Construction de Modèles de Langages par Inférence d'Automates Typés à partir de Données Etiquetées Automatiquement

Christopher Kermorvant*, Colin de la Higuera*, Pierre Dupont†

* EURISE, Université Jean Monnet, Saint-Etienne, France,
{kermorva,cdlh}@univ-st-etienne.fr

† INGI, Université de Louvain, Louvain-la-Neuve, Belgique,
pdupont@info.ucl.ac.be

Résumé :[1] Nous proposons deux façons d'établir des classifications préliminaires en inférence grammaticale d'automates probabilistes, dans le but de construire des modèles de langages. La première consiste à considérer des classes de mots et à inférer sur les classes de mots plutôt que sur les mots eux-mêmes ; la seconde à inférer des automates typés. Nous comparons ces approches en utilisant deux sources d'information, ou deux façons distinctes de classifier les données avant l'inférence grammaticale : un algorithme de classification non supervisée et un *part-of-speech tagger* ont été utilisés pour étiqueter de façon automatique les données soit en fonction d'information statistique, soit en fonction d'information syntaxique. Ces données étiquetées sont ensuite utilisées pour l'inférence. Les meilleurs résultats sont obtenus avec les données étiquetées statistiquement dans un automate typé. Ces résultats sont comparables à ceux obtenus par les meilleurs modèles statistiques (n-grammes) sur les données ATIS (*Air Travel Information System*).

Mots-clés : Modèles de langage, Inférence Grammaticale, Connaissances du domaine

1 Motivation

The possibility to include prior knowledge in models 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 insuffi cient. 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 fi xed set of data, prior knowledge allows to learn more complex functions.

In the specifi c context of language modeling, the importance of prior knowledge has been shown in two applications of Hidden Markov Models. Whereas mathematically

1. To appear in *Conférence d'Apprentissage*, Laval, France, July 2003.

founded methods exist to estimate the parameters of the probability distribution in these models (the EM algorithm (Dempster *et al.*, 1977)), 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 prior 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 (Rabiner & Biing-Hwang, 1993). In computational biology, prior knowledge regarding the mean length of proteins and prior distribution of amino acids is used to design the models (Durbin *et al.*, 1999).

Grammatical inference consists in learning formal grammars for unknown languages when provided with examples of strings belonging (or not) to the languages. Regular grammatical inference, in which the target grammar is supposed to be regular, has received most of the attention. There are many efficient algorithms to process regular grammars and several positive learning results are known for this class. If one is provided with positive and negative examples, algorithm RPNI (Oncina & García, 1992) can be used to infer deterministic finite automata. In the case where only positive examples are available, one may choose to learn a stochastic finite automaton: several algorithms have been proposed (Carrasco & Oncina, 1994; Stolcke & Omohundro, 1994; Thollard *et al.*, 2000) for this task.

The interest of using prior knowledge in grammatical inference is threefold. First, the search space of the deterministic finite automata inference task is well defined (Dupont *et al.*, 1994). This space depends on the learning data available, and can be extremely large. The use of prior 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, prior knowledge can complete the learning data, for example by providing implicit counter-examples: strings known not to belong to the target language. Finally, prior knowledge can introduce in the inference procedure real world constraints that the induced formal language must satisfy.

From a more general point of view, the use of prior knowledge can allow some of the rewriting rules inherent to a grammar to be in some way fed into the learning algorithm, thus allowing, as proposed in (McAllester & Schapire, 2001), to “seed the search with sufficient initial regularities”.

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 prior knowledge to be merely another point of view on the learning data.

Prior knowledge has already been used in grammatical inference by Bernard & de la Higuera (Bernard & de la Higuera, 2001) in their ILP system called GIFT [2]. 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 (Kermorvant &

2. Grammatical Inference For Terms.

de la Higuera, 2002) have proposed a framework to include prior knowledge in the automaton inference algorithms. In this framework, prior knowledge regarding sequences to be modeled can be included in automata using typing.

