EFFICIENT ALGORITHM FOR RATIONAL KERNEL EVALUATION IN LARGE LATTICE SETS

Jan Švec, Pavel Ircing

Department of Cybernetics
University of West Bohemia
Univerzitní 22, Pilsen, Czech Republic
{honzas,ircing}@kky.zcu.cz

ABSTRACT

This paper presents an effective method for evaluation of the rational kernels represented by finite-state automata. The described algorithm is optimized for processing speed and thus facilitates the usage of state-of-the-art machine learning techniques like Support Vector Machines even in the real-time application of speech and language processing, such as dialogue systems and speech retrieval engines. The performance of the devised algorithm was tested on a spoken language understanding task and the results suggest that it consistently outperforms the baseline algorithm presented in the related literature.

Index Terms— Finite-state machines, Kernels, Support Vector Machines, Natural language processing

1. INTRODUCTION

The effectiveness of many speech and natural language processing systems benefits greatly from the usage of state-of-the-art machine learning techniques such as Support Vector Machines (SVMs). However, there are several issues that have to be dealt with when designing such classification algorithms. First, the speech and language processing tools are often designed in a "cascade" fashion where the (generally ambiguous) output of one module – e.g. speech recognizer – is passed into the next one, for example natural language understanding engine. Those intermediate results often constitute a more structure form than fixed-size feature vectors for which the original classification methods were devised. Typically, they take the form of a lattice represented by an acyclic weighted finite-state acceptor (WFSA) [1].

The theoretically well-founded algorithm for computing the kernel function[1] between two acyclic WFSAs was introduced in [2]. However, specifically in the case of SVM usage, the pairwise calculation between classified lattice and lattices representing the support vectors (which can, by definition, cover the whole training set) becomes very inefficient. Therefore we have designed a new method that capitalizes on effective optimization methods for weighted finite-state transducers (WFSTs) available in the OpenFST library [3], especially determinization and minimization algorithms.

[1] An operation essential for efficient usage of many classification algorithms including SVMs.

2. PAIRWISE KERNEL COMPUTATION

2.1. Basic definitions

Using a notation introduced in [3], we can define a *semiring* as a 5-tuple $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ where $\mathbb{K}$ denotes the set of values, operation $\oplus$ is associative and commutative with identity element $\bar{0}$; operation $\otimes$ is associative with identity element $\bar{1}$, $\otimes$ distributes over $\oplus$ and $\bar{0}$ is an annihilator for $\otimes$, i.e. $a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$ for all $a \in \mathbb{K}$.

Then a *weighted finite-state transducer* (WFST) over a semiring $\mathbb{K}$ is an 8-tuple $T = (\mathcal{A}, \mathcal{B}, Q, I, F, E, \lambda, \rho)$ where $\mathcal{A}$ denotes the finite input alphabet of the transducer, $\mathcal{B}$ is the finite output alphabet, Q is the finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, $E \subseteq Q \times (\mathcal{A} \cup \{\epsilon\}) \times (\mathcal{B} \cup \{\epsilon\}) \times \mathbb{K} \times Q$ is a finite set of transitions, $\lambda : I \to \mathbb{K}$ is the initial weight function and $\rho : F \to \mathbb{K}$ is the final weight function; ϵ denotes an empty string.

For a given transition $e \in E$, let us denote by $p[e]$ the origin (previous) state of the transition e; $n(e)$ then denotes its destination (next) state; $i[e]$, $o[e]$ and $w[e]$ further denote the input symbol, output symbol and weight of the transition e, respectively. The path $\pi = e_1 \dots e_k$ is defined as a sequence of consecutive transitions such that $n[e_{i-1}] = p[e_i], i = 2, \dots k$. Functions n, p and w can then be extended to paths – $n[\pi] = n[e_k]$, $p[\pi] = p[e_1]$ and $w[\pi] = \bigotimes_{i=1}^{k} w[e_i]$. The function w can be further extended to an arbitrary set of paths M as $w[M] = \bigoplus_{\pi \in M} w[\pi]$.

