OCR Post-processing Using Weighted Finite-State Transducers

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

A new approach for Stochastic Error-Correcting Language Modeling based on Weighted Finite-State Transducers (WFSTs) is proposed as a method to post-process the results of an optical character recognizer (OCR). Instead of using the recognized string as an input to the transducer, in our approach the complete set of OCR hypotheses, a sequence of vectors of a posteriori *class probabilities, is used to build a WFST that is then composed with independent WFSTs for the error and language models. This combines the practical advantages of a de-coupled (OCR + post-processor) model with the full power of an integrated model.*

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 our extraordinary error recovery power, thanks to the lexical, syntactic, semantic, pragmatic and discursive language constraints we apply.

A common goal of OCR post-processing methods is to maximize the probability that the words or sentences generated as OCR hypotheses are correct in the sense that they are compatible with 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.

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 approximate search in the lexicon. In [6] an excellent survey of string search methods is presented.

Other methods are based on n-grams or on finite-state machines [3, 4, 10], where a candidate string is parsed and the set of transitions with the lowest cost (highest probability) defines the output string. Most of these methods rely on the Viterbi Algorithm [2, 9].

All the aforementioned approaches use as their input a string provided by the OCR engine, but do not take into account other valuable knowledge sources such as the *a posteriori* class probabilities of the output hypothesis of the OCR classifier, or its confusion matrix. In contrast, we propose a flexible method that takes advantage of all the information the OCR can provide.

2. Weighted Finite-State Transducers

A Weighted Finite-State Transducer (WFST) is a generalization of a Finite-State Automaton (FSA) [8, 12]. 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 $(Q, q_0, F, \Sigma, \delta)$ 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. 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.

In contrast, 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

*Work partially supported by *IMPIVA*, Eur. Community grant IMIDIC-2009-204, and Spanish MICINN grants TIN2009-14205-C04-02 and Consolider Ingenio 2010: MIPRCV (CSD2007-00018)

FST (WFST) include a weight in their transitions, used to represent the cost of taking a particular path.

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.

FSTs (and WFSTs) are very flexible and powerful due to some fundamental properties. In particular, the approach presented in this paper relies, among others, on the *composition* operation [11]. 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 $w_1 \otimes w_2$.

The major drawback for composition is the computational cost. To avoid this problem *lazy* composition can be 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 [8]. In our approach, 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.

3. OCR post-processing using WFSTs

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.

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.

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 $n = k$.

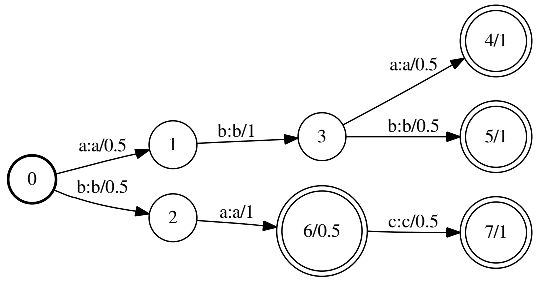

Figure 1. Example of an identity transducer representing a language model.

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. 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 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 $k = 3$. 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 $L(S)$.

3.2. The hypothesis model (HM)

An OCR output can be seen 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.

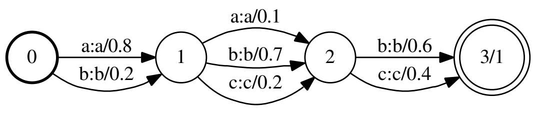

Figure 2. Example of an identity transducer representing a hypothesis model.

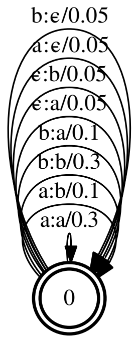

Figure 3. Example of a transducer representing an error model

3.3. The error model (EM)

It is possible that none of the character sequences that can be generated from the OCR hypothesis is compatible with the language model or that a small variation of one of them is more probable than any of the original alternatives. The possible variations allowed and their probabilities are represented by an *error model*.

We have included in our model three types of operations: *substitutions* (including the substitution of a symbol by itself), *insertions* and *deletions*. Given two 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 has a cost in the model. The cost of substitutions is extracted from the confusion matrix associated with the OCR. 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 cost of insertions and deletions is task-dependent and can be empirically estimated. Figure 3 shows an example of a WFST representing an error model.

Using an independent error model that can be modified without affecting the LM or the HM models is an important practical advantage. For example, it is easy to build an error model that allows insertions or deletions only at the beginning or at the end of the string, to restrict the number of error operations, to represent other sources of error like transposition of two adjacent symbols, different confusion matrices, etc.

