Learning with Weighted Transducers

Corinna CORTES [a] and Mehryar MOHRI [b,1]
[a] *Google Research, 76 Ninth Avenue, New York, NY 10011*
[b] *Courant Institute of Mathematical Sciences and Google Research, 251 Mercer Street, New York, NY 10012*

Abstract. Weighted finite-state transducers have been used successfully in a variety of natural language processing applications, including speech recognition, speech synthesis, and machine translation. This paper shows how weighted transducers can be combined with existing learning algorithms to form powerful techniques for sequence learning problems.

Keywords. Learning, kernels, classification, regression, ranking, clustering, weighted automata, weighted transducers, rational powers series.

Introduction

Weighted transducer algorithms have been successfully used in a variety of applications in speech recognition [21, 23, 25], speech synthesis [28, 5], optical character recognition [9], machine translation, a variety of other natural language processing tasks including parsing and language modeling, image processing [1], and computational biology [15, 6]. This paper outlines the use of weighted transducers in *machine learning*.

A key relevance of weighted transducers to machine learning is their use in kernel methods applied to sequences. Weighted transducers provide a compact and simple representation of sequence kernels. Furthermore, standard weighted transducer algorithms such as composition and shortest-distance algorithms can be used to efficiently compute kernels based on weighted transducers.

1. Overview of Kernel Methods

Kernel methods are widely used in machine learning. They have been successfully used to deal with a variety of learning tasks including classification, regression, ranking, clustering, and dimensionality reduction. This section gives a brief overview of these methods.

Complex learning tasks are often tackled using a large number of features. Each point of the input space X is mapped to a high-dimensional feature space F via a non-linear mapping Φ. This may be to seek a linear separation in a higher-dimensional space,

[1]Corresponding Author: Courant Institute of Mathematical Sciences and Google Research, 251 Mercer Street, New York, NY 10012; E-mail: mohri@cs.nyu.edu. Mehryar Mohri's work was partially funded by the New York State Office of Science Technology and Academic Research (NYSTAR).

which was not achievable in the original space, or to exploit other regression, ranking, clustering, or manifold properties easier to attain in that space. The dimension of the feature space F can be very large. In document classification, the features may be for example the set of all trigrams. Thus, even for a vocabulary of just $200{,}000$ words, the dimension of F is 2×10^{15}.

The high dimensionality of F does not affect the generalization ability of large-margin algorithms such as support vector machines (SVMs). Remarkably, these algorithms benefit from theoretical guarantees for good generalization that depend only on the number of training points and the separation *margin*, and not on the dimensionality of the feature space. But the high dimensionality of F can directly impact the efficiency and even the practicality of such learning algorithms, as well as their use in prediction. This is because to determine their output hypothesis or to make predictions, these learning algorithms rely on the computation of a large number of dot products in the feature space F.

A solution to this problem is the so-called *kernel method*. This consists of defining a function $K\colon X \times X \to \mathbb{R}$ called a *kernel*, such that the value it associates to two examples x and y in input space, $K(x, y)$, coincides with the dot product of their images $\Phi(x)$ and $\Phi(y)$ in feature space:

$$\forall x, y \in X, \quad K(x, y) = \Phi(x) \cdot \Phi(y). \tag{1}$$

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. 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 condition equivalent to Mercer's condition is that the kernel K be *positive definite and symmetric* (PDS), that is, in the discrete case, the matrix $(K(x_i, x_j))_{1 \le i,j \le m}$ must be symmetric and positive semi-definite for any choice of n points $x_1, \ldots, x_m$ in X. Thus, the matrix must be symmetric and its eigenvalues non-negative.

The next section briefly describes a general family of kernels for sequences that is based on weighted transducers, *rational kernels*.

2. Rational Kernels

We start with some preliminary definitions of automata and transducers.

2.1. Weighted Transducers and Automata

Finite-state transducers are finite automata in which each transition is augmented with an output label in addition to the familiar input label [8, 16]. Output labels are concatenated along a path to form an output sequence and similarly with input labels. *Weighted transducers* are finite-state transducers in which each transition carries some weight in addition to the input and output labels. The weights of the transducers considered in this paper are real values and they are multiplied along the paths. The weight of a pair of in-

put and output strings (x, y) is obtained by summing the weights of all the paths labeled with (x, y). The following gives a formal definition of weighted transducers.

