
Probabilistic DFA Inference using Kullback-Leibler Divergence and Minimality

Franck Thollard
Pierre Dupont
Colin de la Higuera

THOLLARD@UNIV-ST-ETIENNE.FR
PDUPONT@UNIV-ST-ETIENNE.FR
CDLH@UNIV-ST-ETIENNE.FR

EURISE, Jean Monnet University, 23 rue Dr P Michelon, 42023 Saint-Etienne, France

Abstract

Probabilistic DFA inference is the problem of inducing a stochastic regular grammar from a positive sample of an unknown language. The ALERGIA algorithm is one of the most successful approaches to this problem. In the present work we review this algorithm and explain why its generalization criterion, a state merging operation, is purely local. This characteristic leads to the conclusion that there is no explicit way to bound the divergence between the distribution defined by the solution and the training set distribution (that is, to control globally the generalization from the training sample). In this paper we present an alternative approach, the MDI algorithm, in which the solution is a probabilistic automaton that trades off minimal divergence from the training sample and minimal size. An efficient computation of the Kullback-Leibler divergence between two probabilistic DFAs is described, from which the new learning criterion is derived. Empirical results in the domain of language model construction for a travel information task show that the MDI algorithm significantly outperforms ALERGIA.

1. Introduction

The problem of inferring a formal grammar from a finite set of positive and negative examples of an unknown language has been extensively studied (Sakakibara, 1997; Honavar & Slutzki, 1998). In the identification in the limit framework, Gold (1967) shows that negative examples are required even for learning the class of regular languages, or equivalently for inducing deterministic finite automata (DFAs). When training data includes negative examples, the regular

positive and negative inference (RPNI) algorithm can be used (Oncina & García, 1992). This algorithm was proven to identify in the limit the class of regular languages. Moreover the class of simple DFAs is polynomially learnable under the simple PAC learning model using the same algorithm (Parekh & Honavar, 1999).

Negative information, however, is not always available in practical domains, as in the case of natural language or speech applications. A promising approach to learning DFAs from simple positive examples has been recently proposed (Denis, in press), but real data is also generally noisy, either because of transcription errors or, for instance, because the data itself does not consistently follow a formal syntax. One response is to learn probabilistic deterministic finite automata (PDFAs). One possible approach to learning PDFAs consists of reducing the class of machines of interest to discrete Markov models, also known as N-grams in this context. Several extensions to variable order Markov models have also been proposed (Ron et al., 1994; Saul & Pereira, 1997). All these models, however, form a proper subclass of PDFAs in which the maximal order of dependence between several symbols in a sequence is bounded. The general goal of the work presented here is to be able to avoid this restriction and to learn the entire class of PDFAs.

Several inference algorithms for probabilistic automata have been proposed (Rulot & Vidal, 1988; Carrasco & Oncina, 1994; Ron et al., 1995) but only Carrasco and Oncina's ALERGIA algorithm, a stochastic extension of the RPNI algorithm, is free from the restriction to the learning of acyclic automata (Carrasco & Oncina, 1994; Carrasco & Oncina, 1999). This algorithm has been applied to information extraction from text (Freitag, 1997) or structured documents (Young-Lai & Tompa, in press) and speech language modeling (Dupont & Chase, 1998). A detailed presentation of the ALERGIA algorithm is given in section 3. The generalization operation of this algorithm is local

in the sense that pair of states will be merged if the probabilistic languages associated to their suffixes are close enough. This characteristic leads to the conclusion that, in the context of the ALERGIA algorithm, there is no explicit way to bound the divergence between the distribution defined by the solution and the training set distribution (that is, to control globally the generalization from the training sample).

We propose in section 4 an alternative approach, the MDI algorithm, for which the solution is a probabilistic automaton that trades off minimal divergence from the training sample and minimal size. An efficient computation of the Kullback-Leibler divergence between two probabilistic DFAs is described in section 4.1. The new learning criterion is then detailed in section 4.2

Empirical results in the domain of language model construction for a travel information task are presented in section 5. They show that the MDI algorithm significantly outperforms ALERGIA on this task.

