Learning from Uncertain Data

Mehryar Mohri

AT&T Labs – Research
180 Park Avenue, Florham Park, NJ 07932, USA
mohri@research.att.com

Abstract. The application of statistical methods to natural language processing has been remarkably successful over the past two decades. But, to deal with recent problems arising in this field, machine learning techniques must be generalized to deal with *uncertain data*, or datasets whose elements are distributions over sequences, such as weighted automata. This paper reviews a number of recent results related to this question. We discuss how to compute efficiently basic statistics from a weighted automaton such as the expected count of an arbitrary sequence and higher moments of that distribution, by using weighted transducers. Both the corresponding transducers and related algorithms are described. We show how general classification techniques such as Support Vector Machines can be extended to deal with distributions by using general kernels between weighted automata. We describe several examples of positive definite kernels between weighted automata such as kernels based on counts of common n-gram sequences, counts of common factors or suffixes, or other more complex kernels, and describe a general algorithm for computing them efficiently. We also demonstrate how machine learning techniques such as clustering based on the edit-distance can be extended to deal with unweighted and weighted automata representing distributions.

1 Introduction

The application of statistical methods to natural language processing has been remarkably successful over the past two decades. Many of the components of speech recognition systems, language models, pronunciation models, context-dependency models, Hidden-Markov Models (HMMs), are statistical models [10, 25, 22]. Sophisticated statistical learning techniques have also been used in all other areas of natural language processing from the design of high-accuracy statistical parsers [3, 4, 26] and morphological analyzers to that of accurate text classification systems [27, 11].

As for all machine learning techniques, these methods heavily rely on data and the availability of text and speech corpora in several areas has played a critical role in their success. But, new machine learning techniques in natural language processing must deal with *uncertain data*. To illustrate this, consider the case of large-vocabulary speech recognition systems designed to transcribe broadcast programs. A virtually unlimited amount of unlabeled audio data can

be collected from a television feed. An existing speech recognizer can be applied to this data to produce, for each speech utterance, a set of uncertain alternative transcriptions, typically represented by a weighted automaton [13]. Such an automaton can then be used as a *label* for the corresponding speech utterance and constitute the input data for training acoustic or grammar models.

A similar situation can be found in the case of a complex information extraction system or a complex search engine. These systems are typically not fully certain about the correct response. This may be because of the ambiguities affecting the input query, which are quite common in natural language, or because the complex information sources used by the system just cannot uniquely determine the response. Thus, they generate a range of alternative hypotheses with some associated weights or probabilities used to rank these hypotheses. When the accuracy of a system is relatively low, it is not safe to rely only on the best hypothesis output by the system. It is then preferable to use instead the full range of hypotheses and their weights since that set most often contains the correct response.

These observations apply similarly to many other areas of natural language processing and in fact to other domains such as computational biology. The input data for many machine learning techniques in such contexts is *uncertain* and can be viewed as a set of *distributions*. More specifically, in natural language processing, these are distributions over sequences, such as sequences of words or phonemes. Thus, statistical learning techniques must be generalized to deal with distributions of strings, or sequences.

The data uncertainty and the distributions we are referring to here should not be confused with the sampling distribution inherent to any learning problem – that distribution is typically assumed to be *i.i.d.* and the corresponding assumptions hold similarly in our case. Here, each object of the input data is a distribution. To further clarify the problem we are describing, consider the toy classification problem of predicting gender based on height and weight. The input data for this problem is typically a set of pairs (height, weight) for a large sample of the population. The problem we are facing here is similar to having instead for each member of that population not the exact height or weight but a height or weight distribution.

It is possible to reformulate this problem as a classical machine learning problem if we resort to an approximation. Indeed, we can extract random samples from each distribution and use the samples to augment the feature set. The sample sizes must be sufficiently large to be representative of each distribution. But, this may dramatically affect the efficiency of the learning algorithm by significantly increasing the size of the feature set, while still solving only an approximate problem. In what follows, we are interested in exact solutions and efficient algorithms exploiting the input distributions directly.

In most cases, such as those just discussed, the distributions can be represented by weighted automata, in fact acyclic, since their support is a large but finite number of sequences. Dealing with objects represented by weighted

automata arises many new algorithmic questions. This paper discusses and reviews a number of recent results related to these questions [7, 6, 21].

A general question for any statistical learning technique dealing with weighted automata is that of collecting statistics. How can we count efficiently sequences appearing in weighted automata? How do we take into account the weight or probability associated to each path? We present simple algorithms for computing efficiently the expected count and even higher moments of the count of an arbitrary sequence in a weighted automaton.

