A MACHINE LEARNING FRAMEWORK FOR SPOKEN-DIALOG CLASSIFICATION

Corinna Cortes[1] *Patrick Haffner*[2] *Mehryar Mohri*[3,1]

[1] Google Research
76 Ninth Avenue
New York, NY 10011
corinna@google.com

[2] AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932
haffner@research.att.com

[3] Courant Institute
251 Mercer Street
New York, NY 10012
mohri@cims.nyu.edu

ABSTRACT

One of the key tasks in the design of large-scale dialog systems is classification. This consists of assigning, out of a finite set, a specific category to each spoken utterance, based on the output of a speech recognizer. Classification in general is a standard machine learning problem, but the objects to classify in this particular case are word lattices, or weighted automata, and not the fixed-size vectors learning algorithms were originally designed for. This chapter presents a general kernel-based learning framework for the design of classification algorithms for weighted automata. It introduces a family of kernels, *rational kernels*, that combined with support vector machines form powerful techniques for spoken-dialog classification and other classification tasks in text and speech processing. It describes efficient algorithms for their computation and reports the results of their use in several difficult spoken-dialog classification tasks based on deployed systems. Our results show that rational kernels are easy to design and implement and lead to substantial improvements of the classification accuracy. The chapter also provides some theoretical results helpful for the design of rational kernels.

1. MOTIVATION

A critical problem for the design of large-scale spoken-dialog systems is to assign a category, out of a finite set, to each spoken utterance. These categories help guide the dialog manager in formulating a response to the speaker. The choice of categories depends on the application, they could be for example *referral* or *pre-certification* for a health-care company dialog system, or *billing services* or *credit* for an operator-service system.

To determine the category of a spoken utterance, one needs to analyze the output of a speech recognizer. Figure 1 is taken from a customer-care application. It illustrates the output of a state-of-the-art speech recognizer in a very simple case where the spoken utterance is "Hi, this is my number". The output is an acyclic weighted automaton called a *word lattice*. It compactly represents the recognizer's best guesses. Each path is labeled with a sequence of words and has a score obtained by summing the weights of the constituent transitions. The path with the lowest score is the recognizer's best guess, in this case "I'd like my card number".

This example make evident that the error rate of conversational speech recognition systems is still too high in many tasks to rely only on the one-best output of the recognizer. Instead, one can use the full word lattice which contains the correct transcription in most cases. This is indeed the case in Figure 1, since the top path is labeled with the correct sentence. Thus, in this chapter, spoken-dialog classification is formulated as the problem of assigning a category to each word lattice.

Classification in general is a standard machine learning problem. A classification algorithm receives a finite number of labeled examples which it uses for training, and selects a hypothesis expected to make few errors on future examples. For the design of modern spoken-dialog systems, this training sample

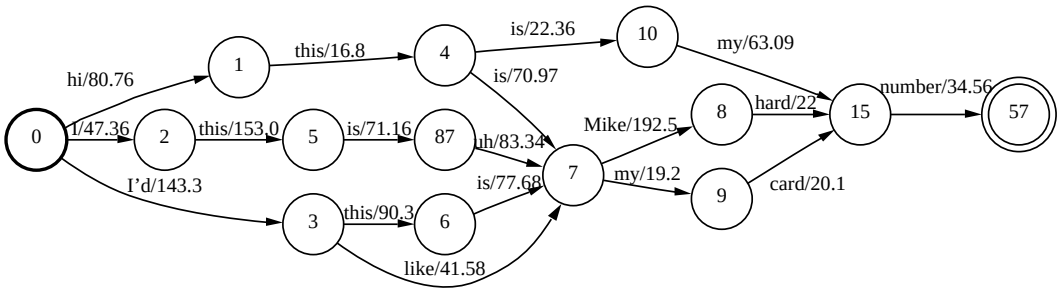


Figure 1: Word lattice output of a speech recognition system for the spoken utterance "Hi, this is my number".

is often available. It is the result of careful human labeling of spoken utterances with a finite number of pre-determined categories of the type already mentioned.

But, most classification algorithms were originally designed to classify fixed-size vectors. The objects to analyze for spoken-dialog classification are word lattices, each a collection of a large number of sentences with some weight or probability. How can standard classification algorithms such as support vector machines [Cortes and Vapnik, 1995] be extended to handle such objects?

This chapter presents a general framework and solution for this problem, which is based on *kernels methods* [Boser et al., 1992, Schölkopf and Smola, 2002]. Thus, we shall start with a brief introduction to kernel methods (Section 2). Section 3 will then present a kernel framework, *rational kernels*, that is appropriate for word lattices and other weighted automata. Efficient algorithms for the computation of these kernels will be described in Section 4. We also report the results of our experiments using these methods in several difficult large-vocabulary spoken-dialog classification tasks based on deployed systems in Section 5. There are several theoretical results that can guide the design of kernels for spoken-dialog classification. These results are discussed in Section 6.

