AUTOMATIC LEARNING OF WORD TRANSDUCERS FROM EXAMPLES

Michel Gilloux
Centre National d'Études des Télécommunications
LAA/SLC/AIA
Route de Trégastel, BP 40
F-22301 Lannion Cedex, FRANCE
e-mail: gilloux@lannion.cnet.fr

ABSTRACT

This paper describes the application of markovian learning methods to the inference of word transducers. We show how the proposed method dispenses from the difficult design of hand-crafted rules, allows the use of weighed non deterministic transducers and is able to translate words by taking into account their whole rather than by making decisions locally. These arguments are illustrated on two examples: morphological analysis and grapheme-to-phoneme transcription.

INTRODUCTION

Several tasks associated with electronic lexicons may be viewed as transductions between character strings. This may be the decomposition of words into morphemes in morphology or the grapheme-to-phoneme transcription in phonology. In the first case, one has for example to decompose the French word "*chronométrage*" into the sequence of affixes "*chrono+mètre+er+age*". In the second, "*abstenir*" should be translated into "abstenir" or "apstenir"[1].

Most of the proposed methods in the domain (Catach 1984; Danlos et al. 1986; Koskenniemi 1983; Laporte 1988; Ritchie et al. 1987; Tufis 1989; Véronis 1988) are based on the availability of local rules whose combination, either through direct interpretation or by being compiled, form the target transducer.

Although these methods make it possible – at least in theory – to design suitable transducers, provided that the rule description language has the right expressive power, they are complex to use because of the difficulty of writing down rules. Moreover, for a given rule language, there may not exist an algorithm for compiling rules into a form better suited to the translation process. Lastly, in numerous cases, the translation procedures are improperly deterministic as shown by the example of "*abstenir*" so that it is not possible to consider several competing hypotheses in parallel not to speak of ranking them according to some certainty factor.

We have designed a program which allows to construct transducers without retaining the above shortcomings. It is no longer necessary to write down translation rules since the transducer is obtained as the result of an automatic learning over a set of examples. The transducer is represented into the language of probabilistic finite state automata (Markov models) so that its use is straightforward. Lastly, it produces results which are assigned a probability and makes it possible to list them by decreasing order of likelihood.

After stating the problem of character strings translation and defining the few

[1]These two tasks are in fact closely related in that (1) the correct phoneme transcription may mirror an underlying morphological structure, like for "*asocial*" whose phonemic form is "`asosjal`" rather than "`azosjal`" due to the decomposition "*a+social*", and (2) the surface form of a derived word may depend on the pronunciation of its component morphemes, like for "*dé+hamacher*" which results in "*déhamacher*" and not "*déshamacher*".

central notions of markovian learning, this paper describes their adaptation to the word translation problem in the learning and translating phases. This adaptation is illustrated through two applications: morphological analysis and grapheme-to-phoneme transcription.

THE TRANSDUCTION PROBLEM

In the context of character strings transduction, we look for an application f: $C^* \to C'^*$ which transforms certain words built over the alphabet C into words over the alphabet C'. For example, in the case of grapheme-to-phoneme transcription, C is the set of graphemes and C' that of phonemes.

It may be appropriate, for example in morphology, to use an auxiliary lexicon (Ritchie et al. 1987; Ritchie 1989) which allows to discard certain translation results. For example, the decomposition "*sage*" $\to$ "*ser+age*" would not be allowed because "*ser*" is not a verb in the French lexicon, although this is a correct result with respect to the splitting of word forms into affixes. The method we propose in this paper is only concerned with describing this last type of regularities leaving aside all non regular phenomena better described on a case-by-case basis such as through a lexicon.

MARKOV MODELS

A *Markov model* is a probabilistic finite state automaton $M = (S, T, A, s_I, s_F, g)$ where S is a finite set of states, A is a finite alphabet, $s_I \in S$ and $s_F \in S$ are two distinguished states called respectively the *initial state* and the *final state*, T is a finite set of transitions, and g is a function $g: t \in T \to \langle O(t), D(t), S(t), p(t)\rangle \in S \times S \times A \times [0, 1]$ such that

$$\forall (s \in S), \sum_{\{t \mid O(t) = s\}} p(t) = 1$$

where p(t) is the probability of reaching state D(t) while generating symbol S(t) starting from state O(t).

