Distances between Distributions: Comparing Language Models

Thierry Murgue[1,2] and Colin de la Higuera[2]

[1] RIM, Ecole des Mines de Saint-Etienne 158, Cours Fauriel
42023 Saint-Etienne cedex 2 – France
[2] EURISE, University of Saint-Etienne 23, rue du Dr Paul Michelon
42023 Saint-Etienne cedex 2 – France
murgue@emse.fr, cdlh@univ-st-etienne.fr

Abstract. Language models are used in a variety of fields in order to support other tasks: classification, next-symbol prediction, pattern analysis. In order to compare language models, or to measure the quality of an acquired model with respect to an empirical distribution, or to evaluate the progress of a learning process, we propose to use distances based on the L_2 norm, or quadratic distances. We prove that these distances can not only be estimated through sampling, but can be effectively computed when both distributions are represented by stochastic deterministic finite automata. We provide a set of experiments showing a fast convergence of the distance through sampling and a good scalability, enabling us to use this distance to decide if two distributions are equal when only samples are provided, or to classify texts.

1 Introduction

A common task to machine translation [1], speech recognition [2], optical character recognition [3] or computational biology [4] is that of constructing a language model. Typical language models are n-grams [5], but HMMs also can be used [6]. Finite automata have been considered as alternative language models for nearly 10 years [7], with an extension to stochastic automata more adapted to the task [8]. In a more general setting, stochastic or probabilistic automata have been used in structural and syntactic pattern recognition for a number of years [9]. How to derive a grammar or an automata from data is usually called grammatical inference or grammar induction: This has been studied in the framework of pattern recognition [10, 11], with applications to textures in images, fingerprints classification, dynamic systems or recognition of pictures of industrial objects. The problem of learning a stochastic finite automaton is generally considered to be a hard but important one [12], with smoothing a decisive component of language modeling. Clearly, a better understanding of stochastic automata, and moerover of their topological properties is necessary in order to better learn them or compare them. Not much work has been done in this direction: Work linked with the results we report here has been done by Fred [13], who computes the weight of a language following a distribution, or Lyngsø *et al.* [4] who prove

© Springer-Verlag Berlin Heidelberg 2004

that computing the L_2-distance is tractable between distributions over finite sets, or Carrasco who computes the Kulback-Leibler divergence between regular distributions [14] or the L_2-distance between distributions over trees [15].

We can thus identify the following tasks related with language modeling: measure how well a sample corresponds to a distribution or how close a hypothesis language is from a target, or even how close two samples are one from the other. Traditionally researchers in the field have used *perplexity* as its measure for the above questions, with the *Kullback-Leibler divergence* closely related to this measure between a true or target distribution $\mathfrak{D}$ and a hypothesis or candidate distribution $\mathfrak{D}'$:

$$d_{KL}(\mathfrak{D}, \mathfrak{D}') = \sum_{w \in \Sigma^\star} \Pr_{\mathfrak{D}}(w) \log \frac{\Pr_{\mathfrak{D}}(w)}{\Pr_{\mathfrak{D}'}(w)}$$

The Kullback-Leibler divergence suffers from a number of drawbacks:

1. It is not a distance. Therefore topological operations are not easy, and using samples to represent a distribution is not reasonable.
2. In the case where some string has a null probability in $\mathfrak{D}'$, but not in $\mathfrak{D}$, then the Kullback-Leibler divergence is infinite. This implies that over-generalization is necessarily going to appear. The language model can only be considered when it is properly *smoothed*. But in this case it is hard to know if (when concerned with testing) what we are measuring is the quality of the model or that of the smoothing or alternatively of both, combined, which may be what we are looking for.

These seems to be good reasons to propose an alternative measure to see how close one distribution is to another. Furthermore this measure should be a distance, computable and easy to calculate, and not require the models to be smoothed for computation to be possible. We give in section 2 the definitions of stochastic deterministic finite automata (DPFA), regular distributions, and the probability functions that are associated. Distances between DPFA are defined and studied in section 3. In section 4, we experimentally study the distances: A first set of experiments (section 4.1) on the well-known Reber grammar [16] show that our measure can give a good estimation of the quality of the learned automata. In section 4.2, we show that the speed of convergence of the distance of a sample to the model is sufficiently fast to be able to decide in practice from which automaton a sample is generated. These experiments are made on artificial data corresponding to parity functions, which are usually considered as a hard case for learning. Then, section 4.3 deals with experiments on real data: firstly, we show the L_2 distance compares favorably to perplexity on a typical speech language modeling task; a second set of experiments (section 4.4) on French poems shows that the distance can be used in classification tasks. They also show the scalability of these methods. In section 5 we discuss further work and conclude.

