Computing The Kullback-Leibler Divergence Between Probabilistic Automata Using Rational Kernels

Rani Nelken
and
Stuart M. Shieber

TR-07-06

Computer Science Group
Harvard University
Cambridge, Massachusetts

Computing The Kullback-Leibler Divergence Between Probabilistic Automata Using Rational Kernels

Rani Nelken and Stuart M. Shieber
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138

March 3, 2006

Abstract

Kullback-Leibler divergence is a natural distance measure between two probabilistic finite-state automata. Computing this distance is difficult, since it requires a summation over a countably infinite number of strings. Nederhof and Satta (2004) recently provided a solution in the course of solving the more general problem of finding the cross-entropy between a probabilistic context-free grammar and an unambiguous probabilistic automaton. We propose a novel solution for two unambiguous probabilistic automata, by showing that Kullback-Leibler divergence can be defined as a rational kernel (Cortes et al., 2004) over the expectation semiring (Eisner, 2002). Using this definition, the computation is performed using the general algorithm for rational kernels, yielding an elegant and efficient solution.

1 Introduction

Kullback-Leibler (KL) divergence (or relative-entropy) is an asymmetric dissimilarity measure between two probability distributions, p and q. Intuitively speaking, it measures the added number of bits required for encoding events sampled from p using a code based on q. The need for such a measure arises quite naturally for probability distributions associated with language models in both theoretical and practical contexts. For instance, (Nederhof, 2005) is concerned with training one language model, such as a probabilistic finite state automaton (PFA) on the basis of another, such as a Probabilistic Context-Free Grammar (PCFG), while minimizing the KL-divergence between them. Quint (2004) describes an algorithm for checking the equivalence of two weighted automata, a property that could be verified by checking for a KL-divergence of 0. In the language modeling approach to Information Retrieval (IR) (Ponte and Croft, 1998), there has been growing interest in using KL-divergence to measure the distance between (the language model associated with) a query and members of a document collection (Lafferty and Zhai, 2001). In the IR literature, this distance is usually computed by restricting the computation to unigrams or other fixed-order *n*-grams, enabling a straightforward finite enumeration of all the *n*-grams. The more general case, however, requires a computation over an infinite set of strings that are assigned a probability by the language models.

Several authors have suggested that computing the KL-divergence between finite-state models cannot be solved analytically, and is difficult to compute numerically, leading them to suggest numerical approximations using Monte-Carlo simulations (Juang and Rabiner, 1985; Falkhausen et al., 1995). Recently, Nederhof and Satta (2004) solved the more general problem of computing the related measure of *cross-entropy* between a PCFG and a deterministic PFA. Since a PFA can be seen as a right linear PCFG, their solution can also be applied to a pair of PFAs, and can even be simplified. The difference between cross-entropy and KL-divergence is the entropy of the first argument. Nederhof and Satta show that this entropy cannot be effectively computed for a PCFG, but poses no problem for a PFA. Hence their algorithm yields a complete solution for finding the KL-divergence between two PFAs.

In this paper we propose an alternative elegant method of computing the KL-divergence between two unambiguous PFAs. Our main observation is that KL-divergence can be recast as an instance of a *rational kernel* (Cortes et al., 2004). Rational kernels provide a general way of specifying and efficiently computing diverse real-valued metrics over variable-length sequences and more generally, over weighted automata. They can be combined with Support Vector Machines (SVMs) (Vapnik, 1998) to efficiently classify weighted automata. Rational kernels are parameterized by several parameters, including a semiring, a transducer, and a function from the semiring to the real numbers. Crucially, we choose to use the *expectation semiring* (Eisner, 2002) as the semiring underlying the rational kernel. By virtue of this choice, the entire computation proceeds directly from (a slight generalization of) the general rational kernel algorithm, yielding a simple and efficient elegant solution.

This paper is structured as follows. We first provide the necessary definitions in Section 2, including semirings, automata (Kuich and Salomaa, 1986), and basic notions from information theory as applied to automata. Section 3 gives an overview of Nederhof and Satta's (2004) solution. Section 4 presents rational kernels (Cortes et al., 2004) and describes the construction of a rational kernel computing the KL-divergence for PFAs. Finally, we describe an implementation of the algorithm in Section 5.