2. INTRODUCTION TO KERNEL METHODS

Let us start with a very simple two-group classification problem illustrated by Figure 2 where one wishes to distinguish two populations, the blue and red circles. In this very simple example, one can choose a hyperplane to correctly separate the two populations. But, there are infinitely many choices for the selection of that hyperplane. There is good theory though supporting the choice of the hyperplane that maximizes the *margin*, that is the distance between each population and the separating hyperplane. Indeed, let $\mathcal{F}$ denote the class of real-valued functions on the ball of radius R in $\mathbb{R}^N$:

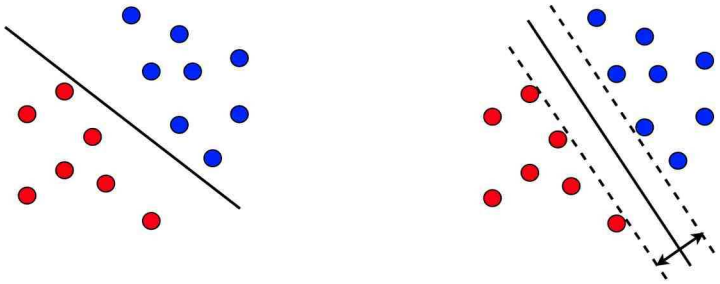
$$\mathcal{F} = \{x \mapsto w \cdot x : \|w\| \leq 1, \|x\| \leq R\}. \quad (1)$$

Then, it can be shown [Bartlett and Shawe-Taylor, 1999] that there is a constant c such that, for all distributions D over X, with probability at least $1-\delta$, if a classifier $\mathrm{sgn}(f)$, with $f \in \mathcal{F}$, has margin at least ρ over m independently generated training examples, then the generalization error of $\mathrm{sgn}(f)$, or error on any future example, is no more than

$$\frac{c}{m}\left(\frac{R^2}{\rho^2}\log^2 m + \log\frac{1}{\delta}\right). \quad (2)$$

This bound justifies large-margin classification algorithms such as support vector machines (SVMs). Let $w \cdot x + b = 0$ be the equation of the hyperplane, where $w \in \mathbb{R}^N$ is a vector normal to the hyperplane and $b \in \mathbb{R}$ a scalar offset. The classifier $\mathrm{sgn}(h)$ corresponding to this hyperplane is unique and can be defined with respect to the training points $x_1, \ldots, x_m$:

$$h(x) = w \cdot x + b = \sum_{i=1}^{m} \alpha_i (x_i \cdot x) + b, \quad (3)$$



(a) (b)

Figure 2: Large-margin linear classification. (a) An arbitrary hyperplane can be chosen to separate the two groups. (b) The maximal-margin hyperplane provides better theoretical guarantees.

where α_is are real-valued coefficients. The main point we are interested in here is that both for the construction of the hypothesis and the later use of that hypothesis for classification of new examples, one needs only to compute a number of dot products between examples.

In practice, non-linear separation of the training data is often not possible. Figure 3(a) shows an example where any hyperplane crosses both populations. However, one can use more complex functions to separate the two sets as in Figure 3(b). One way to do that is to use a non-linear mapping $\Phi : X \to F$ from the input space X to a higher-dimensional space F where linear separation is possible.

The dimension of F can truly be very large in practice. For example, in the case of document classification, one may use as features, sequences of three consecutive words (trigrams). Thus, with a vocabulary of just 100,000 words, the dimension of the feature space F is 10^{15}. On the positive side, as indicated by the error bound of Equation 2, the generalization ability of large-margin classifiers such as SVMs does not depend on the dimension of the feature space but only on the margin ρ and the number of training examples m. However, taking a large number of dot products in a very high-dimensional space to define the hyperplane may be very costly.

A solution to this problem is to use the so-called 'kernel trick' or *kernel methods*. The idea is to define a function $K : X \times X \to \mathbb{R}$ called a *kernel*, such that the kernel function on two examples x and y in input space, $K(x, y)$, is equal to the dot product of two examples $\Phi(x)$ and $\Phi(y)$ in feature space:

$$\forall x, y \in X, \quad K(x, y) = \Phi(x) \cdot \Phi(y). \tag{4}$$

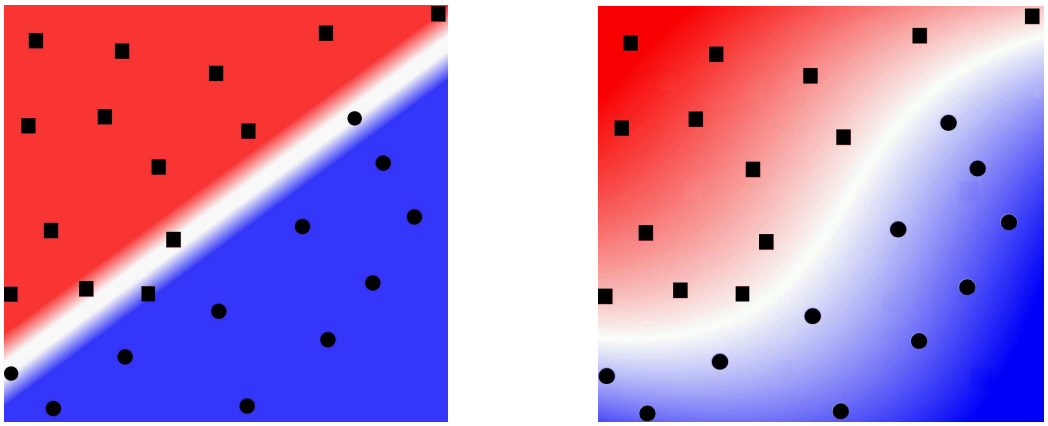
K is often viewed as a similarity measure. A crucial advantage of K is efficiency: there is no need anymore to define and explicitly compute $\Phi(x)$, $\Phi(y)$, and $\Phi(x) \cdot \Phi(y)$. Another benefit of K is flexibility: K can be arbitrarily chosen so long as the existence of Φ is guaranteed, which is called Mercer's condition. This condition is important to guarantee the convergence of training for algorithms such as SVMs.[1]

A condition equivalent to Mercer's condition is that the kernel K be *positive definite and symmetric*, that is, in the discrete case, the matrix $(K(x_i, x_j))_{1 \leq i,j \leq n}$ must be symmetric and positive semi-definite for any choice of n points $x_1, \ldots, x_n$ in X. Said differently, the matrix must be symmetric and its eigenvalues non-negative. Thus, for the problem that we are interested in, the question is how to define positive definite symmetric kernels for word lattices or weighted automata.