2. Preliminaries

A *probabilistic DFA* (PDFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, \gamma)$ in which Q is a finite set of *states*, Σ is a finite *alphabet*, δ is a *transition function*, i.e. a mapping from $Q \times \Sigma$ to Q , q_0 is the *initial state*, γ is the *next symbol probability function*, i.e. a mapping from $Q \times \Sigma \cup \{\#\}$ to $[0, 1]$. A special symbol $\#$, not belonging to the alphabet Σ , denotes the *end of string symbol*. Hence $\gamma(q, \#)$ represents the probability of ending the generation process in state q . The probability function must satisfy the following constraints:

$$\begin{aligned} \gamma(q, a) &= 0 & , \text{ if } \delta(q, a) = \emptyset, \forall a \in \Sigma \\ \sum_{a \in \Sigma \cup \{\#\}} \gamma(q, a) &= 1 & , \forall q \in Q \end{aligned}$$

The probability $P_A(x)$ of *generating* a string $x = x_1 \dots x_n$ from a PDFA $A = (Q, \Sigma, \delta, q_0, \gamma)$ is defined as

$$P_A(x) = \begin{cases} (\prod_{i=1}^n \gamma(q^i, x_i)) \gamma(q^n, \#) & \text{if } \delta(q^i, x_i) \neq \emptyset \text{ with } q^{i+1} = \delta(q^i, x_i), \\ & \text{for } 1 \leq i < n \text{ and } q^1 = q_0 \\ 0, & \text{otherwise} \end{cases}$$

Our definition of probabilistic automaton is equivalent to a stochastic deterministic regular grammar used as a string generator. Thus, $\sum_{x \in \Sigma^*} P_A(x) = 1$. Note that some work on the learning of discrete distributions uses distributions defined on Σ^n (that is $\sum_{x \in \Sigma^n} P(x) = 1$, for any $n \geq 1$), instead of Σ^* .

Let I_+ denote a *positive sample*, i.e. a set of strings belonging to the probabilistic language we are trying

to model. Let $PTA(I_+)$ denote the *prefix tree acceptor* built from a positive sample I_+ . The prefix tree acceptor is an automaton that only accepts the strings in the sample and in which common prefixes are merged together resulting in a tree shaped automaton. Let $PPTA(I_+)$ denote the *probabilistic prefix tree acceptor*. It is the probabilistic extension of the $PTA(I_+)$ in which each transition has a probability proportional to the number of times it is used while generating, or equivalently parsing, the positive sample. Let $C(q)$ denote the count of state q , that is, the number of times the state q was used while generating I_+ from $PPTA(I_+)$. Let $C(q, \#)$ denote the number of times a string of I_+ ended on q . Let $C(q, a)$ denote the count of the transition (q, a) in $PPTA(I_+)$. The $PPTA(I_+)$ is the maximal likelihood estimate built from I_+ . In particular, for $PPTA(I_+)$ the probability estimates are $\hat{\gamma}(q, a) = \frac{C(q, a)}{C(q)}$ and $\hat{\gamma}(q, \#) = \frac{C(q, \#)}{C(q)}$.

2.1 Quotient Automata and Inference Search Space

The probabilistic automaton A/π denotes the automaton *derived from* the probabilistic automaton A *with respect to the partition* π of Q , also called the *quotient automaton* A/π . It is obtained by merging states of A belonging to the same subset B in π . When q results from the merging of the states q' and q'' , the following equality must hold

$$\gamma(q, a) = \frac{C(q', a) + C(q'', a)}{C(q') + C(q'')} \quad , \forall a \in \Sigma \cup \{\#\}$$

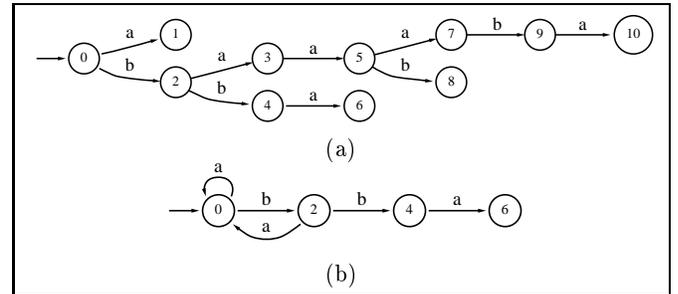


Figure 1. A prefix tree acceptor (a) and a quotient automaton (b).

Let $Lat(PPTA(I_+))$ denote the lattice of automata which can be derived from $PPTA(I_+)$. This lattice is the set of all possible quotient automata of $PPTA(I_+)$, that is, the set of all probabilistic automata that can be derived from $PPTA(I_+)$ by merging some states. It defines the search space of all possible PDFAs that can generalize the training sample (Dupont et al., 1994). Figure 1 shows a prefix tree acceptor built from the sample $I_+ = \{a, bb, bba, baab, baaaba\}$ (probabilities are omitted here for clarity) and a quotient automaton $PPTA(I_+)/\pi$ corresponding to the partition

$\pi = \{\{\underline{0}, 1, 3, 5, 7, 10\}, \{\underline{2}, 8, 9\}, \{\underline{4}\}, \{\underline{6}\}\}$. Each subset of the partition represents a set of merged states and is denoted by its state of minimal rank.