Definition 1. *A* weighted finite-state transducer T *over* $(\mathbb{R}, +, \cdot, 0, 1)$ *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,* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R} \times Q$ *a finite set of transitions,* $\lambda \colon I \to \mathbb{R}$ *the initial weight function, and* $\rho \colon F \to \mathbb{R}$ *the final weight function mapping* F *to* $\mathbb{R}$*.*

For a path π in a transducer, we denote by $p[\pi]$ the origin state of that path and by $n[\pi]$ its destination state. We also denote by $P(I, x, y, F)$ the set of paths from the initial states I to the final states F labeled with input string x and output string y. The weight of a path π is obtained by multiplying the weights of its constituent transitions and is denoted by $w[\pi]$. We shall say that a transducer T is *regulated* if the output weight associated by T to any pair of strings (x, y) by:

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

is in $\mathbb{R} \cup \{\infty\}$ and if this definition does not depend on the order of the terms in the sum. By convention, $T(x, y) = 0$ when $P(I, x, y, F) = \emptyset$. In the absence of ϵ-cycles, the set of accepting paths $P(I, x, y, F)$ is finite for any $(x, y) \in \Sigma^* \times \Delta^*$, and thus T is regulated. The transducers considered in this paper are all regulated. Figure 1 shows an example of a weighted transducer.

The standard rational operations, sum $+$, product or concatenation $\cdot$, and Kleene-closure * can be defined for regulated transducers [27, 18]. For any pair of strings (x, y), and any three weighted regulated transducers T, T_1, T_2,

$$(T_1 + T_2)(x, y) = T_1(x, y) + T_2(x, y) \tag{3}$$

$$(T_1 \cdot T_2)(x, y) = \sum_{\substack{x_1 x_2 = x \\ y_1 y_2 = y}} T_1(x_1, y_1)\, T_2(x_2, y_2) \tag{4}$$

$$T^*(x, y) = \sum_{n=0}^{+\infty} T^n(x, y). \tag{5}$$

For any weighted transducer T, we denote by T^{-1} its *inverse*, that is the transducer obtained from T by swapping the input and output label of each transition. The *composition* of two weighted transducers T_1 and T_2 with matching input and output alphabets Σ, is a weighted transducer denoted by $T_1 \circ T_2$ when the sum:

$$(T_1 \circ T_2)(x, y) = \sum_{z \in \Sigma^*} T_1(x, z)\, T_2(z, y) \tag{6}$$

is well-defined and in $\mathbb{R}$ for all $x, y \in \Sigma^*$ [27, 18]. There exists an efficient algorithm for the composition of two weighted transducers [26, 24]. The worst case complexity of that algorithm is quadratic, that is $O(|T_1||T_2|)$, where $|T_i|$ denotes the size of transducer T_i.

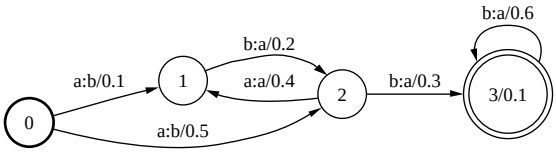

Figure 1. Example of a weighted transducer T. A bold circle indicates an initial state and a double-circle a final state. A final state carries a weight indicated after the slash symbol representing the state number. The initial weights are not indicated in all the examples in this paper since they are all equal to one. There are two paths in the transducer with input label abb and ouput label baa, thus the weight associated by T to the pair (abb, baa) is $T(abb, baa) = .1 \times .2 \times .3 \times .1 + .5 \times .3 \times .6 \times .1$.

2.2. Definition

As mentioned earlier, kernels can be viewed as similarity measures. It is often natural to define a similarity measure between two sequences, e.g., two documents or two biological sequences, as a function of the number of subsequences of some type that they share. These subsequences could be for example n-gram sequences, gappy n-grams, or substrings of any length. A sequence kernel is then typically defined as the sum of the product of the counts of these common subsequences.

Similarity measures of this kind can typically be computed using weighted finite-state transducers. This leads naturally to the following definition of a family of kernels overs strings.

Definition 2. *A kernel function* $K : \Sigma^* \times \Sigma^* \to \mathbb{R}$ *is* rational *when there exists a weighted transducer* U *such that* $K(x, y) = U(x, y)$ *for all sequences* x *and* y*.*

Thus, for a rational kernel defined by U, $U(x, y)$ is the similarity measure between two strings x and y.[2]

2.3. Algorithm

$U(x, y)$ can be computed using the composition of weighted transducers [26, 24]. Let M_x be a trivial weighted transducer representing x, that is a transducer such that $M_x(x, x) = 1$ and $M_x(y, z) = 0$ for $y \neq x$ or $z \neq x$. M_x can be constructed from a linear finite automaton representing x by augmenting each transition with an output label identical to the input label and by setting all transition and final weights to one. Similarly, we can construct a weighted transducer representing M_y. Then, by definition of composition, $(M_x \circ U \circ M_y)(x, y) = M_x(x, x)U(x, y)M_y(y, y) = U(x, y)$ and $(M_x \circ U \circ M_y)(z_1, z_2) = 0$ for $(z_1, z_2) \neq (x, y)$. Thus, $\sum_{u,v}(M_x \circ U \circ M_y)(u, v) = U(x, y)$, that is the sum of the weights of all paths of $M_x \circ U \circ M_y$ is exactly $U(x, y)$.