3. RATIONAL KERNELS

This section introduces a family of kernels for weighted automata, *rational kernels*. We will start with some preliminary definitions of automata and

[1] Some standard Mercer kernels over a vector space are the polynomial kernels of degree $d \in \mathbb{N}$, $K_d(x, y) = (x \cdot y + 1)^d$, and Gaussian kernels $K_\sigma(x, y) = \exp(-\|x - y\|^2/\sigma^2)$, $\sigma \in \mathbb{R}_+$.



(a) (b)

Figure 3: Non-linearly separable case. The classification task consists of discriminating between solid squares and solid circles. (a) No hyperplane can separate the two populations. (b) A non-linear mapping can be used instead.

transducers. For the most part, we will adopt the notation introduced in [Mohri et al., 2006] for automata and transducers.

Figure 4(a) shows a simple weighted automaton. It is a weighted directed graph in which edges or transitions are augmented with a label and carry some weight indicated after the symbol slash. A bold circle indicates an initial state and a double-circle a final state. A final state may also carry a weight indicated after the slash symbol representing the state number.

A path from an initial state to a final state is called a *successful path*. The weight of a path is obtained by multiplying the weights of constituent transitions and the final weight. The weight associated by A to a string x, denoted by $[\![A]\!](x)$, is obtained by summing the weights of all paths labeled with x. In this case, the weight associated to the string abb is the sum of the weight of two paths. The weight of each path is obtained by multiplying transition weights and the final weight 0.1.

Similarly, Figure 4(b) shows an example of a weighted transducer. Weighted transducers are similar to weighted automata but each transition is augmented with an output label in addition to the familiar input label and the weight. The output label of each transition is indicated after the colon separator. The weight associated to a pair of strings (x, y), denoted by $[\![T]\!](x, y)$, is obtained by summing the weights of all paths with input label x and output label y. Thus, for example, the weight associated by the transducer T to the pair (abb, baa) is obtained as in the automata case by summing the weights of the two paths labeled with (abb, baa).

To help us gain some intuition for the definition of a family of kernels for weighted automata, let us first consider kernels for sequences. As mentioned earlier in Section 2, a kernel can be viewed as a similarity measure. In the case of sequences, we may say that two strings are similar if they share many common substrings or subsequences. The kernel could then be for example the sum of the product of the counts of these common substrings. But how can we generalize that to weighted automata?

Similarity measures such as the one just discussed can be computed by using weighted finite-state transducers. Thus, a natural idea is to use weighted transducers to define similarity measures for sequences. We will say a that kernel K is *rational* when there exists a weighted transducer T such that

$$K(x, y) = [\![T]\!](x, y), \tag{5}$$

for all sequences x and y. This definition generalizes naturally to the case where the objects to handle are weighted automata. We say that K is *rational*

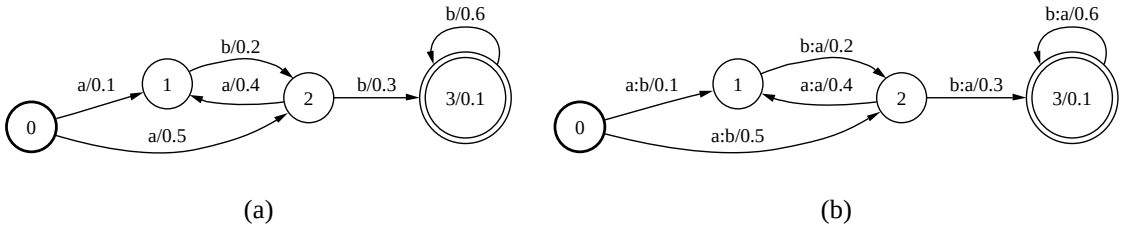


Figure 4: Weighted automata and transducers. (a) Example of a weighted automaton A. $[\![A]\!](abb) = .1 \times .2 \times .3 \times .1 + .5 \times .3 \times .6 \times .1$. (b) Example of a weighted transducer T. $[\![T]\!](abb, baa) = [\![A]\!](abb)$.

when there exists a weighted transducer T such that $K(A, B)$, the similarity measure of two automata A and B, is given by:

$$K(A, B) = \sum_{x,y} [\![A]\!](x) \cdot [\![T]\!](x, y) \cdot [\![B]\!](y). \quad (6)$$

$T(x, y)$ is the similarity measure between two strings x and y. But, in addition, we need to take into account the weight associated by A to x and by B to y, and sum over all pairs (x, y).[2]

Our definition of kernels for weighted automata is a first step towards extending SVMs to handle weighted automata. However, we also need to provide efficient algorithms for computing the kernels. The weighted automata that we are dealing with in spoken-dialog systems are word lattices that may have hundreds of thousands of states and transitions, and millions of paths, thus the efficiency of the computation is critical. In particular, the computational cost of applying existing string kernel algorithms to each pair of paths of two automata A and B is clearly prohibitive.