2 Probability Distributions over Sequences

Definition 1 (Distribution over strings). *Let Σ be an alphabet i.e. a finite set of letter, $\{a, b, \ldots\}$. A string w is an element from Σ^*. The length of a string w is denoted by $|w|$. The unique string with length 0 is λ, $|\lambda| = 0$. A probability function is a function such that $\forall w \in \Sigma^*$, $0 \leq \Pr_{\mathfrak{D}}(w)$ and $\sum_{w\in\Sigma^*} \Pr_{\mathfrak{D}}(w) = 1$.*

Definition 2 (Dpfa). *A* Dpfa *is a tuple $\mathcal{A} = \langle \Sigma_{\mathcal{A}}, Q_{\mathcal{A}}, q_{I_{\mathcal{A}}}, \delta^{\bullet}_{\mathcal{A}}, p^{\bullet}_{\mathcal{A}}, f_{\mathcal{A}} \rangle$ where $\Sigma_{\mathcal{A}}$ is a finite alphabet, e.g. $\{a, b\}$, $Q_{\mathcal{A}}$ is a set of states, $\{q_0, q_1, q_2, \ldots, q_{|\mathcal{A}|-1}\}$, $q_{I_{\mathcal{A}}}$ is the unique initial state, $\delta^{\bullet}_{\mathcal{A}} : Q_{\mathcal{A}} \times \Sigma_{\mathcal{A}} \mapsto Q_{\mathcal{A}}$ defines a transition function, $p^{\bullet}_{\mathcal{A}} : Q_{\mathcal{A}} \times \Sigma_{\mathcal{A}} \mapsto [0,1]$ is a probability function, $f_{\mathcal{A}} : Q_{\mathcal{A}} \mapsto [0,1]$ defines the probability that a state is final.*

Non-deterministic automata exist, but shall not be used in this paper. In the following, when no ambiguity is possible, we will forget subscript $_{\mathcal{A}}$.

To be able to deal with strings, we generalize the automaton probability and transition functions:

Definition 3 (Generalized probability and transition functions). *Let $w \in \Sigma^*$ be a string, let $\mathcal{A}$ be an* Dpfa, *and q a state,*

$$p_{\mathcal{A}}(q, w) = \begin{cases} f_{\mathcal{A}}(q) & \text{if } w = \lambda \\ p^{\bullet}_{\mathcal{A}}(q, a) \cdot p_{\mathcal{A}}(\delta^{\bullet}_{\mathcal{A}}(q, a), x) & \text{else } (w = ax, a \in \Sigma, x \in \Sigma^*) \end{cases}$$

$$\delta_{\mathcal{A}}(q, w) = \begin{cases} q & \text{if } w = \lambda \\ \delta_{\mathcal{A}}(\delta^{\bullet}_{\mathcal{A}}(q, a), x) & \text{else } (w = ax, a \in \Sigma, x \in \Sigma^*) \end{cases}$$

Definition 4. *$\pi_{\mathcal{A}}$ is the prefix probability function:*

$$\forall w \in \Sigma^*, \; \pi_{\mathcal{A}}(w) = \sum_{x\in\Sigma^*} p_{\mathcal{A}}(wx)$$

The definitions are linked as follows:

$$\forall w \in \Sigma^*, \; p(w) = \pi(w) \cdot f(\delta(q_I, w))$$
$$\forall w \in \Sigma^*, \forall a \in \Sigma, \; \pi(wa) = \pi(w) \cdot p^{\bullet}(\delta(q_I, w), a)$$
$$\forall w \in \Sigma^*, \; \pi(w) = p(w) + \sum_{a\in\Sigma} \pi(wa)$$

3 Distances between Two Dpfa

In this section we define two measures over string probability distributions. Technically a first step is to compute, given $\mathcal{A}$ and $\mathcal{A}'$ two Dpfa, the product probability of reaching two states q in $\mathcal{A}$ and q' in $\mathcal{A}'$:

Definition 5. *Let $(q, q') \in Q_{\mathcal{A}} \times Q_{\mathcal{A}'}$,*

$$\eta_{qq'} = \sum_{\substack{w\in\Sigma^*: \\ \delta_{\mathcal{A}}(q_{I_{\mathcal{A}}}, w) = q \\ \delta_{\mathcal{A}'}(q_{I_{\mathcal{A}'}}, w) = q'}} \pi_{\mathcal{A}}(w) \cdot \pi_{\mathcal{A}'}(w)$$

We describe here a method to compute $\eta_{qq'}$. When Σ^* is not a finite set of strings, the above sum may not be easily computable. We adapt the efficient method that computes the d_2 distance over trees [15] to the string case. The method computes iteratively $\eta_{qq'}$.