2.2 Kullback-Leibler Divergence

Let $A = (Q, \Sigma, \delta, q_0, \gamma)$ and $A' = (Q', \Sigma, \delta', q'_0, \gamma')$ be two PDFAs. Then $D(P_A \parallel P_{A'})$ denotes the *Kullback-Leibler divergence* or *relative entropy* (Cover & Thomas, 1991).

$$D(P_A \parallel P_{A'}) \stackrel{\text{notation}}{=} D(A \parallel A') = \sum_{x \in \Sigma^*} P_A(x) \log \frac{P_A(x)}{P_{A'}(x)}$$

The divergence can be interpreted as the additional number of bits needed to encode data generated from P_A when the optimal code is chosen according to $P_{A'}$ instead of P_A .

The divergence between two PDFAs can be decomposed over all pairs of transitions as follows (Carrasco, 1997) :

$$D(A \parallel A') = \sum_{q_i \in Q} \sum_{q'_j \in Q'} \sum_{a \in \Sigma \cup \{\#\}} c_{ij} \gamma(q_i, a) \log \frac{\gamma(q_i, a)}{\gamma'(q'_j, a)} \quad (1)$$

In equation (1), c_{ij} denotes the probability mass in A of the set L_{ij} of prefixes common to state q_i in A and q_j in A' . In general, the divergence becomes infinite as soon as there is one symbol a such that $\gamma(q_i, a) \neq 0$ and $\gamma'(q_j, a) = 0$. As detailed in section 4.1, this will never happen when the divergence between one PDFa and one of its quotient automata is considered.

3. PDFa Inference and the ALERGIA Algorithm

Algorithm 1 depicts in pseudocode the ALERGIA algorithm (Carrasco & Oncina, 1994). It performs an ordered search in the lattice $Lat(PPTA(I_+))$ which, as described above, is the set of quotient automata of $PPTA(I_+)$. This ordered search can be described as follows.

By construction, each of the states of $PPTA(I_+)$ corresponds to a unique prefix. The prefixes may be sorted according to the standard order $<$ on strings¹. This order also applies to the prefix tree states. A partition in the state set of the $PPTA(I_+)$ consists of an ordered set of subsets, each subset receiving the rank of its state of minimal rank. The ALERGIA algorithm proceeds in $N - 1$ steps, where $N = \mathcal{O}(\|I_+\|)$ is

¹According to the standard order, the first strings on the alphabet $\Sigma = \{a, b\}$ are $\lambda < a < b < aa < ab < ba < bb < aaa < \dots$

the number of states of $PPTA(I_+)$. The partition $\pi(i)$ at step i , that is, the quotient automaton obtained at step i , is obtained by merging the two first subsets, according to the standard order, of the partition $\pi(i - 1)$ at step $i - 1$ such that $PPTA(I_+)/\pi(i)$ is a compatible automaton. An automaton is compatible if it is obtained from another compatible automaton by a compatible merging. A compatible merging of two states means that the probability of any of their suffixes are similar within a certain threshold denoted α_A . The formal definition of this compatibility measure is given below.

Algorithm 1 ALERGIA.

```

input
 $I_+$  // A positive sample
 $\alpha_A$  // A precision parameter
output
a PDFa // A probabilistic DFA

begin
//  $N$  is the number of states of  $PPTA(I_+)$ 
// One subset for each prefix in the order  $<$ 
 $\pi \leftarrow \{\{0\}, \{1\}, \dots, \{N - 1\}\}$   $A \leftarrow PPTA(I_+)$ 

// Loop on the subsets of partition  $\pi$ 
for  $i = 1$  to  $|\pi| - 1$ 
// Loop on the subsets of lower rank
for  $j = 0$  to  $i - 1$ 
// Merging of subset  $B_i$  and subset  $B_j$ 
if compatible ( $i, j, \alpha_A$ ) then
 $\pi' \leftarrow \pi \setminus \{B_j, B_i\} \cup \{B_i \cup B_j\}$ 
 $A/\pi' \leftarrow \text{derive}(A, \pi')$ 
 $\pi'' \leftarrow \text{determ\_merge}(A/\pi')$ 
 $A \leftarrow A/\pi''$ 
 $\pi \leftarrow \pi'$ 
break // Break  $j$  loop
end if
end for // End  $j$  loop
end for // End  $i$  loop
return  $A$ 
end ALERGIA

```

The function *derive* (A, π') returns the quotient automaton A with respect to partition π' . The automaton A/π' may be non-deterministic. The function *determ_merge* (A/π') returns the partition π'' obtained by recursively merging all subsets of π' that create a non-determinism. If A/π' is deterministic, the partition π'' is equal to partition π' .