2 Preliminaries

2.1 Semirings and automata

A system $(\mathbb{K}, \odot, e)$ is a *monoid* if $\mathbb{K}$ is closed under the binary relation $\odot$, $\odot$ is associative and e is an identity for $\odot$.

A *monoid morphism* is a function φ from one monoid, $(\mathbb{K}_1, \odot_1, e_1)$, to another, $(\mathbb{K}_2, \odot_2, e_2)$ respecting the monoid's operation, i.e. $\varphi(a \odot_1 b) = \varphi(a) \odot_2 \varphi(b)$, and its identity element, i.e. $\varphi(e_1) = e_2$.

A *semiring* is a system $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ such that: $(\mathbb{K}, \oplus, \overline{0})$ is a commutative monoid where $\overline{0}$ is the identity element of $\oplus$, $(\mathbb{K}, \otimes, \overline{1})$ is a monoid where $\overline{1}$ is the identity element of $\otimes$, $\otimes$ distributes over $\oplus$, and $\overline{0}$ is an annihilator for $\otimes$. A semiring is *closed* if for all $a \in \mathbb{K}$, the infinite sum $\oplus_{n=0}^{\infty} a^n$, written a^*, is well defined and in $\mathbb{K}$, and associativity, commutativity, and distributivity apply to countable sums. A semiring is k-closed if for all $a \in \mathbb{K}$, $\bigoplus_{n=0}^{k+1} a^n = \bigoplus_{n=0}^{k} a^n$.

A *weighted transducer*, T, over the semiring $\mathbb{K}$ is a tuple $(\Sigma, \Delta, Q, I, F, E)$ where[1] Σ is the finite input alphabet, Δ is the finite output alphabet, Q is a finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathbb{K} \times Q$ is a finite set of transitions. A *weighted automaton* is a weighted transducer in which the input and output labels on each transition are identical.

For a transition $e \in E$, let $w[e] \in \mathbb{K}$ be its weight. A path is an element of E^* with consecutive transitions. $Paths_A$ is the set of all paths in A from I to F. The weight of a path $\pi = e_1, \ldots e_k$ is $w[\pi] = w[e_1] \otimes \ldots \otimes w[e_k]$. For a string $x \in \Sigma^*$, $Paths_A(x)$ is the subset of $Paths_A$ labeled by x. The language accepted by A is $L(A) = \{x \in \Sigma^* \mid Paths_A(x) \neq \emptyset\}$. The weight assigned by A to x is $A(x) = \bigoplus_{\pi \in Paths_A(x)} w[\pi]$. We define $A(x) = \overline{0}$ for $x \notin L(A)$.

A weighted automaton is *unambiguous* if for each $x \in \Sigma^*$, $\mid Paths_A(x) \mid \leq 1$. It is *deterministic* if it has a unique initial state and if two transitions leaving any state share the same label.

A probabilistic finite automaton (*PFA*) A is a weighted automaton over the *real semiring* (sometimes called the *probability semiring*), $(\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$, that assigns

[1]Weighted transducers are usually presented with additional start and end weights, which without loss of generality we ignore here.

each string $x \in \Sigma^*$ a probability, $p_A(x) \in [0,1]$. It is *consistent* if $\sum_{x\in\Sigma^*} p_A(x) = 1$, inducing a probability distribution over Σ^*.

2.2 Entropy, relative-entropy, and cross-entropy

Soule (1974) applies the notion of entropy to PCFGs, and the same definitions apply to PFAs. The *entropy* of p_A is:

$$H(p_A) \overset{\text{def}}{=} -\sum_{x\in\Sigma^*} p_A(x)\log p_A(x) = -\sum_{x\in L(A)} p_A(x)\log p_A(x) \quad .$$

The KL-divergence (relative-entropy) from A to B is defined by:

$$D(A\,||\,B) \overset{\text{def}}{=} E_{p_A}\left(\log\frac{p_A(x)}{p_B(x)}\right) = \sum_{x\in\Sigma^*} p_A(x)\cdot\log\left(\frac{p_A(x)}{p_B(x)}\right)$$

where by convention if $p_A(x) = 0$, then $p_A(x)\cdot\log\left(\frac{p_A(x)}{p_B(x)}\right) = 0$. Otherwise, if $p_B(x) = 0$, then it is ∞.