In this paper, we propose to infer typed automata from labeled sequencial 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 (Brill, 1992). 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 is due to the actual limitations of our typing model. The induced types are used by a classical grammatical inference algorithm, Alergia, that constructs a stochastic finite state automaton.

We compare the use of these two kinds of prior 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 is finite but huge (Dupont *et al.*, 1994): a partition lattice defined by the set of states of the prefix tree acceptor (PTA) which is the smallest tree-shaped deterministic automaton accepting exactly I_+. 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 structurally complete sample (*i.e.* a set of strings that makes use of all edges, nodes and final states of the target), this target automaton is guaranteed to belong to the 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. Evidence driven strategies have been proposed (Lang *et al.*, 1998) to cope with the difficulty of the search.

Regular language inference from real data, such as natural language sentences or biological sequences, raises additional difficulties. First, negative information is not always available. Second, real data is generally noisy. If we choose to learn stochastic 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 (Kermorvant & de la Higuera, 2002) a framework that includes prior 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 study how to incorporate background knowledge in stochastic automata inference.

2.1 Typed Stochastic Finite State Automata

We consider stochastic finite state automata (SFA), which provide a stochastic extension of finite state automata. A SFA $\mathcal{A}$ 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 SFA 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* SFA (SDFA). Second, inference algorithms explicitly searching for SDFA, such as Alergia (Carrasco & Oncina, 1994), 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

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

Note that a SFA 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 (Kermorvant & de la Higuera, 2002). A typed SFA $\mathcal{A}$ 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 SFA,
- S is a set of sorts,
- σ is a typing function which associates a single sort to each state of the automaton.

Algorithm 1 Generic SFA induction algorithm

Require:
I_+, training set (sequences)
α, a precision parameter
Ensure: a stochastic finite state automaton
A $\leftarrow$ *build_SPTA*(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)
  end if
end while
return A

A typed stochastic automaton is a stochastic automaton with typed states. There are several ways to define the typing function σ. In (Kermorvant & de la Higuera, 2002), 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 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. To be compatible with finite state automata, the labeling must obey some natural rules: it must be prefix-consistent, *i.e.* a prefix must always have a label, and regular in the sense that this label must not depend of the righthand context. It should be noticed that this is a strong condition: in most cases tags are computed by taking into account both left and righthand contexts.

Hence one can associate various types to a symbol, but only one type to a string. Furthermore this is the case for any prefix. Technically, denoting a finite set of data by I_+, and the type of the prefix u in the context of a string uv by $\sigma_{uv}(u)$, we must have: $\forall u \in I_+, \sigma(u)$ is defined, and $\forall u \in \Sigma^*$, $uv \in I_+ \wedge uw \in I_+ \Rightarrow \sigma_{uv}(u) = \sigma_{uw}(u)$.

2.2 Learning Typed Automata from Automatically Labeled Data

Several algorithms have been proposed to infer SFA from examples using frequencies (Carrasco & Oncina, 1994; Ron *et al.*, 1995; Thollard *et al.*, 2000). 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 stochastic prefix tree acceptor (SPTA). The typed SPTA 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

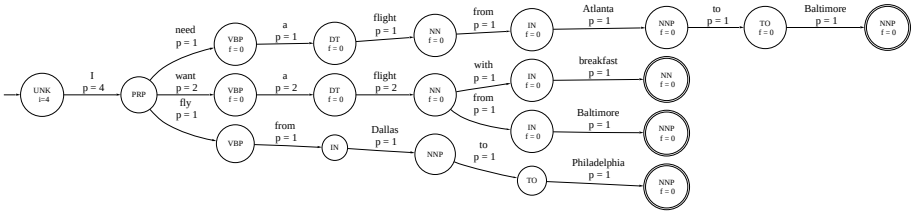

FIG. 1 – *A typed stochastic prefix tree acceptor.*

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 SPTA 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 SPTA, the label L is the type of the state q: $\sigma(q) = L$. Note that a typed SPTA cannot always be built from a given labeling since each state must have a single label and since we only consider structurally deterministic automata. A sufficient condition to be able to construct a typed SPTA from a labeling is that each prefix has always the same label in I_+. We only consider this case in the present work.