We also have to provide effective kernels for spoken-dialog classification and prove that they are indeed positive definite symmetric so they can be combined with SVM for high-accuracy classification systems.

[2]This definition can be generalized to the case of an arbitrary semiring where general operations other than the usual sum and multiplication are applied, which has important theoretical and algorithmic consequences and also interesting software engineering implications [Cortes et al., 2003a].

4. ALGORITHMS

This section describes general and efficient algorithms for the computation of rational kernels between weighted automata and provide examples of effective positive definite symmetric kernels for spoken-dialog classification.

The outline of the algorithm is as follows. The main observation is that the sum defining rational kernels can be viewed as the sum of the weights of all the paths of a single transducer, the one obtained by *composing* A with T with B (see [Mohri et al., 2006]):

$$\sum_{x,y} [\![A]\!](x)[\![T]\!](x, y)[\![B]\!](y) = \sum_{x,y} [\![A \circ T \circ B]\!](x, y). \quad (7)$$

The sum of the weights of all the paths of a transducer can be computed using a general single-source shortest-distance algorithm over the ordinary $(+, \times)$ semiring or the so-called forward-backward algorithm in the acyclic case [Mohri, 2002]. Thus, this leads to the following general algorithm to compute $K(A, B)$ when K is a rational kernel associated to the transducer T.

- Use composition algorithm (see [Mohri et al., 2006]) to compute $U = A \circ T \circ B$ in time $O(|T||A||B|)$.
- Use a general single-source shortest-distance algorithm to compute the sum of the weights of all successful paths of U [Mohri, 2002]. This can be done in linear time ($O(|U|)$) when U is acyclic, which is the case when A and B are acyclic automata (word lattices).

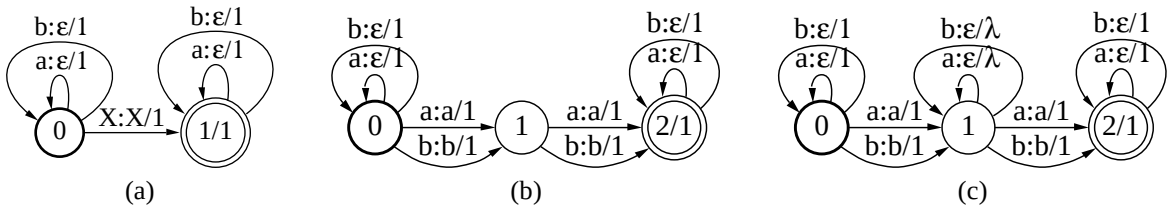


Figure 5: Weighted transducers computing the expected counts of (a) all sequences accepted by the regular expression X; (b) all bigrams; (c) all gappy bigrams with penalty gap λ, over the alphabet $\{a, b\}$.

In combination, this provides a general and efficient algorithm for computing rational kernels whose total complexity for acyclic word lattices is quadratic: $O(|T||A||B|)$.[3]

The next question that arises is how to define and compute kernels based on counts of substrings or subsequences as previously discussed in Section 3. For weighted automata, we need to generalize the notion of counts to take into account the weight of the paths in each automaton and use instead the *expected counts* of a sequence.

Let $|u|_x$ denote the number of occurrences of a substring x in u. Since a weighted automaton A defines a distribution over the set of strings u, the *expected count* of a sequence x in a weighted automaton A can be defined naturally as

$$c_A(x) = \sum_{u \in \Sigma^*} |u|_x [\![A]\!](u), \tag{8}$$

where the sum runs over Σ^*, the set of all strings over the alphabet Σ. We mentioned earlier that weighted transducers can often be used to count the occurrences of some substrings of interest in a string. Let us assume that one could find a transducer T that could compute the expected counts of all substrings of interest appearing in a weighted automaton A. Thus, $A \circ T$ would provide the expected counts of these substrings in A. Similarly, $T^{-1} \circ B$ would provide the expected counts of these substrings in B. Recall that T^{-1} is the transducer obtained by swapping the input and output labels of each transition ([Mohri et al., 2006]). Composition of $A \circ T$ and $T^{-1} \circ B$ matches paths labeled with the same substring in $A \circ T$ and $T^{-1} \circ B$ and multiplies their weights. Thus, by definition of composition (see [Mohri et al., 2006]), we could compute the sum of the expected counts in A and B of the matching substrings by computing

$$A \circ T \circ T^{-1} \circ B, \tag{9}$$

and summing the weights (expected counts) of all paths using a general single-source shortest-distance algorithm.

Comparing this formula with Equation 7 shows that the count-based similarity measure we are interested in is the rational kernel whose corresponding weighted transducer S is:

$$S = T \circ T^{-1}. \tag{10}$$

Thus, if we can determine a T with this property, this would help us naturally define the similarity measure S. In Section 6 we will prove that the kernel corresponding to S will actually be positive definite symmetric, and can hence be used in combination with SVMs.

In the following, we will provide a weighted transducer T for computing the expected counts of substrings. It turns out that there exists a very simple transducer T that can be used for that purpose. Figure 5(a) shows a simple transducer that can be used to compute the expected counts of all sequences accepted by the regular expression X over the alphabet $\{a, b\}$. In the figure, the transition labeled with $X{:}X/1$ symbolizes a finite automaton representing X with identical input and output labels and all weights equal to one.