Next, let us denote by $P(q, q')$ the set of paths from q to q' and by $P(q, x, y, q')$ the set of paths from q do q' with input label $x \in \mathcal{A}^*$ and output label $y \in \mathcal{B}^*$. Again, these definitions can be extended to subsets of states $R, R' \subseteq Q$ as follows: $P(R, R') = \bigcup_{q \in R, q' \in R'} P(q, q')$ and $P(R, x, y, R') = \bigcup_{q \in R, q' \in R'} P(q, x, y, q')$.

The transducer T then assigns to any pair $(x, y) \in \mathcal{A}^* \times \mathcal{B}^*$ the weight defined as

$$T(x, y) = \bigoplus_{\pi \in P(I, x, y, F)} \lambda\left[p[\pi]\right] \otimes w[\pi] \otimes \rho\left[n[\pi]\right] \qquad (1)$$

If $P(I, x, y, F) = \emptyset$, i.e., if there is no successful path from initial to final state labeled with (x, y), the $T(x, y) = \bar{0}$. The *weighted finite-state acceptor* can be viewed as a special case of WFST with identical input and output alphabet $\mathcal{A}$ that defines an identical relation between input and output symbols, i.e., $i[e] = o[e]\ \forall e \in E$. This representation simplifies for example the definition of the composition of transducer and acceptor (see below) and is also in this form implemented in [3]. We will also use the term *finite-state au-*

tomaton in the paper in cases when the explicit distinction between transducer and acceptor is not necessary.

Now let us briefly introduce some of the operations with WFSTs that will be utilized in the algorithms below. The *union* of two transducers $T_1(.,)$ and $T_2(.,.)$ is defined as $T_1 \oplus T_2(x,y) = T_1(x,y) \oplus T_2(x,y)$; the *concatenation* of transducers is defined as $T_1 \otimes T_2(x,y) = \bigoplus_{x=x_1x_2, y=y_1y_2} T_1(x_1,y_1) \otimes T_2(x_2,y_2)$ and the *inverse* of a transducer $T(.,.)$ is defined by $T^{-1}(x,y) = T(y,x)$. The *composition* of $T_1(.,)$ and $T_2(.,.)$ is defined as $T_1 \circ T_2(x,y) = \bigoplus_z T_1(x,z) \otimes T_2(z,y)$ and, finally, the *input* and *output projection* of a transducer is defined as $\Pi_1(T)(x) = \bigoplus_y T(x,y)$ and $\Pi_2(T)(y) = \bigoplus_x T(x,y)$, respectively. [1]

The methods introduced in this paper also take an advantage from the efficient optimization algorithms that change the topology of the transducer to optimize their computational efficiency while maintaining the original transducer relation. Those are the *ϵ-removal* (rmeps) removing all the transitions e with $i[e] = o[e] = \epsilon$ [4], *determinization* (det) creating the equivalent transducer such that for each state $q \in Q$ and each symbol $a \in \mathcal{A}$ there is at most one transition leaving the state q labeled with the input symbol a [1] and *minimization* (min) resulting into the equivalent transducer with the minimum number of states [5].

2.2. Rational kernels

A *kernel* over the sets X and Y is a function $K : X \times Y \to \Re$ such that $X, Y \neq \emptyset$. A *rational kernel* is then a kernel defined over sets of strings or weighted automata.

More precisely, a kernel K over $\mathcal{A}^* \times \mathcal{B}^*$ is called rational if there exists a weighted transducer $S = (\mathcal{A}, \mathcal{B}, Q, I, F, E, \lambda, \rho)$ over the semiring $\mathbb{K}$ and a function $\psi : \mathbb{K} \to \Re$ such that for all $x \in \mathcal{A}^*$ a $y \in \mathcal{B}^*$:

$$K(x,y) = \psi\left(S(x,y)\right) \tag{2}$$

The kernel K is then defined by the pair (ψ, S) where ψ is an arbitrary function mapping $\mathbb{K}$ to $\Re$ [2].

