Learning typed automata from automatically labeled data

Christopher Kermorvant[1], Colin de la Higuera[2], Pierre Dupont[3]

[1] Dept. IRO, Université de Montréal, Canada
`kermorvc@iro.umontreal.ca`
[2] EURISE, Université Jean Monnet, Saint-Etienne, France
`cdlh@univ-st-etienne.fr`
[3] INGI, Université de Louvain, Louvain-la-Neuve, Belgique,
`pdupont@info.ucl.ac.be`

Abstract. In this paper, we propose a way of incorporating additional knowledge in probabilistic automata inference, by using typed automata. We compare two kinds of knowledge that are introduced into the learning algorithms. A statistical clustering algorithm and a part-of-speech tagger are used to label the data according to statistical or syntactic information automatically obtained from the data. The labeled data is then used to infer correctly typed automata. Compared to a previously proposed method which improved grammatical inference, our approach yields better automata. The inference of typed automata with statistically labeled data provides language models competitive with state-of-the-art n-grams on the Air Travel Information System (ATIS) task.

1 Introduction

Probabilistic finite state automata (PFA) have received increasing attention over the past few years. They are used in prediction tasks [GLT01], pattern recognition tasks [LVA$^+$94] and language modeling tasks [SO94], in tasks where data is sequential or structures, regularities in the data have to be described and where the dependencies can be long-term. As models they are richer than purely statistical models (Markov chains, n-grams [RBH93]) but poorer than complex models such as Bayesian Networks [Kol99]. They are equivalent to hidden Markov models (HMMs) [Jel98], but become more constrained if the PFA are deterministic. Determinism, on the other hand, allows for faster parsing (computing the probability of a string or a set of strings) and for the use of a variety of learning algorithms from the *grammatical inference* framework. Moreover, learning algorithms for HMMS and PFAS differ in spirit since the structure (number of states and edges) is usually fixed for HMMS and is learned for PFAS.

Grammatical inference consists in learning formal grammars for unknown languages when provided with examples of strings belonging (or not) to the language. Regular grammatical inference, in which the target grammar is supposed to be regular, has received most of the attention.

If one is provided with positive and negative examples, algorithm RPNI [OG92] can be used to infer deterministic finite automata. Neural approaches have also been proposed [LGF00]. Grammar induction techniques have been used in a variety of tasks and fields: information extraction [Fre97,SMR99], pattern recognition [LVA$^+$94], or time series [GLT01] are some examples.

In the case where only positive examples are available, theoretical results [Gol67] show that the task becomes considerably harder. An alternative is to learn a probabilistic finite automaton from the data, learning the regularities of the distribution rather than those of the language: several algorithms have been proposed [CO94,SO94,TDdlH00,YLT00] for this task.

These algorithms generally perform well on small benchmarks but are not currently able to obtain significant results on real world tasks where the size of the alphabet, the lack of learning data or the noise are serious obstacles. An alternative to better these techniques is to reduce the search space, or to put it like McAllester and Schapire [MS01]: "to seed the search with sufficient initial regularities".

One way to do this is through the inclusion of prior knowledge in models, which is an important and successful feature in machine learning. Very often, the complexity of the intended model is such that the quantity of learning data is insufficient. The success of a model and a learning algorithm depends on their ability to include prior knowledge, in order to compensate for the lack of data. Alternatively, with only a fixed set of data, prior knowledge allows to learn more complex functions. In general prior knowledge is additional external knowledge that is not used during the learning phase. When the knowledge is used twice, first to classify, order or prepare the data, and second for the actual learning, we will use the term of *additional knowledge*. It is generally accepted that prior knowledge is taken from a different source than the actual data the learning process is going to use. Yet in the sequel we will be allowing the additional knowledge to be merely another point of view on the learning data.

In the specific context of probabilistic model learning, the importance of additional knowledge has been shown in applications of Hidden Markov Models; whereas mathematically founded methods exist to estimate the parameters of the probability distribution in these models (the EM algorithm [DLR77]), the quality of these estimations widely depend on the chosen structure of the models (number of states, number of transitions and topology); the success of HMMs in several application domains, like speech recognition or computational biology, is partly due to the use of additional knowledge to design the structure of the models:

– in speech recognition, the knowledge on the phonemic structure of utterances induces the left-to-right structure of the HMM (Bakis models) and also facilitates the learning process with data segmentation [RBH93];
– in computational biology, additional knowledge regarding the mean length of proteins and additional distribution of amino acids is used to design the models [DEKM99].

We believe that the use of additional knowledge in grammatical inference can bring a number of advantages:

- first, the search space of the deterministic finite automata inference task is well defined [DMV94]. This space depends on the learning data available, and can be extremely large. The use of additional knowledge on the structure of the target automaton reduces the search space by excluding from the search automata which do not conform with this knowledge;
- second, additional knowledge can complete the learning data, for example by providing implicit counter-examples: strings known not to belong to the target language;
- finally, real world constraints that the induced formal language must satisfy can be introduced in a simple way.

