Expected Sequence Similarity Maximization

Cyril Allauzen[1], **Shankar Kumar**[1], **Wolfgang Macherey**[1], **Mehryar Mohri**[2,1] **and Michael Riley**[1]
[1]Google Research, 76 Ninth Avenue, New York, NY 10011
[2]Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012

Abstract

This paper presents efficient algorithms for expected similarity maximization, which coincides with minimum Bayes decoding for a similarity-based loss function. Our algorithms are designed for similarity functions that are sequence kernels in a general class of positive definite symmetric kernels. We discuss both a general algorithm and a more efficient algorithm applicable in a common unambiguous scenario. We also describe the application of our algorithms to machine translation and report the results of experiments with several translation data sets which demonstrate a substantial speed-up. In particular, our results show a speed-up by two orders of magnitude with respect to the original method of Tromble et al. (2008) and by a factor of 3 or more even with respect to an approximate algorithm specifically designed for that task. These results open the path for the exploration of more appropriate or optimal kernels for the specific tasks considered.

1 Introduction

The output of many complex natural language processing systems such as information extraction, speech recognition, or machine translation systems is a probabilistic automaton. Exploiting the full information provided by this probabilistic automaton can lead to more accurate results than just using the one-best sequence.

Different techniques have been explored in the past to take advantage of the full lattice, some based on the use of a more complex model applied to the automaton as in rescoring, others using additional data or information for reranking the hypotheses represented by the automaton. One method for using these probabilistic automata that has been successful in large-vocabulary speech recognition (Goel and Byrne, 2000) and machine translation (Kumar and Byrne, 2004; Tromble et al., 2008) applications and that requires no additional data or other complex models is the minimum Bayes risk (MBR) decoding technique. This returns that sequence of the automaton having the minimum expected loss with respect to all sequences accepted by the automaton (Bickel and Doksum, 2001). Often, minimizing the loss function L can be equivalently viewed as maximizing a similarity function K between sequences, which corresponds to a kernel function when it is positive definite symmetric (Berg et al., 1984). The technique can then be thought of as an *expected sequence similarity maximization*.

This paper considers this expected similarity maximization view. Since different similarity functions can be used within this framework, one may wish to select the one that is the most appropriate or relevant to the task considered. However, a crucial requirement for this choice to be realistic is to ensure that for the family of similarity functions considered the expected similarity maximization is efficiently computable. Thus, we primarily focus on this algorithmic problem in this paper, leaving it to future work to study the question of determining how to select the similarity function and report on the benefits of this choice.

A general family of sequence kernels including the sequence kernels used in computational biology, text categorization, spoken-dialog classification, and many other tasks is that of *rational kernels* (Cortes et al., 2004). We show how the expected similarity maximization can be efficiently computed for these kernels. In section 3, we describe more specifically the framework of expected similarity maximization in the case of rational kernels and the corresponding algorithmic problem. In Section 4, we describe both a general method for the computation of the expected similarity maximization, and a more efficient method that can be used with a broad sub-family of rational kernels that verify a condition of non-ambiguity. This latter family includes the class of n-gram kernels which have been previously used to apply MBR to machine translation (Tromble et al., 2008). We examine in more detail the use and application of our algorithms to machine translation in Section 5. Section 6 reports the results of experiments applying our algorithms in several large data sets in machine translation. These experiments demonstrate the efficiency of our algorithm which is shown empirically to be two orders of magnitude faster than Tromble et al. (2008) and more than 3 times faster than even an approximation algorithm specifically designed for this problem (Kumar et al., 2009). We start with some preliminary definitions and algorithms related to weighted automata and transducers, following the definitions and terminology of Cortes et al. (2004).

2 Preliminaries

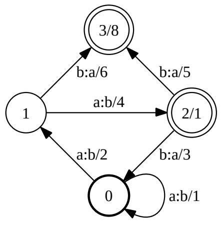
Weighted transducers are finite-state transducers in which each transition carries some weight in addition to the input and output labels. The weight set has the structure of a semiring.