This gives a two-step algorithm to compute $K(x, y) = U(x, y)$: (a) use composition to compute $N = M_x \circ U \circ M_y$; (b) use a *shortest-distance algorithm* or forward-backward algorithm to compute the sum of the weights of all paths of N. We can assume that U does not contain any (ϵ, ϵ) cycle, that is a cycle with input ϵ and output ϵ. Otherwise, an equivalent weighted transducer without ϵ-transitions could be constructed from U by application of an ϵ-removal algorithm [22]. When U contains no ϵ-transition, N is necessarily acyclic since M_x and M_y are acyclic, and the computation of the sum of the

[2] This definition can be generalized to the case of an arbitrary semiring [12].

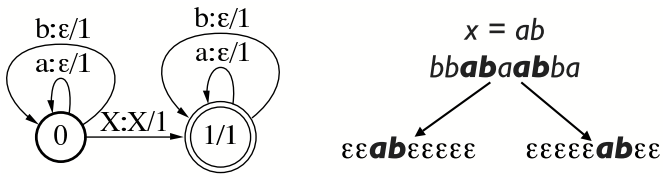

Figure 2. General count-based transducer T, for $\Sigma = \{a, b\}$. The figure illustrates the use of T in the special case where the automaton X accepts only the string $x = ab$, to count the number of occurrences of x in an input sequence such as $bbabaabba$.

weights of its paths can be done in linear time. Thus, the overall complexity of the computation of a rational kernel using that algorithm is $O(|U||M_x||M_y|)$, where $|U|$ remains constant in the calculation of a large number of kernels. In the particular case of many kernels used in practice, the complexity of the composition algorithm is in fact linear, which reduces the total cost of the application of the algorithm to $O(|U|+|M_x|+|M_y|)$. A new and more general n-way composition algorithm can also be used to dramatically improve the computational speed in other cases [2, 4].

2.4. Properties

To guarantee the convergence of algorithms such as support vector machines, the rational kernel K used must be positive definite symmetric. The following theorem gives a general method for constructing a PDS rational kernel from any weighted tranducer.

Theorem 1 ([12])**.** *Let T be an arbitrary weighted transducer, then $U = T \circ T^{-1}$ defines a PDS rational kernel.*

In this construction, the weighted transducer T can be viewed as the mapping from the input space $X = \Sigma^*$ to a high-dimensional feature space, compactly represented by the output of T. The construction of U from T is straightforward and very efficient since it requires only applying composition. Our inspection of the sequence kernels used in computational biology, natural language processing, or other sequence learning tasks, e.g., mismatch kernels [19], gappy n-gram kernels [20], locality-improved kernels [29], convolutions kernels for strings [17], tree kernels [11], n-gram kernels [12], and moment kernels [13], seem to show that they are all rational kernels of the form $T \circ T^{-1}$ [12]. In fact, we have conjectured that all PDS rational kernels are of this form and proven a number of results favoring that thesis [12].

Standard weighted transducer operations can be used to combine simpler PDS rational kernels to form more complex ones, as shown by the following theorem.

Theorem 2 ([12])**.** *PDS rational kernels are closed under sum, product, and closure operations.*

3. Applications

As already pointed out, to the best of our knowledge, the sequence kernels used in practice are all special instances of PDS rational kernels. Here we will briefly describe a

general and important family of rational kernels, *count-based kernels*, and show how a sequence kernel recently introduced in computational biology can be represented by as a weighted transducer.

3.1. Count-based kernels

The definition of many sequence kernels relies on the counts of some subsequences in the sequences x and y to compare. These subsequences can be of different nature, they may be for example arbitrary substrings, n-grams, gappy n-grams, or subsequences of ancestor sequences, where ancestor sequences are defined as sequences with a fixed number of mutations relative to the given sequence [19]. We will refer to such kernels as *count-based kernels*. These sequence kernels can typically be conveniently represented by weighted transducers and form rational kernels. This is because there exists a general weighted transducer that can be used to count the number of occurrences of the sequences described by an arbitrary regular expression.