Cortes et al. further show in [2] that the definition of rational kernels can be extended to kernels over weighted automata and that the kernel K over the automata A and B can be computed using a composition operation:

$$K(A,B) = \psi\left(w[A \circ S \circ B]\right) \tag{3}$$

under the condition that A and B are acyclic weighted automata over a *closed semiring* $\mathbb{K}$.[2]

Many machine learning techniques (including SVMs) require kernels satisfying Mercer's condition, or equivalently, kernels that are positive definite symmetric (PDS). It has been proven in [2] that for an arbitrary weighted transducer $T = (\mathcal{A}, \mathcal{B}, Q, I, F, E, \lambda, \rho)$ which satisfies the condition that a composition $S = T \circ T^{-1}$ is regulated[3], the pair $(\psi, T \circ T^{-1})$ defines a PDS rational kernel over $\mathcal{A}^* \times \mathcal{A}^*$.

2.3. N-gram kernels over lattices

The rational kernels are, broadly speaking, designed to evaluate some kind of a similarity measure between weighted automata. In many NLP applications, we need to assess the similarity between strings (documents, queries, utterances) or, even more frequently, between more complex representations of those strings emerging from the inherent uncertainty present in the NLP algorithm outcomes. Those representation often take form of a lattice, i.e. acyclic WFSA. One of the plausible measures of similarity between lattices is the number of shared n-grams (or, more precisely for the probabilistic setting, the *expectation* of the number of shared n-grams). As it turns out, the transducer T realizing this measure can be quite straightforwardly defined as

$$T_n = (\mathcal{A} \times \{\epsilon\})^* \otimes (\oplus_{x \in \mathcal{A}} \{x\} \times \{x\})^n \otimes (\mathcal{A} \times \{\epsilon\})^* \tag{4}$$

where $*$ denotes the Kleene closure and n represents the n-fold concatenation (in this case, of the union of all symbols from the input alphabet). The more illustrative graphical representation of T_n can be found in [2]. The kernel computing the n-gram similarity over two lattices A and B is then defined as

$$K_n(A,B) = \psi(w[A \circ T_n \circ T_n^{-1} \circ B]) \tag{5}$$

Now suppose that we want to classify an unknown lattice X using the SVM algorithm while having a training set $\mathcal{T}$ consisting of l lattices. It means that we need to evaluate $K_n(X, U_j)$ for $j = 1, \cdots, l$ as we do not know which training set examples will constitute the support vectors. In other words, we have to perform the following operations for each training sample U_j:

1. Construct the composite transducer $C_j = L \circ R_j$ where L is the determinized composition of classified lattice with the transducer T_n ($L = \det\left[\text{rmeps}\ \Pi_2(X \circ T_n)\right]$) and R_j is the determinized composition of the training set lattice with T_n^{-1} ($R_j = \det\left[\text{rmeps}\ \Pi_1(T_n^{-1} \circ U_j)\right]$ – both those partial composition can be precomputed before iterating over the training set.
2. Compute $w[C_j]$ using the shortest distance algorithm [6].

Since the training sets $\mathcal{T}$ quite commonly contain thousands and even tens of thousands of examples, this baseline algorithm turns out to be rather inefficient.

3. OPTIMIZED KERNEL COMPUTATION ALGORITHM

The design of our optimized algorithm was guided by the idea that many n-grams have to be shared across the training set lattices. Therefore, if we were able to devise a clever way of merging the entire training set into a single WFSA while maintaining the information about the original lattice identities, the WFST optimization algorithms would conflate the shared paths (n-grams) and the whole kernel computation described in Section 2.3 would be essentially reduced to a single composition of transducer representing the classified lattice and a large but minimized transducer representing the whole training set. Technically, the layout of the combined "training set" transducer was inspired by the construction of the factor transducer described in [7].