A *semiring* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ verifies all the axioms of a ring except from the existence of a negative element $-x$ for each $x \in \mathbb{K}$, which it may verify or not. Thus, roughly speaking, a semiring is a ring that may lack negation. It is specified by a set of values $\mathbb{K}$, two binary operations $\oplus$ and $\otimes$, and two designated values $\overline{0}$ and $\overline{1}$. When $\otimes$ is commutative, the semiring is said to be *commutative*.

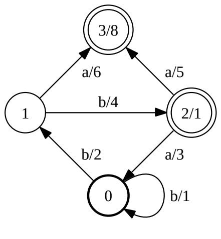
The *real semiring* $(\mathbb{R}_+, +, \times, 0, 1)$ is used when the weights represent probabilities. The *log semiring* $(\mathbb{R} \cup \{-\infty, +\infty\}, \oplus_{\log}, +, \infty, 0)$ is isomorphic to the real semiring via the negative-$\log$ mapping and is often used in practice for numerical stability. The *tropical semiring* $(\mathbb{R}\cup, \{-\infty, +\infty\}, \min, +, \infty, 0)$ is derived from the log semiring via the *Viterbi approximation* and is often used in shortest-path applications.

Figure 1(a) shows an example of a weighted finite-state transducer over the real semiring $(\mathbb{R}_+, +, \times, 0, 1)$. In this figure, the input and output labels of a transition are separated by a colon delimiter and the weight is indicated after the slash separator. A weighted transducer has a set of initial states represented in the figure by a bold circle and a set of final states, represented by double circles. A path from an initial state to a final state is an accepting path.

The weight of an accepting path is obtained by first $\otimes$-multiplying the weights of its constituent



(a)



(b)

Figure 1: (a) Example of weighted transducer T over the real semiring $(\mathbb{R}_+, +, \times, 0, 1)$. (b) Example of weighted automaton A. A can be obtained from T by projection on the output and $T(aab, bba) = A(bba) = 1 \times 2 \times 6 \times 8 + 2 \times 4 \times 5 \times 8$.

transitions and $\otimes$-multiplying this product on the left by the weight of the initial state of the path (which equals $\overline{1}$ in our work) and on the right by the weight of the final state of the path (displayed after the slash in the figure). The weight associated by a weighted transducer T to a pair of strings $(x, y) \in \Sigma^* \times \Sigma^*$ is denoted by $T(x, y)$ and is obtained by $\oplus$-summing the weights of all accepting paths with input label x and output label y.

For any transducer T, T^{-1} denotes its *inverse*, that is the transducer obtained from T by swapping the input and output labels of each transition. For all $x, y \in \Sigma^*$, we have $T^{-1}(x, y) = T(y, x)$.

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 semiring is commutative and the sum:

$$(T_1 \circ T_2)(x, y) = \sum_{z \in \Sigma^*} T_1(x, z) \otimes T_2(z, y) \quad (1)$$

is well-defined and in $\mathbb{K}$ for all x, y (Salomaa and Soittola, 1978).

Weighted automata can be defined as weighted transducers A with identical input and output labels, for any transition. Since only pairs of the form (x, x) can have a non-zero weight associated to them by A, we denote the weight associated by A to (x, x) by $A(x)$ and call it the *weight associated by A to x*. Similarly, in the graph representation of weighted automata, the output (or input) label is omitted. Figure 1(b) shows an example of a weighted automaton. When A and B are weighted automata, $A \circ B$ is called the *intersection* of A and B. Omitting the input labels of a weighted transducer T results in a weighted automaton which is said to be the *output projection of* T.

3 General Framework

Let X be a probabilistic automaton representing the output of a complex model for a specific query input. The model may be for example a speech recognition system, an information extraction system, or a machine translation system (which originally motivated our study). For machine translation, the sequences accepted by X are the potential translations of the input sentence, each with some probability given by X.