The function *compatible* (i, j, α_A) controls the merging of states. It returns TRUE if the states i and j are compatible within the precision defined by the parameter α_A and FALSE otherwise. The compatibility measure derives from the Hoeffding bound (Hoeffding, 1963). Formally, two states q_1 and q_2 are α_A -compatible ($0 < \alpha_A \leq 1$) if the two following con-

ditions hold

$$(2.1) \quad \left| \frac{C(q_1, a)}{C(q_1)} - \frac{C(q_2, a)}{C(q_2)} \right| < \sqrt{\frac{1}{2} \ln \frac{2}{\alpha_A}} \left(\frac{1}{\sqrt{C(q_1)}} + \frac{1}{\sqrt{C(q_2)}} \right),$$

$$\forall a \in \Sigma \cup \{\#\}$$

$$(2.2) \quad \delta(q_1, a) \text{ and } \delta(q_2, a) \text{ are } \alpha_A\text{-compatible, } \forall a \in \Sigma$$

Condition (2.1) defines the compatibility between each pair of transitions outgoing respectively from state q_1 and q_2 . Condition (2.2) requires the compatibility to be recursively satisfied for every pair of successors of these states².

Figure 2 depicts one step of the ALERGIA algorithm. We assume that the quotient automaton at step i is represented at the top of this figure³ and that states 5 and 2 satisfy the compatibility measure. States 5 and 2 are then merged resulting in a new quotient automaton which, in this case, is non-deterministic. Subsequent merging steps are then performed to eliminate non-determinism. This results in the automaton depicted at the bottom of the figure.

The two main features of the ALERGIA algorithm are (1) the order in which candidate states for merging are tried and (2) the compatibility measure between two states. The relevance of this merging order is explained in detail in Carrasco and Oncina (1999).

We argue here that the compatibility measure can be improved. Indeed this measure is only based on distances between suffixes (down to the prefix tree leaves because of the recursive condition (2.2)) of the two candidate states. In particular, it is independent of the probability mass associated to the prefixes of these states. In other words, pairs of states with an associated count that is too low, will always be considered compatible, resulting in inappropriate mergings. This was observed experimentally by Young-Lai and Tompa (in press) who introduced some refinements to this compatibility measure.

Another related problem comes from the fact that there is no explicit way to bound the divergence between the training sample distribution and the distribution associated to the PDFa obtained after merging.

²If one successor state is undefined, as the transition function δ is partial, its associated counts are set to zero. In this case, condition (2.1) can be computed, assuming $\frac{0}{0} \triangleq 0$.

³Probabilities are left out to make the figure more readable.

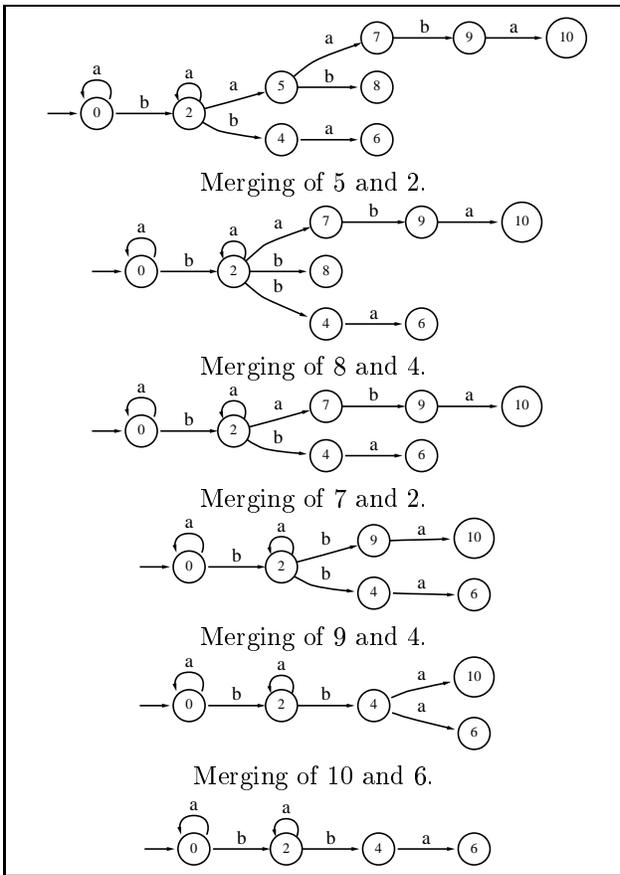


Figure 2. One step of the ALERGIA algorithm.

In other words, there is no way to globally control the generalization from the training sample. This observation leads to a new compatibility measure detailed in section 4.2 and the associated learning algorithm described in section 4.3.