Additional knowledge has already been used in grammatical inference by Bernard & de la Higuera [BdlH01] in their ILP system called GIFT[1]. GIFT learns a tree automaton from a set of terms, which is later translated into a logic program. The algorithm is applied to structured data, and a typing system of this data is also inferred (for instance, rules that state that a person has two parents one of which is male and the other female). Typing is used to avoid impossible situations (for example, in a family relationship, someone with two fathers). Kermorvant & de la Higuera [KdlH02] have proposed a framework to include additional knowledge in the automaton inference algorithms. In this framework, additional knowledge regarding sequences to be modeled can be included in automata using typing.

In this paper, we propose to infer typed automata from labeled sequential data. Typing is performed through the use of an initial parse of the data where either statistical clusters of strings are formed (inducing one type *per* class) or by part-of-speech labeling through an adapted Brill tagger [Bri92]. It should be noticed that the adapted tagging will be limited in the sense that prefixes will have to be tagged in an uniform way. This constraint should be considered as a compromise. By doing so the complexity of the learning algorithm is not worse than the complexity of the original one, as testing whether a given merge is compatible with the typing can be done in constant time. The induced types are used by a classical grammatical inference algorithm, ALERGIA, that constructs a probabilistic finite state automaton.

We compare the use of these two kinds of additional knowledge in the framework of language modeling: on the Air Travel Information System (ATIS) task, the results we report are comparable with those achieved by state-of-the art n-grams models.

2 Regular Language Learning from Tagged Data

The search space for regular languages inference has been studied by Dupont *et al.* [DMV94]. Basically, when inferring a regular language from a sample set I_+,

[1] Grammatical Inference For Terms.

the search space is a partition lattice defined by the set of states of the prefix tree acceptor (PTA) which is the smallest deterministic automaton accepting exactly I_+ and in which each state has at most one predecessor. The least upper bound of the lattice is the partition corresponding to the universal automaton which accepts all strings on the alphabet. Under the hypothesis of the presence of a characteristic set in the sample set, we are insured that the target automaton belongs to the lattice. However, since the size of the lattice is exponential in the size of the sample set, we need a good strategy to explore this lattice. Evidence driven strategies have already been proposed [dOV96,LPP98].

Regular language inference from real data, such as natural language sentences or biological sequences, raises additional difficulties. First, negative information is not always available. Secondly, real data is generally noisy. If we choose to learn probabilistic finite automata we can, in principle, handle both the lack of negative information and the presence of noise. Besides, background knowledge of the application domain is often available. In [KdlH02] a framework that includes additional knowledge in the automaton inference algorithms is proposed. This framework is the application of typing, as known for terms and trees, to finite state automata. In this paper, we propose to infer typed probabilistic automata from tagged data.

2.1 Typed Probabilistic Finite State Automata

We consider probabilistic finite state automata (PFA), which provide a probabilistic extension of finite state automata. A PFA $\mathcal{A}$ is a tuple $< Q, \Sigma, \delta, \tau, q_0, F >$ where:

- Q is a finite set of states,
- Σ is an alphabet,
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function,
- $\tau : Q \times \Sigma \rightarrow]0..1]$ is a function which returns the probability associated with a transition;
- q_0 is the initial state,
- $F : Q \rightarrow [0..1]$ is a function which returns the probability for a state to be final.

Furthermore, we only consider PFA which are structurally deterministic. This means that from any given state q at most one state can be reached on any given alphabet symbol. This has two implications. First, we restrict our attention to the particular class of distributions that can be generated by *deterministic* PFA (PDFA). Second, inference algorithms explicitly searching for PDFA, such as ALERGIA [CO94], can be used.

In order to define a probability distribution on Σ^* (the set of all strings built on Σ), the automaton must be trimmed (without useless states) and τ and F must satisfy the following consistency constraint:

$$\forall q \in Q, \left[\sum_{a \in \Sigma} \tau(q,a)\right] + F(q) = 1$$

A string $a_0 \cdots a_{l-1}$ is generated by an automaton $\mathcal{A}$ *iff* there exists a sequence of states $e_0 \cdots e_l$ such that

- $e_0 = q_0$
- $\forall i \in [0, l-1],\ \delta(e_i, a_i) = e_{i+1}$
- $F(e_l) \neq 0$.

The automaton assigns to the string the probability

$$P_{\text{PFA}}(a_0 \cdots a_{l-1}) = \left[\prod_{i=0}^{l-1} \tau(e_i, a_i)\right] * F(e_l) \qquad (1)$$

Note that a PFA can be seen as a particular case of Markov models with discrete emission probabilities on transitions and with final probabilities.