If there is a word $w \in L(A)\setminus L(B)$, then $D(A\,||\,B) = \infty$, a situation we can easily check for by checking whether $L(A)\cap\overline{L(B)} = \emptyset$. Thus, we assume without loss of generality that $L(A)\subseteq L(B)$.

The cross-entropy from A to B is

$$H(A\,||\,B) \overset{\text{def}}{=} \sum_{x\in\Sigma^*} p_A(x)\frac{1}{\log p_B(x)} \quad .$$

Nederhof and Satta (2004) make use of the fact that relative entropy can be expressed as a sum of the cross-entropy and the entropy of the first argument: $D(A\,||\,B) = H(A\,||\,B) - H(A) =$

$$\sum_{x\in L(A)} p_A(x)\frac{1}{\log p_B(x)} + \sum_{x\in L(A)} p_A(x)\log p_A(x) \tag{1}$$

2.3 The expectation semiring

Eisner (2002) introduced the expectation semiring as a way to compute the E-step of the Expectation Maximization algorithm for training probabilistic transducers. Given a PFA A, each transition e, and by extension each path, π, is assigned not only a probability, $p_A(\pi)$, but also a value, $v_A(\pi)$, an element of some vector space, which we set here to be $\mathbb{R}_{\geq 0}$. The expectation of v_A, $E_{p_A}(v_A)$, can be computed using the expectation semiring: $(\mathbb{R}_{\geq 0}\times\mathbb{R}_{\geq 0}, \oplus, \otimes, \underline{0}, \underline{1})$, where the semiring operations are:

- $\langle p_1, v_1 \rangle \otimes \langle p_2, v_2 \rangle \overset{\text{def}}{=} \langle p_1 p_2, p_1 v_2 + p_2 v_1 \rangle$,
- $\langle p_1, v_1 \rangle \oplus \langle p_2, v_2 \rangle \overset{\text{def}}{=} \langle p_1 + p_2, v_1 + v_2 \rangle$,
- $\langle p, v \rangle^* \overset{\text{def}}{=} \langle p^*, p^* \cdot p^* v \rangle$ if p^* is defined,

and the additive and multiplicative identities are: $\overline{0} \overset{\text{def}}{=} \langle 0, 0 \rangle$ and $\overline{1} \overset{\text{def}}{=} \langle 1, 0 \rangle$.

If the transition weights of A are set to be elements of this semiring of the form $w[e] = \langle p_A(e), p_A(e) v_A(e) \rangle$, then the weight of a set of paths, Π, is

$$\left\langle \sum_{\pi \in \Pi} p_A(\pi)\ ,\ \sum_{\pi \in \Pi} p_A(\pi) v_A(\pi) \right\rangle \tag{2}$$

Thus, accumulated over $Paths_A$, the second coordinate of the weight is $E_{p_A}(v_A)$.

3 Nederhof and Satta's solution

Nederhof and Satta (2004) present an algorithm for computing the cross-entropy between a PCFG, G_p, and a deterministic PFA, M_p, defined by