In general, the transition probabilities p(t) are mutually independent. Yet, in some contexts, it may be useful to have their values depend on others transitions. In this respect, it is possible to define a one-to-one correspondence $\tau_{ss'}$ $\{t \mid O(t) = s'\} \to \{t \mid O(t) = s\}$ such that p(t) is equal to $p(\tau_{ss'}(t))$. States s and s' are then said to be *tied*.

For every word $w = a_1 \ldots a_n \in A^*$, the set of *partial paths compatible with w till i*, $Path_i(w)$, is the set of sequences of i transitions $t_1 \ldots t_i$ such that $O(t_1) = s_I$, $D(t_j) = O(t_{j+1})$, for j = 1, ..., i–1 and $S(t_j) = a_j$, for j = 1, ..., i.

The set of *complete paths compatible with w*, Path(w), is in turn the set of elements in $Path_{|w|}(w)$, where $|w| = n$, the length of word w, such that $D(t_n) = s_F$.

The probability for the model M of emitting the word w is

$$Prob_M(m) = \sum_{path \in Path(w)} \prod_{t \in path} p(t)$$

A Markov model for which there exist at most one complete path for a given word is said to be *unifilar*. In this case, the above probability is reduced to

$$Prob_M(w) = \prod_{t \in path} p(t), \quad \text{if } Path(w) = path$$
$$Prob_M(w) = 0, \quad \text{if } Path(w) = \emptyset$$

Thus the probability $P_M(w)$ may be generally computed by adding the probabilities observed along every path compatible with w. In practice, this renders computationally expensive the algorithm for computing $P_M(w)$ and it is tempting to assume that the model is unifilar. Practical studies have shown that this sub-optimal method is applicable without great loss (Bahl et al. 1983).

Under this hypothesis, the probability $P_M(w)$ may be computed through the Viterbi dynamic programming algorithm. Indeed, the probability $P_M(w, i, s)$, maximal probability of reaching state s with the i first transitions in a path compatible with w is

$$\begin{cases} P_M(w, i, s) = \max \prod_{j=1}^{i} p(t_j), \quad i = 1 \ldots n \\ \text{where } (path \in \{t_1 \ldots t_i \in Path_i(w) \mid D(t_i) = s\}) \\ P_M(w, 0, s_I) = 1 \\ P_M(w, 0, s) = 0, \text{ if } (s \neq s_I) \end{cases}$$

therefore

$$P_M(w, i+1, s) = \max_{path} (p(t_{i+1}) \cdot P_M(w, i, D(t_i)))$$
$$\text{where } path \in \{t_1 \ldots t_{i+1} \in Path_{i+1}(w) \mid D(t_{i+1}) = s\}$$

whereby

$$P_M(w, i+1, s) = \max_t (p(t) \cdot P_M(w, i, O(t)))$$
$$\text{where } (t \in \{t \mid D(t) = s \text{ and } S(t) = a_{i+1}\})$$

with

$$Prob_M(w) = P_M(w, |w|, s_F) = \prod_{t \in MaxPath(w)} p(t)$$

It is therefore possible to compute $P_M(w, i, s)$ recursively for $i = 1, \ldots, n$ until $Prob_M(w)$.

Automatic learning of Markov models

Given a training set TS made of words in A* and a number N > 2 of states, that is the set S, learning a Markov model consists in finding a set T of transitions such that the joint probability P of the examples in the training set

$$P(TS) = \prod_{w \in TS} P_M(w)$$

is maximal.