The application of discriminant classification algorithms to weighted automata arises other issues. We briefly describe a general kernel framework, *rational kernels*, that extends kernel methods to the analysis of weighted automata. These kernels can be computed efficiently and include many of the kernels introduced in text classification and computational biology [9, 30, 16, 14]. They have been used successfully in applications such as spoken-dialog classification. We give several examples of positive definite rational kernels and illustrate their generality.

We also show how machine learning techniques such as clustering based on the edit-distance can be generalized to deal with unweighted and weighted automata representing distributions. We extend the definition of the edit-distance to automata and describe efficient algorithms for the computation of the edit-distance and the longest common subsequence of two automata. We start with some basic definitions and notation related to weighted automata and transducers.

2 Preliminaries

Definition 1 ([12]). *A system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is a* semiring *if* $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid with identity element* $\overline{0}$*,* $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid with identity element* $\overline{1}$*,* $\otimes$ *distributes over* $\oplus$*, and* $\overline{0}$ *is an annihilator for* $\otimes$*: for all* $a \in \mathbb{K}, a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$*.*

Thus, a semiring is a ring that may lack negation. Some familiar examples are the Boolean semiring $\mathcal{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$, or the probability semiring $\mathcal{R} = (\mathbb{R}_+, +, \times, 0, 1)$ used to combine probabilities. Two semirings often used in natural language processing are: the *log semiring* $\mathcal{L} = (\mathbb{R} \cup \{\infty\}, \oplus_{\log}, +, \infty, 0)$ [20] which is isomorphic to $\mathcal{R}$ via a log morphism with: $\forall a, b \in \mathbb{R} \cup \{\infty\}, a \oplus_{\log} b = -\log(\exp(-a) + \exp(-b))$ (by convention: $\exp(-\infty) = 0$ and $-\log(0) = \infty$), and the *tropical semiring* $\mathcal{T} = (\mathbb{R}_+ \cup \{\infty\}, \min, +, \infty, 0)$ which is derived from the log semiring using the Viterbi approximation.

Definition 2. *A* weighted finite-state transducer T *over a semiring* $\mathbb{K}$ *is an 8-tuple* $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ *where* Σ *is the finite input alphabet of the transducer,* Δ *is the finite output alphabet,* Q *is a finite set of states,* $I \subseteq Q$ *the set of initial states,* $F \subseteq Q$ *the set of final states,* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ *a finite set of transitions,* $\lambda : I \to \mathbb{K}$ *the initial weight function, and* $\rho : F \to \mathbb{K}$ *the final weight function mapping* F *to* $\mathbb{K}$*.*

A *Weighted automaton* $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ is defined in a similar way by simply omitting the input or output labels. We denote by $L(A) \subseteq \Sigma^*$ the set of strings accepted by an automaton A and similarly by $L(X)$ the strings described by a regular expression X.

Given a transition $e \in E$, we denote by $i[e]$ its input label, $p[e]$ its origin or previous state and $n[e]$ its destination state or next state, $w[e]$ its weight, $o[e]$ its output label (transducer case). Given a state $q \in Q$, we denote by $E[q]$ the set of transitions leaving q.

A *path* $\pi = e_1 \cdots e_k$ is an element of E^* with consecutive transitions: $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend n and p to paths by setting: $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. A *successful path* in a weighted automaton or transducer is a path from an initial state to a final state. We denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, q')$ and $P(q, x, y, q')$ the set of paths from q to q' with input label $x \in \Sigma^*$ and output label y (transducer case). These definitions can be extended to subsets $R, R' \subseteq Q$, by: $P(R, x, R') = \cup_{q \in R,\, q' \in R'} P(q, x, q')$. The labeling functions i (and similarly o) and the weight function w can also be extended to paths by defining the label of a path as the concatenation of the labels of its constituent transitions, and the weight of a path as the $\otimes$-product of the weights of its constituent transitions: $i[\pi] = i[e_1] \cdots i[e_k]$, $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. We also extend w to any finite set of paths Π by setting: $w[\Pi] = \bigoplus_{\pi \in \Pi} w[\pi]$, and even to an automaton or transducer M by $w[M] = \bigoplus_{\pi \in \Pi_M} w[\pi]$, where Π_M is the set of successful paths of M. An automaton A is *regulated* if the output weight associated by A to each input string $x \in \Sigma^*$:

$$[\![A]\!](x) = \bigoplus_{\pi \in P(I, x, F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) \tag{1}$$

is well-defined and in $\mathbb{K}$. This condition is always satisfied when A contains no ϵ-cycle since the sum then runs over a finite number of paths. It is also always satisfied with *k-closed* semirings such as the tropical semiring [20]. $[\![A]\!](x)$ is defined to be $\overline{0}$ when $P(I, x, F) = \emptyset$.

