Improving probabilistic automata learning with Additional Knowledge

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[4], 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. 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

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, the RPNI algorithm [OG92] can be used to infer deterministic finite automata.

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] for this task.

These algorithms generally perform well on small tasks but are not currently able to obtain significant results on real world tasks where the size of the alphabet and the noise are serious obstacles. Furthermore, very often the complexity of the intended model is such that the quantity of learning data is insufficient. The

[4] To appear in Proc. of the Joint IAPR International Workshops on Syntactical and Structural Pattern Recognition (SSPR 2004) and Statistical Pattern Recognition (SPR 2004), Lisbon, Portugal, 2004

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 the specific context of probabilistic model learning, 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 and in computational biology, additional knowledge regarding the mean length of proteins and additional distribution of amino acids are used to design the models.

We believe that the use of additional knowledge in grammatical inference can bring a number of advantages. Firstly to reduce the search space by excluding automata which do not conform with this knowledge; secondly to complete the learning data with additional knowledge, for example by providing implicit counter-examples: strings known not to belong to the target language; thirdly to introduce in a simple way real world constraints that the induced formal language must satisfy.

Kermorvant & de la Higuera [KdlH02] have proposed a framework based on state typing to include additional knowledge into the automaton inference process. State typing both reduces the search space of the probabilistic finite automata (PFA) induction (hence decreasing the practical complexity of learning), and guarantees the compatibility of the learned models with this knowledge. The additional knowledge considered in the present paper either comes from statistical clusters or part-of-speech tags.

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 the problem of regular languages inference is finite but huge [DMV94]: it is a partition lattice defined by the set of states of the prefix tree acceptor (PTA). The PTA is the smallest tree-shaped deterministic automaton accepting exactly I_+. Under the hypothesis of the presence of a structurally complete sample (*i.e.* a set of strings that makes use of all edges, nodes and final states of the target), the target automaton is guaranteed to belong to this lattice. A negative sample is generally used to control the generalization while searching for the target. However, since the size of the lattice is exponential in the size of the sample set, a good strategy is required to explore this lattice. We choose to learn probabilistic finite automata so that we can, in principle, handle two additional difficulties raised by real data: 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.

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, $\delta : Q \times \Sigma \to Q$ is a transition function, $\tau : Q \times \Sigma \to]0..1]$ is a function which returns the probability associated with a transition; q_0 is the initial state, $F : Q \to [0..1]$ is a function which returns the probability for a state to be final. A typed PFA [KdlH02] is defined with the addition of S, a set of types and σ, a typing function which associates a single type to each state of the automaton.

Furthermore, we only consider PFA which are structurally deterministic and which define define a probability distribution on Σ^* (trimmed and satisfy the consistency constraint: $\forall q \in Q, \left[\sum_{a \in \Sigma} \tau(q, a)\right] + F(q) = 1$) The probability assigned by the automaton to a string is classically the product of the transition probabilities along any accepting path and the final probability.

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 strings where the symbols are tagged for example according to a part-of-speech tagger for natural language sentences.

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

- if L is the set of all possible strings: $\forall u \in L, \sigma(u)$ is defined;
- if we denote by $\sigma_{uv}(u)$ the label of prefix u in the context of string uv: $\forall uv, uw \in L \Rightarrow \sigma_{uv}(u) = \sigma_{uw}(u)$.

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 relaxing this condition.

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.

Algorithm 1 Generic PFA induction algorithm

Require:
I_+, training set (sequences)
α, a precision parameter
Ensure: a probabilistic finite state automaton
A $\leftarrow$ *build_PPTA*(I_+)
while $(q_i, q_j) \leftarrow$ *choose_states*(A) **do**
if *is_compatible*(q_i, q_j, α) **then**
merge(A,q_i, q_j)
return A

Given a set of labeled positive examples I_+, the algorithm first builds the typed *probabilistic prefix tree acceptor* (PPTA). 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.Let $C(q)$ (respectively $C(q,a)$ and $C_f(q)$) denotes the number of times the state q (respectively the transition (q,a) and the final state q) is used while parsing I_+. An estimate $\hat{\tau}$ (resp. $\hat{F}$) of the function τ (resp. F) can be computed from these counts:

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

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.

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 testing the compatibility of their associated provabilities.

By using only admissible typing functions, the introduction of type constraints in the learning algorithm is straightforward. Every time a merging operation is considered, we check whether the states have the same type. This constraint can easily be implemented 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, the type of any string in the inferred language is guaranteed to be consistent with the types of the strings in the learning set.

3 Experiments

We have tested our approaches on a language modeling task with the Air Travel Information System (ATIS) corpus. This corpus has been widely used in the speech recognition community and specifically for probabilistic automaton induction tasks (see *e.g.* [DC98,TDdlH00,LVC02]). This corpus consists in information requests performed in American English.