Typed automata are introduced in [KdlH02]. A typed PFA $\mathcal{A}$ is defined as a tuple $< Q, \Sigma, \delta, \tau, q_0, F, S, \sigma >$ where:

- $Q, \Sigma, \delta, \tau, q_0, F$ are the same as in classical PFA,
- S is a set of sorts,
- σ is a typing function which associates a single sort to each state of the automaton.

A typed probabilistic automaton is a probabilistic automaton with typed states. There are several ways to define the typing function σ. In [KdlH02], it was proposed to define the typing function σ by a type automaton constructed by an expert, on the basis of some knowledge he may have of the domain. In this paper, we propose to automatically infer the typing function from labeled sequential data. Labeled sequential data corresponds to strings where the symbols that appear are tagged. Such data is very often available and can take the form of a tagging over the sequences, for example part-of-speech tagging for natural language sentences or secondary structure for protein sequences.

We define the extension of σ to words such that $\sigma(w) = \sigma(\delta(q_0, w))$ for all $w \in \text{Pref}(L)$, where $\text{Pref}(L)$ the set of all the prefixes of all the strings recognized by the typed automaton. We define $\sigma_{uv}(u)$ as the sort of prefix u in the context of string uv.

Typing functions could, in theory, be as complex as one may want. Practically we do not want typing to be a burden to learning. The choice in this paper is to make type-checking easy, even if that introcuces limits for the typing function. The typing function (σ) must be able to type all states, and therefore possible strings in a regular manner. Therefore, if L is the set of all possible strings, two conditions must be met:

- $\forall u \in L, \sigma(u)$ is defined;
- $\forall uv, uw \in L \Rightarrow \sigma_{uv}(u) = \sigma_{uw}(u)$.

A typing function σ is *admissible* if the above conditions hold.

Hence one can associate various types to a symbol, but only one type to a string. It should be noticed that this is a strong condition: in usual cases tags are computed by taking into account both left and right-hand contexts. In section 4 we discuss various ways of lifting this condition.

Algorithm 1 Generic PFA induction algorithm

```
INPUTS
$I_+$, training set (sequences)
$\alpha$, a precision parameter
OUTPUT
a probabilistic finite state automaton

A ← $build\_PPTA(I_+)$
while $(q_i, q_j) \leftarrow choose\_states(A)$ do
  if $is\_compatible(q_i, q_j, \alpha)$ then
    $merge(A, q_i, q_j)$
return A
```

2.2 Learning Typed Automata from Automatically Labeled Data

Several algorithms have been proposed to infer PFA from examples using frequencies [CO94,RST95,TDdlH00]. All these algorithms are based on a similar scheme, which is presented in algorithm 1. Our inference algorithm for typed automata from labeled data is also derived from this scheme that we explicit below.

Given a set of labeled positive examples I_+ , the algorithm first builds the typed *probabilistic prefix tree acceptor* (PPTA). A set of labeled examples is a set a examples with a label (a sort) for every prefix of every examples. The typed PPTA is an automaton accepting all examples of I_+, in which the states corresponding to common prefixes are merged and such that a training count is attached to each state and each transition. This count denotes the number of times this state, or transition, is used while parsing the sample. An estimate $\hat{\tau}$ (resp. $\hat{F}$) of the function τ (resp. F) can be computed from these counts. $C(q)$ denotes the number of times the state q is used while parsing I_+, $C(q,a)$ denotes the number of times the transition (q,a) is used while parsing I_+, and $C_f(q)$ denotes the number of times q is used as final state in I_+. For each state $q \in Q$, we have:

$$\forall a \in \Sigma, \hat{\tau}(q,a) = \frac{C(q,a)}{C(q)} \qquad \hat{F}(q) = \frac{C_f(q)}{C(q)}$$

The typing function of the typed PPTA is defined by the labels of the strings in I_+. When a string with label l is used to reach a state q of the typed PPTA, the label l is the type of the state q: $\sigma(q) = l$. Note that the fact that the typing function is admissible implies that a typed PPTA can always be built from a given labeling.

In the framework of language modeling, text corpora manually or automatically annotated with Part-of-Speech (POS) tags are available. An example of an annotated corpus of English sentences is:

I/PRP fly/VBP from/IN Dallas/NNP to/TO Philadelphia/NNP ./.
I/PRP want/VBP a/DT flight/NN from/IN Baltimore/NNP ./.

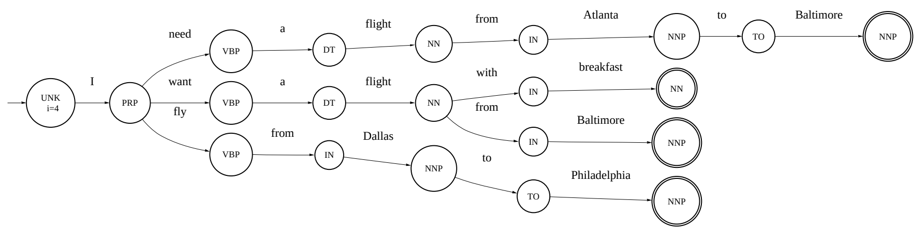

Fig. 1. A typed probabilistic prefix tree acceptor.