Similarly, a transducer T is *regulated* if the output weight associated by T to any pair of input-output string (x, y) by:

$$[\![T]\!](x, y) = \bigoplus_{\pi \in P(I, x, y, F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) \tag{2}$$

is well-defined and in $\mathbb{K}$. $[\![T]\!](x, y) = \overline{0}$ when $P(I, x, y, F) = \emptyset$. In the following, we will assume that all the automata and transducers considered are regulated. We denote by $|M|$ the sum of the number of states and transitions of an automaton or transducer M.

For any transducer T, we denote by $\Pi_2(T)$ the automaton obtained by projecting T on its output, that is by omitting its input labels.

3 Collecting Statistics

Statistical methods used in natural language processing are typically based on very large amounts of data, e.g., statistical language models are derived from text corpora of several million words. The first step for the creation of these models is the computation of the counts of some sequences. For language models used in speech recognition, these sequences are often n-gram sequences, but other applications may require the computation of the counts of non-contiguous units or even that of sequences given by a regular expression.

In new applications such as task adaptation, one needs to construct these models from distributions of sequences. For example, one may need to construct a language model based on the output of a speech recognition system, which is an acyclic weighted automaton called a *word* or a *phone lattice*. In traditional language modeling tasks, the count of the relevant sequences is computed from the input text. To deal with distributions of sequences instead of just text, we need to compute efficiently the *expected count* of a sequence or a set of sequences.[1]

A weighted automaton contains typically a very large set of alternative sequences with corresponding weights or probabilities. Collecting such statistics cannot be done by counting the number of occurrences of the sequence considered in each path since the number of paths of even a small automaton may be more than a billion. We present a simple and efficient algorithm for computing efficiently the expected count and higher moments of the count of an arbitrary sequence appearing in a weighted automaton.

3.1 Definition

Let $A = (Q, I, F, \Sigma, \delta, \sigma, \lambda, \rho)$ be an arbitrary weighted automaton over the probability semiring and let X be a regular expression defined over the alphabet Σ. We are interested in *counting* the occurrences of the sequences $x \in L(X)$ in A while taking into account the weight of the paths where they appear.

When A is *stochastic*, i.e. when it is deterministic and the sum of the weights of the transitions leaving any state is 1, it can be viewed as a probability distribution P over all strings Σ^*.[2] The weight $[\![A]\!](u)$ associated by A to a string $u \in \Sigma^*$ is then $P(u)$. Thus, we define the *expected count* of the sequence x in A, $c(x)$, as:

$$c(x) = \sum_{u \in \Sigma^*} |u|_x \, P(u) = \sum_{u \in \Sigma^*} |u|_x \, [\![A]\!](u) \tag{3}$$

where $|u|_x$ denotes the number of occurrences of x in the string u. We will define the count of x as above regardless of whether A is stochastic or not. More

[1] Many modeling algorithms can be naturally generalized to deal with input distributions by replacing the quantity X originally derived from text by its expectation $E[X]$ based on the probability distribution considered.

[2] There exist a general weighted determinization and a weight pushing algorithm that can be used to create a deterministic and stochastic automaton equivalent to an input weighted automaton [17].

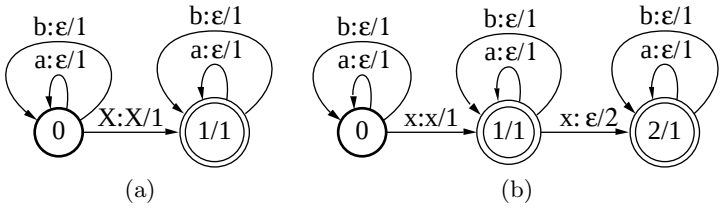

Fig. 1. Weighted transducers over the probability semiring used for counting with the alphabet $\Sigma = \{a, b\}$. (a) Transducer T_1 used to compute the expected counts of the sequences $x \in L(X)$. The transition weights and the final weight at state 1 are all equal to 1. (b) Weighted transducer T_2 used to compute $c_2(x)$, the second moment of the counts of an aperiodic sequence $x \in L(X)$.

generally, for $m \geq 1$, the *m-th moment of the count* of the sequence x is defined by:

$$c_m(x) = \sum_{u \in \Sigma^*} |u|_x^m \, [\![A]\!](u) \tag{4}$$

In many applications, the weighted automaton A is acyclic, e.g., it is the output of a speech recognition system. But our algorithm is general and does not assume A to be acyclic.