3.4. Composing LM, EM and HM

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 lowest cost 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. Then, the transducer $T_2 = T_1 \odot LM$ accepts only strings belonging to L_2, and the result of the transduction with the lowest cost path is the final corrected string. Since we encode a probability in the cost of each transition t of these transducers, we can compute the probability of the output string as the product of the probabilities of the corresponding transitions in HM, LM and EM. Assuming independence:

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

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 $t_1 \dots t_n$ is the sequence of transitions that transduces x into y. Since the influence of each model is generally not known, the parameters λ_e and λ_h are defined [7].

We 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. Experiments

A set of experiments was conducted to compare the proposed approach with our earlier Stochastic Error Correcting Parsing method [10] and with the uncorrected OCR output. 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 $k = 25$ (the length of the largest surname) was used in the LM, so only known surnames were accepted as

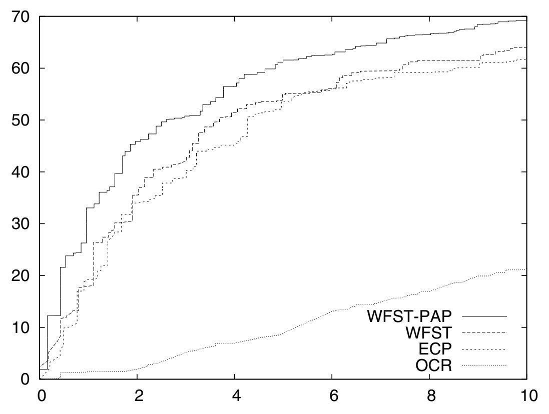

Figure 4. ROC curve comparison for different approaches (only first 10% shown).

corrected output. The OpenFST library was used for the experiments [8, 1].

The corpus was divided into a training (15%) and a test (85%) set using a hold-out scheme. 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).

In form-processing tasks 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 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.

The computational cost is another important issue. In our case, for the language model described, with $64K$ states and $140K$ transitions, the average correction time is 270 ms per string in an Intel Xeon 2.5 GHz.

Table 1 shows the recognition rate for three error rates and four different approaches: a) the proposed method using multiple hypotheses and *a posteriori* probabilities in HM (WFST-PP), the same approach using only the OCR strings (WFST), Stochastic Error Correcting Parsing [10] (ECP) and the uncorrected OCR output (OCR). Figure 4 shows the first 10% of the ROC curves for these experiments.

Table 1. Experimental results

Error rate (%)	Recognition rate (%)			
	WFST-PP	WFST	ECP	OCR
0.1	30.6	25.1	24.7	0.1
0.25	45.5	40.5	41.2	0.6
0.5	61.5	54.6	53.9	0.7

5. Conclusions

A new method to post-process OCR results has been proposed and tested with handwritten data of a real task from the data entry industry. The method uses the sequence of vectors of *a posteriori* class probabilities from the OCR to build a WFST that is then composed with the error and language WFSTs. The results show an important reduction in the errors of the OCR classifier and a clear improvement on our previous approach.

References

[1] C. Allauzen, M. Riley, and et al. Openfst: A general and efficient weighted finite-state transducer library. In *Proceedings of CIAA, LNCS*, pages 11–23, 2007.

[2] J. Amengual and E. Vidal. Efficient error-correcting viterbi parsing. *IEEE Trans. on PAMI*, 20(10):1109–1116, 1998.

[3] H. L. Berghel. A logical framework for the correction of spelling errors in electronic documents. *Information Processing and Management*, 23(5):477–494, 1987.

[4] F. Farooq, D. Jose, and V. Govindaraju. Phrase-based correction model for improving handwriting recognition accuracies. *Patt. Recog.*, 42(12):3271–3277, 2009.

[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, 1990.

[6] P. Hall and G. Dowling. Approximate string matching. *ACM Surveys*, 12(4):381–402, 1980.

[7] R. Llobet, J. Navarro, and et al. Efficient ocr post-processing combining language, hypothesis and error models. In *Accepted for publication in SSPR*, 2010.

[8] M. Mohri, F. Pereira, and M. Riley. The design principles of a weighted finite-state transducer library. *Theoretical Computer Scienc*, 231(1):17–32, 2000.

[9] D. Neuhoff. The viterbi algorithm as an aid in text recognition. *IEEE Trns. Inf. Theory*, 21:222–226, 1975.

[10] J. Perez-Cortes, J. Amengual, J. Arlandis, and R. Llobet. Stochastic error correcting parsing for ocr post-processing. In *ICPR*, volume 4, pages 405–408, 2000.

[11] M. Riley, F. Pereira, and M. Mohri. Transducer composition for context-dependent network expansion. *in Proc. Eurospeech 97*, 1997.

[12] E. Vidal, F. Thollard, and et al. Probabilistic finite-state machines - parts i and ii. *IEEE Trans. on PAMI*, 27(7):1013–1039, 2005.