Eurise

UNIVERSITE JEAN MONNET Saint-Etienne

Use of grammatical inference in natural speech recognition

F. Thollard *

EURISE, Université Jean Monnet
23, rue P. Michelon
42023 Saint-Etienne Cédex – France
thollard@univ-st-etienne.fr

[a]Work supported by France Telecom CNET under contract 97 1B 004

Abstract

This paper presents the application of stochastic grammatical inference to speech recognition.

In speech recognition, the acoustic signal process produces a set of words which are combinating to build sentences. Language models are then used to lead the speech recognition application to the most pertinent combination. Up to now, statistical language models are used. We suggest to use stochastic formal grammars instead of statistical models. Theses stochastic grammars will be build by machine learning algorithms.

We will first show that unaided grammatical inference cannot be used for speech recognition. We will then make manifest that smoothing is necessary and show the gain that one can obtain by using a basic smoothing. We finally put up a smoothing technic dedicates to stochastic formal grammars.

1 Introduction

Our aim is to use stochastic grammatical inference for natural speech recognition. The main difference between validations of grammatical inference algorithms and real applications comes from the size of the vocabulary.

Our final goal is the following : a user phones to a vocal server and tries to get, by way of a dialogue with the machine, the horary of the cinema of its choice. So, our framework is speaker independent natural speech recognition in noisy context. But the application knows which language domain it has to deal with.

The size of our vocabulary is roughly 1000 words. Although it is small in regard of other applications[1], a vocabulary of 1000 words troubles grammatical inference and its use in natural speech recognition.

The language models are used to predict a word given his context : given the beginning of a sentence, (the context of the word), we aim to guess the next word. In other words, we aim to guess the probabilities of *each* word of the vocabulary given the beginning of the sentence.

Our experimentations bring that a lexicon size of 1000 precludes good results with unaided grammatical inference.

First, we present our quality criterion (the perplexity). We then present a well known technic of grammatical inference and display the results obtained by the algorithm *Alergia* [CO94] when processing real data[2]. We then show the importance of smoothing when using grammatical inference on "big" vocabularies. At least, we will present some new smoothing technics dedicated to stochastic formal grammars.

The grammatical inference literature mainly deals with regular grammars. We work in this framework, and in the following grammars must be considered as regular formal grammars. More precisely, we infer stochastic deterministic finite automata : SDFA.

2 The quality criterion

Evaluating a grammatical inference algorithm can be done trough 3 ways :

1. We use theorical criterion to guarantee that the algorithm results will be correct. Knowned criterion are : identification in the limit with probability 1 [Hor69] : a probabilistic extension of Gold's criterion [Gol67], PAC criterion (from valiant [Val84]), ϵ-approximation from Ron [RYT95] and simple PAC [DG97]. All theses criteria guarantee a good (or exact) identification when algorithm process with infinite data sets. They provide no information on the speed of the identification : no information on what happens if we have small data sets.

2. We use benchmarks : algorithms work on artificial data hopping that if algorithm A is more efficient than algorithm B on theses data then final applications that use algorithm A will have better results. Moreover, benchmarks usually aren't under copyright as real data. But, one could develop algorithm that use feature of benchmarks. Theses algorithms can then be very efficient on benchmarks but not on real data.

[1] For information, the data base of wall street journal has a lexicon size of 65000 words.

[2] Our experimentations were realized on data given by France Telecom CNET which support author's thesis under contract 97 1B 004.

3. Our algorithms are used in real applications. We then compare the efficiency of the final application. In natural speech recognition, results are given in term of error rate (or success rate if you are optimist).

We are lying between benchmarks and real application. Because of implementation questions, our language models are not used by the speech recognition application. However, they are build on the same data set ; good results on those data allows us to be hopeful on the results in the final application.

Because we are not in real condition we have to use a criterion quality to estimate the result of the infered automata (as far as we cannot use the rate of success). The criterion must satisfy the following conditions :

- It must provide numerical result to allow comparison between different language models.
- It must represent the quality of the machine learning process.
- It must be easily computable.

Stochastic grammatical inference can be view as the learning of probability distributions over a given alphabet.