Here is how transducer T counts the occurrences of a substring x recognized by X in a sequence u. State 0 reads a prefix u_1 of u and outputs the empty

[3] Here $|T|$ is a constant independent of the automata A and B for which $K(A, B)$ needs to be computed.

string ϵ. Then, an occurrence of x is read and output identically. Then state 1 reads a suffix u_2 of u and outputs ϵ. In how many different ways can u be decomposed into $u = u_1\, x\, u_2$? In exactly as many ways as there are occurrences of x in u. Thus, T applied to u generates exactly $|u|_x$ successful paths labeled with x. This holds for all strings x accepted by X. If T is applied to (or composed with) a weighted automaton A instead, then for any x, it generates $|u|_x$ paths for each path of A labeled with u. Furthermore, by definition of composition, each path generated is weighted with the weight of the path in A labeled with u. Thus, the sum of the weights of the paths generated that are labeled with x is exactly the expected count of x:

$$[\![A \circ T]\!](x) = c_A(x). \tag{11}$$

Thus, there exists a simple weighted transducer T that can count, as desired, the expected counts of all strings recognized by an arbitrary regular expression X in a weighted automaton A. In particular, since the set of bigrams over an alphabet Σ is a regular language, we can construct a weighted transducer T computing the expected counts of all bigrams. Figure 5(b) shows that transducer, which has only three states, regardless of the alphabet size.

In some applications, one may wish to allow for a gap between the occurrences of two symbols and view two weighted automata as similar if they share such substrings (*gappy bigrams*) with relatively large expected counts. The gap or distance between two symbols is penalized using a fixed penalty factor λ, $0 \leq \lambda < 1$. A sequence kernel based on these ideas was used successfully by Lodhi et al. [2001] for text categorization. Interestingly, constructing a kernel based on gappy bigrams is straightforward in our framework. Figure 5(c) shows the transducer T counting expected counts of gappy bigrams from which the kernel can be efficiently constructed. The same can be done similarly for higher-order gappy n-grams or other gappy substrings.

The methods presented in this section can be used to construct efficiently and, often in a simple manner, relatively complex weighted automata kernels K based on expected counts or other ideas. A single general algorithm can then be used as described to compute efficiently $K(A, B)$ for any two weighted automata A and B, without the need to design a new algorithm for each new kernel, as previously done in the literature. As an example, the gappy kernel used by Lodhi et al. [2001] is the rational kernel corresponding to the 6-state transducer $S = T \circ T^{-1}$ of Figure 6.

5. EXPERIMENTS

This section describes the applications of the kernel framework and techniques to spoken-dialog classification.

In most of our experiments, we used simple n-*gram rational kernels*. An n-*gram kernel* k_n for two weighted automata or lattices A and B is defined by:

$$k_n(A, B) = \sum_{|x|=n} c_A(x)\, c_B(x). \tag{12}$$

As described in Section 4, k_n is a kernel of the form $T \circ T^{-1}$ and can be computed efficiently. An n-*gram rational kernel* K_n is simply the sum of kernels k_m, with $1 \leq m \leq n$:

$$K_n = \sum_{m=1}^{n} k_m$$

Thus, the feature space associated with K_n is the set of all m-gram sequences with $m \leq n$. As discussed in the previous section, it is straightforward, using the same algorithms and representations, to extend these kernels to kernels with gaps and to many other more complex rational kernels more closely adapted to the applications considered.

We did a series of experiments in several large-vocabulary spoken-dialog tasks using rational kernels with a twofold objective [Cortes et al., 2004]: to improve classification accuracy in those tasks, and to evaluate the impact on classification accuracy of the use of a word lattice rather than the one-best output of the automatic speech recognition (ASR) system.

The first task we considered was that of a deployed customer-care application (HMIHY 0300). In this task, users interact with a spoken-dialog system via the telephone, speaking naturally, to ask about their bills, their calling plans, or other similar topics. Their responses to the open-ended prompts of the system are not constrained by the system, they

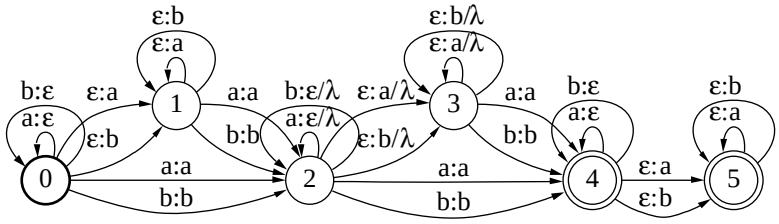


Figure 6: Simple weighted transducer corresponding to the gappy bigram kernel used by Lodhi et al. [2001] obtained by composition of the transducer of Figure 5(c) and its inverse.

may be any natural language sequence. The objective of the spoken-dialog classification is to assign one or several categories or call-types, e.g., *Billing Credit*, or *Calling Plans*, to the users' spoken utterances. The set of categories is pre-determined, and in this first application there are 64 categories. The calls are classified based on the user's response to the first greeting prompt: "*Hello, this is AT&T. How may I help you?*".

Table 1 indicates the size of the HMIHY 0300 datasets we used for training and testing. The training set is relatively large with more than 35,000 utterances, this is an extension of the one we used in our previous classification experiments with HMIHY 0300 [Cortes et al., 2003b]. In our experiments, we used the n-gram rational kernels just described with $n = 3$. Thus, the feature set we used was that of all n-grams with $n \leq 3$. Table 1 indicates the total number of distinct features of this type found in the datasets. The word accuracy of the system based on the best hypothesis of the speech recognizer was 72.5%. This motivated our use of the word lattices, which contain the correct transcription in most cases. The average number of transitions of a word lattice in this task was about 260.