Let Σ be the alphabet for the task considered, e.g., words of the target language in machine translation, and let $L\colon \Sigma^* \times \Sigma^* \to \mathbb{R}$ denote a loss function defined over the sequences on that alphabet. Given a reference or hypothesis set $H \subseteq \Sigma^*$, minimum Bayes risk (MBR) decoding consists of selecting a hypothesis $x \in H$ with minimum expected loss with respect to the probability distribution X (Bickel and Doksum, 2001; Tromble et al., 2008):

$$\widehat{x} = \operatorname*{argmin}_{x \in H} \operatorname*{E}_{x' \sim X}[L(x, x')]. \tag{2}$$

Here, we shall consider the case, frequent in practice, where minimizing the loss L is equivalent to maximizing a similarity measure $K\colon \Sigma^* \times \Sigma^* \to \mathbb{R}$. When K is a sequence kernel that can be represented by weighted transducers, it is a *rational kernel* (Cortes et al., 2004). The problem is then equivalent to the following *expected similarity maximization*:

$$\widehat{x} = \operatorname*{argmax}_{x \in H} \operatorname*{E}_{x' \sim X}[K(x, x')]. \tag{3}$$

When K is a positive definite symmetric rational kernel, it can often be rewritten as $K(x, y) = (T \circ T^{-1})(x, y)$, where T is a weighted transducer over the semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$. Equation (3) can then be rewritten as

$$\begin{aligned} \widehat{x} &= \operatorname*{argmax}_{x \in H} \operatorname*{E}_{x' \sim X}[(T \circ T^{-1})(x, x')] & (4)\\ &= \operatorname*{argmax}_{x \in H} \|A(x) \circ T \circ T^{-1} \circ X\|, & (5) \end{aligned}$$

where we denote by $A(x)$ an automaton accepting (only) the string x and by $\|\cdot\|$ the sum of the weights of all accepted paths of a transducer.

4 Algorithms

4.1 General method

Equation (5) could suggest computing $A(x) \circ T \circ T^{-1} \circ X$ for each possible $x \in H$. Instead, we can compute a composition based on an automaton accepting all sequences in H, $A(H)$. This leads to a straightforward method for determining the sequence maximizing the expected similarity having the following steps:

1. compute the composition $X \circ T$, project on the output and optimize (epsilon-remove, determinize, minimize (Mohri, 2009)) and let Y_2 be the result;[1]

2. compute the composition $Y_1 = A(H) \circ T$;

3. compute $Y_1 \circ Y_2$ and project on the input, let Z be the result;[2]

4. determinize Z;

5. find the maximum weight path with the label of that path giving $\widehat{x}$.

While this method can be efficient in various scenarios, in some instances the weighted determinization yielding Z can be both space- and time-consuming, even though the input is acyclic. The next two sections describe more efficient algorithms.

Note that in practice, for numerical stability, all of these computations are done in the log semiring which is isomorphic to $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$. In particular, the maximum weight path in the last step is then obtained by using a standard single-source shortest-path algorithm.

4.2 Efficient method for n-gram kernels

A common family of rational kernels is the family of n-gram kernels. These kernels are widely use as a similarity measure in natural language processing and computational biology applications, see (Leslie et al., 2002; Lodhi et al., 2002) for instance.

The n-gram kernel K_n of order n is defined as

$$K_n(x, y) = \sum_{|z|=n} c_x(z) c_y(z), \tag{6}$$

where $c_x(z)$ is the number of occurrences of z in x. K_n is a positive definite symmetric rational kernel since it corresponds to the weighted transducer $T_n \circ T_n^{-1}$ where the transducer T_n is defined such that $T_n(x, z) = c_x(z)$ for all $x, z \in \Sigma^*$ with $|z| = n$.

[1]Equivalent to computing $T^{-1} \circ X$ and projecting on the input.

[2]Z is then the projection on the input of $A(H) \circ T \circ T^{-1} \circ X$.