Our experimental ideal framework is the following : we have a multi-set L of strings[3] generated by the target SDFA T_{SDFA} ; we use L to learn a SDFA L_{SDFA}. We then use a kind of distance between L_{SDFA} and T_{SDFA} to estimate the quality of the inference. As a multi-set can be view as a distribution over a certain alphabet, we "just" have to compare two distributions of probabilities. Such a distance exists : the relative entropy[4].

In practical cases, we only have two multi-sets of data L and T which are supposed to have been generated by T_{SDFA}. Our goal is to infer a SDFA L_{SDFA} using the multi-set L. We then use the multi-set T to estimate the quality of L_{SDFA}. This could be done by estimating a kind of distance between L_{SDFA} and T. In the following, we will note D_L the distribution given by L_{SDFA} and D_T the distribution given by T_{SDFA}. We can now compute an estimation of the relative entropy between L_{SDFA} and T_{SDFA} by estimating an approximation of the relative entropy between L_{SDFA} and T.

We will note Σ the alphabet, Σ^* the set of strings on Σ, x a string of Σ^*, $D(D_L||D_T)$ the relative entropy of D_L with regard to D_T, and $P_L(x)$ ($resp$ $P_T(x)$) the probability of the string x given the distribution D_L ($resp.$ D_T). We then have :

$$
\begin{aligned}
D(D_T||D_L) &\stackrel{def}{=} \sum_{x\in\Sigma^*} P_T(x)\ \log\frac{P_T(x)}{P_L(x)} \\
&= \underbrace{-\sum_{x\in\Sigma^*} P_T(x)\ \log(P_L(x))}_{Cross\ entropy\ beetween D_L\ et\ D_T} \underbrace{-\Big(-(\sum_{x\in\Sigma^*} P_T(x)\ \log(P_T(x)))\Big)}_{H_{D_T}}
\end{aligned}
$$

[3] In languages theory, the alphabet is a set of letters and automatons generate some words. In speech recognition, the alphabet is a set of words and the automatons generate some sentences. In the following we'll call *symbol* the elements of the alphabet and *string* the ones which are generated by automatons

[4] The relative entropy is not a mathematical distance as it isn't symmetric. however a small relative entropy between two distributions means that they are not very different.

First bloc can be seen as the cross entropy between D_L and D_T. The second as the entropy of D_T. If experimentations are conducted on the same data, (*i.e.* in order to infer the same distribution D_T) H_{D_T} will be a constant.

Let now focus on the cross entropy :

$$-\sum_{x\in\Sigma^*} P_T(x)\log(P_L(x))$$

As said before, practical cases can only offer a multi-set T which is supposed to have been generated by distribution D_T. We will approximate the cross entropy between D_L and D_T by the cross entropy by symbol of D_L in regard of T :

$$H(D_L,T) = -\sum_{x\in T} \tilde{P}_T(x)\log(P_L(x)) \tag{1}$$

where $\tilde{P}_T(x)$ represents an estimation of $P_T(x)$, computed with the count of string x in multi-set T. Let us note $C_T(x)$ the count of x in T and n the number of string of T. Equation 1 becomes :

$$H(D_L,T) = -\sum_{x\in T} \frac{C_T(x)}{n}\ \log(P_L(x)) \tag{2}$$

In our case, P_L is represented by a SDFA and $P_L(x)$ is then the probability that the SDFA generates the string x. If *symbols* are numbered from 1 to $\|T\|$, equation 2 becomes :

$$H(D_L,T) = -\frac{1}{\|T\|}\sum_{i=1}^{\|T\|}\ \log(p(a_i|q^i)) \tag{3}$$

where a_i is the i^{nth} symbol in T and $p(a_i|q^i)$ the probability that the automaton generates a_i when in state q_i. State q_i is the current state when parsing the multi-set T.

The cross entropy by symbol is an approximation of the cross entropy between P_T and P_L. It represents the cross entropy by considering only the string of T. When logarithm is taken in base 2, the unit of cross entropy by symbol is the bit. We now define the perplexity as :

$$\begin{aligned} Q(D_L,T) &= 2^{H(D_L,T)} \\ &= 2^{\left[-\frac{1}{\|T\|}\sum_{i=1}^{\|T\|}\ \log_2(P_L(a_i|q^i))\right]} \end{aligned} \tag{4}$$

From formula 4, we can infer that the smaller the cross entropy is, the bigger the adequation between the automaton and the multi-set T is.