3.2 Algorithm

We describe an algorithm for computing the expected counts of the sequences $x \in L(X)$. Let A_1 be a weighted automaton over the probability semiring representing the weighted regular expression (or *formal power series* [12]) $\Sigma^* x \Sigma^*$. Figure 1(a) shows a weighted transducer whose input automaton is A_1 if $X = x$ and $\Sigma = \{a, b\}$.

Lemma 1. *For all $u \in \Sigma^*$, $[\![A_1]\!](u) = |u|_x$.*

Proof. Let $u \in \Sigma^*$. For any occurrence of x in u, u can be decomposed into $u = u_1 x u_2$, with $u_1, u_2 \in \Sigma^*$. Thus, for any occurrence of x in u, A_1 contains one distinct path labeled with u and with weight 1. Since $[\![A_1]\!](u)$ is the sum of the weights of all these paths, this proves the lemma. □

Since X is regular, the weighted transduction defined by $(\Sigma \times \{\epsilon\})^*(X \times X)(\Sigma \times \{\epsilon\})^*$ is rational. Thus, by the theorem of Schützenberger [28], there exists a weighted transducer T_1 defined over the alphabet Σ and the probability semiring realizing that transduction. Figure 1(a) shows the transducer T_1 in the particular case of $\Sigma = \{a, b\}$.

Proposition 1. *Let A be a weighted automaton over the probability semiring, then:*

$$[\![\Pi_2(A \circ T_1)]\!](x) = c(x)$$

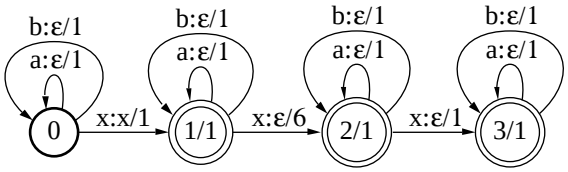

Fig. 2. Weighted transducer T_3 used to compute $c_3(x)$, the third moment of the counts of an aperiodic sequence $x \in L(X)$.

Proof. By definition of T_1, for any $u \in \Sigma^*$, $[\![T_1]\!](u, x) = [\![A_1]\!](x)$, and by Lemma 1, $[\![A_1]\!](x) = |u|_x$. Thus, by definition of composition:

$$[\![\Pi_2(A \circ T_1)]\!](x) = \sum_{\pi \in P(I,F),\, u = i[\pi]} [\![A]\!](u) \times |u|_x = \sum_{u \in \Sigma^*} |u|_x \, [\![A]\!](u) = c(x)$$

This ends the proof of the proposition. □

The weighted automaton $B = \Pi_2(A \circ T_1)$ contains ϵ-transitions. A general ϵ-removal algorithm can be used to compute an equivalent weighted automaton with no ϵ-transition [19]. The computation of $[\![B]\!](x)$ for a given x is done by composing B with an automaton representing x and by using a simple shortest-distance algorithm [20] to compute the sum of the weights of all the paths of the result.

The proposition gives a simple algorithm for computing the expected counts of X in a weighted automaton A based on several general and well-studied algorithms: composition algorithm for weighted transducers [23], projection of weighted transducers, ϵ-removal of weighted transducers, forward-backward algorithm or shortest-distance algorithm.

The algorithm is also based on the transducer T_1 which is quite easy to construct. The size of T_1 is in $O(|\Sigma| + |A_X|)$, where A_X is a finite automaton accepting X. With a lazy implementation of T_1, only one transition can be used instead of $|\Sigma|$, thereby reducing the size of the representation of T_1 to $O(|A_X|)$. The worst case complexity of composition is quadratic and that of projection is linear, thus the time complexity of the construction of the automaton B giving the expected counts of sequences $x \in L(X)$ is $O(|A||A_X|)$.

One can compute other moments of the count of a sequence x in a similar way. Indeed, let A_m, $m \geq 1$, be the automaton obtained by composing or intersecting A_1 with itself $m-1$ times, and let T_m be a weighted transducer corresponding to $A_m \times \{x\}$, then T_m can be used to compute the *m-th* moment of the count of x.