Indeed, let X be an arbitrary finite automaton and thus representing an arbitrary regular expression. Then, the transducer defined by Figure 2 can be used to count all occurrences of the sequences accepted by X in any input sequence x. This is illustrated in the special case where X represents the single sequence $x = ab$ and for an input sequence $bbabaabba$.

The loop at the first state of T maps input symbols to ϵ until a match with a sequence in X is found. Then the sequence matched is mapped to itself and the remaining suffix of the input sequence mapped to ϵ at the final state of T. In the case of the sequence $bbabaabba$ and for the particular X considered, there are two possible occurrences of ab and thus two possible matches. The figure shows the alternative outputs generated by T. Since two paths are generated, each with weight one, the total weight associated by T to the input sequence is the sum two, which is the expected and correct count.

By theorem 1, transducer T can be used to construct a PDS rational kernel $U = T \circ T^{-1}$. This gives a very general method for the definition and construction of count-based sequence kernels. In fact, many sequence kernels successfully used in practice coincide precisely with this construction. This includes in particular n-gram kernels or gappy n-gram kernels.

3.2. Locality-improved kernel

A family of kernels was introduced by Zien et al. for the problem of *recognition of translation initiation sites* in computational biology. This problem consists of determining whether a start codon position in a DNA sequence is a translation initiation site (TIS).

The *locality-improved kernel* introduced by [29] is based on matching scores over windows of length $2l + 1$. It is defined as follows.

Definition 3 ([29])**.** *Let l and d be positive integers and w_j, $j \in [-l, +l]$ the weights associated to a match at position j in a window of size $2l+1$. Then, the locality-improved kernel for two sequences x and y of length m is defined by*

$$K(x,y) = \sum_{p=1}^{m} \mathrm{win}_p(x,y), \quad \textit{with} \quad \mathrm{win}_p(x,y) = \left(\sum_{j=-l}^{+l} w_j \mathrm{match}_{p+j}(x,y) \right)^d . \quad (7)$$

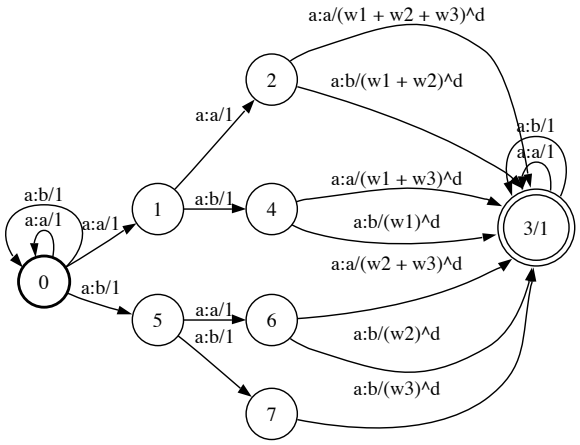

Figure 3. Fraction of the locality-improved kernel represented as a $U = T \circ T^{-1}$ rational kernel for the alphabet $\Sigma = \{a, b\}$ and $l = 1$. For the full transducer, all symmetric paths (obtained by exchanging a and b) should be added.

This kernel can be naturally combined with polynomial kernels to form more complex kernels and can be straightforwardly represented by a weighted transducer. Figure 3 shows the corresponding weighted transducer $U = T \circ T^{-1}$ for $\Sigma = \{a, b\}$ and $l = 1$ for the input sequence $x = aaa$. The corresponding weighted transducer T has the same topology as U, but the output label of all the transitions from state 0 to 3 with a mismatch between input and output should instead be a special mismatch symbol, say z, and the transition weight should be the square root of the weight in U.

The loops of state 0 and 3 allow for arbitrary prefixes and suffixes around a window in which the mismatches are evaluated. U contains a unique path from state 0 to state 3 for each possible sequence of matches and mismatches. The weight of each sequence is marked at the last transition and equals the sum of the weights of the matching symbols taken to the power of d, the other transitions weights being one.

With this locality-improved kernel, Zien et al. obtain a 25% performance improvement over previous results on a task with about $13{,}500$ sequences of which $3{,}300$ are positive TIS examples and the rest are considered negative examples.