Nota bene : If the automaton cannot parse a given string x (that is, $P_L(x) = 0$) the perplexity will take a infinite value. To be able to take information from perplexity, the infered language model *must* be able to give a non null probability to any string of Σ^*. In other words, we say that any string of Σ^* can appear in the test set T. In words of speech recognition, we say that any word can appear in any sentence, even if the sentence means nothing or is syntacticly incorrect.

3 Building and smoothing language models

3.1 Statistic language models

Definition : Statistic language models are called n-gramms [BPd+92]. A n-gramm is a set of couples (T_i,p_i) where T_i is a n-uplet of symbols $< s_n, ..., s_2, s_1 >$ and p_i is the probability of symbol s_1 given the (n-1)-uplet $s_n, ..., s_3, s_2$. As p_i represents a probability, it must respects the consistency condition $\sum_{i=1}^{M} p_i = 1$ where M is the number of couples (T_i , p_i) of the n-gramm.

In practice, bigramms and trigramms are used as statistical language models.

We now present how stochastic language models are built and smoothed. Without loosing generality, we can present trigramms.

Building : A trigram is built by extracting the triplet T_i from the learning multi-set L. The probabilities p_i are then computed using the count of each triplet T_i. If $C_L(T_i)$ is the count of T_i in L one write :

$$p_i = \frac{C_L(T_i)}{M}$$

Smoothing : The aim of the smoothing is to be able to give a non null probability for any triplet T_i that can be built using symbols of the lexicon. To do so, we substracts to each p_i a small quantity ϵ. We obtain an amount of probability (namely $M \times \epsilon$) that we will be able to redistribute on the missing triplets (*i.e.* triplets that can be built on the lexicon, and which do not appear in the training multi-set L). The smoothing technics depend on the way they rearrange the probabilities. The one used in speech recognition applications is the back-off technic proposed by Katz[Kat87] : let $< s_3, s_2, s_1 >$ be a triplet of probability firstly estimated to 0. Katz smoothes the trigramm by giving a probability to $< s_3, s_2, s_1 >$ *computed*[5] on the probability of couple $< s_2, s_1 >$.

In the same way, if couple $< s_2, s_1 >$ has a null probability (that is symbol s_1 never appear after symbol s_2 in the learning multi-set L), one rearrange the probabilities by giving a probability linked to the value of $< s_1 >$ in the smoothed unigramm. A symbol can appear in the multi-set T whereas it does not appear in L. To manage this problem, we smooth the unigramm by adding a word UNK to the lexicon. This new word represents the unknowned words, . By the same technic, word UNK will have a non null probability (namely, $n \times \epsilon$ if n is the size of the lexicon).

3.2 Stochastic deterministic finite automata

Many algorithms are based on state merging [CO94], [RYT95], [SO94]. They operate in 2 steps. First, a SDFA is built. It represents the learning multi-set L. This automaton is called the PTA (Prefix Tree Acceptor). Then, states are compared (in a predefined order) and merged if considered compatible. The resulting automaton can be non deterministic. States involved in the non determinism are then merged.

[5]Because of the constrain of consistency, we cannot give to the triplet $< s_3, s_2, s_1 >$ the probability of $< s_2, s_1 >$.

Algorithms differ by their compatibility function and the predefine order they use to schedule the state comparisons.

We are now going to deal with algorithm *Alergia* [CO94].

The compatible criterion is based on Hoeffding bound [Hoe63] which gives the confidence range of a Bernoulli variable of probability p, observed f times in a set of n elements.

$$Pr\Big(\quad |p - \frac{f}{n}| < \sqrt{\frac{1}{2n} \log(\frac{2}{\alpha})} \quad \Big) > 1 - \alpha \tag{5}$$

α becomes a parameter of the algorithm which allows users to adjust the notion of state similarity. The smaller α is, the smaller the final automaton (in term of state number) will be. Let A be a SDFA and A' a SDFA derived from A[6]. If A generates language L and A' language L', then [Dup96] $L \subseteq L'$.

In other words, the more α is small, the bigger the generalization will be.

