Learning Regular Languages from Positive Evidence

Laura Firoiu[†] [1] **(lfiroiu@cs.umass.edu)**
Tim Oates[†] [2] **(oates@cs.umass.edu)**
Paul R. Cohen[†] **(cohen@cs.umass.edu)**
[†] Computer Science Department, LGRC, University of Massachusetts, Box 34610
Amherst, MA 01003-4610

Abstract

Children face an enormously difficult task in learning their native language. It is widely believed that they do not receive or make little use of negative evidence (Marcus, 1993), and yet it has been proven that many classes of languages less powerful than natural languages cannot be learned in the absence of negative evidence (Gold, 1964). In this paper we present an approach to learning good approximations to members of one such class of languages, the regular languages, based on positive evidence alone.

1. Introduction

The ability to communicate through spoken language is widely regarded as the hallmark of human intelligence. Children acquire their native tongue with remarkable ease, mastering the vast majority of that language before they enter school. However, the facility with which children acquire language belies the complexity of the task. For example, children clearly receive positive evidence (examples of sentences in the language), but it is widely believed that children do not receive negative evidence (examples of sentences that are not in the language and that are somehow marked as such) (Marcus, 1993). The difficulty with respect to learnability arises with a now famous theorem due to Gold(1967). He proved that several classes of languages, including regular, context free and context sensitive, can be identified in the limit when the learner has access to both positive and negative evidence. However, those same classes of languages cannot be learned from positive evidence alone. How do children overcome Gold's theoretical hurdle?

Difficulties such as the one above led Chomsky(1975) to suggest that language is innate, that it is not learned per se but that facility with language grows and matures much in the same way that one's organs are genetically predetermined to grow and mature. In this paper we explore the possibility that language is not innate by developing algorithms for learning good approximations of regular languages from positive evidence alone. Although no natural language is strictly regular, large subsets of natural languages are regular, and this class of languages is the simplest one covered by Gold's theorem. As such, it seemed like a good place to start our investigations into learnability.

Our approach to learning regular languages begins with a class of languages, called Szilard (Makinen, 1997), that can be learned from positive evidence alone. Given examples of sentences generated by an arbitrary regular language, we assume that the language is Szilard, yielding a representation of the language that simply "memorizes" the input and does no generalization. We then apply heuristic techniques to create increasingly more compact representations that maintain the Szilard property and that become better approximations to the target language.

The remainder of the paper is organized as follows. Section 2 briefly reviews deterministic finite automata, their relationship to regular grammars, and how a Szilard language can be learned from positive evidence alone. Section 3 explains our algorithm for learning a good approximation of an arbitrary regular language from positive evidence. Section 4 describes the implementation of the algorithm and Section 5 presents experiments and results. Finally, Section 6 concludes and points to future research directions.

2. Finite automata, trivial DFAs and Szilard regular grammars

Deterministic finite automata(DFA) are well known, simple computing devices that are described by the tuple:
⟨Set_of_States, Alphabet, Transition_Function
Start_State, Set_of_Final_States ⟩

The finite automata are equivalent to regular grammars and thus recognize precisely the class of regular languages. Their functioning is described by a transition function:
$\delta(current\_state, current\_input\_symbol) = next\_state$
which is equivalent to a set of productions:
$\{current\_state \rightarrow current\_input\_symbol \;\; next\_state\}$.
The input symbols are elements of the alphabet and are usually called terminals. A DFA can be visualized by its associated graph, as in figure 1: the nodes are the states and the arcs represent the transitions. The labels on the arcs are the input symbols. A sentence is said to be accepted by the au-

[1]This research is supported by DARPA/AFOSRF and DARPA under contracts No. F49620-97-1-0485 and No. N66001-96-C-8504. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the or the U.S. Government.

[2]This research is supported by a National Defense Science and Engineering Graduate Fellowship.