$$H(G_p \,||\, M_p) = \sum_x p_G(x) \frac{1}{\log p_M(x)} \quad , \tag{3}$$

which combines the probabilities of each string being accepted by G_p and by M_p. Nederhof and Satta manage to reduce this to an expression involving the individual transitions of the PFA, of the form $s \overset{a}{\mapsto} t$. Obviously, the PFA's log probabilities for such transitions are known, but finding the expectation of these probabilities over the derivations of G_p is more challenging. To do so, they construct a PCFG, $G_{\cap,p}$, representing the weighted intersection (Nederhof and Satta, 2003)—a weighted extension of the classical unweighted construction (Bar-Hillel et al., 1961)—of G_p and the non-probabilistic automaton M underlying M_p. In particular, for each transition of M, $G_{\cap,p}$ includes a non-terminal $\langle s, a, t \rangle$ together with the rule, $\langle s, a, t \rangle \to a$ with probability 1. A derivation of $G_{\cap,p}$ simulates a derivation of the original PCFG as well as a path through the automaton, assigning it the same probability as G_p would. What we would like to know is the expected frequency of using the $\langle s, a, t \rangle \to a$ rule over all $G_{\cap,p}$ derivations. This is done by recursively defining *in* and *out* probabilities for each non-terminal of $G_{\cap,p}$, similarly to the decomposition used by the inside-outside algorithm (Lari and Young, 1990; Lari and Young, 1991). The required expected frequency is shown to be exactly $out_{G_{\cap,p}}(\langle s, a, t \rangle)$. They show that the cross-entropy can be given by

$$H(G_p \,||\, M_p) = \sum_{s \overset{a}{\mapsto} t} out_{G_{\cap},p}(\langle s, a, t \rangle) \cdot \log \frac{1}{p_M(s \overset{a}{\mapsto} t)} \tag{4}$$

Using Equation 1, the KL-divergence between G_p and M_p can be expressed as the difference between the cross-entropy of the PCFG and the PFA, and the entropy of the PCFG. Unfortunately, Nederhof and Satta show that this entropy term cannot be effectively computed for a PCFG. What can be computed is the *derivational entropy*, $H_d(G_p)$, which is the expectation of the information over all complete derivations of G_p. In general, $H(G_p) \leq H_d(G_p)$ with equality if and only if G is unambiguous (Soule, 1974). Since ambiguity of CFGs is undecidable, we cannot hope to be able to compute $H(G_P)$.

Since a PFA can be viewed as a right linear PCFG, the algorithm above can also compute the cross-entropy for a pair of deterministic PFAs. Given that the entropy of a PFA *can* be computed, this approach therefore yields a complete solution for computing the KL-divergence between a pair of PFAs. This is summarized in Figure 1. Moreover, the algorithm can be simplified for this case (Nederhof, 2005) to use the weighted intersection of two PFAs (Pereira and Riley, 1997). The computation in Equation 3 can be decomposed more efficiently in this case to states rather than transitions, using the *forward* and *backward* probabilities. These can be computed exactly by solving a system of linear equations.

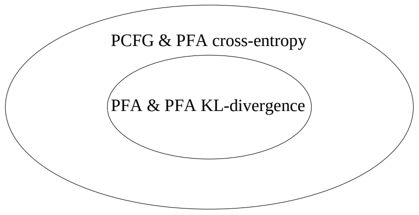


Figure 1: Relations computed by Nederhof and Satta's algorithm

What is the run-time complexity of this algorithm? The dominant factor is the solution of the system of linear equations. There are a pair of *forward* and *backward* equations for each state, for a total of $O(|\, Q_A \,| \cdot |\, Q_B \,|)$ linear equations.

Gaussian elimination takes cubic time, but faster matrix inversion algorithms based on sub-cubic matrix multiplication (Strassen, 1969) are known. These algorithms have complexity $O(n^{\alpha})$ for matrices of size n and $\alpha < 3$, making the overall complexity of Nederhof and Satta's algorithm $O((|Q_A| \cdot |Q_B|)^{\alpha})$. The best known exponent to date is $\alpha = 2.376$ (Coppersmith and Winograd, 1990).

We now turn to present rational kernels and our algorithm.

4 Rational kernels

Rational kernels (Cortes et al., 2004) generalize many distance measures over weighted automata. Let A and B be two weighted automata over the alphabets Σ and Δ, respectively, and the same semiring $\mathbb{K}$. A *rational kernel* is a function parameterized by a weighted transducer $T = (\Sigma, \Delta, Q, I, F, E)$ over $\mathbb{K}$, and a function $\psi : \mathbb{K} \to \mathbb{R}$ as follows:

$$K(A,B) = \psi\Big(\bigoplus_{\substack{x \in \Sigma^* \\ y \in \Delta^*}} A(x) \otimes T(x,y) \otimes B(y)\Big) \tag{5}$$