4. MDI Algorithm

We present here the Minimal Divergence Inference (MDI) algorithm, which aims at inducing PDFAs while trading off minimal divergence from the training sample distribution and minimal size. This algorithm can be considered as a new formalization of Stolcke’s (1994) Bayesian learning method. Indeed a possible solution with null divergence is the prefix tree acceptor $PPTA(I_+)$ itself. It is also the maximal likelihood estimate built from the training sample. On the other hand, favoring small automata, or equivalently automata derived from $PPTA(I_+)$ with a large number of mergings, defines a *prior* probability proportional to the solution size. Trading off both effects is thus equivalent to maximizing the *posterior* probability of the model given the training data.

Apart from a clear interpretation in terms of KL di-

vergence, the main advance over Stolcke’s work is the reduced complexity of the MDI algorithm. As detailed in section 4.3, the total number of candidates for merging which are evaluated is $O(N^2)$, where N denotes the size of $PPTA(I_+)$. The greedy search proposed by Stolcke requires the same complexity *at each step* of the inference, resulting in an overall complexity of $O(N^3)$. All experiments reported in Stolcke (1994), including those with additional heuristics to speed up the search, only deal with very small problems (alphabets containing a few symbols, training sample containing typically less than 100 strings). Experiments reported in section 5.2 deal with significantly larger problems and never took more than a few hours of CPU time on a personal computer.

4.1 Computation of the KL Divergence

Let $A_0 = (Q^0, \Sigma, \delta^0, q_0^0, \gamma^0)$ be the probabilistic prefix tree acceptor $PPTA(I_+)$ built from a positive sample I_+ and $A_0/\pi_{01} = A_1 = (Q^1, \Sigma, \delta^1, q_0^1, \gamma^1)$, a quotient automaton of A_0 . By definition of a quotient automaton (see section 2.1), each state q_i in A_0 exactly corresponds to one state $q_{\underline{i}} \triangleq B_{\pi_{01}}(q_i)$ in A_1 . In other words, $B_{\pi_{01}}(q_i)$ denotes the subset of the partition π_{01} to which the state q_i belongs.

The set L_{ij} of common prefixes to state q_i in A_0 and any state of A_1 is necessarily empty for all states but $q_{\underline{i}}$. Thus the probability mass c_{ij} associated with L_{ij} is zero for all j but \underline{i} . Let $c_i \triangleq c_{i\underline{i}}$ denote this probability mass. Moreover, as A_0 is a tree, there is only one prefix for each state q_i and c_i simply denotes the probability of reaching q_i from the tree root.

These properties lead to a simplified version of equation (1) to compute the divergence between A_0 and A_1 :

$$\begin{aligned} D(A_0||A_1) &= \sum_{q_i \in Q_0} \sum_{a \in \Sigma \cup \{\#\}} c_i \gamma_0(q_i, a) \log \frac{\gamma_0(q_i, a)}{\gamma_1(q_{\underline{i}}, a)} \\ &= -H(A_0) - \sum_{q_i \in Q_0} \sum_{a \in \Sigma \cup \{\#\}} c_i \gamma_0(q_i, a) \log \gamma_1(q_{\underline{i}}, a) \end{aligned} \quad (3)$$

where $H(A_0)$ denotes the entropy of A_0 . The divergence $D(A_0||A_1)$ is always finite in this case as $\gamma_1(q_{\underline{i}}, a) \neq 0$ if $\gamma_0(q_i, a) \neq 0$. Let $A_2 = A_1/\pi_{12}$ be a quotient automaton of A_1 . By construction, A_2 is also a quotient automaton of A_0 for some partition π_{02} . Thus the divergence $D(A_0||A_2)$ can be incrementally computed as follows:

$$\begin{aligned} D(A_0||A_2) &= D(A_0||A_1) \\ &+ \sum_{q_i \in Q_{012}} \sum_{a \in \Sigma \cup \{\#\}} c_i \gamma_0(q_i, a) \log \frac{\gamma_1(q_{\underline{i}}, a)}{\gamma_2(q_{\underline{i}}, a)} \end{aligned} \quad (4)$$

where $Q_{012} = \{q_i \in Q_0 \mid B_{\pi_{01}}(q_i) \neq B_{\pi_{02}}(q_i)\}$ denotes

Algorithm 2 MDI.

```

input
 $I_+$  // A positive sample
 $\alpha_M$  // A precision parameter
output
a PDFA // A probabilistic DFA

begin
//  $N$  is the number of states of  $PPTA(I_+)$ 
// One subset for each prefix in the order <
 $\pi \leftarrow \{\{0\}, \{1\}, \dots, \{N-1\}\}$   $A \leftarrow PPTA(I_+)$ 

// Loop on the subsets of partition  $\pi$ 
for  $i = 1$  to  $|\pi| - 1$ 
// Loop on the subsets of lower rank
for  $j = 0$  to  $i - 1$ 
// Merging of subset  $B_i$  and subset  $B_j$ 
 $\pi' \leftarrow \pi \setminus \{B_j, B_i\} \cup \{B_i \cup B_j\}$ 
 $A/\pi' \leftarrow \text{derive}(A, \pi')$ 
 $\pi'' \leftarrow \text{determ\_merge}(A/\pi')$ 
// Compute the size difference
 $nmerge \leftarrow (|A| - |A/\pi''|)$ 
// Confirm merging if compatible
if  $\text{compatible}(\pi, \pi'', \alpha_M, nmerge)$  then
 $A \leftarrow A/\pi''$ 
 $\pi \leftarrow \pi''$ 
break // Break  $j$  loop
end if
end for // End  $j$  loop
end for // End  $i$  loop
return  $A$ 
end MDI