Table 1 reports similar information for two other datasets, VoiceTone1, and VoiceTone2. These are more recently deployed spoken-dialog systems in different areas, e.g., VoiceTone1 is a task where users interact with a system related to health-care with a larger set of categories (97). The size of the VoiceTone1 datasets we used and the word accuracy of the recognizer (70.5%) make this task otherwise similar to HMIHY 0300. The datasets provided for VoiceTone2 are significantly smaller with a higher word error rate. The word error rate is indicative of the difficulty of classification task since a higher error rate implies a more noisy input. The average number of transitions of a word lattice in VoiceTone1 was about 210 and in VoiceTone2 about 360.

Each utterance of the dataset may be labeled with several classes. The evaluation is based on the following criterion: it is considered an error if the highest scoring class given by the classifier is none of these labels.

We used the AT&T FSM Library [Mohri et al., 2000] and the GRM Library [Allauzen et al., 2004] for the implementation of the n-gram rational kernels K_n used. We used these kernels with SVMs, using a general learning library for large-margin classification (LLAMA), which offers an optimized multi-class recombination of binary SVMs [Haffner et al., 2003]. Training time took a few hours on a single processor of a 2.4GHz Intel Pentium processor Linux cluster with 2GB of memory and 512 KB cache.

In our experiments, we used the trigram kernel K_3 with a second-degree polynomial. Preliminary experiments showed that the top performance was reached for trigram kernels and that 4-gram kernels, K_4, did not significantly improve the performance. We also found that the combination of a second-degree polynomial kernel with the trigram kernel significantly improves performance over a linear classifier, but that no further improvement could be obtained with a third-degree polynomial.

We used the same kernels in the three datasets previously described and applied them to both the speech recognizer's single best hypothesis (one-best results), and to the full word lattices output by the speech recognizer. We also ran, for the sake of

Dataset	Number of classes	Training size	Testing size	Number of n-grams	ASR word accuracy
HMIHY 0300	64	35551	5000	24177	72.5%
VoiceTone1	97	29561	5537	22007	70.5%
VoiceTone2	82	9093	5172	8689	68.8%

Table 1: Key characteristics of the three datasets used in the experiments. The fifth column displays the total number of unigrams, bigrams, and trigrams found in the one-best output of the ASR for the utterances of the training set, that is the number of features used by BoosTexter or SVMs used with the one-best outputs. The training and testing sizes reported in columns 3 and 4 are described in the number of utterances, or equivalently the number of word lattices.

comparison, the BoosTexter algorithm [Schapire and Singer, 2000] on the same datasets by applying it to the one-best hypothesis. This served as a baseline for our experiments.

Figure 7(a) shows the result of our experiments in the HMIHY 0300 task. It gives classification error rate as a function of rejection rate (utterances for which the top score is lower than a given threshold are rejected) in HMIHY 0300 for: BoosTexter, SVM combined with our kernels when applied to the one-best hypothesis, and SVM combined with kernels applied to the full lattices.

SVM with trigram kernels applied to the one-best hypothesis leads to better classification than BoosTexter everywhere in the range of 0-40% rejection rate. The accuracy is about 2-3% absolute value better than that of BoosTexter in the range of interest for this task, which is roughly between 20% and 40% rejection rate. The results also show that the classification accuracy of SVMs combined with trigram kernels applied to word lattices is consistently better than that of SVMs applied to the one-best alone by about 1% absolute value.

Figures 7(b)-(c) show the results of our experiments in the VoiceTone1 and VoiceTone2 tasks using the same techniques and comparisons. As observed previously, in many regards, VoiceTone1 is similar to the HMIHY 0300 task, and our results for VoiceTone1 are comparable to those for HMIHY 0300. The results show that the classification accuracy of SVMs combined with trigram kernels applied to word lattices is consistently better than that of BoosTexter, by more than 4% absolute value at about 20% rejection rate. They also demonstrate more clearly the benefits of the use of the word lattices for classification in this task. This advantage is even more manifest for the VoiceTone2 task for which the speech recognition accuracy is lower. VoiceTone2 is also a harder classification task as can be seen by the comparison of the plots of Figure 7(b). The classification accuracy of SVMs with kernels applied to lattices is more than 6% absolute value better than that of BoosTexter near 40% rejection rate, and about 3% better than SVMs applied to the one-best hypothesis.

Thus, our experiments in spoken-dialog classification in three distinct large-vocabulary tasks demonstrated that using rational kernels with SVMs consistently leads to very competitive classifiers. They also show that their application to the full word lattices instead of the single best hypothesis output by the recognizer systematically improves classification accuracy.