Corollary 1. *Let A be a weighted automaton over the probability semiring, then:*

$$[\![\Pi_2(A \circ T_m)]\!](x) = c_m(x)$$

Proof. In view of lemma 1, $[\![A_1]\!](u) = |u|_x$. Thus, by definition of composition:

$$\forall u \in \Sigma^*, [\![A_m]\!](u) = |u|_x^m \tag{5}$$

The result follows. □

In the worst case, the size of the composition of an automaton with itself may be quadratic in the size of the original automaton. But, it can be shown that there exists a compact representation of A_m whose size is in $O(m(|x| + \Sigma))$, thus the size of a lazy representation of T_m is in $O(m|x|)$. Figure 1(a) shows the transducer T_2 and Figure 2 the transducer T_3 for an aperiodic string x, when the alphabet is reduced to $\Sigma = \{a, b\}$.

These transducers can be used to compute efficiently the moments of the counts of sequences appearing in weighted automata, which are needed for the design of statistical models derived from distributions of sequences given by weighted automata.

4 Classification

Classification is a key task in many natural language processing applications. In document or text classification, it may consist of assigning a specific topic to each document, or to a set of sentences. Typically, features such as some specific sequences are extracted from these sentences and used by a machine learning algorithm. In spoken-dialog systems, the task consists of assigning to each speech utterance a category, out of a finite set, based on the output of a speech recognizer for that utterance. Due to the word error rate of conversational speech recognizers, it is preferable to use to full output of the recognizer which is a weighted automaton containing a large set of alternative transcriptions with their corresponding weights.

Recently, we introduced a general kernel framework based on weighted transducers, *rational kernels*, to extend kernel methods to deal with weighted automata [7, 6]. These kernels can be used in combination with a statistical learning technique such as Support Vector Machines (SVMs) [2, 8, 29] for efficient classification of weighted automata. This section briefly introduces rational kernels and describes several examples of positive definite rational kernels that can be used in a variety of applications.

4.1 Definition

Let Ω denote the set of weighted automata over the alphabet Σ.

Definition 3. *$K : \Omega \times \Omega \to \mathbb{R}$ is a* rational kernel *if there exists a weighted transducer T and a function $\psi : \mathbb{R} \to \mathbb{R}$ such that for all $X, Y \in \Omega$:*

$$K(X, Y) = \psi(w[X \circ T \circ Y]) \tag{6}$$

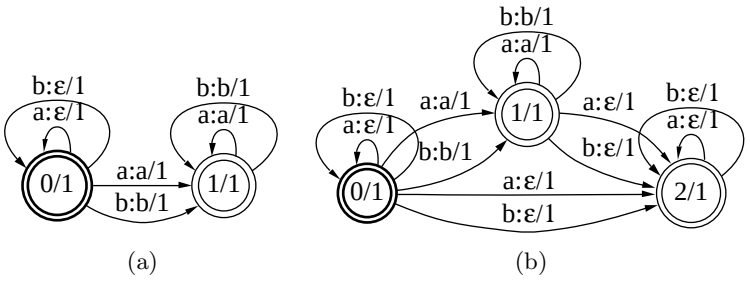

Fig. 3. (a) Weighted transducer T_s mapping any sequence x to the set of suffixes of x. (b) Weighted transducer T_f mapping any sequence x to the set of factors of x.

No special assumption is made about the function ψ in this definition. In most cases of interest, however, ψ is simply the identity function. Since the worst cost complexity of composition is quadratic, assuming that ψ can be computed in constant time, the cost of the computation of $K(X, Y)$ is in $O(|T||X||Y|)$. This complexity can be improved to be linear in $|X|$ and $|Y|$ in the case of automata representing strings by using failure functions.

Rational kernels define a general set of kernels based on weighted transducers that can be computed efficiently using composition. But not all rational kernels are *positive definite symmetric* (PDS), or equivalently verify Mercer's condition [1], a condition that guarantees the convergence to a global optimum of discriminant classification algorithms such as Support Vector Machines (SVMs). A kernel K is a PDS kernel iff the matrix $K(x_i, x_j)_{i,j \leq n}$ for all $n \geq 1$ and all $\{x_1, \ldots, x_n\} \subseteq X$ is symmetric and all its eigenvalues are non-negative. There exists however a systematic method for constructing a PDS kernel. Assume that the function ψ in the definition of rational kernels is a continuous morphism. Then, the following result [6] shows that one can in fact construct a PDS rational kernel from an arbitrary weighted transducer T. The construction is based on the composition of T with its inverse T^{-1}, that is the transducer obtained from T by swapping input and output labels of each transition.