The resulting optimized algorithm works as follows:

1. Create the composite transducers $R_j = T_n^{-1} \circ U_j$ for $j = 1, \cdots, l$
2. Add the training set indexes $1, \cdots, l$ into the symbol alphabet $\mathcal{A}$ and concatenate all the transducers R_j projected on the input with a simple acceptor $I(j) = \{j\} \times \{j\}$, creating $E_j = \Pi_1(R_j) \otimes I(j)$ for $j = 1, \cdots, l$. In other words, $I(j)$ adds the identifier of the individual training set lattices into the combined transducer and ensures the preservation of the lattice identities throughout the subsequent WFST optimization process.

[2] The tropical semiring fulfills the definition of the closed semiring. In the case of the probability semiring, the Eq. 3 can be used only when A, B and T represent probability distributions.

[3] A transducer is said to be regulated if its transduction weight defined by the Eq. 1 is well-defined and in $\mathbb{K}$ for any pair (x, y).

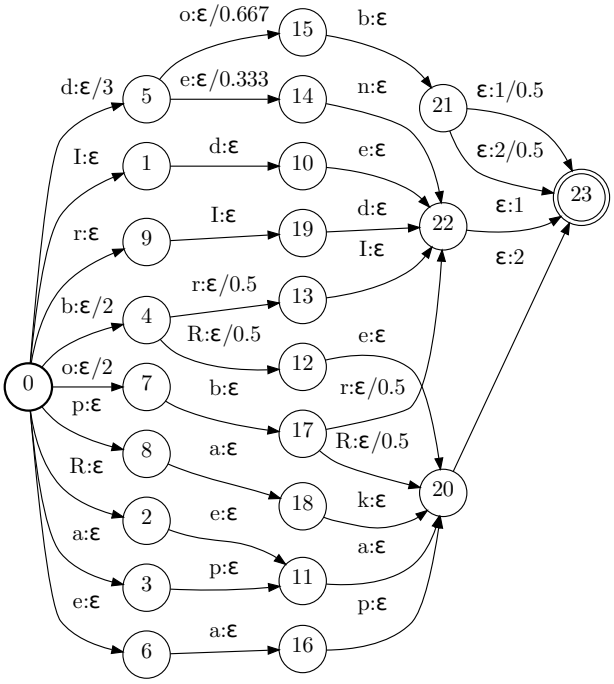


Fig. 1. Example of the combined transducer R representing the entire training set (two utterances). R is defined over the real semiring, the kernel was defined using (ψ_I, T_3) where ψ_I is an identity function.

3. Take the union of all extended acceptors $E_j, j = 1, \cdots, l$ and perform ϵ-removal, determinization and minimization of the resulting acceptor – i.e., create $\bar{R} = \min[\det[\text{rmeps} \bigoplus_j E_j]]$. The $\bar{R}$ is now a minimal WFSA over $\mathcal{A} \cup \{j\}_{j=1}^{l}$.
4. Convert the acceptor $\bar{R}$ into a transducer R by first initializing R with $\bar{R}$[4] and then modifying all transitions e as follows:
   - $i[e] = \epsilon$ if $i[e] \in \{j = 1, 2, \ldots, l\}$
   - $o[e] = \epsilon$ otherwise

The evaluation of the kernel function over the classified lattice and the whole training set is consequently reduced to a single composition $C = L \circ R$ and computing of the $K_n(X, U_j)$ values by traversing the transducer C such that:

$$K_n(X, U_j) = \psi(\bigoplus_{\pi \in P(I,F): o(\pi)=j} \lambda[p[\pi]] \otimes w[\pi] \otimes \rho[n[\pi]]) \quad (6)$$

An example of the optimized combined transducer R created by running the algorithms described above on the training set containing just two phoneme lattices U_1 and U_2 is given in Figure 1.