We would like to generalize the definition by allowing the semiring of the automata to differ from the kernel's semiring. In particular, we will use weighted automata over the real semiring, but perform the kernel's computation using the expectation semiring. Generally, if A and B are both defined over the semiring $\mathbb{K}'$ we extend the definition by adding a pair of $\otimes$-monoid morphisms $\varphi_A, \varphi_B : \mathbb{K}' \to \mathbb{K}$:

$$K(A,B) = \psi\Big(\bigoplus_{\substack{x \in \Sigma^* \\ y \in \Delta^*}} \varphi_A(A(x)) \otimes T(x,y) \otimes \varphi_B(B(y))\Big) \tag{6}$$

Cortes et al. give a general algorithm for computing rational kernels over weighted automata. The $\otimes$ operation in Equation 5 can be computed using the weighted composition (Pereira and Riley, 1997), of $A \circ T \circ B$ in time

$$O((|Q_A| + |E_A|)(|Q_B| + |E_B|))$$

(taking T to be of constant size). The $\oplus$ operation accumulates all the weights over all paths of the composition, and can be performed using the Floyd-Warshall algorithm for all-pairs-shortest-paths (Cormen et al., 1989) in time $O(|Q|^3)$ assuming constant time for computing the semiring's operations. Thus on the composition it runs in time $O((|Q_A| \cdot |Q_B|)^3)$. Clearly, this is the dominant factor in the time complexity.

Mohri (2002) presents a more efficient single-source shortest paths algorithm. Its exact complexity depends on several factors including the semiring and the queuing discipline. When A and B are guaranteed to be acyclic, the algorithm is linear in the size ($| Q | + | E |$) of the composed automaton.

Unfortunately, the algorithm can only be applied exactly to closed or k-closed semirings, which the expectation semiring is not. Mohri introduces a simple modification to the algorithm, which provides an approximate solution for general semirings, which is in practice orders of magnitude faster than the all-pairs-shortest-paths algorithm (Mohri, 2002).

4.1 The KL rational kernel

Let A and B be unambiguous PFAs (see Section 4.2) over the same alphabet Σ. We define the KL rational kernel to be:

$$K(A,B) = \psi(\bigoplus_{x,y \in \Sigma^*} \varphi_A(A(x)) \otimes I(x,y) \otimes \varphi_B(B(y))) \tag{7}$$

where I is the identity transducer over Σ^*, $\psi(\langle p, v \rangle) = v$, and the multiplicative monoid morphisms are defined by:

- $\varphi_A(p_A) = \langle p_A, p_A \log p_A \rangle$,
- $\varphi_B(p_B) = \begin{cases} \langle 0,0 \rangle & \text{if } p = 0 \\ \langle 1, -\log p_B \rangle & \text{otherwise.} \end{cases}$

It is easy to verify that these morphisms do indeed respect the multiplicative operations and the identity element.

With these morphisms, for each transition e_A in A and e_B in B labeled by the same symbol, $x \in \Sigma$, the weights on the composition $A \circ I \circ B$ are set to:

$$\varphi_A(p_A(e_A)) \otimes \varphi_B(p_B(e_B)) = \langle p_A(e_A), p_A(e_A) \log(\frac{p_A(e_A)}{p_B(e_B)}) \rangle \tag{8}$$

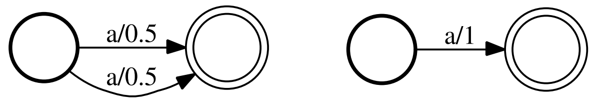
By Equation 2, the accumulation of all these weights over all possible paths yields

$$\langle \sum_{\pi \in \Pi} p_A(\pi), \sum_{\pi \in \Pi} p_A(\pi) \log(\frac{p_A(\pi)}{p_B(\pi)}) \rangle \quad .$$

Thus we get the total probability mass of p_A in the first coordinate and the KL-divergence in the second coordinate, which ψ proceeds to extract as the final result.