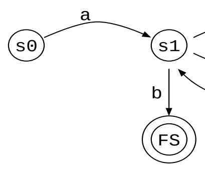


Figure 1: A small deterministic finite automaton.

tomaton if there is a path from the start state to a final state, labeled with the words of the sentence. DFA induction is the problem of finding the automaton that describes a given language from a set of examples: sentences whose membership to the language is known. Gold's famous theorem on language learnability states that any class of regular languages (and other languages) can be learned from positive and negative examples, but not from positive examples only, if at least one infinite language exists in the class.

Any finite language can be represented by a *trivial DFA*, that has a distinct state for each word occurrence in each sentence, meaning that the learner memorizes all the sentences.

A Szilard regular grammar has the property that its productions have the form $A_i \rightarrow a_{ij}A_j$, meaning that each terminal appears on one arc only. It follows that the number of terminals equals the number of productions. The number of states (without the start state) is at most equal to the number of productions. The inference algorithm that finds a Szilard grammar from a set of positive examples, is:

- assume that the number of states is equal to the number of productions; associate each state with a terminal a and name it accordingly, A: $A \rightarrow aX$, where "X" stands for the unknown next state; if a terminal is the first one in a sentence, then its associated state is the start state; if it is the last one in the sentence, it is followed by the final state.
- starting from the start state, follow the derivations for the given examples and merge the states which follow the same terminal; continue the process until a Szilard grammar is obtained.

The algorithm finds the target DFA, provided that it sees all the possible consecutive transitions $\langle a_i, a_j\rangle$. For example, the DFA in figure 1 can be learned from three examples {aceb, adeb, ab}, even if its language is infinite. The first step yields the state sequences: ⟨S0 C E B FS⟩ , ⟨S0 D E B FS ⟩ and ⟨S0 B FS⟩. Because the states B, C and D all follow the terminal "a", they are merged into state ⟨BCD⟩ and the desired DFA is obtained. An even simpler grammar which can be inferred from positive examples only has the property that each terminal uniquely identifies the next state. The inference algorithm is immediate:

- assign one variable A to each terminal a to obtain productions of the type $X \rightarrow aA$, where "X" is the unknown current state;
- from the start state, for a sequence ⟨ a b . . . ⟩, recover the productions: S0 →a A, A →b B, . . .

Through an abuse of notation, we will denote this second grammar Szilard* and it and the first one collectively Szilard. It can be noticed though, that the Szilard regular languages are included in the Szilard* regular languages, because the property that a terminal uniquely identifies a transition between a pair of states implies that it also identifies the next state.

We do not know if a natural language or at least a part of it can be described by a Szilard grammar, but it can be regarded as such for the purpose of learning a grammar from positive examples. The trivial DFA for a set of examples has the Szilard property if each terminal occurrence is considered distinct. For example, in the sentences {"the_1 boys see the_2 cat", "the_3 girls walk"} the three occurrences of "the" are considered different. Unfortunately this does not solve the problem yet, because we got the desired Szilard property, but the trivial DFA is not the language representation we want to learn. The two sentences above suggest that actually not all occurrences of "the" should be deemed different. While "the_1" and "the_2" are different, one determining the noun in the subject and the other in the direct object, "the_1" and "the_3" both determine the subject noun and so should belong to the same terminal. It follows that we need an algorithm that partitions word occurrences into classes that can be associated with grammar terminals.

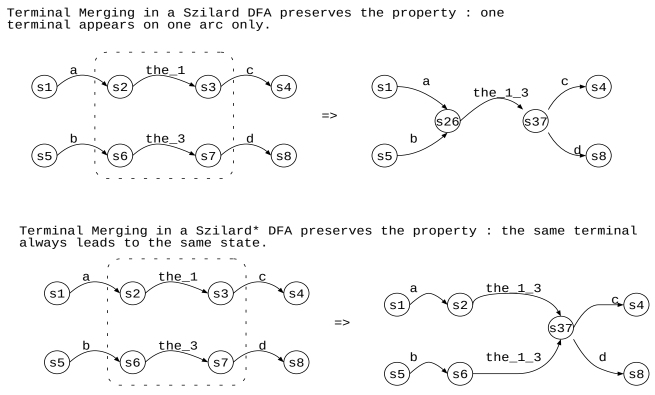

Terminal Merging in a Szilard DFA preserves the property : one terminal appears on one arc only.
s1
a
s2
the_1
s3
c
s4
s5
b
s6
the_3
s7
d
s8
=>
s1
a
s26
the_1_3
s37
c
s4
s5
b
d
s8
Terminal Merging in a Szilard* DFA preserves the property : the same terminal always leads to the same state.
s1
a
s2
the_1
s3
c
s4
s5
b
s6
the_3
s7
d
s8
=>
s1
a
s2
the_1_3
s37
c
s4
s5
b
s6
the_1_3
d
s8