Note that λ is the only null-length prefix and $\pi_{\mathcal{A}}(\lambda) = \pi_{\mathcal{A}'}(\lambda) = 1$. That allows us to write:

$$\eta_{q_{I_\mathcal{A}} q_{I_{\mathcal{A}'}}} = 1 + \sum_{q \in Q_\mathcal{A}} \sum_{q' \in Q_{\mathcal{A}'}} \sum_{\substack{a \in \Sigma: \\ \delta_\mathcal{A}(q,a) = q_{I_\mathcal{A}} \\ \delta_{\mathcal{A}'}(q',a) = q_{I_{\mathcal{A}'}}}} \eta_{q,q'} \cdot p^{\bullet}_{\mathcal{A}}(q,a) \cdot p^{\bullet}_{\mathcal{A}'}(q',a)$$

and $\forall q \in Q_\mathcal{A}, q' \in Q_{\mathcal{A}'} \neq (q_{I_\mathcal{A}}, q_{I_{\mathcal{A}'}})$

$$\eta_{qq'} = \sum_{s \in Q_\mathcal{A}} \sum_{s' \in Q_{\mathcal{A}'}} \sum_{\substack{a \in \Sigma: \\ \delta_\mathcal{A}(s,a) = q \\ \delta_{\mathcal{A}'}(s',a) = q'}} \eta_{s,s'} \cdot p^{\bullet}_{\mathcal{A}}(s,a) \cdot p^{\bullet}_{\mathcal{A}'}(s',a)$$

The above relation corresponds to a system of linear equations. To solve it efficiently, when convergence is clear, terms are computed iteratively during a predefined number k of steps. Complexity then is $\mathcal{O}(|Q_\mathcal{A}| \cdot |Q_{\mathcal{A}'}| \cdot |\Sigma| \cdot k)$.

In order to compare two distributions, we first evaluate the co-emission probability of each string, *i.e.* the probability that $\mathcal{A}$ and $\mathcal{A}'$ generate independently a same string.

Definition 6. *The* Co-emission probability *of $\mathcal{A}$ and $\mathcal{A}'$ is*

$$\begin{aligned}\mathrm{CoEm}(\mathcal{A},\mathcal{A}') &= \sum_{w \in \Sigma^*} (p_\mathcal{A}(w) \cdot p_{\mathcal{A}'}(w)) \\ &= \sum_{q \in Q_\mathcal{A}} \sum_{q' \in Q_{\mathcal{A}'}} \eta_{qq'} \cdot f_\mathcal{A}(q) \cdot f_{\mathcal{A}'}(q')\end{aligned}$$

If comparing samples, co-emission may be null when large vocabulary and long strings are used. It is then reasonable to compare not only the whole strings, but their prefixes:

Definition 7. *As above, the* Prefixial Co-emission probability *of $\mathcal{A}$ and $\mathcal{A}'$ is*

$$\mathrm{CoEmPr}(\mathcal{A},\mathcal{A}') = \sum_{w \in \Sigma^*} (\pi_\mathcal{A}(w) \cdot \pi_{\mathcal{A}'}(w))$$

Note that CoEmPr *is directly linked to η coefficients.*

The above co-emission measures allow us to define two distances for the L_2 norm:

Definition 8 (d_2 distance between two models). *The* distance for the L_2 norm*, denoted by* d_2 *is defined as:*

$$\mathrm{d}_2(\mathcal{A},\mathcal{A}') = \sqrt{\sum_{w \in \Sigma^*} (p_\mathcal{A}(w) - p_{\mathcal{A}'}(w))^2}$$

which can be computed easily by using:

$$d_2(\mathcal{A},\mathcal{A}') = \sqrt{\mathrm{CoEm}(\mathcal{A},\mathcal{A}) + \mathrm{CoEm}(\mathcal{A}',\mathcal{A}') - 2\,\mathrm{CoEm}(\mathcal{A},\mathcal{A}')}$$