I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NNP to/TO Baltimore/NNP ./.
I/PRP want/VBP a/DT flight/NN with/IN breakfast/NN ./.

The typed PPTA constructed from these labeled examples is presented in Figure 1.

The second step of the algorithm consists in visiting the states of the PPTA (function *choose_ states(A)*), and testing whether the states are compatible and can be merged. The compatibility criterion (defined in algorithm 1) by function *is_ compatible*(q_i, q_j, α) depends on a precision parameter α. If the states are compatible, they are merged (function *merge(A,*q_i, q_j*)*). Usually, several consecutive merging operations are made in order to maintain the deterministic structure of the automaton. The algorithm halts when no more merging is possible. In the case of algorithm ALERGIA [CO94], the compatibility of two states is based on different tests: the compatibility of their outgoing probabilities on the same letter, the compatibility of their probability to be final and the recursive compatibility of their successors. Statistically, these tests are derived from Hoeffding bounds [Hoe63].

By using only admissible typing functions, the introduction of type constraints in the learning algorithm is straightforward. We must only, each time a merging operation is considered, check if the states have the same type. This constraint can easily be checked in constant time: it is sufficient to test the equality of the types of q_i and q_j in function *is_ compatible(*q_i, q_j, α*)*. With this constraint, we ensure that the type of all the strings in the inferred language is consistent with the tagger which provided the tagged learning set.

3 Experiments

3.1 The ATIS Language Modeling Task

We have tested our approaches on a language modeling task. A language model is used any application dealing with text : speech recognition, machine translation, handwriting recognition. This model is used to assign a probability to sequences of words. A good language model must be able to predict accurately the next word in a sequence.

We have used the Air Travel Information System (ATIS) corpus to learn and test our models. This corpus consists in information requests performed in American English. The sentences have been collected in Wizard-of-Oz conditions in which a human secretly replace the speech recognition systems in an automated dialog system. We use the ATIS-2 sub corpus which is composed of a training set containing 13,044 utterances (130,773 tokens) and two test sets containing respectively 974 utterances (10,636 tokens) and 1001 utterance (11,703 tokens). We use the first test set as a validation set to tune the parameters of the algorithms and the second one as an independent test set. The task vocabulary is composed of 1,296 different words. This corpus has been widely used in the speech recognition community and specifically for probabilistic automaton induction tasks (see e.g. [DC98,TDdlH00,LVC02]).

All the automata are inferred on the train set and evaluated on the ATIS test set. The usual quality measure in language modeling tasks is the average (per word) log-likelihood (LL) of the words in the sequences of the test set S. A directly related measure is known as the *test set perplexity*:

$$PP = 2^{LL} = 2^{-\frac{1}{||S||}\sum_{s=w_0\cdots w_{l-1}\in S}\log_2 P(w_0\cdots w_{l-1})}$$

The smaller the perplexity the better the automaton can predict the next word. It is generally agreed that perplexity is a good quality criterion for language models. In order to guarantee that every word can be predicted with a non null probability, the inferred automaton must be smoothed. We interpolate the automaton with a unigram model. The unigram defines the probability $P_1(w)$ of each word w in the training set, independently of its context. The probability of a word w assigned by the smoothed automaton is then (with the notations of equation 1)

$$P(w_0\cdots w_{l-1}) = \left[\prod_{i=0}^{l-1}\beta\cdot\tau(e_i,w_i) + (1-\beta)P_1(w_i)\right] * [\beta\cdot F(e_l) + (1-\beta)P_1(\#)]$$

where $P_1(\#)$ is the unigram probability of the end symbol. Since this smoothing technique is very rudimentary, it best reflects the quality of the induced PFA alone. As some words of the application vocabulary may not occur in the training set[2], the unigram probability itself is smoothed by absolute discounting [NEK94]. Let $C(w)$ denote the count of word w in the training set containing a total of N tokens. The smoothed unigram probability is defined as follows:

$$P_1(w) = \begin{cases} \frac{C(w)-d}{N} & \text{, if } C(w) > 0 \\ \frac{D}{N_0} & \text{otherwise} \end{cases}$$

where d is a *discounting parameter* (set here to 0.5), D is the total discounted probability mass and N_0 is the number of unseen words in the training sample:

$$D = \sum_{\{w \mid C(w)>0\}} \frac{d}{N} \qquad N_0 = \sum_{\{w \mid C(w)=0\}} 1.$$

[2] This is the case for 131 out of 1,296 words.

