Inducing SFA with ϵ-Transitions Using Minimum Description Length

Yasemin Altun **Mark Johnson**

Cognitive and Linguistic Sciences
Brown University

Email: Yasemin_Altun@brown.edu
Mark_Johnson@brown.edu

1 Introduction

Induction of the description of a language can be defined as a problem that consists of two parts: learning the structure and learning the parameters of that structure. Considering the difficulty of the problem, most of the studies in this field focus on inducing the parameters of a given structure, as opposed to inducing the structure itself. My project aims to address the more challenging part of the problem, the induction of the structure. I developed an algorithm for the induction of a Stochastic Finite State Automaton (SFA) with ϵ-transitions for a regular language which takes as input a set of strings from that language. Working within a Bayesian framework, my algorithm, whose structure is similar to (Stolcke and Omohundro, 1994), induces the SFA using Minimum Description Length (MDL) Principle for model selection.

The automata with ϵ-transitions capture some linguistic generalities that the automata without ϵ-transitions fail to capture. They are also more readable and simpler. The automata of the English auxiliary system with and without ϵ-transitions are good examples that demonstrate these properties. The MDL framework is a natural setting of the induction of automata with ϵ-transitions. I applied the algorithm to the problems of the induction of SFA of the English auxiliary system and of Turkish morphology. The results show that in the MDL framework, inducing SFA with ϵ-transitions requires less data and also decreases the number of learning steps.

2 Minimum Description Length and Learning ϵ-Transitions

Viewing induction as a search problem and using a Bayesian framework, an automata inducer aims to find the automaton that maximizes a combination of the prior probability of the automaton and the probability of the data set given the automaton. One can use the Minimum Description Length principle, which has cognitive plausibility, to define a probability distribution on the set of possible automata, such that simpler machines are more probable than more complex ones. The following objective function, in Bayesian terms, expresses the problem:

$$a = \arg\min_{a \in A} (\log p(O|a) + \alpha \log p(a)) \qquad (1)$$

where a is stochastic finite state automaton, A is the set of possible automata, O is the observed data, and α is a factor comprimising between the size and the accuracy of the automaton. The prior probability is assigned according to the description length l of a:

$$p(a) = \exp^{-l(a)}$$

Using the Viterbi assumption, the likelihood is calculated as follows:

$$p(O|a) = \prod_{i}^{N} \theta_i^{w_i}$$

where θ_i is a parameter of a (the transition or ending probability) and w_i is the evidence from O related with the parameter θ_i. The MDL principle defines only an approximate, not an exact prior, since a plausible inductive system should not be too sensitive to the encoding scheme. We represent this observation using the α parameter in Equation 1, which effectively allows us to adjust the weight given the prior.

Automata with ϵ-transitions can be exponentially smaller than equivalent automata without ϵ-transitions that accept exactly the same language. If the training data is in fact generated by a small automaton with ϵ-transitions, one might expect that a learner that can posit automata with ϵ-transitions may succeed when a learner that cannot posit ϵ-transitions fails. Automata with ϵ-transitions also capture some linguistic generalities that automata without ϵ-transitions fail to capture. The MDL principle, which prefers simpler automata to more complex ones favors the automata with ϵ-transitions. Therefore, the MDL framework is a very natural setting of the problem of the induction of automata with ϵ-transitions.

3 The Algorithm

The algorithm first builds the tree-structured automaton a that generates all and only the strings in the data set with the probabilities equal to their relative frequencies and no other strings. Every state q_i has the record of the number of strings in O arriving q_i, the number of strings ending in q_i and the number of strings traversing each arc of q_i. The algorithm repeatedly searches for the best generalization of a. Merging two states of a and their children recursively is one generalization technique. Another one is inserting an ϵ-transition from one state to another and merging their children recursively. Both of these techniques results in smaller machines that may produce more (in some cases infinite number of) strings that are not seen in the data set. These machines are evaluated using the objective function, given in (1), where the description length of a automata is a simple function of the number of states and arcs of a. The likelihood of the data is calculated using the counts recorded in every state (the number of strings entering the states, ending in the states, and traversing the arcs of the states). So, the algorithm does not need to parse the observed data every time, but just uses the recorded counts. It, then, selects the best generalization of a and repeats the search process if the generalized machine performs better in terms of the objective function than a. The algorithm stops the search otherwise.