4. ALGORITHM EVALUATION

We evaluated the presented algorithm on a spoken language understanding task. In this task, the input utterance represented by a lattice X is classified using a set of SVM classifiers into semantic classes called semantic tuples. This discriminative model called Semantic

[4]Remember that the acceptor is represented as a transducer with $i[e] = o[e]\ \forall e \in E$

	Word lattices		Phoneme lattices	
	mean	*std*	*mean*	*std*
# of states	8.3	10.4	37.1	53.0
# of trans.	11.1	20.3	47.1	84.1
# of paths	102.3	1200.8	282.8	1681.6

Table 1. Characteristics of HHTT lattices (mean and standard deviation) computed over 56k lattices.

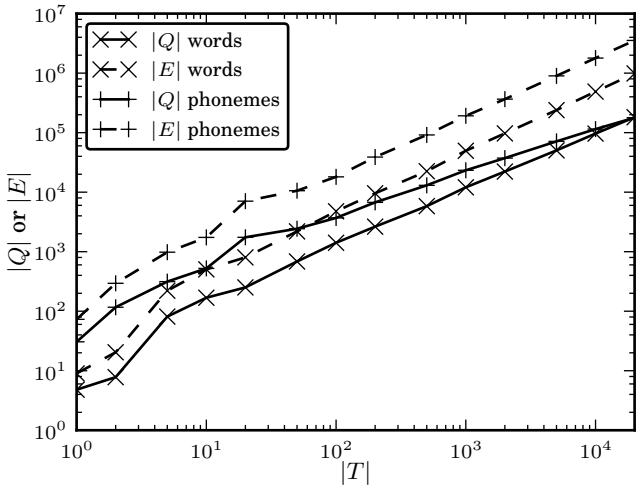


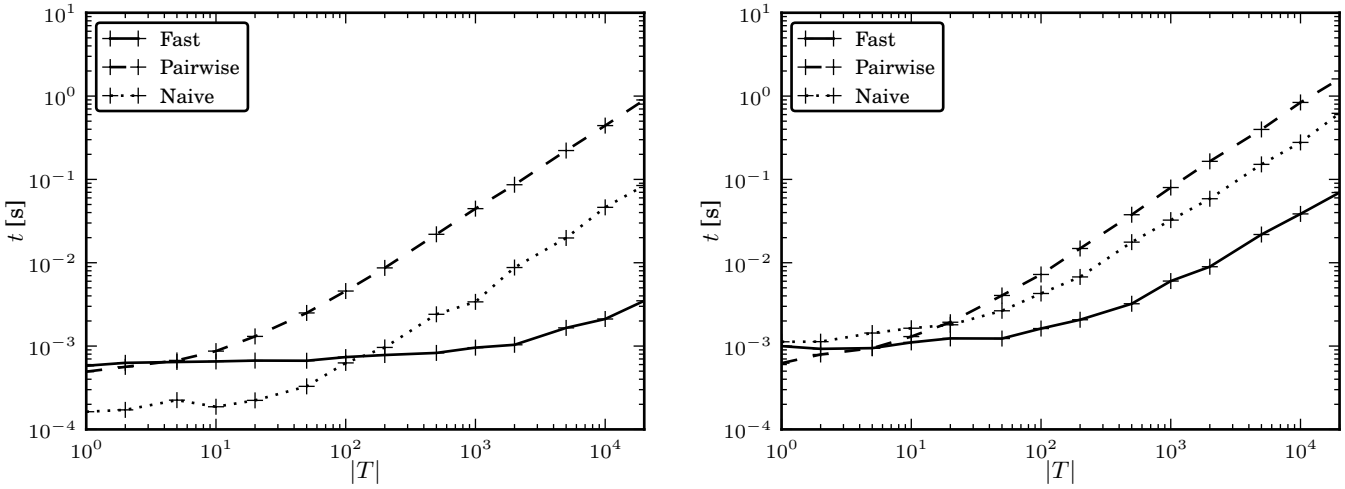
Fig. 2. Dependency of the size of transducer R on the number of training utterances $|T|$, $|Q|$ denotes the number of states, $|E|$ the number of transitions.