In general, the set T is composed a priori of all possible transitions between states in S producing a symbol in A. The determination of probabilities p associated with these transitions is equivalent to the restriction of T to elements with non null probability which induces the structure of the associated automaton. In this case, the model is said to be *hidden* because it is hard to attach a meaning to the states in S. On the contrary, it is possible to force those states to have a clear-cut interpretation by defining them, for example, as n-grams which are sequences of n elements in A which encode the last n symbols produced by the model to reach the state. It is clear that then only some transitions are meaningful. In dealing with problems like those studied in the present paper it is preferable to use hidden models which allow states to stand for arbitrarily complex predicates.

The learning algorithm (Bahl et al. 1983) is based upon the following remark: given a model M whose transitions probabilities are known a priori, the a posteriori probability of a transition t may be estimated by the relative frequency with which t is used on a training set.

The number of times a transition t is used on TS is

$$freq(t) = \sum_{w \in TS} \sum_{t' \in MaxPath(w)} \delta(t, t')$$
$$\text{where } \delta(t, t') = 1 \text{ if } t = t', 0 \text{ otherwise}$$

The relative frequency of using t on TS is

$$\text{rel-freq}(t) = \frac{freq(t)}{\left(\sum_{\{t' \mid (O(t') = O(t))\}} freq(t') \right)}$$

The learning algorithm consists then in setting randomly the probability distribution p(t) and adjusting iteratively its values through the above formula until the adjustment is small enough to consider the distribution as stationary. It has been shown (Bahl et al. 1983) that this algorithm does converge towards a stationary value of the p(t) which maximizes locally[1] the probability P of the training set depending on the initial random probability distribution.

[1]In order to find a global optimum, we used a kind of simulated annealing technique (Kirkpatrick et al. 1983) during the learning process.

The stationary distribution defines the Markov model induced from the examples in TS[1].

TRANSDUCTION MODEL

To be applied in both illustrative examples, the general structure of Markov models should be related, by means of a shift in representation, to the problem of strings translation. The model of two-level morphological analysis (Koskenniemi 1983) suggests the nature of this shift. Indeed, this method, which was successfully applied to morphologically rich natural languages (Koskenniemi 1983), is based upon a two-level rule formalism for which there exist a way to compile them into the language of finite state automata (FSA) (Ritchie 1989). This result validates the idea that FSAs are reasonable candidates for representing transduction rules, at least in the case of morphology[2].

The shift in representation is designed so as to define the alphabet A as the set of pairs c:– or –:c' where $c \in C$ and $c \in C'$, – standing for the null character, $- \notin C$, $- \notin C'$. The mapping between the transducer f and the associated Markov model M is now straightforward:

$w' \in f(w)$ iff
$\exists\, x = x_1 \ldots x_n \in (C \cup \{-\})^*$,
$y = y_1 \ldots y_n \in (C' \cup \{-\})^*$
such that
$x_i{:}y_i$ is of the form –:c' or c:–,
for $i = 1, \ldots, n$,
$\mathrm{Prob}_M(x_1{:}y_1 \ldots x_n{:}y_n) \neq 0$,
$w = \mathrm{delete}(x)$ and $w' = \mathrm{delete}(y)$

where the function delete is defined as

delete(λ) = λ, (λ is the empty string),
delete(–Z) = delete(Z) and
delete(zZ) = z.delete(Z) if $z \in C$ or $z \in C'$

Given a training set TS = {<w, w'> | $w \in C^*$, $w' \in C'^*$}, the problem is thus to find the model M that maximizes the probability

$$P = \prod_{(w,w') \in TS} \max_{(x,y)} \mathrm{Prob}_M(x_1{:}y_1 \ldots x_n{:}y_n)$$
$$\text{where delete}(x) = w \text{ and delete}(y) = w'$$

This formula makes it clear what is the new difficulty with this type of learning, namely the indetermination of words x and y, that is of the alignment induced by them between w and its translation w'. The notions of partial and complete compatible paths should thus be redefined in order to take this into account.