Figure 1 and Figure 2 present examples of merging states and inserting an ϵ-transition re-

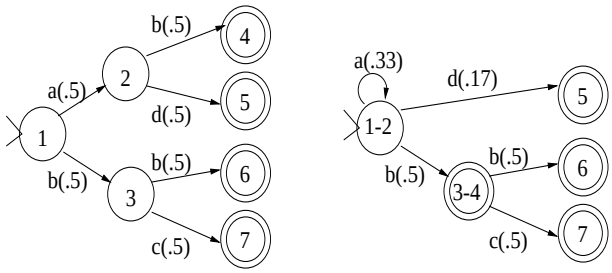

Figure 1: State merging example. Merge State1 and State2

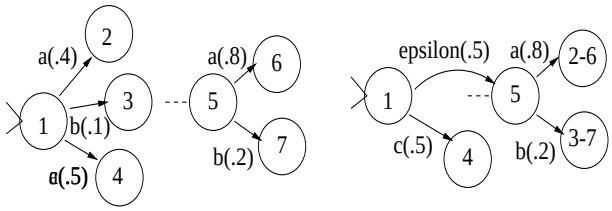

Figure 2: Inserting ϵ-transition example. From State1 to State5

spectively. The algorithm merges states q_i and q_j, when the probability distribution of the strings generated by q_i and q_j are similar enough that the description length of the system decreases. The algorithm inserts an ϵ-transition from q_i to q_j, when the probability distribution of the strings generated from some arcs of q_i is very similar to the probability distribution of the string generated from q_j, however the probability distribution of the strings generated from the rest of the arcs of q_i is so different than the rest that if q_i and q_j are merged, the description length of the system increases. One can think of ϵ-transition insertion as merging some arcs of q_i with the arcs of q_j and merging the states that are reached by the merged arcs.

When merging states q_i and q_j, the algorithm simply adds the related counts recorded in q_i and q_j and calculates the relative frequencies. When inserting an ϵ-transition from q_i to q_j, the algorithm inserts an ϵ-transition with the count equal to the sum of the counts of the merged arcs of q_i and increment the related counts of q_j accordingly.

When the search space is intractable, the algorithm prunes the search space by restricting attention to states that are close to the start state. The rationale is that operations on the states that are far from the start state do not improve the objective function as much as the

states that are closer to the start state. Since the part of the machine under these states are much smaller that the part of the machine under the states that are closer to the start state, one wouldn't gain much in terms of the objective function.

During the calculation of the likelihood of the data, the algorithm makes a Viterbi assumption and uses the stored counts as opposed to parsing the observed data over and over again. Viterbi assumption is crucial for the algorithm to be intractable. In each learning step n^2 automata are generated and evaluated, where n is the number of states. Evaluation of the automata involves calculating the likelihood of the observed data given the automaton. Parsing the data over and over again is unaffordable. The algorithm uses the counts stored in the states to calculate the likelihood. The Viterbi assumption holds for the initial automaton and continues to hold for the generalizations of it if the possible generations are deterministic. For this reason, no nondeterminism is allowed in the induced machines. Since after ϵ-transition insertion, the child states are merged, there is always one possible path for each string.

4 Applications

4.1 English Auxiliary System

English auxiliary system can be summarized as (Chomsky, 1957):

AUX $\rightarrow$ (Modal) (have) (be) (be) VERB

Any or all of the auxiliary components preceding the verb can be omitted.

I extracted out the uninflected form of the auxiliary sequences from sections 2-21 of the Penn TreeBank corpus and generated the SFA that would describe this data best. I generated 1000 random corpora of different sizes of 1000 to 12000 each and presented the algorithms that learn SFA with ϵ-transitions and that learn SFA without ϵ-transitions. The results show that the SFA with ϵ-transitions is accurately induced with small data sets (data sets of size 3500), whereas the algorithm that learns SFA without ϵ-transitions needs more data (data sets of size of 9500) to converge to an accurate SFA. The accuracy of the induced automata is measured by the likelihood ratio test, where the null hypothesis is that the languages generated by the two machines, the induced one and the one that we started off with, have the same underlying distribution. The algorithm that learns SFA with ϵ-transitions also requires fewer generalization step, whereas learning the SFA without ϵ-transitions require around twice more steps.

4.2 Turkish Morphology

Turkish morphology defines a regular language that has an infinite number of strings and the FSA of this system is much more complex than the FSA of English auxiliary system (Oflazer et al., 1994). I used a segmented and labelled Turkish newspaper articles corpus which was morphologically analyzed by (Hakkani-Tur et al., 2000) to induce the SFA of Turkish morphology system. I presented the algorithms that learn SFA with ϵ-transitions and that learn SFA without ϵ-transitions different sizes of corpora (Figure 3). The algorithm that learns SFA with ϵ-transitions requires a training set of 240000 words to induce the correct automata, whereas the algorithm that learns SFA without ϵ-transitions requires a training set of 720000 words [1]. The algorithm that learns SFA with ϵ-transitions takes less learning steps than the algorithm that learns SFA without ϵ-transitions (Table 4) and the difference increases as the training data size increases.

5 Related Work

The use of Bayesian framework approach dates back to late 60's (Horning, 1969). Chen performs a model splitting approach using MDL Principle when inducing a probabilistic context-free grammar (Chen, 1996). Stolcke and Omohundro use model merging, which is a common technique used in automata theory (Hopcroft and Ullman, 1979), for induction and apply it to regular and context free languages (Stolcke and Omohundro, 1994). Carrasco and Oncina use a statistical test in model merging and apply it to regular languages (Carrasco and Oncina, 1994).