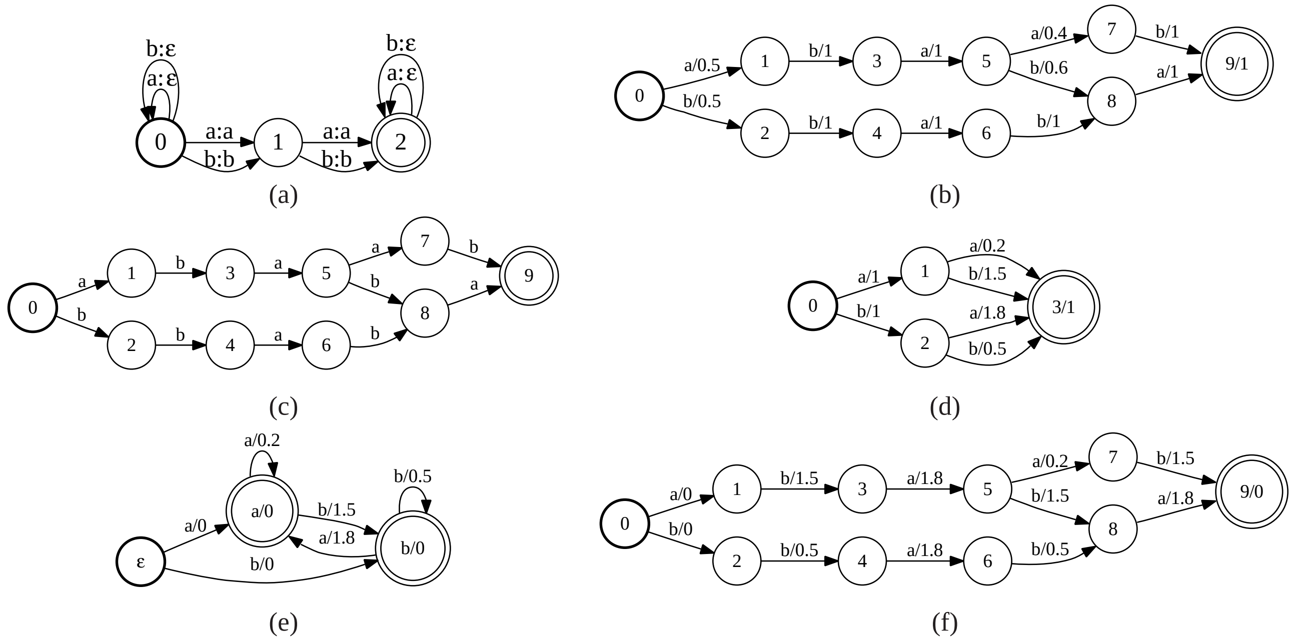


Figure 2: Efficient method for bigram kernel: (a) Counting transducer T_2 for $\Sigma = \{a, b\}$ (over the real semiring). (b) Probabilistic automaton X (over the real semiring). (c) The hypothesis automaton $A(H)$ (unweighted). (d) Automaton Y_2 representing the expected bigram counts in X (over the real semiring). (e) Automaton Y_1: the context dependency model derived from Y_2 (over the tropical semiring). (f) The composition $A(H) \circ Y_1$ (over the tropical semiring).

The transducer T_2 for $\Sigma = \{a, b\}$ is shown in Figure 2(a).

Taking advantage of the special structure of n-gram kernels and of the fact that $A(H)$ is an unweighted automaton, we can devise a new and significantly more efficient method for computing $\widehat{x}$ based on the following steps.

1. *Compute the expected n-gram counts in X:* We compute the composition $X \circ T$, project on output and optimize (epsilon-remove, determinize, minimize) and let Y_2 be the result. Observe that the weighted automaton Y_2 is a compact representation of the expected n-gram counts in X, i.e. for an n-gram w (i.e. $|w| = n$):

$$\begin{aligned} Y_2(w) &= \sum_{x \in \Sigma^*} X(x) c_x(w) \\ &= \mathop{\mathrm{E}}_{x \sim X}[c_x(w)] = c_X(w). \end{aligned} \tag{7}$$