Definition 9 (d_{2p} distance between two models). *The* prefixial distance for the L_2 norm*, denoted by* d_{2p} *is defined as:*

$$d_{2p}(\mathcal{A},\mathcal{A}') = \sqrt{\sum_{w\in\Sigma^*} \left(\pi_{\mathcal{A}}(w) - \pi_{\mathcal{A}'}(w)\right)^2}$$

Theorem 1. d_2 *and* d_{2p} *are distances over* Σ^* *(see [17] for the proof).*

4 Experiments – Results

4.1 Identification Task

The Reber artificial grammar [16] has been used to prove that human language learning is implicit (no conscious rules). Figure 1 shows a DPFA representing this grammar. We compare here distance d_2 and perplexity pp between a learned automaton and the target one.

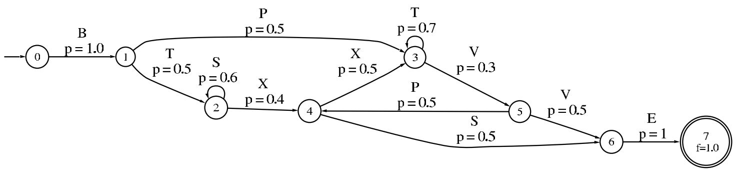

Fig. 1. Reber grammar

We built learning samples with different sizes (100, 500, 1000 strings). Algorithm MDI [8] is used to learn an automaton from these samples. MDI is a state merging algorithm with currently the best results for language modeling by DPFA tasks; It makes use of a parameter (α) to adjust learning.

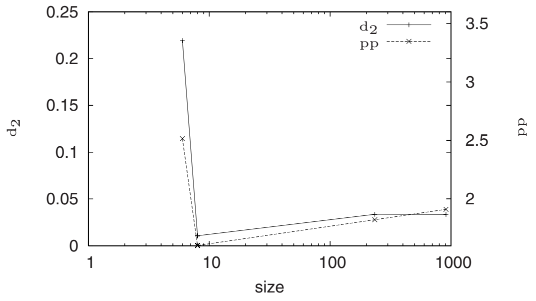

Fig. 2. Comparison between perplexity and distance d_2 in the exact identification task

Figure 2 shows the results for a sample of size 500, but other sizes give similar results: The structure of the automaton is well identified, but the probabilities are better estimated when sample size becomes larger. Size is drawn in log-scale format. The point corresponding to a 8 states automaton is actually reached by 5 automata learned with 5 different settings of the α parameter (in the range $\{0.005, 0.08\}$). Both pp and d_2 are minimal with the same values of α (*i.e.* on the same learned automaton) corresponding to the case where the target structure is obtained. Over- and under-generalization (which does not reach the correct structure) yields worse results. Again, it should be noted that the d_2 results have been obtained without smoothing the automaton; This is an advantage of the technique.

4.2 Discrimination and Sample Robustness

In this section, the aim is to show that the d_2 distance is robust for sampling, or, in other words, is able to help us deciding if a sample is generated from one automaton or another. A set of experiments and evaluation of the distance behavior was carried out on artificial data. Parity functions [18] have been used in various occasions as examples of functions that are hard to learn. A parity function accepts strings if the total number of bs in a certain number of predefined positions is even. To define this kind of functions as automata, we use *inversion positions expressions* which are strings of n digits from $\{0, 1\}$. They classify strings over a 2-letter alphabet $\Sigma = \{a, b\}$. A parity function f will either accept a string of length n with an even number of bs in inversion positions (marked by a 1), or a string of length $n + 1$ with an odd number of bs in those positions.

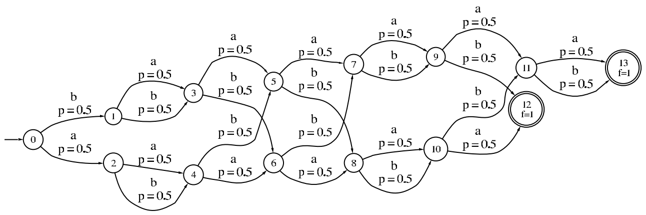

Fig. 3. f_1 DPFA representation

For the experiment we created three functions/languages: f_1 which represents expression 101101 (represented Figure 3). f_2 corresponds to expression 011010 and f_3 has the same structure as f_1 with small differences on transition probabilities. Computation of the distance d_2 is made on 15 samples (s_i) of different sizes (from 10 to 10000 strings).