Figure 2: Terminal merging

An interesting property of the Szilard regular grammars is that if we conflate two terminals and then merge their associated states, the grammar remains Szilard. As can be seen in figure 2, if we merge the terminals "the_1" and "the_3" into one terminal "$the_{1,3}$" and the states $\{s_2, s_6\} \rightarrow s_{2,6}$, $\{s_3, s_7\} \rightarrow s_{3,7}$ for the Szilard DFA, the resulted automaton retains its defining property, namely that each terminal appears on just one arc. The Szilard* property is also preserved if the two terminals "the_1" and "the_3" and the states $\{s_2, s_6\}$ are merged. This leads to the following strategy for addressing the problem of grammar induction, by breaking it into two subproblems:

- "Szilard-ify" the language by immediately constructing the trivial DFA.
- "compact" the trivial DFA, by merging word instances, and the associated states.

3. Approach and Algorithms

It follows that a device is needed that can distinguish word occurrences up to classes associated with the grammar's terminals. From these classes, a Szilard regular grammar can then be inferred, in the form of a deterministic finite automaton (DFA). It was reported in (Elman, 1990), that an Elman-type recurrent neural network(*rnn*) can classify the word instances in a partition that reflects grammatical categories. We will use this device to extract representations of word instances which are suitable for the task of identifying terminal classes. These representations will then serve as input to a DFA extraction algorithm. The generic Elman-type *rnn* is presented in figure 3: the hidden layer encodes the current network state and the context (recurrent) units maintain a copy of the previous state. The output layer encodes the probability distribution of the next word for the current input symbol. The network as a dynamical system is described by the state function F and its output is given by the function O:

$$next\_state = F(current\_input\_symbol, current\_state)$$
$$\mathcal{P}(next\_word|current\_word) = O(current\_state)$$

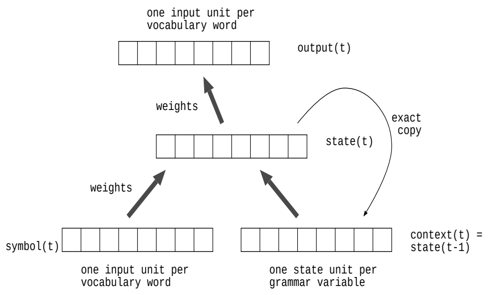

one input unit per vocabulary word
output(t)
weights
state(t)
exact copy
weights
symbol(t)
context(t) = state(t-1)
one input unit per vocabulary word
one state unit per grammar variable

Figure 3: Elman recurrent network.

The *rnn* state function F is similar with the transition function of a finite automaton, and so it appears to be convenient to draw a one-to-one correspondence between *rnn* and DFA states, thus solving the problem of grammar induction. Unfortunately this is not immediately possible, the main reason being that the network states belong to a continuous space, while the DFA states are discrete. One method used so far for DFA inference from *rnn*(Giles et al. 1992) is to assume that the network states which are close in the continuous space form clusters that represent the automaton states. This procedure is immediately equivalent to extracting the Szilard* DFA, where the terminal classes uniquely identify the states. Both positive and negative examples were used in the mentioned work. Kolen(1994) argued that the network states cannot be mapped directly onto the DFA states because of an instability of the dynamic system represented by the recurrent network, that makes the DFA state encoding in the hidden layer shift in time. We can address this problem by resetting the network before the beginning of each sentence. Another problem is that of local minima: due to its huge parameter space, the network can use similar state vectors for either the same or different DFA states and still encode the desired output.