[1] The difference between the set of strings generated by the SFA with ϵ-transitions induced with 240000 words and the set of strings generated by the SFA with ϵ-transitions induced with 1200000 words is not statistically significant. This is also true for the set of strings generated by the SFA with ϵ-transitions induced with 360000 words and the set of strings generated by the SFA with ϵ-transitions induced with 1200000.

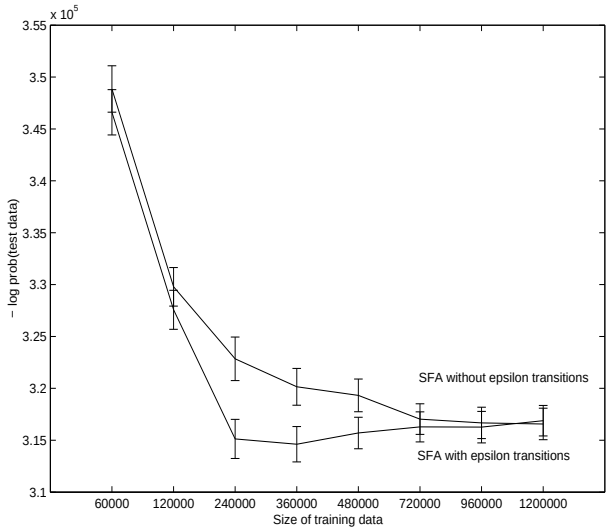

Figure 3: The negative loglikelihood of the test data wrt SFA with and without ϵ-transitions

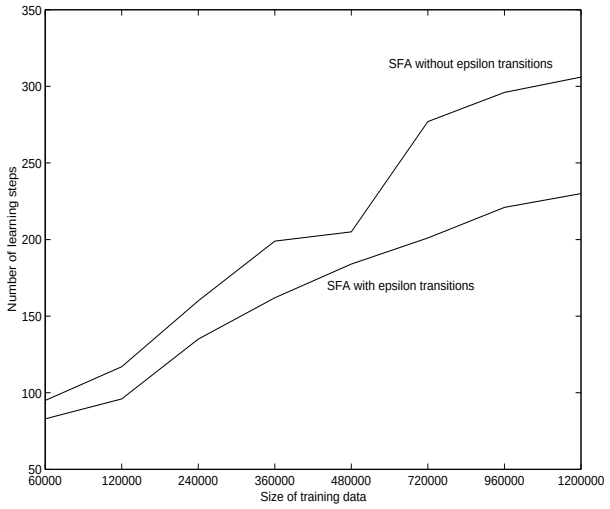

Figure 4: The number of learning steps for each training data size

According to my knowledge learning SFA with ϵ-transitions has not been studied before.

Inducing the finite state automaton of the English auxiliary system has been studied previously (Pilato and Berwick, 1985). In this study, the algorithm makes use of the $k-reversibility$ of the English auxiliary system to constraint the induction problem. The induced automaton is larger than the automaton induced by the algorithm presented above, since it does not have ϵ-transitions. According to my knowledge, learning the SFA of Turkish morphology defined as a problem of learning both the structure and the parameters has not been studied before. However, it has been studied as a problem learning the parameters of a given structure (Hakkani-Tur et al., 2000).

6 Conclusion

In this study, I developed an algorithm to induce a stochastic finite state automaton, with ϵ-transitions for a regular language, which takes as input a set of strings generated from the language. One can induce SFA with ϵ-transitions within the MDL framework very naturally. The results show that the algorithm that learns automata with ϵ-transitions requires less data and fewer learning steps than the algorithm that learns SFA without ϵ-transitions requires.

References

R. C. Carrasco and J. Oncina. 1994. Learning stochastic regular grammars by means of a state merging method. In *International Conference on Grammatical Inference.*

S. Chen. 1996. *Building Probabilistic Models For Natural Language.* Ph.D. thesis, Harvard University.

C. Chomsky. 1957. *Syntactic Structures.* The Hague.

D. Z. Hakkani-Tur, K. Oflazer, and G. Tur. 2000. Statistical morphological disambiguation for agglutinative languages. In *Proceedings of the 18th International Conference on Computational Linguistics.*

J.E. Hopcroft and J.D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley.

J.J. Horning. 1969. A study of grammatical inference. Technical report, Computer Science Department, Stanford University.

K. Oflazer, E. Gocmen, and C. Bozsahin. 1994. An outline of turkish morphology. Technical report, Computer Engineering Department, METU, Turkey.

S. F. Pilato and R. C. Berwick. 1985. Reversible automata and induction of english auxiliary system. In *Proc. of the 23th ACL*, pages 70–75, Chicago, IL.

A. Stolcke and S. Omohundro. 1994. Inducing probabilistic grammars by bayesian model merging. In *International Conference on Grammatical Inference.*