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 utterances (11,703 tokens). The task vocabulary is composed of 1,296 different words. All the automata are inferred on the train set, the parameters are tuned on the first test set (named validation set), and the second test set (named test set) is used for independent final evaluation.

The usual quality measure in language modeling tasks is *test set perplexity*:

$$PP = 2^{LL} = 2^{-\frac{1}{||S||}\sum_{w \in S} \log_2 P(w)}$$

where $P(w)$ denotes the predicted probability of word w and $||S||$ denotes the number of words in the test set. The smaller the perplexity the better the automaton can predict the strings observed in the test set. 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, which 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:

$$P(w) = \beta P_{PFA}(w) + (1 - \beta)P_1(w)$$

This smoothing technique is very rudimentary but, as such, it best reflects the quality of the induced PFA alone. Finally, as some words of the application vocabulary may not occur in the training set[5], the unigram probability itself is smoothed by absolute discounting [KN95].

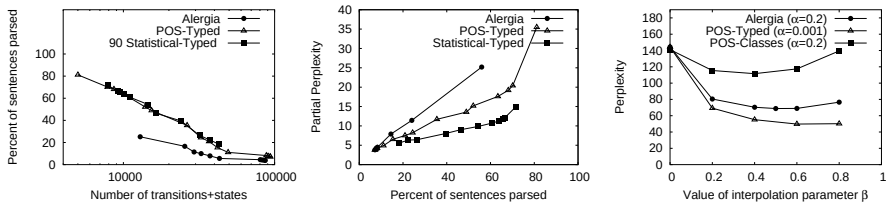
3.1 Comparison of two Typing Functions for Automata Inference

In this section, we compare the use of two kind of additional knowledge for typed automata inference : POS tags and statistical clustering. Our baseline is a standard PFA inference algorithm (ALERGIA) not using state typing.

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. 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

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



(a) (b) (c)

Fig. 1. Results on the validation set : Percentage of sentences correctly parsed versus the number of parameters of the automaton 1(a), perplexity of the sentences correctly parsed 1(b), and perplexity with inferred typed automaton and unigram interpolation 1(c).

clustering algorithm iteratively constructs the classes so that the average mutual information between the classes is maximized. Values for the number of clusters ranging from 10 to 1000 have been tested.

The best standard automata inferred with ALERGIA yields a perplexity of 66 on the ATIS test set. On the same test set, typed automaton inferred using POS tags typing yields a perplexity of 57 and the best perplexity score, 42, is obtained with the typed automaton using statistical clusters. The influence of 3 learning parameters was studied 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 is 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 1(a) 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 contain 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 inferred with ALERGIA.

Figure 1(b) 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 1(b) 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 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 1(c) 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 $\beta = 0.8$.

3.2 Comparison with Class-Based Inference

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

Dupont & Chase [DC98] proposed to use statistical clustering of symbols to improve grammatical inference on large vocabularies. The first step of their approach consists in building classes of symbols from the learning samples. Once the classes are defined, each symbol is associated to a class and the probability of each symbol 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 symbols 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 symbols w such that $g(w) = G$. The probability estimates $\hat{\tau}(q, w)$ of these transitions are given by $\hat{\tau}(q, w) = \hat{\tau}(q, G) \cdot \hat{P}(w|G)$.

We propose to use the same scheme but with Part-of-Speech classes instead of statistical clusters. The class automaton is inferred with ALERGIA on POS tags and expanded to words afterward. This yields to compare four approaches that are summarized in Table 1. The perplexity results of these four approaches is also shown on Table 1. Our approach, based on typed automata yields better results than the approach based on class inference, both when using POS tags or statistical clusters.

3.3 Improved Smoothing Methods

The smoothing technique used in the evaluations described in section 3.1 and 3.2 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.

	typed automata inference	inference on classes + expansion
POS tags	POS-typed automata perplexity : 57	POS-class automata perplexity : 112
Statistical clusters	cluster-typed automata perplexity : 42	cluster-class automata perplexity : 52

Table 1. Two approaches to use two different kinds of information and their best perplexity results on the test set with interpolation to unigram.

A very competitive 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.

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. 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. Firstly inference algorithms could be developed to infer (possibly) non-deterministic structures. Secondly extended 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 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

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

[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 *LNAI*, 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 *LNAI*, pages 232 – 243. Springer-Verlag, 1998.

[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 *LNAI*, pages 25–37. Springer-Verlag, 1994.

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

[KdlH02] C. Kermorvant and C. de la Higuera. Learning languages with help. In *Proc. Int. Coll. on Grammatical Inference*, volume 2484 of *LNAI*, 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, 1995.

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

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

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

[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.