Since our goal is to extract grammatical categories from the network, we will look at ways of distinguishing word instances, based on the *rnn* states, rather than trying directly to extract DFA states. The DFA can be extracted afterwards, due to the assumed Szilard property. If we consider each word occurrence to be represented by the hidden layer as in (Elman, 1990), we might lose the information encoded in the previous state. The example in figure 4 illustrates how two different words can get similar representations, because of the constraints imposed on the hidden layer by the output function: $(h = F(see, A_i)) \approx (h' = F(sees, A_j))$ because $O(h)$ must equal $O(h')$, so "see" and "sees" can be clustered together.

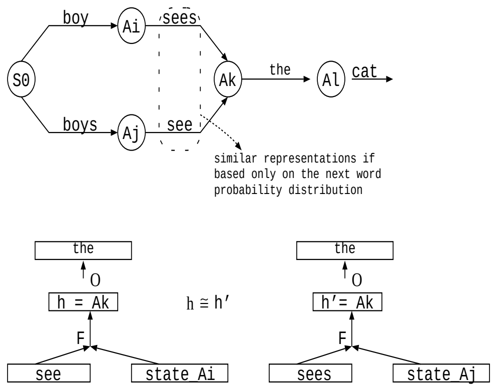


Figure 4: How "see" and "sees" can get the same representation. The weights can encode a function F such that $F(see, A_i) \approx F(sees, A_j)$

This problem can be viewed as a miss-representation of the DFA states in the hidden layer. It turns out that states, and thus word occurrences, can be better distinguished by the different paths that lead to them. A simple way of achieving this is by considering the concatenation of the context layer (previous network state) with the hidden layer (current state) as the representation of word instances. Even so, the problems of non-discrete and eventually falsely similar vectors of word occurrences remain. It follows that two other processing steps are needed:

- network state clustering, using the Euclidian distance, for detecting the potentially similar word occurrence representations
- merging the word instances considered similar by the previous step, but only if they also conform to a criterion other than their vector distance

A *distributional criterion*, albeit weak, that allows discrimination of word instances, is the probability distribution of the previous and next word occurrences. While the representations of words reflect local information (consecutive states in a dynamical system), the probability distributions carry global information that spans sentences.

The overall processing is:

1. obtain vector representations of word occurrences, using an Elman-type *rnn*
2. hierarchically cluster the word instances, using the Euclidian distance, obtaining a binary tree
3. create initial classes of words from the leaves of the tree belonging to the same subtree, at a certain low level in the tree
4. "climb" the tree only if the two children of the current internal node represent two classes that can be merged according to the distributional criterion
5. extract the DFA from the classes of terminals obtained at the previous step, by considering that the target grammar is Szilard

4. Implementation

4.1. The language

We wrote a small context free grammar (CFG), similar to the one used in (Elman, 1992). From this CFG we obtained a regular grammar by expanding the start symbol with all possible productions, up to an arbitrary depth in the derivation trees. This regular grammar can generate only a subset of the original language. Furthermore, the regular grammar is used to generate sentences no longer than a chosen length. The target of our learning system is this finite regular language, which is exhaustively presented to the learner. The fact that the language is finite has no influence on the learning algorithm, which neither builds the trivial DFA, nor does it assume that it has seen the entire language or that the language is finite.

The initial grammar we used is given in figure 5. The grammar encodes no context constraints, so it can generate sentences like "John hears John", which are unlikely from the semantic point of view. We are not concerned here with the semantic content of words, but want to test that the two occurrences of John, which are syntactically diferrent, are considered so by the learning system.

4.2. The recurrent network

For the experiments we used the package "tlearn" (Plunkett & Elman, 1997). We used one input and one output unit, respectively, for each word in the language, as in (Elman, 1990). An extra output unit encoded the ⟨End_of_Sentence⟩ marker. The number of hidden units was equal to the number of states in the target DFA. The network was trained on the classification problem: predict the next word or ⟨End_of_Sentence⟩. We used the cross-entropy function as the error function. The cross-entropy function, when applied to classification tasks, was shown by Rumelhart et al.(1993) to make a network learn the probability distribution over the output vectors. The network state was reset before the start of each sentence, in order to avoid the instability phenomenon mentioned in (Kolen, 1994). In our experiments this training regime gave the best results in terms of word clustering.