Tuple Classifiers (STC) was described in [8]. In our research we were interested in the use of both the word lattices and phoneme lattices to classify the semantics of an utterance. The original STC model used a feature vector composed from lexico-syntactic features computed from the word sequence (one best hypothesis). The algorithm for the fast computation of rational kernels was developed to evaluate directly the kernel function between X and the whole training set U_j instead of computing the explicit feature vector representation. The rapid computation is necessary to allow real-time processing of utterances and to provide a natural dialog.

	Word lattices		Phoneme lattices	
Operation	[ms]	[%]	[ms]	[%]
$A = X \circ T$	0.513	11.56%	0.449	0.54%
$B = \Pi_2(A)$	0.027	0.61%	0.135	0.16%
$C = \text{rmeps } B$	0.107	2.41%	2.290	2.77%
$L = \det C$	0.038	0.86%	0.302	0.37%
$L \circ R$	0.630	14.20%	18.079	21.86%
traverse $L \circ R$	3.121	70.36%	61.446	74.30%
Σ	4.436	100.00%	82.701	100.00%

Table 2. Mean computational times of partial operations during the rational kernel functions evaluation. Absolute and relative values, word and phoneme lattices, $|T| = 2 \cdot 10^4$.

We experimentally evaluated the performance of the algorithm on the described task. We varied the size of the training set. The



(a) Word lattices

(b) Phoneme lattices

Fig. 3. Medians of computational times required to evaluate $|T|$ kernel functions between one test utterance and $|T|$ training utterances. Utterances were represented using word and phoneme lattices. The described algorithm is denoted as fast (solid line), baseline algorithms were pairwise computation (dashed line) and naive implementation (dotted line). Smaller times are better, logarithmic axes.

computational times required to evaluate Eq. 6 against all training utterances was measured. The median of the computational time in dependency on the size of the training set is depicted in Fig. 4. We used word and phoneme lattices generated from our in-house speech recognizer [9] and the Human-Human Train Timetable (HHTT) dialog corpus [10]. The corpus contains about 56k utterances which were recognized into word and phoneme lattices. The characteristics of these lattices are summarized in 1.

For each size of training set, the algorithm was performed 5-times for randomly chosen training utterances from the whole corpus. 100 utterances were selected as test set during each repetition. Then for each test utterance the kernel functions between the test lattice and the lattices from training set were computed and the computational times were measured. In the experiment, we used n-gram transducer defined by $T = \bigoplus_{n=1}^{N} T_n$. The automaton T transduces the output symbols of path from the lattice X into a set of n-grams of length $1, 2, \ldots N$. We used $N = 5$ in the experiments presented in this paper. The logarithmic semiring and the mapping $\psi(w) = \mathrm{e}^{-w}$ was used in order to ensure the numerical stability of transducer optimization algorithms. The kernel $K(X, U_j)$ is then defined by the tuple $(\psi(w), T)$.

The performance of the optimized algorithm (solid line) is compared with a pairwise kernel computation (dotted line) and with an naive algorithm (dashed line). The naive algorithm uses explicit feature vector representation composed from a sparse feature vector. The sparse vector contains expected counts of n-grams and the dot product of these vectors is equivalent to computation of n-gram rational kernel.

While the pairwise computation is linearly dependent on the number of training samples, the computational demands of an optimized algorithm are growing much more slowly. For the word lattices and the training set size of 20k utterances, the median of computational times was only about 3.50ms (4.69 mean). In the case when the phoneme lattices were used as an input, we achieved median time of about 69.6ms (86.9ms mean) for 20k utterances.

The Fig. 2 depicts the dependency of the number of states $|Q|$ and the number of transitions $|E|$ of the transducer R on the number of training lattices used to construct R.