In *Alergia*, state are numbered in a "breath first" order. Each state is compared with all the states which have a smaller number. This merging scheduling guarantee that, one (at least) of the two states candidate for a merge is the root of a tree. This propriety is strongly used by the similarity function.

4 The results

We now present the inference of algorithm *Alergia*.

Characteristics of data (learning multi-set, test multi-set, size of the vocabulary) are shown by figure 1.

Corpus	learning	tests	Vocabulary
Number of string	9852	726	-
Number of symbols	49610	3584	824

Figure 1: Data sets characteristics

To avoid a infinite value of perplexity computed between the infered SDFA and the test multi-set, the automaton must be smoothed. We used a linear interpolation between the infered automaton and a smoothed unigramm. If one notes $P(x|SDFA)$ the probability of x given by the infered automaton and $P(x|Uni)$ the probability of x given by the smoothed unigramm, the linear interpolation between the two models can be noted :

$$IL(x) = \lambda \; P(x|SDFA) + (1 - \lambda) \; P(x|Uni)$$

At least one can tune with two parameters to obtain the final language model : parameter α which controls the generalization of the infered SDFA and λ which controls the relative importance of the two language models.

Figure 2 shows the results (in term of perplexity) obtained by tuning each parameter.

[6] A automaton A' is derived from a automaton A if there is a partition π of the set of state of A such that A' is obtained by merging the state of A belonging to the same subset in π.

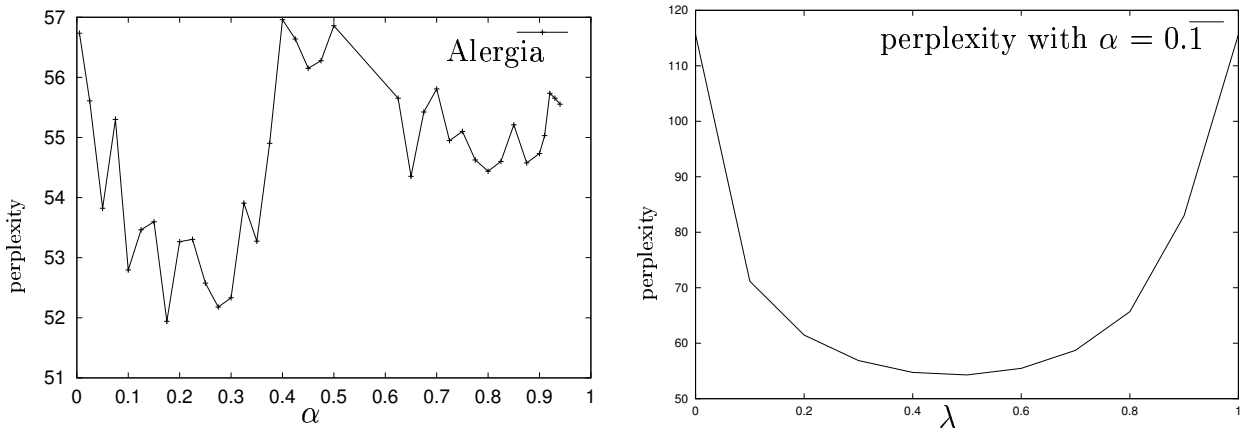

Figure 2: Perplexity behavior

We can see on the first plot of figure 2 that we must not generalize two much the infered language model : while trying a big generalization, the algorithm seems to allow bad merges : perplexity grows.

The second plot of figure 2 shows the gain of perplexity provided by the smoothing : up to 50 %.

4.0.1 Grammatical inference limits

Important point brought up by continuous speech recognition is the lexicon size. Besides the computing constrains, the vocabulary size forces the use of smoothing. Smoothing is necessary when the parsing of a string x of T cannot be done : to parse x, we start from the initial state of the infered automaton. We then follow the outgoing transition labeled by the first symbol of x and set p (the probability of generating x) to be the probability of the above transition. We try then to parse the second symbol of x. If p' is the probability of the second transition we have $p = p \times p'$. And so on. If we cannot find a transition from the current state labeled by the current symbol, we have to smooth to define p to a non null value. In figure 3 page 8, smoothing will be used to parse string *bcdda* in automaton A1 : state 3 of automaton A1 has no outgoing transition labeled by *d*.