For small languages we inspected the output units and as expected, they encoded a close approximation of the probability distribution over the next words. For larger languages, the approximation became less accurate.

S	→	NP_hs VP_hs . | NP_hp VP_hp .
S	→	NP_as VP_as . | NP_ap VP_ap .
NP_hs	→	the {boy, girl} | {John, Mary}
NP_hs	→	the {boy, girl} RC_hs | the {boy, girl} RC_hs
NP_hp	→	{John, Mary} and {John, Mary}
NP_hp	→	the {boys, girls} RC_hp
VP_hs	→	chases OBJ | feeds OBJ
VP_hs	→	{walks, lives} | {sees, hears}
VP_hp	→	{see, hear} OBJ
VP_hp	→	chase OBJ | feed OBJ
VP_hp	→	{walk, live} | {see, hear}
OBJ	→	NP_hs | NP_hp | NP_as | NP_ap
NP_as	→	the {cat, dog} | the {cat, dog} RC_as
NP_ap	→	the {cats, dogs} | the {cats, dogs} RC_ap
VP_as	→	{walks, lives} | {sees, hears}
VP_as	→	chases OBJ | {sees, hears} OBJ
VP_ap	→	{walk, live} | {see, hear}
VP_ap	→	chase OBJ | {see, hear} OBJ
RC_hs	→	who VP_hs RC_hp → who VP_hp
RC_as	→	who VP_as RC_ap → who VP_ap

Figure 5: hs stands for human_singular, as stands for animal_singular, hp stands for human_plural and ap stands for animal_plural

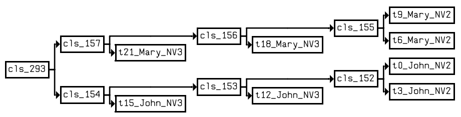
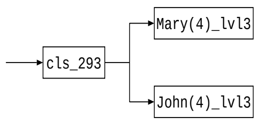


Figure 6: Fragment of initial tree

4.3. The merging algorithm

The algorithm relies on the initial clustering of word instances, based on their vector representation. This vector was obtained by concatenating the context and hidden layers in the network. We used a simple hierarchical clustering algorithm that yields a binary tree. The word instances that have almost identical vectors are placed by the algorithm, in subtrees at low levels in the tree, as illustrated in figure 6. These subtrees form the initial classes of word instances.

After the initialization stage the tree can be viewed as in figure 7. The merging algorithm proceeds then by trying to merge classes of words associated with sibling nodes in the



There are 4 instances of "Mary" with identical vectors. All were placed in a subtree at the third level from the fringe and now form one class. The same holds for "John".

Figure 7: Fragment of the tree after the initialization step.

tree. For the example in figure 7, the two classes “John” and “Mary” are proposed. The criterion for merging is the similarity of their probability distributions over the next and previous word classes. That is, “John” and “Mary” are merged into one class if they tend to be preceded and followed by the same word classes. We use the G statistic, (Cohen, 1995) to test if there is a statistically significant difference between the two probability distributions.

The G statistic has a χ^2 distribution whose formula is:
$G = 2 \sum_{all\ cells} f_{observed} \frac{f_{observed}}{f_{expected}}$.

The merging process continues until no more merges are possible.

4.4. The DFA extraction algorithm

We do not know exactly what information is encoded in the word representations obtained by the *rnn*. We can only assume that these word instances uniquely identify either transitions or states in the target DFA. Both DFA extraction algorithms, as presented in section 2, can be applied after the word classes that define the grammar terminals are formed during the previous stage. It can be immediately observed that if there is a one-to-one correspondence between the obtained *rnn* states and the states of the original DFA, then both extraction algorithm will recover the target automaton.