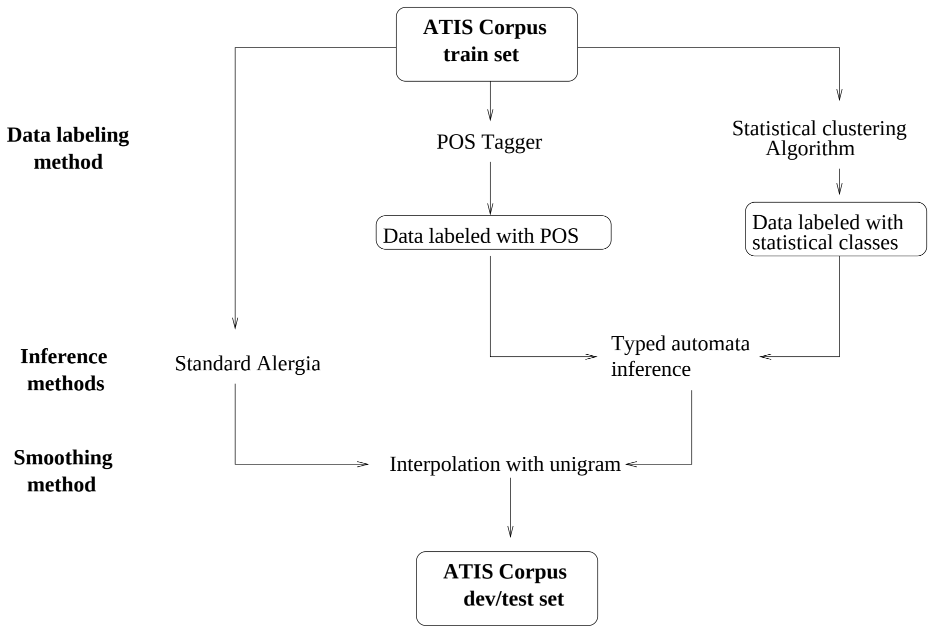

Fig. 2. Experimental protocol : a comparison of two typing functions.

The influence of the various learning parameters was first evaluated on the validation set. Final results are reported on an independent test set. Note that a better methodology would rely on n-fold cross-validation. We did not follow this methodology in the present case because comparative results exist mainly on the same splitting between validation and test sets (e.g. see [DC98]). It has been observed experimentally that the test set is closer to the training set than the validation set is.

3.2 Comparison of two Typing Functions for Automata Inference

In this section, we compare the use of two kind of additional knowledge for typed automata inference : Pos tags and statistical clustering. The baseline is a standard inference algorithm (Alergia). The experimental protocol is shown on Figure 2.

The Pos information was obtained by tagging the training set using the Brill tagger [Bri92]. As a first approach, each word was tagged with its most likely tag, disregarding the context rules. It has been shown on a French corpus, that this tagging method can yield up to 90% of correct tags [VG98]. The resulting tagged training set contains 32 different Pos types.

The statistical information leading to the class tagging was obtained by the clustering algorithm presented in [DC98]. For a given number of clusters, the clustering algorithm iteratively constructs the classes so that the average mutual

Table 1. Best perplexity results on the ATIS test set with interpolation to unigram for standard and typed automata

	ALERGIA	Typed automata	
		POS tags	Statistical clusters
Perplexity	66	57	42

information between the classes is maximized. Values for the number of clusters ranging from 10 to 1000 have been tested.

Table 1 shows the perplexity results on the ATIS test set for the best standard automata infered with ALERGIA and the best typed automata. The perplexity of the typed automata is better and the best perplexity is obtained with a typed automaton using statistical clusters typing.

We now look into more details the influence of the different learning parameters and other quality criteria.

Three learning parameters were tuned on the ATIS development set :

- the precision parameter α which controls the compatibility criterion and therefore the number of compatible merging operations,
- the number k of distinct types,
- the interpolation parameter β.

Note that k can only be tuned when the types correspond to statistically induced classes. In this case, the optimal number of classes had been found to be 90. In the case of POS tagging, the number of distinct types is defined *a priori* by the tagger and cannot be tuned. The parameters α and k control the degree of generalization allowed during the typed automaton induction. Hence these parameters control the number of parameters of the inferred model (number of states and transitions of the PFA). The parameter β controls the weight of the induced PFA in the combined smoothed automaton.

Figure 3 shows the percentage of sentences from the validation set fully parsed by the inferred typed automaton for the two kind of additional knowledge with respect to the number of parameters of the typed automaton (number of states and number of transitions). In this case, no smoothing is used. The typed PPTA parses only 7% of the validation set whereas the universal automaton which accepts all the sentences built with words of the training set, parses 94% of the sentences in the validation set (6% of the sentences are not fully parsed since they contains out-of-vocabulary words). For a fixed number of parameters, the use of typed automata increase the number of sentences that can be parsed compared to standard automata infered with ALERGIA.