4.2 Ambiguity

We have restricted the algorithm to input PFAs that are both unambiguous. The correctness of the expectation computation is dependent on having only one path for each accepted string. A simple counter-example where one of the PFAs is ambiguous is shown in Figure 4.2. The composition $A \circ B$ would have two paths labeled with a, each with weight $\langle 0.5, 0.5 \log \frac{0.5}{1} \rangle$, the $\oplus$-sum of which is $\langle 1, -1 \rangle$ instead of the expected $\langle 1, 0 \rangle$.



(a) A (b) B

Figure 2: Two automata accepting the same language with the same probability distribution.

Strictly speaking, this is a more stringent requirement than the one imposed by Nederhof and Satta's algorithm, which allows the first argument (PCFG or PFA) to be ambiguous, but requires the second argument PFA to be unambiguous as summarized in Table 1. This difference stems from the way the probabilities and their logarithms are combined. Whereas in our approach they are intertwined, Equation 4 separates the logarithms from the *out* probabilities. Arguably, the extra unambiguity requirement imposed by our algorithm is not a major one, since both PFAs are likely to represent related languages; if one PFA can be guaranteed to be unambiguous, presumably so can the other.

PFA_2 / PFA_1	Ambiguous	Unambiguous
Ambiguous	-	Nederhof & Satta
Unambiguous	-	Nederhof & Satta and this paper

Table 1: Sensitivity of the algorithms to ambiguity of the inputs

In principle, given an ambiguous PFA, we can attempt to apply determinization, bearing in mind that (in contrast to the unweighted case) not every PFA is determinizable. Fortunately, determinizability can be determined efficiently (Allauzen and Mohri, 2003). The explosion in the number of states, however, makes determinization practical only for relatively small PFAs. Thus, the unambiguity requirement is essential.

4.3 ε-transitions

In the presence of ε-transitions, determinism does not imply unambiguity. For unambiguous automata with ε-transitions, ε-removal is not required, and the algorithm can be applied directly. This is a non-trivial property of the algorithm. We have to maintain the invariant that the computed weights of the ε-transition are of the form in Equation 8. Recall that ε-transitions can arise in the composition automaton in two ways (Pereira and Riley, 1997). They can be the result of composing an ε-transition, e_A, from A, with an ε-transition, e_B, from B, in which case the computation proceeds exactly as if it was a non-ε transition. They may also arise, however, from following an ε-transition in one of the PFAs, while remaining in the same state in the other. In this case, Pereira and Riley add a silent (τ) self-transition with weight 1. We have to verify that both $\varphi_A(1) \otimes \varphi_B(p_B(e_B))$ and $\varphi_A(p_A(e_A)) \otimes \varphi_B(1)$ are of the correct form. We take advantage of the fact that φ_A and φ_B were both defined as multiplicative monoid morphisms. Hence, $\varphi_A(1) \otimes \varphi_B(p_B(e_B)) = \overline{1} \otimes \varphi_B(p_B(e_B)) = \langle 1, 1 \cdot \log \frac{1}{p_B(e_B)} \rangle$ which is of the required form. Similarly for the symmetric case.

5 Implementation

The simplicity of our algorithm leads to a straightforward implementation, which we have proceeded to carry out. We used the Aachen open-source FSA toolkit (Kanthak and Ney, 2004), implemented in C++. The toolkit conveniently parameterizes the standard weighted automata operations such as composition by a semiring class. Thus, by introducing a class implementing the expectation semiring's operations, we automatically extend the automata operations to also work over this semiring.

To compute shortest paths, we can run the single-source-shortest-path algorithm of Mohri (2002) discussed in Section 4. Fortunately, this algorithm is already implemented as part of the toolkit's ε-removal algorithm (Mohri, 2000). Recall that the algorithm works only for closed or k-closed semirings, but that a small modification—which we made—yields an approximate solution for general semirings. As suggested by Eisner (2002) one can make use of the ε-removal operation directly by changing all the transition labels to ε, and applying ε-removal, which is what we did. (See Eisner (2002) for additional possible optimizations). All that remains is implementing the morphisms, which we did using some Python code and the AT&T FSM library (Mohri et al., 2000).

