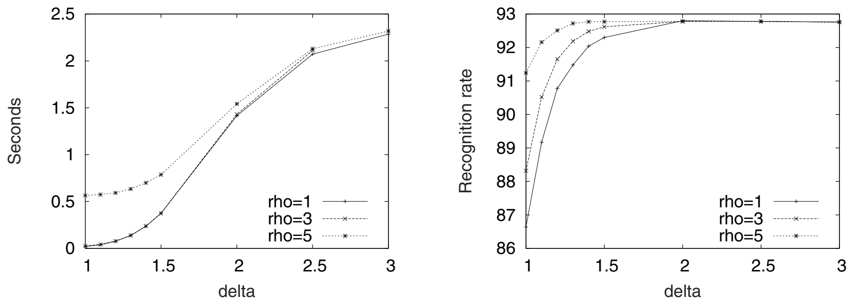

Fig. 6. Average correction time (left) and percentage of well corrected words at zero rejection rate (right) for different values of δ and ρ

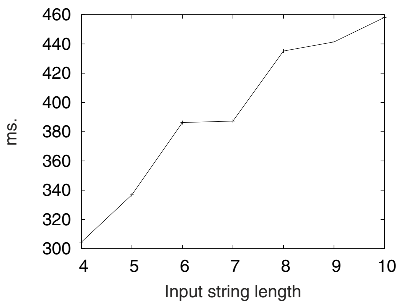

Fig. 7. Average correction time (ms.) against input string length for $\delta = 1.5$ and $\rho = 3$

language model built from 99157 unique words. For larger language models, the computational cost grows sub-linearly. Figure 7 plots the average computational cost for $\delta = 1.5$ and $\rho = 3$, against the length of the input OCR hypothesis.

5 Conclusions

A post-processing method for OCR using WFSTs to encode the set of classifier hypotheses, an error model and a language model implementing a k-Testable Language has been proposed. The lowest cost path in the composed transducer gives the most probable string compatible with the language, the hypothesis and the error models. According to the tests conducted with handwritten data, significant improvements over previous approaches can be obtained efficiently.

Finally, in our view, using independent error, language and OCR models that can be modified without affecting the other parts of the system offers important practical advantages over other more closely coupled paradigms.

References

1. Mohri, M., Pereira, F., Riley, M.: The design principles of a weighted finite-state transducer library. Theoretical Computer Science 231, 17–32 (2000)
2. Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M.: OpenFst: A General and Efficient Weighted Finite-State Transducer LIbrary. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 11–23. Springer, Heidelberg (2007)
3. Perez-Cortes, J.C., Amengual, J.C., Arlandis, J., Llobet, R.: Stochastic Error Correcting Parsing for OCR Post-processing. In: Proceedings of the ICPR, vol. 4, pp. 405–408 (2000)
4. Vidal, E., Thollard, F., de la Higuera, C., Casacuberta, F., Carrasco, R.: Probabilistic Finite-State Machines - Parts I and II. IEEE Trans. on Pattern Analysis and Machine Intelligence 27, 1013–1039 (2005)
5. Garcia, P., Vidal, E.: Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. IEEE Trans. on PAMI 12, 920–925 (1990)
6. Riley, M., Pereira, F., Mohri, M.: Transducer composition for context-dependent network expansion. In: Proc. of Eurospeech 1997 (1997)
7. Amengual, J., Vidal, E.: Efficient error-correcting viterbi parsing. IEEE Trans. on PAMI 20, 1109–1116 (1998)
8. Neuhoff, D.: The viterbi algorithm as an aid in text recognition. IEEE Trns. on Inf. Theory 21, 222–226 (1975)
9. Berghel, H.L.: A logical framework for the correction of spelling errors in electronic documents. Information Processing and Management 23, 477–494 (1987)
10. Hall, P., Dowling, G.: Approximate string matching. ACM Surveys 12, 381–402 (1980)
11. Beaufort, R., Mancas-Thillou, C.: A Weighted Finite-State Framework for Correcting Errors in Natural Scene OCR. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition, vol. 2, pp. 889–893 (2007)
12. Farooq, F., Jose, D., Govindaraju, V.: Phrase-based correction model for improving handwriting recognition accuracies. Pattern Recognition 42, 3271–3277 (2009)