Figure 4 presents the perplexity obtained by the inferred typed automata with respect to the number of sentences parsed. The best results are situated in the bottom right corner of figure 4 as they correspond to high coverage and small perplexity. The smoothed unigram parses 100% of the sentences but yields a perplexity of 145. For a given number of parsed sentences, both the POS-based and cluster-based typed automata yield a smaller perplexity and the typed automaton inferred with statistical classes yields the smallest perplexity. It should

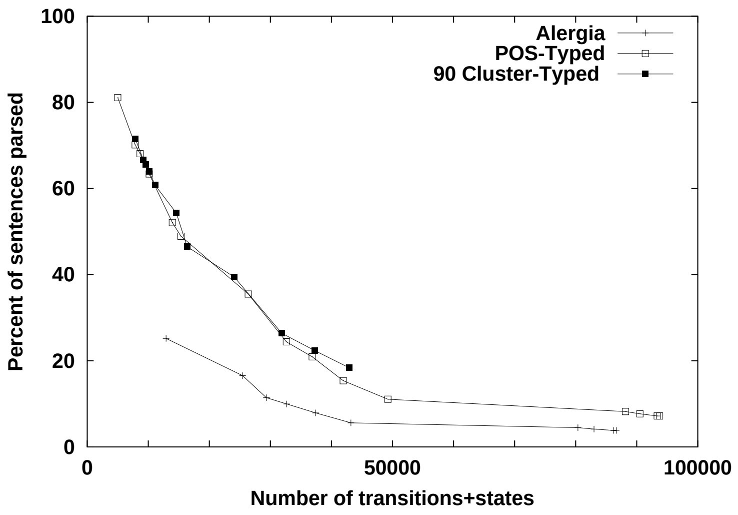

Fig. 3. Percentage of sentences correctly parsed in the validation set versus the number of parameters of the automaton

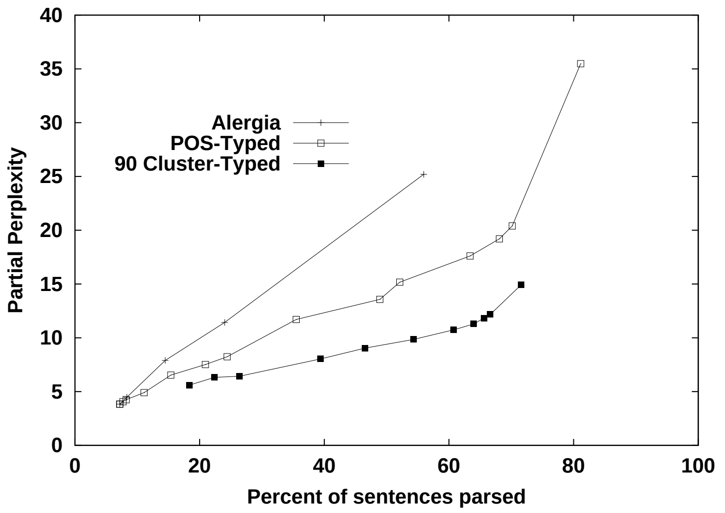

Fig. 4. Perplexity of the sentences correctly parsed in the validation set

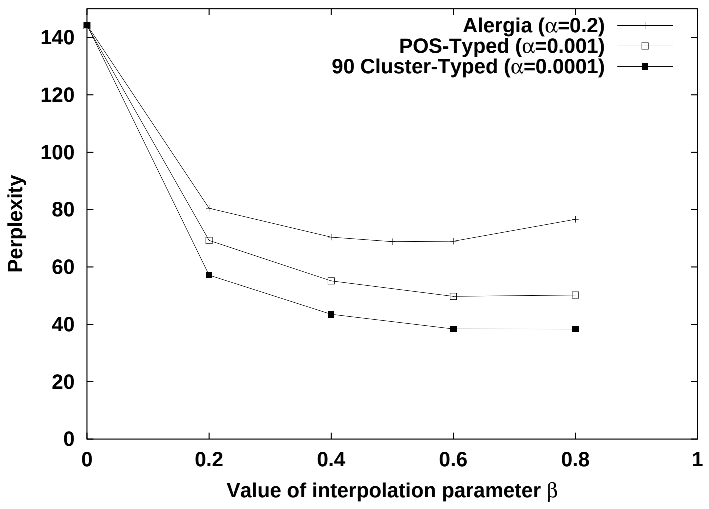

Fig. 5. Perplexity on the ATIS validation set: inferred typed automaton with unigram interpolation.

be stressed that, as no smoothing is performed in this case, the perplexity is only partial as it is computed over those strings that can be parsed.

Figure 5 shows the perplexity obtained by the inferred typed automaton interpolated with the smoothed unigram. The best perplexity reduction (39% as compared with standard Alergia) is obtained when using typed inference with 90 statistically defined classes with inference parameter $\alpha = 1.10^{-4}$ and interpolation parameter $\beta = 0.8$.