```

the set of states in A_0 which have been merged to derive A_2 from A_1 .

4.2 MDI Compatibility Criterion

Assume A_1 is a temporary solution of a PDFA inference algorithm and A_2 is a tentative new solution derived from A_1 . $\Delta(A_1, A_2) = D(A_0||A_2) - D(A_0||A_1)$ denotes the divergence increment while going from A_1 to A_2 . The new solution A_2 is considered to be compatible with the training data if the divergence increment relative to the size reduction, that is, the reduction of the number of states, is small enough. Formally, let α_M denote a compatibility threshold. The compatibility is satisfied if:

$$\frac{\Delta(A_1, A_2)}{|A_1| - |A_2|} < \alpha_M \quad (5)$$

4.3 Description of the MDI Algorithm

Algorithm 2 depicts in pseudocode the MDI algorithm. It follows the same search order as the ALERGIA algorithm. The value of $nmerge$ in the inner loop on j gives the reduction of automaton size while deriving A/π'' from A . The function $\text{compatible}(\pi, \pi'', \alpha_M, nmerge)$

implements the criterion (5). It returns TRUE if this compatibility measure falls below the threshold α_M and FALSE otherwise. All other functions are identical to those described in section 3. The overall complexity of this algorithm, evaluated as the number of state pairs which are considered for merging, is $O(N^2)$ where N denotes the size of $PPTA(I_+)$.

5. Experiments

Evaluation of non-probabilistic inference methods is usually based on correct classification rates of new positive and negative data (Lang et al., 1998). In the case of PDFA inference, alternative measures are possible, since the problem is often stated as estimating a probability distribution over the set of possible strings.

The quality of the PDFA $A = (Q, \Sigma, \delta, q_0, \gamma)$ is measured by the *per symbol log-likelihood* of strings x belonging to a test sample according to the distribution defined by the solution $P_A(x)$ computed on a test sample S :

$$LL = \left(-\frac{1}{\|S\|} \sum_{j=1}^{|S|} \sum_{i=1}^{|x_j|} \log P(x_i^j | q^i) \right)$$

where $P(x_i^j | q^i)$ denotes the probability of generating x_i^j , the i -th symbol of the j -th string in S , given that the generation process was in state q^i . This average log-likelihood is also related to the KL divergence between an unknown target distribution and the proposed solution by considering the test sample as the empirical estimate of the unknown distribution.

The *test sample perplexity* PP is most commonly used for evaluating language models of speech applications. It is given by $PP = 2^{LL}$. The minimal perplexity $PP = 1$ is reached⁴ when the next symbol x_i^j is always predicted with probability 1 from the current state q^i (i.e. $P(x_i^j | q^i) = 1$) while $PP = |\Sigma|$ corresponds to random guessing from an alphabet of size $|\Sigma|$.

In order to guarantee that the probability of predicting any symbol from any given state is always strictly positive, the PDFA must be smoothed. We use here linear interpolation with a unigram model⁵ $P_1(x_i) = C(x_i)/T$ where $C(x_i)$ denotes the count of symbol x_i in a training sample of T symbols:

$$P(x_i | q^i) = \beta \gamma(q^i, x_i) + (1 - \beta) P_1(x_i)$$

This smoothing technique is very rudimentary but, be-

⁴Such a perfectly informed model cannot be constructed in general.

⁵If some alphabet symbol never occurs in the training sample, the unigram model itself needs to be smoothed.

cause it is so simple, it best reflects the quality of the PDFA itself.

Section 5.1 explains how the learning data is split into three sets. PDFA inference with the ALERGIA and MDI algorithms is performed on the training sample. The compatibility threshold α_A and α_M , and optimal interpolation coefficients β 's are adjusted in order to minimize the perplexity on a separate validation set. Finally the performance of both algorithms is assessed on an independent test set.

5.1 The ATIS task

The Air Travel Information System (ATIS) corpus (Hirschman, 1992) was developed under a DARPA speech and natural language program that focussed on developing language interfaces to information retrieval systems. The corpus consists of speakers of American English making information requests such as,

“Uh, I'd like to go from, uh, Pittsburgh to Boston next Tuesday, no wait, Wednesday”.