The asymptotic complexity of the described algorithm is influenced by the fact that input lattices are acyclic automata. Therefore the asymptotic complexity of partial operations of this algorithm is better then for a generic automaton. We assume that the transducer R is precomputed during the training phase. In the case of n-gram rational kernels, the automaton L is in fact the subset of factor automaton. The size of this automaton is linear with the size of X and, moreover, the automaton is constructible in linear time [11]. For a fixed R, the time and memory complexity of $L \circ R$ is $\mathcal{O}(|Q_L| \cdot |D_L|)$ where $|D_L|$ is a maximum out-degree of L because the terms dependent on R are just a multiplicative constant [12].

5. CONCLUSION AND FUTURE WORK

The paper presented an optimized algorithm for computing rational kernels over the acyclic finite state automata. It was experimentally shown that its speed substantially outperforms the baseline pairwise computation and, for phoneme lattices and larger word lattice training sets, also the method that employs explicit evaluation of the dot product of the feature vectors. The optimized kernel evaluation is used in a spoken language understanding module [13] employing SVMs where the reaction time highly influences the dialog system user experience.

Even though the paper reported the results of experiments performed with an n-gram kernel within a spoken language understanding task, the algorithm should be just as effective with other rational kernels and in other related tasks. We are currently planning to test its performance within a topic detection system.

6. ACKNOWLEDGEMENTS

This research was supported by the Grant Agency of the Czech Republic, project No. GAČR P103/12/G084.

7. REFERENCES

[1] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[2] Corinna Cortes, Patrick Haffner, and Mehryar Mohri, "Rational kernels: Theory and algorithms," *The Journal of Machine Learning*, vol. 5, pp. 1035–1062, 2004.

[3] Cyril Allauzen, Michael Riley, and Johan Schalkwyk, "OpenFst: A general and efficient weighted finite-state transducer library," *Implementation and Application of Automata*, vol. 4783, pp. 11–23, 2007.

[4] Mehryar Mohri, "Generic epsilon-removal and Input epsilon-normalization Algorithms for Weighted Transducers," *International Journal of Foundations of Computer Science*, vol. 13, no. 1, pp. 129–143, 2002.

[5] Mehryar Mohri, "Minimization algorithms for sequential transducers," *Theoretical Computer Science*, vol. 234, no. 1-2, pp. 177–201, 2000.

[6] Mehryar Mohri, "Semiring frameworks and algorithms for shortest-distance problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, pp. 321–350, 2002.

[7] Dogan Can and Murat Saraclar, "Lattice Indexing for Spoken Term Detection," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 8, pp. 2338–2347, 2011.

[8] François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young, "Spoken language understanding from unaligned data using discriminative classification models," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.*, Taipei, 2009, pp. 4749–4752, IEEE.

[9] Aleš Pražák, Josef V. Psutka, Jan Hoidekr, Jakub Kanis, Luděk Müller, and Josef Psutka, "Automatic online subtitling of the Czech parliament meetings," *Text, Speech and Dialogue*, vol. 4188, no. 1, pp. 501–508, 2006.

[10] Filip Jurčíček, Jií Zahradil, and Libor Jelínek, "A human-human train timetable dialogue corpus," *Proceedings of EUROSPEECH, Lisboa*, pp. 1525–1528, 2005.

[11] Mehryar Mohri, Pedro Moreno, and Eugene Weinstein, "Factor automata of automata and applications," *Implementation and Application of Automata*, vol. 4783, pp. 168–179, 2007.

[12] Cyril Allauzen, Michael Riley, and Johan Schalkwyk, "Filters for efficient composition of weighted finite-state transducers," *Implementation and Application of Automata*, vol. 6482, pp. 28–38, 2011.

[13] Jan Švec, Luboš Šmídl, and Pavel Ircing, "Hierarchical Discriminative Model for Spoken Language Understanding," in *IEEE International Conference on Acoustics Speech and Signal Processing*, Vancouver, Canada, 2013, IEEE.