We further explored the use of kernels based on other moments of the counts of substrings in sequences, generalizing n-gram kernels [Cortes and Mohri, 2005]. Let m be a positive integer. Let $c_A^m(x)$ denote the *m-th moment of the count* of the sequence x in A defined by:

$$c_A^m(x) = \sum_{u \in \Sigma^*} |u|_x^m \, [\![A]\!](u). \tag{13}$$

We can define a general family of kernels, denoted by K_n^m, $n, m \geq 1$ and defined by:

$$K_n^m(A, B) = \sum_{|x|=n} c_A^m(x) \, c_B^m(x), \tag{14}$$

which exploit the m-th moment of the counts of substrings x in weighted automata A and B to define their similarity. Cortes and Mohri [2005] showed that

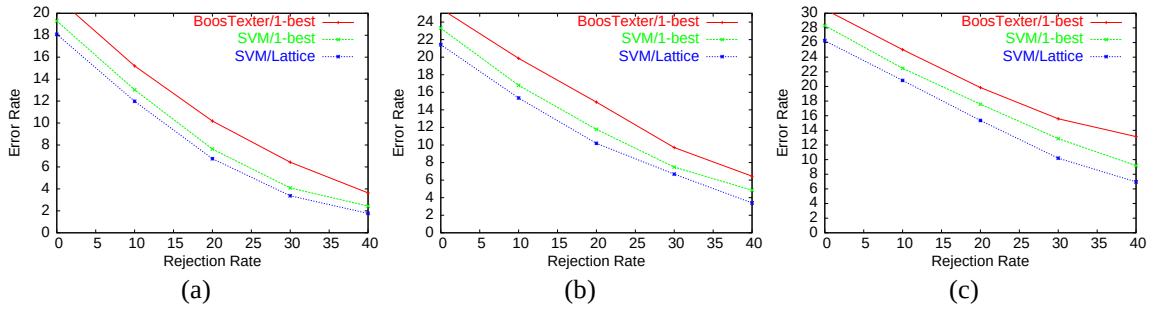


Figure 7: Classification error rate as a function of rejection rate in (a) HMIHY 0300, (b) VoiceTone1, and (c) VoiceTone2.

there exist weighted transducers T_n^m that can be used to compute efficiently $c_A^m(x)$ for all n-gram sequence x and weighted automaton A. Thus, these kernels are rational kernels and their associated transducers are $T_n^m \circ T_n^{m-1}$:

$$K_n^m(A,B) = \sum_{x,y} [\![A \circ [T_n^m \circ (T_n^m)^{-1}] \circ B]\!](x,y). \quad (15)$$

Figure 8 shows the weighted transducer T_1^m for aperiodic strings x, which has only $m|x|+1$ states.[4] By Equation 15, the transducer T_1^m can be used to compute $K_n^m(A,B)$ by first computing the composed transducer $A \circ [T_n^m \circ (T_n^m)^{-1}] \circ B$ and then summing the weights of all the paths of this transducer using a shortest-distance algorithm [Mohri, 2002].

The application of moment kernels to the HMIHY 0300 task resulted in a further improvement of the classification accuracy. In particular, at 15% rejection rate, the error rate was reduced by 1% absolute, that is about 6.2% relative, which is significant in this task, by using *variance kernels*, that is moment kernels of second-order ($m = 2$).

6. THEORETICAL RESULTS FOR RATIONAL KERNELS

In the previous sections, we introduced a number of rational kernels, e.g., kernels based on expected counts or moments of the counts and applied them to spoken-dialog tasks. Several questions arise in relation with these kernels. As pointed out earlier, to guarantee the convergence of algorithms such as SVMs, the kernels used must be positive definite symmetric (PDS). But, how can we construct PDS rational kernels? Are n-gram kernels and similar kernels PDS? Can we combine simpler PDS rational kernels to create more complex ones? Is there a characterization of PDS rational kernels?

All these questions have been investigated by Cortes et al. [2003a]. The following theorem provides a general method for constructing PDS rational kernels.

Theorem 1 *Let T be a weighted finite-state transducer over $(+,\times)$. Assume that the weighted transducer $T \circ T^{-1}$ is regulated,[5] then $S = T \circ T^{-1}$ defines a PDS rational kernel.*

Proof. We give a sketch of the proof. A full proof is given in [Cortes et al., 2003a]. Let K be the kernel associated to $S = T \circ T^{-1}$. By definition of T^{-1} and composition,

$$\forall x, y \in X,\ K(x,y) = \sum_z [\![T]\!](x,z)[\![T]\!](y,z), \quad (16)$$

where the sum is over all strings z. Let K_n be the function defined by restricting the sum to strings of

[4] An aperiodic string is a string that does not admit a non-empty prefix as a suffix. The m-th moment of other strings (periodic strings) can also be computed using weighted transducers.

[5] A weighted transducer T is said to be *regulated* when $[\![T]\!](x,y)$, the sum of the weights of the paths with input x and output y is well-defined.

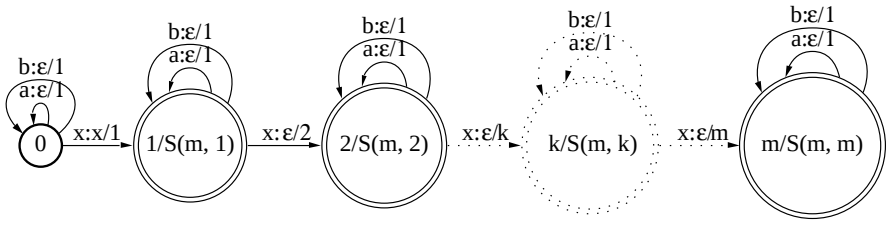


Figure 8: Weighted transducer T^m for computing the m-th moment of the count of an aperiodic substring x. The final weight at state k, $k = 1, \ldots, m$, indicated after "/", is $S(m, k)$, the Stirling number of the second kind, that is the number of ways of partitioning a set of m elements into k nonempty subsets, $S(m, k) = \frac{1}{k!}\sum_{i=0}^{k}\binom{k}{i}(-1)^i(k-i)^m$. The first-order transducer T_1 coincides with the transducer of Figure 5(a), since $S(1, 1) = 1$.