The partial paths compatible with w and w' till i and j are now the set of sequences $t_1 \ldots t_{i+j}$, $\mathrm{Path}_{i,j}(w, w')$ such that $O(t_1) = s_I$, $D(t_k) = O(t_{k+1})$, for $k = 1, \ldots, i+j-1$, $S(t_k) = x_k{:}y_k$, for $k = 1, \ldots, i+j$, delete($x_1 \ldots x_{i+j}$) = $w_1 \ldots w_i$ and delete($y_1 \ldots y_{i+j}$) = $w'_1 \ldots w'_j$. A partial path is also complete as soon as $i = |w|$, $j = |w'|$ and $D(t_{|w|+|w'|}) = s_F$.

As before, we can define the probability $P_M(w, i, w', j, s)$ of reaching state s along a partial path compatible with w and w' and generating the first i symbols in w and j first symbols in w'.

$$P_M(w, i, w', j, s) = \max_{t_1 \ldots t_{i+j}} \prod_{k < i+j} p(t_k)$$
$$\text{where } (t_1 \ldots t_{i+j} \in \{\mathrm{Path}_{i,j}(w, w') \mid D(t_{i+j}) = s\})$$
$$P_M(w, 0, w', 0, s_I) = 1$$
$$P_M(w, 0, w', 0, s) = 0, \quad \text{if } s \neq s_I$$

---

[1]In practice, the number N = Card(S) of states for the model to be learned on a training set is not known. When N is small, the model has a tendency to generate much more character strings that were in TS due to an overgeneralization. At the other end of the spectrum, when N is large, the learned model will describe the examples in TS and them only. So, it is among the intermediate values of N that an optimum has to be looked for.

[2]Ritchie (1989) has even shown that the generative power of two-level morphological analyzers is strictly bound by that of finite state automata. He proved that all languages L generated by these analyzers are such that whenever E_2, E_3 and $E_1E_2E_3E_4$ belong to L, then E_2E_3 also belongs to L. Although this point was not considered in the present study, we may suppose that constraining the learned automaton to respect this last property, for example by means of tying states, would improve the overall results by augmenting in a sound way the generalization from examples.

Here again, this probability is such that $\mathrm{Prob}_M(w, w') = P_M(w, |w|, w', |w'|, s_F)$ and may be computed through dynamic programming according to the formula

$$P_M(w, i+1, w', j+1, s) =$$

$$\max \left(\begin{array}{l} \max_{t_1} P_M(w, i, w', j+1, O(t_1)) \\ \max_{t_2} P_M(w, i+1, w', j, O(t_2)) \end{array} \right.$$

$$\text{where } (t_1 \in \{t \in T \mid D(t) = s \text{ et } S(t) = w_{i+1}{:}-\})$$

$$\text{and } (t_2 \in \{t \in T \mid D(t) = s \text{ et } S(t) = -{:}w_{j+1}\})$$

It is now possible to compute for every training example the optimal path corresponding to a given probability distribution p(t). This path not only defines the crossed states but also the alignment between w and w'. The learning algorithm applicable to general markovian models remains valid for adjusting iteratively the probabilities p(t).

EXPERIMENTS

Morphological analysis

As a preliminary experiment, the morphological analysis automaton was learned on a set of 738 French words ending with the morpheme "*isme*" and associated with their decomposition into two morphemes, the first being a noun or an adjective. For example, we had the pair <"*athlétisme*","*athlète+isme*">. With a 400 states only automaton, the correct decomposition was found amongst the 10 most probable outputs for 97.6% of the training examples[1].

Grapheme-to-phoneme transcription

The case of grapheme-to-phoneme transcription is a straightforward application of the transduction model. String w is the graphemic form, e.g. "*abstenir*" and w' is its transcription into phonemes, e.g. "apsteniʀ" or "absteniʀ". Here the training set may feature such pairs as <w, w'> and <w, w"> where w' ≠ w".

The automaton was learned on a set of 1170 acronyms associated to their phonemic form which was described in a coarse phonemic alphabet where, for example, open or closed /o/ are not distinguished. Acronyms raise an interesting problem in that some should be spelled letter by letter ("*ACL*") whereas others may be pronounced ("*COLING*"). This experiment was thus intended to show that the model may take into account its input as a whole. With a 400 states only automaton, more than 50% of the training examples were correctly transcribed when only the most probable output was considered. This figure may be improved by augmenting the number of states in which case the learning phase becomes much longer.