Results are shown on Figure 4. X axis represents the size of the data sets for each function. Both X and Y axis are in logarithmic scale.

Figure 4 shows that $\lim_{|s_i|\to\infty} d_2(s_i, f_j) = d_2(f_i, f_j)$. Moreover the convergence is fast. A second point is that with $n \geq 200$ we have a real difference between $d_2(s_1, f_1)$ and the other computations. That allows us to hope to be able to detect if a set has been sampled from a given automaton. We can also,

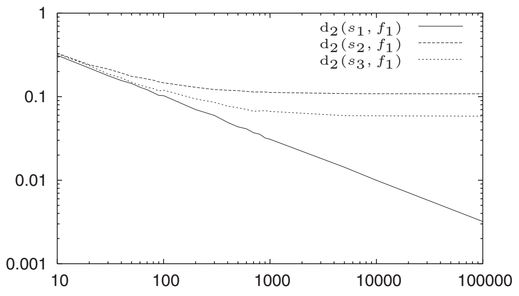

Fig. 4. Convergence of d_2

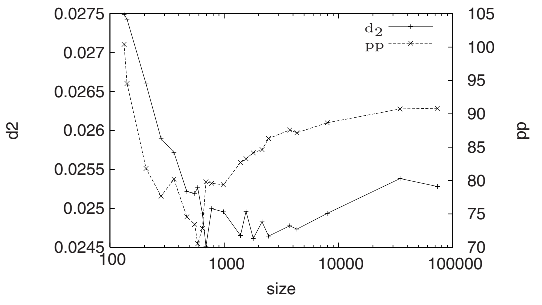

Fig. 5. Behavior of the perplexity and the d_2 distance on `ATIS` database

as a relative result, note that the $d_2(s_3, f_1)$ curve is always under the one for $d_2(s_2, f_1)$. We note that f_1 is closer to f_3 than to f_2. Actually f_1 and f_3 represent the same language with small differences on string probabilities.

4.3 Evaluation of the Quality of a Language Model

In this section, we show that the d_2 distance can be a used as a good measure to evaluate if a learned automaton is a good language model or not. As described in the introduction, perplexity is often used to estimate this by using a test sample for validation. Here, we compare d_2 and perplexity between a learned automaton and a test sample. We use the ATIS database [19]. The learning algorithm is this time `Alergia` [20]: The merging process depends on Hœffding bounds (a statistical test is performed) and a generalization parameter α. The original algorithm is slightly modified in order to learn only acyclic automata. After learning automata with different α values, we compute perplexity and d_2 between the learned automaton and a test sample. To compute perplexity we need to smooth the automaton; we choose a simple smoothing method: interpolation with a 1-gram. The results of the experiment are synthesized on Figure 5. X axis is displayed in log scale format.

Table 1. Success rates

Collection	d_2 rate	d_{2p} rate	# poems	# strings	# symbols
$c_{H\,1}$	2.30 %	70.11 %	87	2381	46686
$c_{H\,2}$	8.82 %	75.00 %	68	3067	61322
$c_{M\,1}$	100.00 %	73.33 %	15	1079	27243
$c_{M\,2}$	100.00 %	53.33 %	30	1398	26483

Globally, d_2 behavior is close to that of pp. Moreover, we note that the optimal automata for d_2 and pp have similar sizes. Thus, distance d_2 can be considered as a good measure to estimate the quality of language learned model. Again, we do not need to smooth the automaton to use this measure, which allows it to be more objective when we only want to compare learning methods.

4.4 Classifying Authors

In this section, we use distance d_2 to classify texts from different authors. We used four collections of poems, two from Victor HUGO ($c_{H\,1}, c_{H\,2}$) and the two others from Alphonse de LA MARTINE ($c_{M\,1}, c_{M\,2}$). Each collection is used to learn a language model (intended to represent the poet). From each collection units of text (poems) are extracted. We then pair the collections ($c_{H\,i}$ with $c_{M\,i}$) and for each poem from collection 1 (resp. 2) compare it with poets $c_{H\,2}$ and $c_{M\,2}$ (resp. $c_{H\,1}$ and $c_{M\,1}$). The experiment is successful if the distance between the poem and the poet is less than the distance from the poem to the other language model.