4. Conclusion

Weighted transducers give a general framework for the representation and computation of sequence kernels. All sequence kernels used in natural language processing, computational biology, and other sequence-related tasks are special instances of rational kernels. This has an important algorithmic advantage since a single general and efficient algorithm can be used to compute such kernels. State-of-the-art implementations of the general algorithms for the use of weighted transducers are available as part of the open-source software library OpenFst [7] and the algorithms for their use as sequence kernels exist as part of the open-source project OpenKernel library [3], freeing up the machine learning practitioner to focus on designing effective kernels for the problem at hand. The OpenKernel library interfaces with the popular software package LIBSVM [10] for easy experimentation with novel PDS rational kernels for classification and regression tasks.

Any weighted transducer T can be used to define a PDS sequence kernel by composing it with its inverse, and existing PDS rational kernels can be combined via standard rational operations to defined more complex PDS rational kernels. This has an important consequence for the design and improvement of sequence kernels. Furthermore, the graphical representation of rational kernels makes it convenient to augment or modify them. For all these reasons, we believe that rational kernels constitute just the *right* algorithmic and representational framework for sequence kernels. Furthermore, sample points can be used to *learn* rational kernels themselves [14]. This helps optimally selecting the specific rational kernel, or the proper transition weights, for the learning task considered.

References

[1] Jürgen Albert and Jarkko Kari. Digital image compression. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of weighted automata*, EATCS Monographs on Theoretical Computer Science. Springer, 2009.

[2] Cyril Allauzen and Mehryar Mohri. N-way composition of weighted finite-state transducers. Technical Report TR2007-902, Courant Institute of Mathematical Sciences, New York University, August 2007.

[3] Cyril Allauzen and Mehryar Mohri. OpenKernel Library, 2007. http://www.openkernel.org.

[4] Cyril Allauzen and Mehryar Mohri. 3-way composition of weighted finite-state transducers. In *Proceedings of the 13th International Conference on Implementation and Application of Automata (CIAA 2008)*, volume 5148 of *Lecture Notes in Computer Science*, pages 262–273, San Francisco, California, July 2008. Springer-Verlag, Heidelberg, Germany.

[5] Cyril Allauzen, Mehryar Mohri, and Michael Riley. Statistical modeling for unit selection in speech synthesis. In 42*nd Meeting of the Association for Computational Linguistics (ACL 2004), Proceedings of the Conference*, Barcelona, Spain, July 2004.

[6] Cyril Allauzen, Mehryar Mohri, and Ameet Talwalkar. Sequence kernels for predicting protein essentiality. In *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, July 2008.

[7] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of the 12th International Conference on Implementation and Application of Automata (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23, Prague, Czech Republic, July 2007. Springer-Verlag, Heidelberg, Germany.

[8] Jean Berstel. *Transductions and Context-Free Languages*. Teubner Studienbücher: Stuttgart, 1979.

[9] Thomas M. Breuel. The OCRopus open source OCR system. In *Proceedings of IS&T/SPIE 20th Annual Symposium*, 2008.

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[11] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *NIPS 14*, Cambridge, MA, 2002. MIT Press.
[12] Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.
[13] Corinna Cortes and Mehryar Mohri. Moment Kernels for Regular Distributions. *Machine Learning*, 60(1-3):117–134, September 2005.
[14] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning sequence kernels. In *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2008), (invited lecture)*, Cancún, Mexico, October 2008.
[15] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge UK, 1998.
[16] Samuel Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974.
[17] David Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.
[18] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin-New York, 1986.
[19] Christina Leslie, Eleazar Eskin, Jason Weston, and William S. Noble. Mismatch String Kernels for SVM Protein Classification. In *NIPS 2002*. MIT Press, 2003.
[20] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. In *NIPS 2000*, pages 563–569. MIT Press, 2001.
[21] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:2, 1997.
[22] 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.
[23] Mehryar Mohri. Statistical natural language processing. In M. Lothaire, editor, *Applied Combinatorics on Words*. Cambridge University Press, 2005.
[24] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted automata in text and speech processing. In *Proceedings of the 12th biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended finite state models of language*, Budapest, Hungary, 1996. John Wiley and Sons, Chichester.
[25] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In Larry Rabiner and Fred Juang, editors, *Handbook on speech processing and speech communication, Part E: Speech recognition*. Springer-Verlag, Heidelberg, Germany, 2008.
[26] Fernando Pereira and Michael Riley. *Finite State Language Processing*, chapter Speech Recognition by Composition of Weighted Finite Automata. The MIT Press, 1997.
[27] Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag: New York, 1978.
[28] Richard Sproat. A finite-state architecture for tokenization and grapheme-to-phoneme conversion in multilingual text analysis. In *Proceedings of the ACL SIGDAT Workshop, Dublin, Ireland*. ACL, 1995.
[29] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and KR. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 9(16):799–807, 2000.