The context free grammar in figure 5 was expanded at depths 1, 2 and 3 in order to obtain regular grammars that approximate the original grammar. These regular grammars, named “elm_r1”and “elm_r2”, were then used to generate sentences of up to 3, 4, 5 and 6 words. The resulting languages are “elm_r1_d3”, “elm_r1_d4” for sentences of 3 and 4 words, from the regular grammar “elm_r1”, and “elm_r2_d5” and “elm_r3_d6”, respectively.

Both Szilard and Szilard* regular grammars were induced for all languages. The results are shown in table 1.

The original and the induced automata for the language “elm_r1_d4” can be seen in figure 8.

In all the induced automata, there could be observed classes formed from the same words. For example “sees hears walks lives” appear on different transitions in figure 8. From figure 8.b. it follows that there are three such classes, associated with the states “q2”, “q9” and “q11”. These classes are formed from non-overlapping sets of occurrences of the four words which were not merged, and so are considered distinct classes. They were not merged because their vector representations, as extracted from the network, are not similar.

For the slightly larger languages, “elm_r2_d5” and “elm_r3_d6”, the induced grammars no longer recognize exactly the target languages, but supersets of them. A sample of correct and incorrect sentences can be seen in table 2. From the sentences listed there it can be noticed that some distinctions which were encoded in the original grammar are not learned. Such is the distinction between verbs that require a direct object and verbs for which it is optional. This type of error yields incorrect sentences, like “the boys feed”. Another

language name	Original DFA	Szilard* DFA	Szilard DFA
elm_r1_d3	6 states	16 states	8 states
40 sent.		finite lang.	finite lang.
		40 corr.sent. 0 err.sent	40 corr.sent. 0 err.sent
elm_r1_d4	8 states	20 states	12 states
56 sent.		finite lang.	finite lang.
		56 corr.sent. 0 err.sent	56 corr.sent. 0 err.sent
elm_r2_d5	22 states	47 states	31 states
512 sent.		finite lang.	finite lang.
		512 corr.sent. 43 err.sent	512 corr.sent. 258 err.sent
elm_r3_d6	31 states	80 states	63 states
1184 sent.		infinite lang.	infinite lang.
		* 1184 corr.sent. 1133 err.sent	* 1184 corr.sent. 4340 err.sent

Table 1: The original and induced automata for the four increasingly complex sub-languages of the grammar in figure 5. * The sentences were obtained by imposing a limit of 7 words on the sentence length.

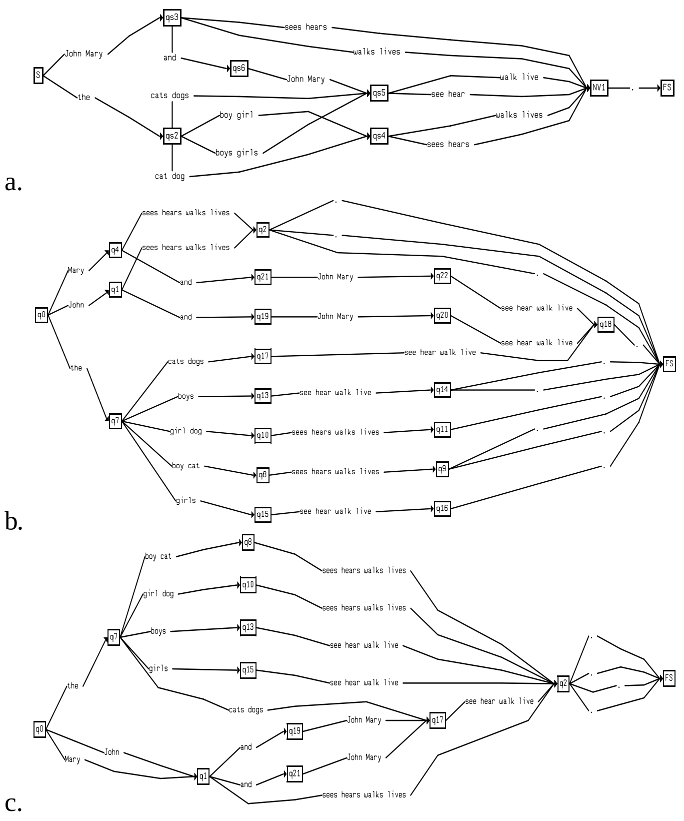


Figure 8: a. Original DFA for the language elm_r1_d4. b. Induced Szilard* DFA . c. Induced Szilard DFA .