2. *Construct a context-dependency model:* We compute the weighted automaton Y_1 over the tropical semiring as follow: the set of states is $Q = \{w \in \Sigma^* \,|\, |w| \leq n \text{ and } w \text{ occurs in } X\}$, the initial state being ϵ and every state being final; the set of transitions E contains all 4-tuple (origin, label, weight, destination) of the form:
   - $(w, a, 0, wa)$ with $wa \in Q$ and $|w| \leq n - 2$ and
   - $(aw, b, Y_2(awb), wb)$ with $Y_2(awb) \neq 0$ and $|w| = n - 2$

   where $a, b \in \Sigma$ and $w \in \Sigma^*$. Observe that $w \in Q$ when $wa \in Q$ and that $aw, wb \in Q$ when $Y_2(awb) \neq 0$. Given a string x, we have

$$Y_1(x) = \sum_{|w|=n} c_X(w) c_x(w). \tag{8}$$

   Observe that Y_1 is a deterministic automaton, hence $Y_1(x)$ can be computed in $O(|x|)$ time.

3. *Compute $\widehat{x}$:* We compute the composition $A(H) \circ Y_1$. $\widehat{x}$ is then the label of the accepting path with the largest weight in this transducer and can be obtained by applying a shortest-path algorithm to $-A(H) \circ Y_1$ in the tropical semiring.

The main computational advantage of this method is that it avoids the determinization of Z in the

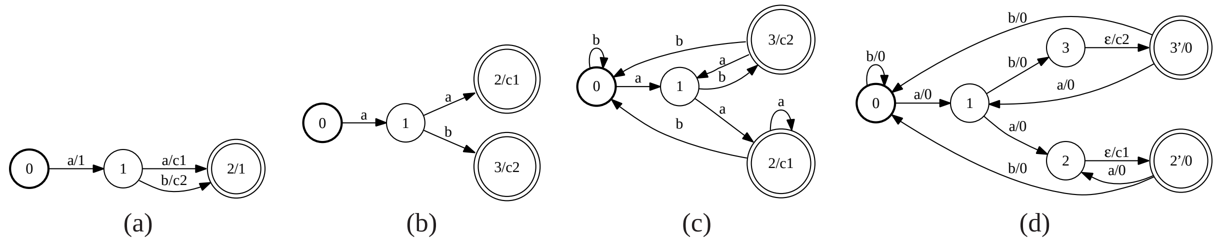


Figure 3: Illustration of the construction of Y_1 in the unambiguous case. (a) Weighted automaton Y_2 (over the real semiring). (b) Deterministic tree automaton Y_2' accepting $\{aa, ab\}$ (over the tropical semiring). (c) Result of determinization of $\Sigma^* Y_2'$ (over the tropical semiring). (d) Weighted automaton Y_1 (over the tropical semiring).

$(+, \times)$ semiring, which can sometimes be costly. The method has also been shown empirically to be significantly faster than the one described in the previous section.

The algorithm is illustrated in Figure 2. The alphabet is $\Sigma = \{a, b\}$ and the counting transducer corresponding to the bigram kernel is given in Figure 2(a). The evidence probabilistic automaton X is given in Figure 2(b) and we use as hypothesis set the set of strings that were assigned a non-zero probability by X; this set is represented by the deterministic finite automaton $A(H)$ given in Figure 2(c). The result of step 1 of the algorithm is the weighted automaton Y_2 over the real semiring given in Figure 2(d). The result of step 2 is the weighted automaton Y_1 over the tropical semiring is given in Figure 2(e). Finally, the result of the composition $A(H) \circ Y_1$ (step 3) is the weighted automaton over the tropical semiring given in Figure 2(f). The result of the expected similarity maximization is the string $\widehat{x} = ababa$, which is obtained by applying a shortest-path algorithm to $-A(H) \circ Y_1$. Observe that the string $\overline{x}$ with the largest probability in X is $\overline{x} = bbaba$ and is hence different from $\widehat{x} = ababa$ in this example.

4.3 Efficient method for the unambiguous case

The algorithm presented in the previous section for n-gram kernels can be generalized to handle a wide variety of rational kernels.