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 ./.
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 SPTA constructed from these labeled examples is presented on Figure 1.

The second step of the algorithm consists in visiting the states of the SPTA (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 the Alergia algorithm (Carrasco & Oncina, 1994), 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 (Hoeffding, 1963).

In our framework, the introduction of type constraints in the learning algorithm is straightforward. We must only consider for a merging operation states with 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 insure that the type of all the strings in the inferred language is consistent with the tagger which provided the tagged learning set.

2.3 Learning Stochastic Automata from Clustered Data

Dupont & Chase (Dupont & Chase, 1998) 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. For a given number of clusters, the clustering algorithm iteratively constructs the classes so that the average mutual information between the classes is maximized (see (Dupont & Chase, 1998) for details). 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 $\hat{\tau}(q, w) = \hat{\tau}(q, G) \cdot \hat{P}(w|G)$.

2.4 Mixed Strategies

Instead of inferring an automaton on the class labels, one can use the class information as a type to infer a typed automaton directly on symbols. On the other hand, the POS information can be used as classes instead of statistically induced clusters. This yields a total of four approaches that are summarized in Table 1. We study in the sequel the comparative performances of these four approaches.

	typed automata inference	inference on classes + expansion
POS tag	POS-typed automata	POS-class automata
Statistical clusters	Statistical-typed automata	Statistical-class automata

TAB. 1 – *Two approaches to use two different kinds of information.*

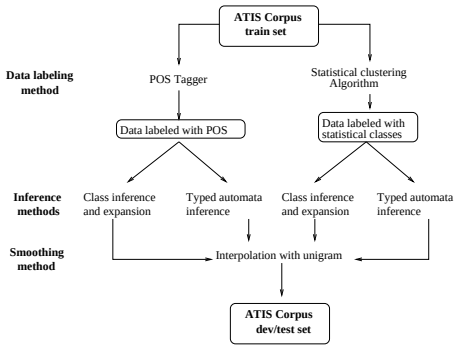

FIG. 2 – *Overview of the experimental protocol.*

3 Experiments

3.1 The ATIS Task

We have tested our approach on the Air Travel Information System (ATIS) corpus. 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 (see section 3.2) 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 stochastic automaton induction tasks (see e.g. (Dupont & Chase, 1998); (Thollard *et al.*, 2000); (Llorens *et al.*, 2002)).

3.2 The Experimental Setting

We compare the different approaches presented in Table 1 following the experimental protocol described in Figure 2.

All the inferred automata are evaluated on the ATIS test set. The usual quality measure in language modeling tasks is the average (per symbol) 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*:

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

The smaller the perplexity the better the automaton can predict the next symbol. 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 defi nes 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:

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

This smoothing technique is very rudimentary but, as such, it best reflects the quality of the induced SFA alone. As some words of the application vocabulary may not occur in the training set[3], the unigram probability itself is smoothed by absolute discounting (Ney *et al.*, 1994). Let $C(w)$ denote the count of word w in the training set containing a total of N tokens. The smoothed unigram probability is defi ned 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:

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

The ATIS validation set is used to tune the learning parameters:

- 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 the case of POS tagging, the number of distinct types is defi ned *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 SFA). The parameter β controls the weight of the induced SFA in the combined smoothed automaton.

The POS information was obtained by tagging the training set using the Brill tagger (Brill, 1992). As a fi rst 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 (Vergne & Giguet, 1998). 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 (Dupont & Chase, 1998). Values for the number of clusters ranging from 10 to 1000 have been tested.