missed distinction is between verbs that require a human subject and those for which it is optional: the same “feed”, but in “the cats feed Mary”. Some other sentences are grammatically correct generalizations, that are longer than the number of words allowed by their original grammars. Such is “the boys who see hear John” in table 2, which is generated by the “elm_r2_d5” Szilard DFA. Although this sentence is not present in the original language, it actually appears in the larger language, “elm_r3_d6”.

elm_r2_d5	
correct sentences	incorrect sentences
	Szilard*
“the boys feed the cats.”	“the boys feed.”
“John and Mary see.”	“John and Mary walk Mary.”
“the dogs chase Mary.”	“the dogs chase.”
“the dogs who see live.”	“the cats feed Mary.”
	Szilard
	“the boys who see hear John.”
	“the girl walks the dog.”
	“the dog lives the boy.”

elm_r3_d6	
correct sentences	incorrect sentences
	Szilard*
“the girl feeds John and John.”	“Mary and John feed.”
“the girl who walks feeds John.”	“the girl chases the girl feeds John.”
“the cats who walk chase John.”	“the girl who walks walks Mary.”
	“the girls who live feed the dogs.”
	Szilard
“the boys who see hear John.”	“John and John chase John and Mary.”
“the boy who John feeds lives.”	“John and Mary live the girl.”
“Mary and John see the girls.”	“the boy chases the boy hears Mary.”

Table 2: Samples of correct and incorrect sentences for languages elm_r2_d5 and elm_r3_d6.

6. Conclusions

We showed that good approximations of regular grammars can be learned from positive examples by considering each word occurrence unique and then merging these occurrences into classes of grammar terminals. While the results for the languages presented were quite good, we expect the learning system to perform less well for more complex grammars. There are at least two reasons for this to happen. The first one is due to the behaviour of the recurrent network which due to its huge parameter space can almost always find a function that predicts the probability distribution over the next words, but the hidden units do not encode the DFA states. The second reason is that the distributional criterion is weak: the previous and next word probability distributions do not encode enough information to distinguish words. For example, because of sentences like “Mary sees” and “John and Mary see”, the instances of “see” and “sees” can be merged, if such a merge is proposed by the *rnn*.

Human language learners have access to a vital source of information that is unavailable to our algorithms, the context in which a sentence is uttered. We hypothesize that by adding additional information about the states, in terms of semantic content of the current word, the network search space can be reduced such that the network is more likely to find the desired function. Furthermore, if the word instances are distinguished by additional information, we have better grounds to treat the original language as Szilard. It is appealing to consider that this is actually the case with natural languages, where it is the context that makes the distinction between word occurrences.

References

[1] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, 1995.

[2] J. L. Elman. Finding structure in time. *Cognitive science*, 14:179–211, 1990.

[3] J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 1992.

[4] C. L. Giles, C. B. Miller, D. Chen, G. Z. Sun, H. H. Chen, and Y. C. Lee. Extracting and learning an *unknown* grammar with recurrent neural networks. In *Advances in Neural Information Processing Systems 4*. 1992.

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

[6] John F. Kolen. Fool’s gold: Extracting finite state machines from recurrent network dynamics. In *Advances in Neural Information Processing Systems 6*, 1994.

[7] E. Makinen. Inferring regular languages by merging nonterminals. Technical Report A-1997-6, Department of Computer Science, University of Tampere, 1997.

[8] G. F. Marcus. Negative evidence in language acquisition. *Cognition*, 46:53–85, 1993.

[9] K. Plunkett and J. L. Elman. *Exercises in Rethinking Innateness*. The MIT Press, 1997.

[10] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. Backpropagation: The basic theory. In *Backpropagation: Theory, architectures, and applications*. Erlbaum, 1993.