Proposition 2 ([6]). *Let T be a weighted finite-state transducer and assume that the weighted transducer $T \circ T^{-1}$ is well-defined, then $T \circ T^{-1}$ defines a PDS rational kernel over $\Sigma^* \times \Sigma^*$.*

4.2 Examples of Rational Kernels

A kernel can be viewed as a measure of similarity between two elements. A common method used for the definition of kernels is based on the idea that two sequences are similar if they share many common subsequences of some kind. This idea can be generalized to weighted automata by considering the expected counts of the subsequences of the paths of an automaton instead of the

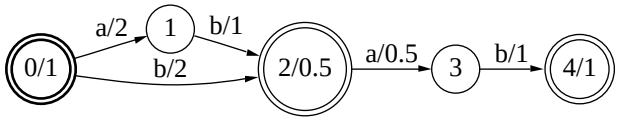

Fig. 4. Result of the application of the transducer T_s to an acceptor for the sequence *abab*. This weighted automaton contains all the suffixes of the string *abab* with their multiplicities.

counts. We present several examples of construction of PDS rational kernels for automata.

Suffix kernel. The transduction from Σ^* to Σ^* which associates to each string the set of its suffixes is rational. Figure 3(a) shows a weighted transducer, T_s, realizing that transduction. By Proposition 2, we can use T_s to define a PDS rational kernel K_s measuring the similarity of two weighted automata X and Y based on the expected counts of the common suffixes of the sequences of X and Y by

$$K_s(X, Y) = w[X \circ (T_s \circ {T_s}^{-1}) \circ Y] \tag{7}$$

We can define similarly a PDS rational kernel based on prefixes. Figure 4 shows the result of the application of the transducer T_s to the input sequence *abab* after ϵ-removal and determinization.

Factor kernel. Similarly, the transduction from Σ^* to Σ^* which associates to each string the set of its factors is rational.[3] Figure 3(b) shows a weighted transducer, T_f, realizing that transduction. By Proposition 2, we can use T_f to define a PDS rational kernel K_f measuring the similarity of two weighted automata X and Y based on the expected counts of the common factors of the sequences of X and Y.

Gappy n-gram kernels. Measures of similarity are often based on common n-gram sequences. Since the set of n-gram sequences over a finite alphabet is a regular language, by Proposition 1, there exists a simple weighted transducer that can be used to compute the expected counts of n-gram sequences of a weighted automaton A. That transducer can be slightly modified to compute the expected counts of non-contiguous or gappy n-grams using a decay factor λ, $0 \leq \lambda < 1$, penalizing long gaps. Figure 5 shows a weighted transducer over the alphabet $\Sigma = \{a, b\}$, T_λ, that computes the expected counts of gappy trigrams with a decay factor λ as defined by [16]. By Proposition 2, the composed transducer $T_\lambda \circ T_\lambda^{-1}$ defines a PDS kernel measuring the similarity of two weighted automata based on the expected counts of their common gappy n-grams.

[3] A *factor* f of a string x is a subsequence of x with contiguous symbols. Thus, if f is a factor of x, x can be written as $x = x_1 f x_2$ for some $x_1, x_2 \in \Sigma^*$.

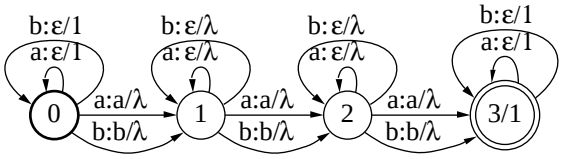

Fig. 5. Weighted transducer T_λ over the probability semiring computing the expected counts of gappy trigrams with a decay factor $0 \leq \lambda < 1$.

More powerful n-gram kernels. The results of Section 3 can be used to design more complex PDS rational kernels. The previous examples were all based on the expected counts of some subsequences. By Corollary 1, one can use a weighted transducer to compute other moments of the counts of n-gram sequences. That transducer can be composed with its inverse to construct a PDS rational kernel. Thus, we can define more general rational kernels based on both the expectation and the variance of common n-gram sequences between two automata. These kernels are likely to lead to a more refined classification.

We have reported elsewhere the results of experiments using SVMs with n-gram kernels in a spoken-dialog classification task [7]. The results show the benefits of the use of kernels applied to weighted automata over the use of just the best paths of these machines.

5 Clustering

The significant increase in size of text and speech datasets, in some cases data generated by continuous streams, has created an even stronger need for clustering, in particular because of its use for data summarization. Clustering consists of grouping objects of a large dataset into classes based on a metric, or a similarity measure. Many of the kernels presented in the previous section can be used as a similarity measure for clustering algorithms. Another metric relevant to clustering often used in natural language processing applications is that of the *edit-distance*, that is the minimal cost of a series of edit operations (symbol insertions, deletions, or substitutions) transforming one sequence into the other [15], e.g., the accuracy of a speech recognition system for an input speech utterance is often measured by the edit-distance between the output of the recognizer and the correct transcription.