We present in table 1 the success rates of good classification for all the four collections, with percentages of good classification and number of poems, sentences (strings), word (symbols) of each collection. Globally, distance d_2 provides poor results: this is due to the fact that individual verses (the *string* unit) are seldom repeated in different poems. In this case, the basic d_2 classifier returns the poem collection with the smallest co-emission probability. Distance d_{2p} obtains more convincing results: Common prefixes are reused by a poet. It should be noted that results are only preliminar; We expect a distance measuring common sub-strings to obtain better results.

5 Conclusion

The results reported in this paper indicate that distance d_2 is an alternative to perplexity for language modeling tasks. A certain number of questions arise from our experiments: A theoretical study of the convergence rate would be helpfull; Identification of the type of automata for which each distance is best suited is an important task; We would like to continue the work on authors in order to enter the debate over affiliations; And finally it would be nice to extend the prefix distance, corresponding to a sub-string distance, and (this is harder) compute it in a similar way?

References

1. Amengual, J.C., Sanchis, A., Vidal, E., Benedí, J.M.: Language Simplification Through Error-Correcting and Grammatical Inference Techniques. Machine Learning Journal **44** (2001) 143–159
2. Ney, H.: Stochastic Grammars and Pattern Recognition. In: Proc. of the NATO Advanced Study Institute. Springer-Verlag (1992) 313–344
3. Lucas, S., Vidal, E., Amari, A., Hanlon, S., Amengual, J.C.: A Comparison of Syntactic and Statistical Techniques for Off-Line OCR. In: Proc. of ICGI 94. (1994) 168–179
4. Lyngsø, R.B., Pedersen, C.N.S., Nielsen, H.: Metrics and Similarity Measures for Hidden Markov Models. In: Proc. of ISMB '99. (1999) 178–186
5. Jelinek, F.: Statistical Methods for Speech Recognition. The MIT Press (1998)
6. Morgan, N., Bourlard, H.: Continuous Speech Recognition: an Introduction to the Hybrid HMM/connectionnist Approach. IEEE Signal Processing Magazine **12** (1995) 24–42
7. García, P., Segarra, E., Vidal, E., Galiano, I.: On the Use of the Morphic Generator Grammatical Inference (MGGI) Methodology in Automatic Speech Recognition. International Journal of Pattern Recognition and Artificial Intelligence **4** (1994) 667–685
8. Thollard, F., Dupont, P., de la Higuera, C.: Probabilistic DFA Inference using Kullback-Leibler Divergence and Minimality. In: Proc. of ICML '00. (2000) 975–982
9. Bunke, H., Sanfeliu, A., eds.: Syntactic and Structural Pattern Recognition, Theory and Applications. Volume 7. World Scientific (1990)
10. Fu, K.S., Booth, T.L.: Grammatical Inference: Introduction and Survey. Part I and II. IEEE Transactions on Syst. Man. and Cybern. **5** (1975) 59–72 and 409–423
11. Miclet, L.: Grammatical Inference. In: Syntactic and Structural Pattern Recognition, Theory and Applications. World Scientific (1990) 237–290
12. McAllester, D., Schapire, R.: Learning theory and language modeling. In: Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann (2002)
13. Fred, A.: Computation of Substring Probabilities in Stochastic Grammars. In: Grammatical Inference: Algorithms and Applications. Volume 1891. Springer-Verlag (2000) 103–114
14. Carrasco, R.C.: Accurate Computation of the Relative Entropy Between Stochastic Regular Grammars. RAIRO **31** (1997) 437–444
15. Carrasco, R.C., Rico-Juan, J.R.: A Similarity Between Probabilistic Tree Languages: Application to XML Document Families. Pattern Recognition, in press **36** (2002) 2197–2199
16. Reber, A.S.: Implicit Learning of Artificial Grammars. Journal of verbal learning and verbal behaviour **6** (1967) 855–863
17. Murgue, T., de la Higuera, C.: Distances Between Distributions: Comparing Language Models. Technical Report RR-0104, EURISE, Saint-Etienne, France (2004)
18. Kearns, M., Valiant, L.: Cryptographic Limitations on Learning Boolean Formulae and Finite Automata. In: 21st ACM Symposium on Theory of Computing. (1989) 433–444
19. MADCOW: Multi-Site Data Collection for a Spoken Language Corpus. In: Proc. DARPA Speech and Natural Language Workshop 92. (1992) 7–14
20. Carrasco, R., Oncina, J.: Learning Stochastic Regular Grammars by Means of a State Merging Method. In: Proc. of ICGI'94. (1994) 139–150