length at most k:

$$\forall x, y \in X,\ K_k(x, y) = \sum_{|z|\leq k} [\![T]\!](x, z)[\![T]\!](y, z). \quad (17)$$

Consider any ordering of all strings of length at most k: $z_1, \ldots, z_l$. For any set of n strings $x_1, \ldots, x_n$, let A be the matrix defined by $A = ([\![T]\!](x_i, z_j))_{i\in[1,n], j\in[1,l]}$. Then, the eigenvalues of the matrix M_n defined by

$$M_n = (K_n(x_i, x_j))_{i,j\in[1,n]} \quad (18)$$

are necessarily non-negative since $M_n = AA^\top$. Thus, for any $n \leq 0$, K_n is a PDS kernel. Since K is a pointwise limit of K_n, K is also PDS [Berg et al., 1984, Cortes et al., 2003a]. □

The theorem shows that the rational kernels we considered in previous sections, e.g., count-based similarity kernels, n-gram kernels, gappy n-gram kernels are all PDS rational kernels, which justifies a posteriori their use in combination with SVMs. Conversely, we have conjectured elsewhere that all PDS rational kernels are rational kernels associated to transducers of the type $S = T \circ T^{-1}$ [Cortes et al., 2003a] and proved several results in support of that conjecture. In particular, for acyclic transducer, this provides indeed a characterization of PDS rational kernels.

It can also be shown that a finite sum of PDS rational kernels is a PDS rational kernel, which we used for defining n-gram kernels. More generally, the following theorem holds [Cortes et al., 2003a].

Theorem 2 *PDS rational kernels are closed under sum, product, and Kleene-Closure.*

Thus, one can use rational operations to create complex PDS rational kernels from simpler ones.

7. CONCLUSION

Rational kernels form an effective tool and framework for spoken-dialog classification. They are based on a general theory that guarantees in particular the positive definiteness of rational kernels based on an arbitrary $(+, \times)$-weighted transducer and thus the convergence of training for algorithms such as SVMs. General and efficient algorithms can be readily used for their computation.

Experiments in several large-vocabulary spoken-dialog tasks show that rational kernels can be combined with SVMs to form powerful classifiers and that they perform well in several difficult tasks. They also demonstrate the benefits of the use of kernels applied to word lattices.

Rational kernels form a rich family of kernels. The kernels used in the experiments we described are only special instances of this general class of kernels. Rational kernels adapted to a specific spoken-dialog task can be designed. In fact, it is often straightforward to craft prior knowledge about a task in the transducer defining these kernels. One may for example exclude some word sequences or regular expressions from the similarity measure defined by these

kernels or emphasize the importance of others by increasing their respective weight in the weighted transducer.

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. A General Weighted Grammar Library. In *Proceedings of the Ninth International Conference on Automata (CIAA 2004)*, Kingston, Ontario, Canada, July 2004.

Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In *Advances in kernel methods: support vector learning*, pages 43–54. MIT Press, Cambridge, MA, USA, 1999.

Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag: Berlin-New York, 1984.

Bernhard E. Boser, Isabelle Guyon, and Vladimir 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.

Corinna Cortes and Mehryar Mohri. Moment Kernels for Regular Distributions. *Machine Learning*, 60(1-3):117–134, September 2005.

Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Positive Definite Rational Kernels. In *Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2003)*, volume 2777 of *Lecture Notes in Computer Science*, pages 41–56, Washington D.C., August 2003a. Springer, Heidelberg, Germany.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels. In *Advances in Neural Information Processing Systems (NIPS 2002)*, volume 15, Vancouver, Canada, March 2003b. MIT Press.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research (JMLR)*, 5:1035–1062, 2004.

Patrick Haffner, Gokhan Tur, and Jeremy Wright. Optimizing SVMs for complex Call Classification. In *Proceedings ICASSP'03*, 2003.

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, *NIPS 2000*, pages 563–569. MIT Press, 2001.

Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3): 321–350, 2002.

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.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. *Speech Handbook*, volume to appear, chapter Speech Recognition with Weighted Finite-State Transducers. Springer-Verlag, 2006.

Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press: Cambridge, MA, 2002.

Index

algorithm
  composition, 5
  forward-backward, 5
  large-margin classification, 2, 3
  shortest-distance, 5, 6, 10
AT&T FSM library, 8

BoosTexter, 9

composition algorithm, 5
customer-care application, 1

dot product, 3

forward-backward algorithm, 5

GRM library, 8

kernel methods, 2, 3
kernel trick, *see* kernel methods
kernels
  n-gram, 7
  dot product, 3
  expected counts, 6, 10, 11
  gappy, 7, 8
  Mercer's condition, 3
  positive definite symmetric, 10
  rational, 2, 3
  sequence learning, 4
  variance of counts, 10

large-margin classification algorithms, 2, 3, 8
LLAMA, 8

machine learning, 1
Mercer's condition, 3

rational kernels, 2, 3

semiring, 5
  probability, 5
shortest-distance algorithm, 5, 6, 10
software library
  AT&T FSM library, 8
  GRM library, 8
  LLAMA, 8
spoken-dialog classification, 1, 7
support vector machines, 2, 3, 8, 9
SVMs, *see* support vector machines

variance kernels, 10

weighted automata, 1, 4
weighted transducer, 4
word lattice, 1, 2, 5