6 Conclusion

We have presented an elegant approach to computing the KL-divergence between two PFAs. Like Nederhof and Satta, our algorithm includes the two basic elements of intersection/composition and the computation of an expectation, but rather than solving a large system of linear equations, the rational kernels algorithm performs all the work. The computational complexity of both approaches is comparable. Nederhof and Satta's solution is sub-cubic in $\mid Q_A \mid \cdot \mid Q_B \mid$. An exact solution for our approach using the Floyd-Warshall algorithm is cubic. A practical approximate solution can be achieved significantly faster using Mohri's algorithm.

By defining the computation of the KL-divergence as a rational kernel, we can directly use it for classifying weighted automata using SVMs. Moreno et al. (2004) suggests using a kernel based on KL-divergence for classifying image or speech data, but used Monte Carlo simulation to approximate the divergence.

Finally, viewing KL-divergence as a rational kernel suggests interesting flexible variants. For instance, recall that A and B have to be defined on the same alphabet, a situation that is unrealistic if they represent language models trained on different corpora. We can relax this assumption by replacing the identity transducer, I, in Equation 7 with a more sophisticated transducer; e.g., one that maps words from A's language to words from B's or maps out-of-vocabulary items to a special symbol, with some appropriate penalty.

Acknowledgments

We thank Christopher Wickens for his help with implementing the algorithm described in this paper, and Stephen Kanthak for his help with the Aachen FSA toolkit. This work was supported in part by grant BCS-0236592 from the National Science Foundation.

References

Cyril Allauzen and Mehryar Mohri. 2003. Efficient algorithms for testing the twins property. *Journal of Automata, Languages and Combinatorics*, 8(2):117–144.

Yehoshua Bar-Hillel, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwis-*

senschaft und Kommunikationsforschung, 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

Don Coppersmith and Shmuel Winograd. 1990. Matrix multiplication via arithmetic programming. *Journal of Symbolic Computing*, 9:251–280.

Thomas Cormen, Charles Leiserson, and Ronald Rivest. 1989. *Introduction to Algorithms*. The MIT Press, Cambridge, MA.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. 2004. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, August.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, July.

M. Falkhausen, H. Reininger, and D. Wolf. 1995. Calculation of distance measures between hidden Markov models. In *Proceedings of Eurospeech '95*, pages 1487–1490, Madrid.

Biing-Hwang Juang and Lawrence R. Rabiner. 1985. A probabilistic distance measure for hidden Markov models. *AT&T Technical Journal*, 64(2):391–408, February.

Stephan Kanthak and Hermann Ney. 2004. FSA: An efficient and flexible C++ toolkit for finite state automata using on-demand computation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 510–517.

Werner Kuich and Arto Salomaa. 1986. *Semirings, Automata, Languages*. Springer-Verlag, Berlin, Germany. EATCS Monographs On Theoretical Computer Science; Vol. 5.

John D. Lafferty and ChengXiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR*, pages 111–119. ACM.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

K. Lari and S. J. Young. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 5:237–257.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Mehryar Mohri. 2000. Generic Epsilon-removal algorithm for weighted automata. In *CIAA: International Conference on Implementation and Application of Automata, LNCS*.

Mehryar Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos. 2004. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Mark-Jan Nederhof and Giorgio Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, pages 137–148, LORIA, Nancy, France, April.

Mark-Jan Nederhof and Giorgio Satta. 2004. Kullback-Leibler distance between probabilistic context-free grammars and probabilistic finite automata. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING 2004*, pages 71–77, Geneva, Switzerland, August.

Mark-Jan Nederhof. 2005. A general technique to train language models on language models. *Computational Linguistics*, 31(2):173–185, June.

Fernando C. N. Pereira and Michael D. Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, pages 431–453. MIT Press, Cambridge, Massachusetts.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 275–281. ACM.

Julien Quint. 2004. On the equivalence of weighted finite-state transducers. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 182–185, Barcelona, Spain, July. Association for Computational Linguistics.

Stephen Soule. 1974. Entropies of probabilistic grammars. *Information and Control*, 25:57–74.

Volker Strassen. 1969. Gaussian elimination is not optimal. *Numerische Mathematik*, 14(3):354–356.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Springer-Verlag, New York.