Two different cases must be dealt with :

1. Symbol is not in the lexicon : it does not appear in the learning multi-set L. This problem must be managed separately by defining a "a priori" lexicon that contains all the vocabulary Σ. In practice, even this technic cannot supply all the symbols of T ; as seen before, a special symbol (UNK) is created for this kind of unknowned symbol.

2. Symbol is in the lexicon : the infered model language has not been generalized enough.

As seen in figure 2, a too strong generalization is not a good generalization. The solution is to build a not too generalized language model and to smooth to deal with the weak of generalization.

5 Smoothing adapted to SDFA

The notion of language model hierarchy is underlying in back-off technic (showned in paragraph 3.1 page 5). A mapping between n-gramms and (n-1)-gramms can indeed be define. This mapping is implicitly used to find which (n-1)-gramm will be used to compute the back-off value of a given n-gramm.

Our goal is to build a SDFA hierarchy to compute a back-off like smoothing on SDFA. Figure 3 presents a SDFA hierarchy. Let SDFA *A1* be the infered automaton, and *bcdda* the string to parse. We can see that *A1* cannot parse string *bcdda* because of the first symbol *d*. Thanks to the mapping, we can find the more specific automaton such that string *dda* can be parsed. On figure 3, this automaton is *Ai*. Like in back-off technic, we will compute the value of the smoothed transition and then be able to terminate the parsing in automaton *Ai*.

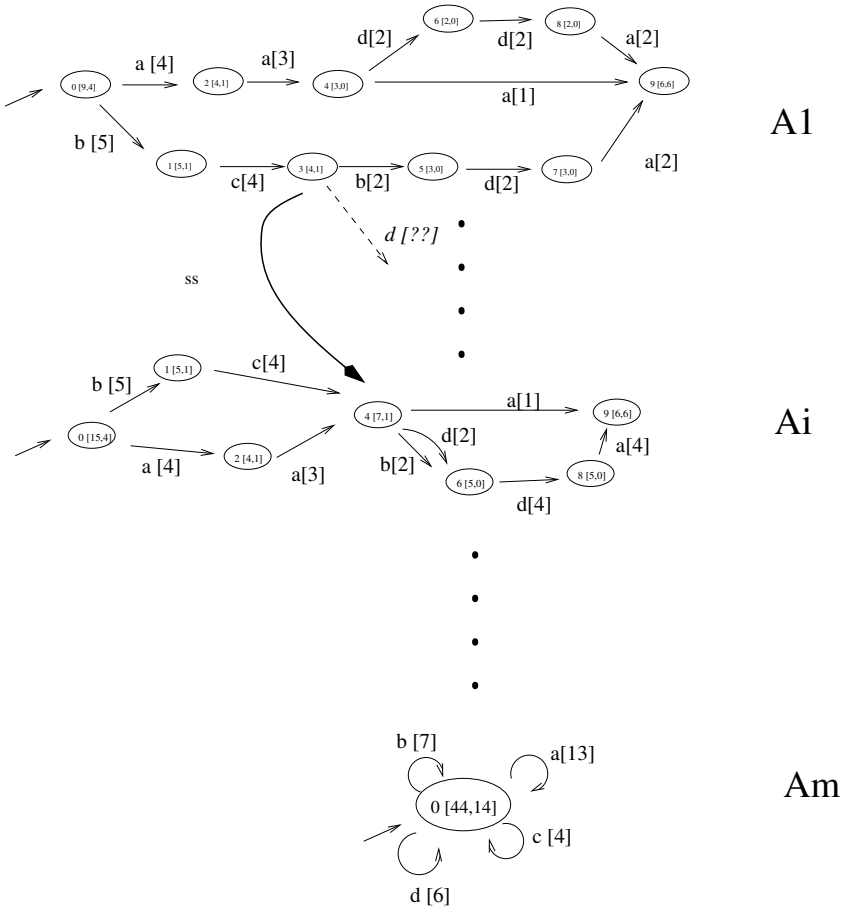

Figure 3: SDFA hierarchy

Putting this idea into practice will not be easy :

- Building the hierarchy supposes the ability to generalize a non acyclic SDFA. The literature algorithms cannot provide this feature since merging and compatibility functions work only if at least one of the two states to be merged is the root of a tree.