3. This is the case for 131 out of 1,296 symbols.

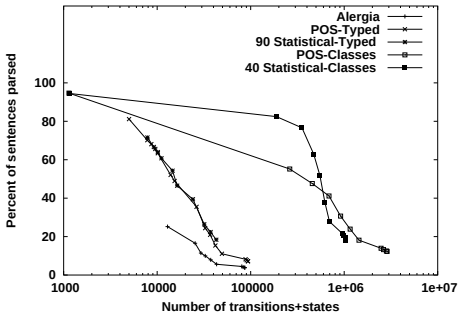

FIG. 3 – *Percentage of sentences correctly parsed in the validation set versus the number of parameters of the automaton*

3.3 Results

The influence of the various learning parameters is first evaluated on the validation set. Final results are reported next 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 (Dupont & Chase, 1998)). It has been observed experimentally that the test set is closer to the training set than the validation set is.

Figure 3, 4 and 5 compare the four selected methods on the ATIS validation set. For the methods based on statistical classes, the results are given for the optimal number of classes. In the case of typed automata inferred on statistically labeled data, the optimal number of classes is 90. In the case of automata inferred at the class level and expanded to words, the optimal number of classes is 40, as in (Dupont & Chase, 1998). In all cases, the results are given for the inference parameter α that was observed to be optimal for this particular setting.

Figure 3 shows the percentage of sentences from the validation set fully parsed by the inferred automaton for the compared techniques with respect to the number of parameters of the automaton (number of states and number of transitions). In this case, no smoothing is used. The typed SPTA 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 four methods increase the number of sentences that can be parsed compared to standard Alergia inference. However, the number of parameters of the automata inferred by class expansion based methods is significantly higher than the number for typed automata. Summarizing, class expansion based techniques generalize more but create less compact automata than type-based methods.

Figure 4 presents the perplexity obtained by the inferred 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

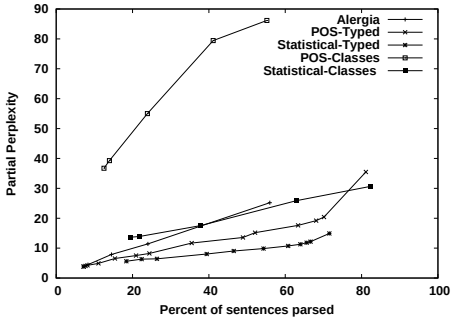

FIG. 4 – *Perplexity of the sentences correctly parsed in the validation set*

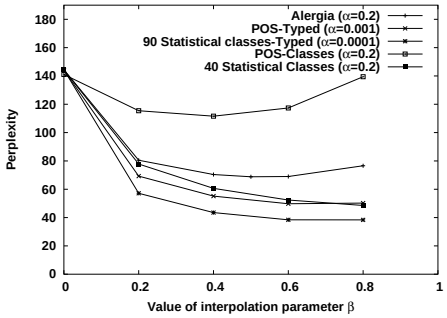

FIG. 5 – *Perplexity on the ATIS validation set: inferred automaton with unigram interpolation.*

of parsed sentences, both the POS-based and statistical-classes-based typed automata yield a smaller perplexity. On the contrary, inferring on POS classes gives expanded automata with higher perplexity. For a given number of fully parsed sentences, 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 5 shows the perplexity obtained by the inferred 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 $\beta = 0.8$. Table 2 summarizes the test set perplexity values for the four methods.

3.4 Improved smoothing techniques

The smoothing technique used in the evaluations described in section 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 SFA alone. Howe-

	typed automata inference	class inference + expansion
POS tag	57	112
Statistical classes	42	52

TAB. 2 – *Best perplexity results on the ATIS test set with interpolation to unigram for the four inference methods.*

ver, if the objective is to minimize test set perplexity, more sophisticated smoothing techniques are required.

The best trigram model on the ATIS task is obtained with Kneser-Ney back-off smoothing (Kneser & Ney, 1995). 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 (but preliminary) 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 (Llorens *et al.*, 2002)) 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 (Dupont & Chase, 1998) that the use of statistical class information improves the quality of stochastic automata used as language models. The present work illustrates that this is even more true when statistically induced classes are combined with typed SFA 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 (Charniak, 2001). 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 SPTA, which is a deterministic SFA as explained in section 2.2. Extensions of the present approach to infer (possibly) non-deterministic structures could lead to further improvements.