Let K be an arbitrary rational kernel defined by a weighted transducer T. Let X_T denote the regular language of the strings output by T. We shall assume that X_T is a finite language, though the results of this section generalize to the infinite case. Let $\overline{\Sigma}$ denote a new alphabet defined by $\overline{\Sigma} = \{\#_x : x \in X_T\}$ and consider the simple grammar G of context-dependent batch rules:

$$\epsilon \rightarrow \#_x / x\_\epsilon. \tag{9}$$

Each such rule inserts the symbol $\#_x$ immediately after an occurrence x in the input string. For batch context-dependent rules, the context of the application for all rules is determined at once before their application (Kaplan and Kay, 1994). Assume that this grammar is *unambiguous* for a parallel application of the rules. This condition means that there is a unique way of parsing an input string using the strings of X_T. The assumption holds for n-gram sequences, for example, since the rules applicable are uniquely determined by the n-grams (making the previous section a special case).

Given an acyclic weighted automaton Y_2 over the tropical semiring accepting a subset of X_T, we can construct a deterministic weighted automaton Y_1 for $\Sigma^* L(Y_2)$ when this grammar is unambiguous. The weight assigned by Y_1 to an input string is then the sum of the weights of the substrings accepted by Y_2. This can be achieved using weighted determinization.

This suggests a new method for generalizing Step 2 of the algorithm described in the previous section as follows (see illustration in Figure 3):

(i) use Y_2 to construct a deterministic weighted tree Y_2' defined on the tropical semiring accepting the same strings as Y_2 with the same weights, with the final weights equal to the total weight given by Y_2 to the string ending at that leaf;

(ii) let Y_1 be the weighted automaton obtained by first adding self-loops labeled with all elements of Σ at the initial state of Y_2' and then determinizing it, and then inserting new transitions leaving final states as described in (Mohri and Sproat, 1996).

Step (ii) consists of computing a deterministic weighted automaton for $\Sigma^* Y_2'$. This step corresponds to the Aho-Corasick construction (Aho and Corasick, 1975) and can be done in time linear in the size of Y_2'.

This approach assumes that the grammar G of batch context-dependent rules inferred by X_T is unambiguous. This can be tested by constructing the finite automaton corresponding to all rules in G. The grammar G is unambiguous iff the resulting automaton is unambiguous (which can be tested using a classical algorithm). An alternative and more efficient test consists of checking the presence of a *failure* or *default* transition to a final state during the Aho-Corasick construction, which occurs if and only if there is ambiguity.

5 Application to Machine Translation

In machine translation, the BLEU score (Papineni et al., 2001) is typically used as an evaluation metric. In (Tromble et al., 2008), a Minimum Bayes-Risk decoding approach for MT lattices was introduced.[3] The loss function used in that approach was an approximation of the log-BLEU score by a linear function of n-gram matches and candidate length. This loss function corresponds to the following similarity measure:

$$K_{LB}(x, x') = \theta_0 |x'| + \sum_{|w| \le n} \theta_{|w|} c_x(w) 1_{x'}(w). \tag{10}$$

where $1_x(w)$ is 1 if w occurs in x and 0 otherwise.

(Tromble et al., 2008) implements the MBR decoder using weighted automata operations. First, the set of n-grams is extracted from the lattice. Next, the posterior probability $p(w|X)$ of each n-gram is computed. Starting with the unweighted lattice $A(H)$, the contribution of each n-gram w to (10) is applied by iteratively composing with the weighted automaton corresponding to $\overline{w}(w/(\theta_{|w|} p(w|X))\overline{w})^*$ where $\overline{w} = \Sigma^* \setminus (\Sigma^* w \Sigma^*)$. Finally, the MBR hypothesis is extracted as the best path in the automaton. The above steps are carried out one n-gram at a time. For a moderately large lattice, there can be several thousands of n-grams and the procedure becomes expensive. This leads us to investigate methods that do not require processing the n-grams one at a time in order to achieve greater efficiency.

[3]Related approaches were presented in (DeNero et al., 2009; Kumar et al., 2009; Li et al., 2009).