We use the ATIS-2 sub-corpus in the experiments reported here. This portion of the corpus was developed under Wizard-of-Oz conditions in which a human being secretly replaced the speech recognition component of an otherwise fully automated dialogue system.

The ATIS-2 collection is officially defined as containing a training set and two evaluation sets. The training set contains 13,044 utterances (130,773 tokens). The vocabulary contains 1,294 words. We used the first evaluation set (Feb92, 974 utterances, 10636 tokens) together with the training set to randomly generate 10 independent data sets. The first 90 % were used for PDFA inference and the last 10 % were used as validation sets to tune our free parameters ($\alpha_A, \alpha_M, \beta$'s). The second evaluation set (Nov92, 1001 utterances, 11703 tokens) was used as our independent test set. In the context of these experiments, alphabet symbols represent *words* from the ATIS vocabulary and strings represent *utterances*.

5.2 Results

Figure 3 reports the average perplexity obtained with ALERGIA and MDI over 10 independent validation sets as a function of the number of strings in the training sets. In each case, the value of the free parameters was adjusted to minimize the perplexity. The optimal value α_A is approximately 0.25 in all cases while optimal α_M ranges from 2.10^{-3} for small samples ($|\Sigma|=1000$) down to 3.10^{-4} for large samples ($|\Sigma|=12000$). With α 's being set to their optimal values, less than 2% relative perplexity difference is ob-

served while changing the interpolation parameter β in the range $[0.5, 0.7]$. This illustrates the robustness of the smoothing mechanism with respect to this parameter.

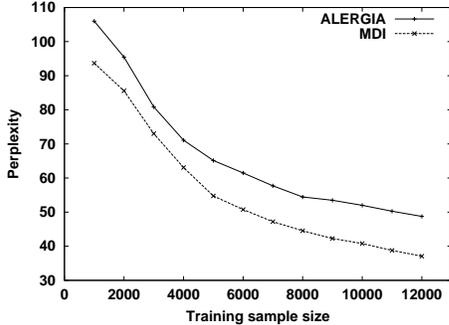


Figure 3. Learning curves: ALERGIA vs MDI.

Table 1 summarizes the results obtained on the independent test set for different models. The 1-gram model denotes the unigram which was interpolated with the PDFAs obtained with ALERGIA and MDI. The 3-gram (1) model denotes a trigram smoothed with the same technique. The 3-gram (2) model is smoothed with back-off to lower order estimates (Katz, 1987) with modified back-off distributions as proposed by Kneser and Ney (1995). The second column reports the test set perplexity. It shows that MDI significantly outperforms ALERGIA and is slightly better than a 3-gram using the same smoothing mechanism. The dramatic perplexity reduction obtained with back-off models indicates that more sophisticated smoothing techniques for PDFAs are required. Preliminary results show that interpolating about 10 different PDFAs obtained with various α_M gives a test set perplexity of 41.

Table 1. Independent test set results

Model	PP	# param	% parsed
1-gram	148	1163	100%
ALERGIA	71	13468	53%
MDI	56	8981	62%
3-gram (1)	59	30813	31%
3-gram (2)	14	45224	31%

The third column in table 1 reports the number of estimated parameters.⁶ The last column gives a generalization measure computed as the number of strings accepted without smoothing by the various models. It shows that MDI offers a better generalization than ALERGIA. Note that a unigram model gives a perfect generalization according to this criterion at the cost of a much higher perplexity.

⁶This figure corresponds to the number of transitions for PDFAs and the number of non-zero counts for N-gram models. When a given model combines several models using either interpolation or back-off, the total number of parameters is reported.

6. Discussion

In this paper we study the problem of PDFa inference from a positive sample of an unknown stochastic language. Several approaches to this problem have been proposed, among which the ALERGIA algorithm and its variants are most successful (Carrasco & Oncina, 1994; Young-Lai & Tompa, in press). We argue that its generalization criterion, a state merging operation, can be significantly improved as it does not globally control the level of generalization from the learning sample. We propose an alternative approach, the MDI algorithm, which relies on a new formalization of Stolcke’s (1994) Bayesian learning method. In particular, the MDI algorithm aims at inducing a PDFa that trades off minimal divergence from the training sample and minimal size. Empirical results in the domain of language model construction for a travel information task show that the MDI algorithm offers a 21% test set perplexity reduction as compared with ALERGIA.

Several heuristics have been proposed for improving the quality of DFA inference to overcome the lack of learning data (Lang et al., 1998; Juillé & Pollack, 1998). Even though PDFa inference is a distinct problem, as it not only requires inference of the DFA structure but also correct estimation of its probabilities, it would be worth investigating whether these heuristics would be useful in conjunction with the MDI algorithm.