3.3 Comparison with Class-Based Inference

In this section we compare our approach with a method previously used to improve automata inference.

Dupont & Chase [DC98] proposed to use statistical clustering of words to improve grammatical inference on large vocabularies. The first step of their approach consists in building classes of words from the learning samples. Once the classes are defined, each word is associated to a class and the probability of each word w in its class $g(w)$, denoted by $\hat{P}(w|g(w))$, can easily be computed. The learning samples are then relabeled in terms of classes and an automaton is inferred on the class labels using a classical inference algorithm such as ALERGIA. Finally the automaton is expanded by replacing each class by all the words it contains. More formally, once an automaton is inferred on the classes, each transition (q, G) from a state q with label G is replaced by as many transitions as there are words w such that $g(w) = G$. The probability estimates $\hat{\tau}(q, w)$ of these transitions are given by $\hat{\tau}(q, w) = \hat{\tau}(q, G) \cdot \hat{P}(w|G)$.

Table 2. Two approaches to use two different kinds of information.

	typed automata inference	inference on classes + expansion
Pos tags	Pos-typed automata	Pos-class automata
Statistical clusters	cluster-typed automata	cluster-class automata

Table 3. Best perplexity results on the Atis test set with interpolation to unigram for the four inference methods.

	typed automata inference	inference on classes + expansion
Pos tags	57	112
Statistical clusters	42	52

We propose to use the same scheme but with Part-of-Speech classes instead of statistical clusters. The class automaton is infered with Alergia on Pos tags and expanded to words afterward. This yields to compare four approaches that are summarized in Table 2.

The perplexity results of these four approaches is shown on Table 3. Our approach, based on typed automata yields better results than the approach based on class inference, both when using Pos tags or statistical clusters.

3.4 Improved Smoothing Methods

The smoothing technique used in the evaluations described in section 3.2 et 3.3 is rudimentary. We argued that interpolation with a smoothed unigram guarantees to bound the perplexity while best reflecting the predictive power of the inferred Pfa alone. However, if the objective is to minimize test set perplexity, more sophisticated smoothing techniques are required.

The best language model on the Atis task is a trigram model with Kneser-Ney back-off smoothing [KN95]. This smoothed trigram model combines a trigram model and two back-off distributions, respectively based on a bigram and a unigram model. The Atis test set perplexity of this combined model is 14.

Current results for the best typed automata inferred with 90 statistically defined classes and smoothed with a simple back-off to unigram (a simplified version of the smoothing scheme described in [LVC02]) gives a perplexity of 20. The trigram model smoothed with the same method (back-off to unigram) gives a perplexity of 17. Further improvements of the smoothing techniques for automata should therefore decrease the perplexity.

A language model can alternatively be viewed as a compressed version of the learning data. In this case, the actual size of the model is very important. It should be noted that the number of parameters needed by the best typed automata combined with a smoothed unigram is $1.1 * 10^5$. The trigram model with Kneser-Ney smoothing to both bigram and unigram needs $6 * 10^5$ parameters. The smoothed typed automata needs less parameters to obtain a similar perplexity on this task.

4 Discussion

It has been shown in [DC98] that the use of statistical class information improves the quality of probabilistic automata used as language models. The present work illustrates that this is even more true when statistically induced classes are combined with typed PFA inference.

The results obtained when using POS tag information are less convincing, even though it has been shown that grammatical information can help language models [Cha01]. Let us stress however that we did not use here the full information provided by the POS tagger as each word was tagged according to its most likely tag, disregarding the contextual rules. This approximation was required to construct a typed PPTA, which is a deterministic PFA as explained in section 2.2. In order to fully take into account POS information, several extensions of the present approach are possible. First, inference algorithms could be developed to infer (possibly) non-deterministic structures, as proposed by [ELDD02]. Second, an extension typing functions allowing several types per states and inducing multi-typed automata could be developed.

Finally, the framework of typed automata is general and could easily be adapted to other grammatical inference algorithms (MDI [TDdlH00] or MALERGIA [KD02]) which have been shown to have better performances than ALERGIA.

5 Conclusion

We have proposed a way to use additional knowledge in grammatical inference with typed automata. When manually or automatically labeled data is available, the labels can be used as types and the inference algorithm we have proposed guarantees that the inferred automaton is compatible with the labeled data. We have compared the use of two kinds of labeling for probabilistic typed automata inference. Part-of-speech labeling provided by a POS tagger and statistical clustering of words have been compared as labeling for natural language data. The use of statistical word classes information allows us to infer better automata. Our approach provides models which are competitive with state-of-the-art n-grams with similar smoothing techniques while being more compact and needing less parameters.

References

[BdlH01] M. Bernard and C. de la Higuera. Apprentissage de programmes logiques par inférence grammaticale. *Revue d'Intelligence Artificielle*, 14(3/4):375–396, 2001.