To use the notion of edit-distance and apply clustering algorithms to uncertain data where each element is represented by a set of alternatives, we need to extend the edit-distance to that of a distance between two languages, the sets of sequences accepted by two automata and provide an efficient algorithm for its computation. Let $d(x, y)$ denote the edit-distance between two strings x and y over the alphabet Σ. A natural extension of the definition of the edit-distance which coincides with the usual definition of the distance between two subsets of a metric space is given by the following.

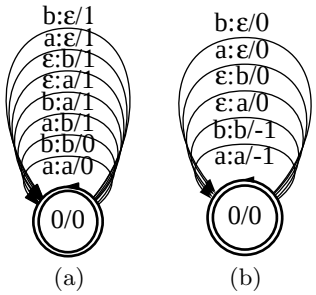

Fig. 6. (a) Weighted transducer T_e over the tropical semiring computing the edit-distance of two strings or automata. (b) Weighted transducer T_s over the tropical semiring computing the longest common subsequence of two strings of two automata.

Definition 4 ([21]). *The* edit-distance of two languages $X \subseteq \Sigma^*$ and $Y \subseteq \Sigma^*$ *is denoted by* $d(X, Y)$ *and defined by:*

$$d(X, Y) = \inf \{d(x, y) : x \in X,\, y \in Y\} \tag{8}$$

This definition can be naturally generalized to (unweighted) automata: the edit-distance of two automata A_1 and A_2 is that of the languages accepted by these automata $d(L(A_1), L(A_2))$. The general problem of the computation of the edit-distance between two languages is not trivial. In fact, it can be proven that this problem is *undecidable* for arbitrary context-free languages [21]. But the edit-distance of two automata can be computed efficiently using a weighted transducer over the tropical semiring [24, 21].

Let U be the weighted transducer over the tropical semiring defined by:

$$\forall a, b \in \Sigma \cup \{\epsilon\}\,,\ [\![U]\!](a, b) = \begin{cases} 1 & \text{if } (a \neq b) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

and define T_e by $T_e = U^*$. Figure 6(a) shows that transducer for $\Sigma = \{a, b\}$.

Proposition 3 ([24, 21]). *Let* A_1 *and* A_2 *be two (unweighted) finite automata over the alphabet* Σ*. Then,*

$$d(A_1, A_2) = w[A_1 \circ T_e \circ A_2] \tag{10}$$

Proof. By definition of U, $A_1 \circ T_e \circ A_2$ contains a successful path corresponding to each alignment of a sequence accepted by A_1 and a sequence accepted by A_2 and the weight of that path is exactly the cost of that alignment, i.e. the number of mismatches in that alignment. Since $A_1 \circ T_e \circ A_2$ is defined over the tropical semiring, $w[A_1 \circ T_e \circ A_2]$ is the minimum alignment cost, which is the edit-distance. □

Note that the best path of $A_1 \circ T_e \circ A_2$ provides also the best alignment of the two automata A_1 and A_2. The proposition leads to an efficient algorithm for computing the edit-distance of two automata based on the composition algorithm and a classical single-source shortest-paths algorithm. This computation

can be done on-the-fly since composition admits a natural on-the-fly implementation. Since the worst case complexity of composition is quadratic, and since the transducer T_e can be constructed on-demand (not all symbols of Σ might be needed for a specific choice of A_1 and A_2), the time and space complexity of the construction of $A_1 \circ T_e \circ A_2$ is $O(|A_1||A_2|)$. When A_1 and A_2 are acyclic, the result of composition is also an acyclic transducer and the shortest path can be found using the classical linear-time single-source shortest paths algorithm [5]. Thus, the total time complexity of the algorithm is $O(|A_1||A_2|)$.

The notion of edit-distance can be similarly extended to measure the similarity between two distributions of sequences given by weighted automata. It can be defined as the expected edit-distance of the strings accepted by A_1 and A_2.

Definition 5 ([21]). *Let A_1 and A_2 be two acyclic weighted automata over the probability semiring. Then the* edit-distance of A_1 and A_2 *is denoted by $d(A_1, A_2)$ and defined by:*

$$d(A_1, A_2) = \sum_{x,y} d(x, y)[\![A_1]\!](x)[\![A_2]\!](y) \tag{11}$$

It can be shown that this generalized definition of the edit-distance can also be computed *exactly* using standard weighted automata algorithms such as weighted determinization, synchronization, and ϵ-removal [21].

A notion closely related to the edit-distance of two strings is that of the *longest common subsequence*, that is the longest subsequence of symbols (not necessarily contiguous) common to two strings. Let $lcs(x, y)$ denote the longest common subsequence of x and y. This definition can be naturally extended to two languages or two finite automata.