We believe that improved smoothing techniques is a key issue in applying PDFa inference methods on real data. The smoothing mechanism detailed in section 5 is particularly simple. Clustering alphabet symbols before PDFa inference was shown to reduce perplexity on new data (Dupont & Chase, 1998). Back-off smoothing is a very powerful technique to improve the perplexity obtained with N-gram models. Adaptation of this smoothing technique to PDFAs that cannot be reduced to N-grams is one of our major research goals.

Acknowledgements

We wish to thank Lin Chase for her careful reading of this manuscript.

References

- Carrasco, R. (1997). Accurate computation of the relative entropy between stochastic regular grammars. *Theoretical Informatics and Applications*, 31, 437–444.
- Carrasco, R., & Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. *Grammatical Inference and Applications*,

- ICGI'94 (pp. 139–150). Alicante, Spain: Springer Verlag.
- Carrasco, R., & Oncina, J. (1999). Learning deterministic regular grammars from stochastic samples in polynomial time. *Theoretical Informatics and Applications*, 33, 1–19.
- Cover, T., & Thomas, J. (1991). *Elements of information theory*. New York: John Wiley and Sons.
- Denis, F. (in press). Learning regular languages from simple positive examples. In *Machine Learning*.
- Dupont, P., & Chase, L. (1998). Using symbol clustering to improve probabilistic automaton inference. *Grammatical Inference and Application, ICGI'98* (pp. 232–243). Ames, Iowa: Springer Verlag.
- Dupont, P., Miclet, L., & Vidal, E. (1994). What is the search space of the regular inference? *Grammatical Inference and Application, ICGI'94* (pp. 25–37). Alicante, Spain: Springer Verlag.
- Freitag, D. (1997). Using grammatical inference to improve precision in information extraction. *Proceedings of the ICML'97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*. Nashville, Tennessee.
- Gold, E. (1967). Language identification in the limit. *Information and Control*, 10, 447–474.
- Hirschman, L. (1992). Multi-site data collection for a spoken language corpus. *Proceedings of DARPA Speech and Natural Language Workshop* (pp. 7–14). Arden House, NY.
- Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58, 13–30.
- Honavar, V., & Slutzki, G. (Eds.). (1998). *Grammatical inference*. No. 1433 in Lecture Notes in Artificial Intelligence. Ames, Iowa: Springer-Verlag.
- Juillé, H., & Pollack, J. (1998). A stochastic search approach to grammar induction. *Grammatical Inference* (pp. 126–137). Ames, Iowa: Springer-Verlag.
- Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 35, 400–401.
- Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling. *International Conference on Acoustic, Speech and Signal Processing* (pp. 181–184). Detroit, Michigan.
- Lang, K., Pearlmutter, B., & Price, R. (1998). Results of the abbingo one DFA learning competition and a new evidence-driven state merging algorithm. *Grammatical Inference* (pp. 1–12). Ames, Iowa: Springer-Verlag.
- Oncina, J., & García, P. (1992). Inferring regular languages in polynomial update time. In N. Pérez de la Blanca, A. Sanfeliu and E. Vidal (Eds.), *Pattern recognition and image analysis*, vol. 1 of *Series in Machine Perception and Artificial Intelligence*, 49–61. Singapore: World Scientific.
- Parekh, R., & Honavar, V. (1999). Simple DFA are polynomially probably exactly learnable from simple examples. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 298–306). Bled, Slovenia: Morgan Kaufmann.
- Ron, D., Singer, Y., & Tishby, N. (1994). Learning probabilistic automata with variable memory length. *Proceedings of the Seventh Annual Conference on Computational Learning Theory*. New Brunswick, NJ: ACM Press.
- Ron, D., Singer, Y., & Tishby, N. (1995). On the learnability and usage of acyclic probabilistic automata. *Proceedings of the Eighth Annual Conference on Computational Learning Theory* (pp. 31–40). Santa Cruz, CA: ACM Press.
- Rulot, H., & Vidal, E. (1988). An efficient algorithm for the inference of circuit-free automata. *Advances in Structural and Syntactic Pattern Recognition* (pp. 173–184). Springer-Verlag.
- Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science*, 185, 15–45.
- Saul, L., & Pereira, F. (1997). Aggregate and mixed-order Markov models for statistical language processing. *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing* (pp. 81–89). Somerset, New Jersey: Association for Computational Linguistics.
- Stolcke, A. (1994). *Bayesian learning of probabilistic language models*. Doctoral dissertation, Division of Computer Science, University of California, Berkeley.
- Young-Lai, M., & Tompa, F. (in press). Stochastic grammatical inference of text database structure. In *Machine Learning*.