[Bri92] E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 152–155, 1992.

[Cha01] E. Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 116–123, 2001.

[CO94] R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *Proc. Int. Coll. on Grammatical Inference*, volume 862 of *Lecture Notes in Artificial Intelligence*, pages 139–152. Springer-Verlag, 1994.

[DC98] P. Dupont and L. Chase. Using symbol clustering to improve probabilistic automaton inference. In *Proc. Int. Coll. on Grammatical Inference*, volume 1433 of *Lecture Notes in Artificial Intelligence*, pages 232 – 243. Springer-Verlag, 1998.

[DEKM99] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1999.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[DMV94] P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference? In *Proc. Int. Coll. on Grammatical Inference*, volume 862 of *Lecture Notes in Artificial Intelligence*, pages 25–37. Springer-Verlag, 1994.

[dOV96] C. de la Higuera, J. Oncina, and E. Vidal. Identification of DFA: data-dependent versus data-independent algorithms. In *Proc. Int. Coll. on Grammatical Inference*, volume 1147 of *Lecture Notes in Artificial Intelligence*, pages 313–325. Springer-Verlag, September 1996.

[ELDD02] Y. Esposito, A. Lemay, F. Denis, and P. Dupont. Learning probabilistic residual finite state automata. In *Proc. Int. Coll. on Grammatical Inference*, volume 2484 of *Lecture Notes in Artificial Intelligence*, pages 77–91. Springer-Verlag, 2002.

[Fre97] D. Freitag. Using grammatical inference to improve precision in information extraction. In *Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, Fourteenth International Conference on Machine Learning*, Nashville, Tennessee, 1997.

[GLT01] C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning Journal*, 44(1):161–183, 2001.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[Jel98] F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, Cambridge, Massachusetts, 1998.

[KD02] C. Kermorvant and P. Dupont. Stochastic grammatical inference with multinomial tests. In *Proc. Int. Coll. on Grammatical Inference*, volume 2484 of *Lecture Notes in Artificial Intelligence*, pages 149–160. Springer-Verlag, 2002.

[KdlH02] C. Kermorvant and C. de la Higuera. Learning languages with help. In *Proc. Int. Coll. on Grammatical Inference*, volume 2484 of *Lecture Notes in Artificial Intelligence*, pages 161–173. Springer-Verlag, 2002.

[KN95] R. Kneser and H. Ney. Improved backing-off for N-gram language modeling. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, pages 181–184, May 1995.

[Kol99] D. Koller. Probabilistic relational models. In S. Dzeroski and P. Flach, editors, *Inductive Logic Programming (ILP-99)*, pages 3–13. Springer-Verlag, 1999.

[LGF00] S. Lawrence, C. L. Giles, and S. Fong. Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 42(1):126–140, 2000.

[LPP98] K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In Springer-Verlag, editor, *Proc. Int. Coll. on Grammatical Inference*, volume 1433 of *Lecture Notes in Artificial Intelligence*, pages 1–12, 1998.

[LVA$^+$94] S. Lucas, E. Vidal, A. Amari, S. Hanlon, and J. C. Amengual. A comparison of syntactic and statistical techniques for off-line ocr. In *Proc. Int. Coll. on Grammatical Inference*, volume 862 of *Lecture Notes in Artificial Intelligence*, pages 168–179. Springer-Verlag, 1994.

[LVC02] D. Llorens, J. M. Vilar, and F. Casacuberta. Finite state language models smoothed using N-grams. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):275–289, 2002.

[MS01] D. McAllester and R. E. Schapire. Learning theory and language modeling. In G. Lakemeyer and B. Nebel, editors, *Proc. Int. Joint Conf. on Artificial Intelligence*. Morgan Kaufmann, 2001.

[NEK94] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

[OG92] J. Oncina and P. García. Identifying regular languages in polynomial time. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition: Proc. of the International Workshop*, pages 99–108. World Scientific, 1992.

[RBH93] L. Rabiner and J. Biing-Hwang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[RST95] D. Ron, Y. Singer, and N Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 31–40. ACM Press, 1995.

[SMR99] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, pages 37–42, Orlando, Florida, 1999.

[SO94] A. Stolcke and S. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *Proc. Int. Coll. on Grammatical Inference*, LNAI, pages 106–118. Springer-Verlag, 1994.

[TDdlH00] F. Thollard, P Dupont, and C. de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. Int. Conf. on Machine Learning*, pages 975–982. Morgan Kaufmann, 2000.

[VG98] J. Vergne and E. Giguet. Regards théoriques sur le tagging. In *Proceedings of the conference Le Traitement Automatique des Langues Naturelles*, 1998.

[YLT00] M. Young-Lai and F .W. Tompa. Stochastic grammatical inference of text database structure. *Machine Learning Journal*, 40(2):111–137, 2000.