Definition 6. *The* longest common subsequence of two languages $X \subseteq \Sigma^*$ and $Y \subseteq \Sigma^*$ *is denoted by $lcs(X, Y)$ and defined by:*

$$lcs(X, Y) = \sup \{lcs(x, y) : x \in X,\, y \in Y\} \tag{12}$$

The longest common subsequence of two finite automata A_1 and A_2 over the alphabet Σ is denoted by $lcs(A_1, A_2)$ and defined by $lcs(A_1, A_2) = lcs(L(A_1), L(A_2))$.

Let U' be the weighted transducer over the tropical semiring defined by:

$$\forall a \in \Sigma,\ [\![U']\!](a, \epsilon) = [\![U']\!](\epsilon, a) = 0 \tag{13}$$

$$[\![U']\!](a, a) = -1 \tag{14}$$

and define T_s by $T_s = U'^*$. Figure 6(b) shows that transducer for $\Sigma = \{a, b\}$.

Proposition 4. *Let A_1 and A_2 be two (unweighted) finite automata over the alphabet Σ. Then,*

$$lcs(A_1, A_2) = -w[A_1 \circ T_s \circ A_2] \tag{15}$$

Proof. The proof is similar to that of Proposition 4. $A_1 \circ T_s \circ A_2$ contains a path corresponding to each subsequence common to a string accepted by A_1 and a string accepted by A_2, and the weight of that path is the negative length of that subsequence. Since $A_1 \circ T_s \circ A_2$ is defined over the tropical semiring, $-w[A_1 \circ T_s \circ A_2]$ is the maximum length of a common subsequence, which is $lcs(A_1, A_2)$. □

As in the case of the edit-distance, the longest common subsequence of two acyclic automata A_1 and A_2 can be computed in $O(|A_1||A_2|)$ using the composition algorithm and a linear-time single-source shortest path algorithm.

6 Conclusion

We discussed several algorithms related to the use of statistical learning techniques in natural language processing when the input data is given as a set of distributions over strings, each compactly represented by a weighted automaton. We described efficient algorithms for computing the expected count of an arbitrary sequence and other moments of that distribution from a weighted automaton. We also demonstrated how several general similarity measures between weighted automata can be efficiently computed using algorithms based on weighted transducers.[4] This extends machine learning techniques such as SVMs and clustering to deal with distributions.

References

1. Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag: Berlin-New York, 1984.
2. B. E. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*, volume 5, pages 144–152, Pittsburg, 1992. ACM.
3. Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*, pages 598–603, 1997.
4. Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. In 35*th Meeting of the Association for Computational Linguistics (ACL '96), Proceedings of the Conference, Santa Cruz, California*, 1997.
5. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press: Cambridge, MA, 1992.
6. Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Positive Definite Rational Kernels. In *Proceedings of The Sixteenth Annual Conference on Computational Learning Theory (COLT 2003)*, Washington D.C., August 2003.
7. Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels. In *Advances in Neural Information Processing Systems (NIPS 2002)*, volume 15, Vancouver, Canada, March 2003. MIT Press.
8. Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.
9. David Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.
10. Frederick Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, 1998.

[4] Efficient implementations of many of the general algorithms that we used are incorporated in two software libraries, the AT&T FSM library [24] and the GRM Library [18], whose binary executables are available for download for non-commercial use.

11. Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.
12. Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1986.
13. Fabrice Lefevre, Jean-Luc Gauvain, and Lori Lamel. Towards task-independent speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, 2001.
14. Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch String Kernels for SVM Protein Classification. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*. MIT Press, March 2003.
15. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10:707–710, 1966.
16. Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, pages 563–569. MIT Press, 2001.
17. Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:2, 1997.
18. Mehryar Mohri. Weighted Grammar Tools: the GRM Library. In Jean claude Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*, pages 165–186. Kluwer Academic Publishers, The Netherlands, 2001. http://www.research.att.com/sw/tools/grm.
19. Mehryar Mohri. Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143, 2002.
20. Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.
21. Mehryar Mohri. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 2003.
22. Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted Finite-State Transducers in Speech Recognition. *Computer Speech and Language*, 16(1):69–88, 2002.
23. Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted automata in text and speech processing. In *ECAI-96 Workshop, Budapest, Hungary*, 1996.
24. Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. The Design Principles of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17–32, January 2000. http://www.research.att.com/sw/tools/fsm.
25. Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
26. Brian Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001.
27. Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
28. Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.
29. Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, NY, 1998.
30. Chris Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Royal Holloway, University of London, 1999.