5 Conclusion

We have proposed a way to use prior 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 stochastic automata inference in the framework of natural language modeling. Part-of-speech labeling provided by a POS tagger and statistical clustering of words have been compared as labeling for the data. The use of statistical word classes information when inferring typed automata provides models which are competitive with state-of-the-art n-grams with similar smoothing techniques while being more compact and needing less parameters.

Références

BERNARD M. & DE LA HIGUERA C. (2001). Apprentissage de programmes logiques par inférence grammaticale. *Revue d'Intelligence Artificielle*, **14**(3/4), 375–396.

BRILL E. (1992). A simple rule-based part-of-speech tagger. In *Proceedings of the Conference on Applied Natural Language Processing*, p. 152–155.

CARRASCO R. C. & ONCINA J. (1994). 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*, p. 139–152: Springer.

CHARNIAK E. (2001). Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, p. 116–123.

DEMPSTER A. P., LAIRD N. M. & RUBIN D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, **39**, 1–38.

DUPONT P. & CHASE L. (1998). Using symbol clustering to improve probabilistic automaton inference. In *Proc. Int. Coll. on Grammatical Inference*, volume 1433 of *Lecture Notes in Artificial Intelligence*, p. 232 – 243: Springer.

DUPONT P., MICLET L. & VIDAL E. (1994). What is the search space of the regular inference? In *Proc. Int. Coll. on Grammatical Inference*, volume 862 of *Lecture Notes in Artificial Intelligence*, p. 25–37: Springer.

DURBIN R., EDDY S., KROGH A. & MITCHISON G. (1999). *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

HOEFFDING W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, **58**(301), 13–30.

KERMORVANT C. & DE LA HIGUERA C. (2002). Learning languages with help. In *Proc. Int. Coll. on Grammatical Inference*, volume 2484 of *Lecture Notes in Artificial Intelligence*, p. 161–173: Springer.

KNESER R. & NEY H. (1995). Improved backing-off for N-gram language modeling. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, p. 181–184.

LANG K., PEARLMUTTER B. & PRICE R. (1998). Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In SPRINGER, Ed., *Proc. Int.*

Coll. on Grammatical Inference, volume 1433 of *Lecture Notes in Artificial Intelligence*, p. 1–12.

LLORENS D., VILAR J. & CASACUBERTA F. (2002). Finite state language models smoothed using n-grams. *International Journal of Pattern Recognition and Artificial Intelligence*, **16**(3), 275–289.

MCALLESTER D. & SCHAPIRE R. (2001). Learning theory and language modeling. In G. LAKEMEYER & B. NEBEL, Eds., *Proc. Int. Joint Conf. on Artificial Intelligence*: Morgan Kaufmann.

NEY H., ESSEN U. & KNESER R. (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, **8**, 1–38.

ONCINA J. & GARCÍA P. (1992). Identifying regular languages in polynomial time. In H. BUNKE, Ed., *Advances in Structural and Syntactic Pattern Recognition: Proc. of the International Workshop*, p. 99–108. World Scientific.

RABINER L. & BIING-HWANG J. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

RON D., SINGER Y. & TISHBY N. (1995). On the learnability and usage of acyclic probabilistic finite automata. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, p. 31–40: ACM Press.

STOLCKE A. & OMOHUNDRO S. (1994). Inducing probabilistic grammars by bayesian model merging. In *Proc. Int. Coll. on Grammatical Inference*, p. 106–118: Springer.

THOLLARD F., DUPONT P. & DE LA HIGUERA C. (2000). Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. Int. Conf. on Machine Learning*, p. 975–982: Morgan Kaufmann.

VERGNE J. & GIGUET E. (1998). Regards théoriques sur le tagging. In *Proceedings of the conference Le Traitement Automatique des Langues Naturelles*.