In [Car97], Carrasco gives a way to compute an approximation of the relative entropy between two SDFA. This technic supposes that the two SDFA recognize languages joined up by an inclusion bond (none strict inclusion). Since, on one hand, languages recognized by two arbitrary states of a SDFA [7] are not linked by an inclusion relationship, and, on the other hand, compatibility function takes states as input, this technic cannot be used to compute a similarity relationship between two states. However, we saw above (section 3.2 page 6) that a SDFA and its derived automaton recognize languages joined up by an inclusion bond. A algorithm based on the Carrasco technic computed on a SDFA and its derived automata can probably be defined. This algorithm is under study.

- Lets come back in the n-gramm case : while parsing the test set T, one will use the smooth only when the n-gramm does not exist. He can then resume the parsing of current string in the n-gramm language model. In automaton case, we cannot come back in automaton *A1* after smoothing : Lets try to parse string *abdda* with automaton *A1* of figure 3. As seen before, one can go in automaton Ai to deal with the first symbol *d*. Current state is then state E_4^i (state 4 of automaton Ai). The question is then : How to proceed the parsing ? The basic idea is to finish the parsing in automaton Ai. A most seductive idea is to come back in the infered automaton (that is *A1*) to a *right* state. As far as there is a mapping between automata, we can find more than one state of *A1* that matches with the current state of Ai (in figure 3, state E_4^i matches with states E_3^1, E_4^1). The more elegant idea is to define a way of finding, given a state of Ai, the "best" matching state in *A1*. The parsing can be terminated in *A1* ; in fig 3, we would like to finish the parsing of suffix *dda* from state E_5^1.

6 Conclusions and further works

Our goal was to use grammatical inference in natural speech recognition.

Experimentations conducted on real data have shown that smoothing is necessary to use grammatical inference in natural speech recognition.

At the moment, statistical model of languages are used in natural speech recognition applications. When they are smoothed with back-off smoothing, they provide better result than the linear interpolation beetween a SDFA and a smothed unigramm. However, they do not provide better result when they are smoothed by a linear interpolation with a smoothed unigramm.

We think that a SDFA smoothed with a adapted technic can perform good results.

We now have to finish the definition and implementation of the automaton hierarchy. We will then be able to try different way of smoothing : finish parsing in automaton Ai, trying different way of choosing a state in *A1* to be able to finish parsing in *A1*. We will then be able to compare our smoothed model of language and the trigramms smoothed by back-off technic.

[7] One can define the language recognized by a state E of a given SDFA A as the language recognized by A having E as initial state.

References

[BPd+92] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December 1992.

[Car97] R. C. Carrasco. Accurate computation of the relativ entropy beetween stochastics regulars grammars. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, vol. 31(num. 5):p 437–444, 1997.

[CO94] R. Carrasco and J. Oncina. Learning stochastic reagular grammars by means of a state merging method. *ICGI-94*, Lecture Notes in Artificial Intelligence 862:139–152, 1994. Subseries of Lectures Notes in Computer Science.

[DG97] F. Denis and R. Gilleron. Pac learning under helpful distributions. In *Proceedings of the Eight International Workshop on Algorithmic Learning Theory*, Japan, 1997. Sendai.

[Dup96] P. Dupont. *Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue.* PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1996.

[Gol67] E. M. Gold. Langage indentification in the limite. *Information and control*, 10(5):447 – 474, 1967.

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

[Hor69] J. J. Horning. *A Study of Grammatical Inference.* PhD thesis, Standford University, Computer Science Department, California, 1969.

[Kat87] S.M Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 35(num 3):400–401, 1987.

[RYT95] D. Ron, Y.Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. pages 31–40, Santa Cruz CA USA, 1995. COLT'95.

[SO94] A. Stolke and S. Omohundro. Inducting probabilistic grammars by bayesian model merging. In Lecture Notes in Artifitial Intelligence, editor, *Proceedings of International Colloquium on Gramatical Inference*, number 862, pages 106–118. ICGI-94, 1994.

[Val84] L. Valiant. A theory of the learnable. *Comunication of the ACM*, 27(11):1134 – 1142, 1984.