CONCLUSION

We have proposed a method for learning transducers for the tasks of morphological analysis and grapheme-to-phoneme transcription. This method may be favorably compared to others solutions based upon writing rules in the sense that it does not oblige to identify rules, it provides a result which is directly usable as a transducer and it allows to list translations according to a decreasing order of probability. Yet, the learned automaton does not lend itself to an interpretation in the form of symbolic rules – provided that such rules exist –. Moreover, some learning parameters are set only as the results of empirical or random choices: number of states, initial probability distribution, etc. Yet, other advantages weigh for the proposed method. The automaton may take into account the whole word to be translated rather than a limited part of it – this justifies that a set of equivalent symbolic rules is hard to obtain –. For example, the grapheme-to-phoneme transcription may recognize the original language of a word while translating it (Oshika et al. 1988): the "French" nouns "*meeting*" and "*carpaccio*" have kept respectively their original English and Italian form

[1]We are aware that a more precise assessment of the method would use a test set different from the training set. We plan to perform such a test in the near future.

and pronunciation, etc. The learned automaton is symmetrical, thus it is also reversible. In other words, the morphological analysis automaton may also be used as a generator and the grapheme-to-phoneme automaton may become a phoneme-to-grapheme transducer. Another remark is in order: since the automaton is reversible, it may be composed with its inverse to form, for example, a grapheme-to-grapheme translator that keeps the phonemic form constant without actually computing it. Now, it has been shown elsewhere (Reape and Thompson 1988) that the transducer that would result is also describable in the formalism of finite state automata and that its number of states has a upper bound which is the square of the number of states in the base automaton. (Reape and Thompson 1988) also describes an algorithm for computing the resulting automaton. Lastly, other functions than morphological analysis or grapheme-to-phoneme transcription may be envisioned like, for example, the decomposition of words into syllables or the computation of abbreviations by contraction.

REFERENCES

Bahl, L. R.; Jelinek, F.; and Mercer, R. L. 1983. "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(2):179-190.

Catach, N. 1984."La phonétisation automatique du français," CNRS.

Danlos, L.; Laporte, E.; and Emerard, F. 1986 "Synthesis of Spoken Messages from Semantic Representations (Semantic-Representation-to-Speech-System)," *Proc. of the 11th Intern. Conf. on Computational Linguistics*, 599-604.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. "Optimization by Simulated Annealing," *Science*, 220:671-680.

Koskenniemi, K. 1983. "Two-Level Model for Morphological Analysis", *Proc. of the Eighth Intern. Joint Conf. on Artificial Intelligence*, 683-685.

Laporte, E. 1988. *Méthodes algorithmiques et lexicales de phonétisation de textes*, Thèse de doctorat, Université Paris 7.

Oshika, B. T.; Evans, B.; Machi, F.; and Tom, J. 1988. "Computational Techniques for Improved Name Search," *Proc. of the Second Conf. on Applied Natural Language Processing*, 203-210.

Reape, W. and Thompson, H. 1988. "Parallel Intersection and Serial Composition of Finite State Transducers", *Proc. of COLING'88*, 535-539.

Ritchie G. D.; Pulman, S. G.; Black, A. W.; and Russell, G. J. 1987. "A Computational Framework for Lexical Description", *Computational Linguistics*,13(3-4):290-307.

Ritchie, G. 1989. "On the Generative Power of Two-Level Morphological Rules," *Fourth Conf. of the European Chapter of the ACL*, 51-57.

Tufis D. 1989. "It Would Be Much Easier If WENT Were GOED," *Proc. of the Fourth Conf. of the European Chapter of the ACL*, 145-152.

Véronis, J. 1988. "Correction of Phonographic Errors in Natural Language Interfaces", *11th ACM-SIGIR Conf.*, 101-115.