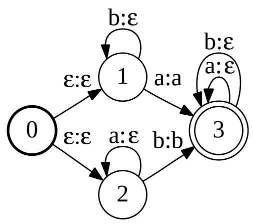


Figure 4: Transducer $\overline{T}_1$ over the real semiring for the alphabet $\{a, b\}$.

The first idea is to approximate the K_{LB} similarity measure using a weighted sum of n-gram kernels. This corresponds to approximating $1_{x'}(w)$ by $c_{x'}(w)$ in (10). This leads us to the following similarity measure:

$$\begin{aligned} K_{NG}(x, x') &= \theta_0 |x'| + \sum_{|w| \le n} \theta_{|w|} c_x(w) c_{x'}(w) \\ &= \theta_0 |x'| + \sum_{1 \le i \le n} \theta_i K_i(x, x') \end{aligned} \tag{11}$$

Intuitively, the larger the length of w the less likely it is that $c_x(w) \neq 1_x(w)$, which suggests computing the contribution to $K_{LB}(x, x')$ of lower-order n-grams ($|w| \le k$) exactly, but using the approximation by n-gram kernels for the higher-order n-grams ($|w| > k$). This gives the following similarity measure:

$$\begin{aligned} K^k_{NG}(x, x') = \theta_0 |x'| &+ \sum_{1 \le |w| \le k} \theta_{|w|} c_x(w) 1_{x'}(w) \\ &+ \sum_{k < |w| \le n} \theta_{|w|} c_x(w) c_{x'}(w) \end{aligned} \tag{12}$$

Observe that $K^0_{NG} = K_{NG}$ and $K^n_{NG} = K_{LB}$.

All these similarity measures can still be computed using the framework described in Section 4. Indeed, there exists a transducer $\overline{T}_n$ over the real semiring such that $\overline{T}_n(x, z) = 1_x(z)$ for all $x \in \Sigma^*$ and $z \in \Sigma^n$. The transducer $\overline{T}_1$ for $\Sigma = \{a, b\}$ is given by Figure 4. Let us define the similarity measure $\overline{K}_n$ as:

$$\overline{K}_n(x, x') = (T_n \circ \overline{T}_n^{-1})(x, x') = \sum_{|w|=n} c_x(w) 1_{x'}(w). \tag{13}$$

Observe that the framework described in Section 4 can still be applied even though $\overline{K}_n$ is not symmetric. The similarity measures K_{LB}, K_{NG} and K^k_{NG}

	zhen					aren				
	nist02	nist04	nist05	nist06	nist08	nist02	nist04	nist05	nist06	nist08
no mbr	38.7	39.2	38.3	33.5	26.5	64.0	51.8	57.3	45.5	43.8
exact	37.0	39.2	38.6	34.3	27.5	65.2	51.4	58.1	45.2	45.0
approx	39.0	39.9	38.6	34.4	27.4	65.2	52.5	58.1	46.2	45.0
ngram	36.6	39.1	38.1	34.4	27.7	64.3	50.1	56.7	44.1	42.8
ngram1	37.1	39.2	38.5	34.4	27.5	65.2	51.4	58.0	45.2	44.8

Table 1: BLEU score (%)

	zhen					aren				
	nist02	nist04	nist05	nist06	nist08	nist02	nist04	nist05	nist06	nist08
exact	3560	7863	5553	6313	5738	12341	23266	11152	11417	11405
approx	168	422	279	335	328	504	1296	528	619	808
ngram	28	72	34	70	43	85	368	105	63	66
ngram1	58	175	96	99	89	368	943	308	167	191

Table 2: MBR Time (in seconds)

can then be expressed as the relevant linear combination of K_i and $\overline{K}_i$.

6 Experimental Results

Lattices were generated using a phrase-based MT system similar to the alignment template system described in (Och and Ney, 2004). Given a source sentence, the system produces a word lattice A that is a compact representation of a very large N-best list of translation hypotheses for that source sentence and their likelihoods. The lattice A is converted into a lattice X that represents a probability distribution (i.e. the posterior probability distribution given the source sentence) following:

$$X(x) = \frac{\exp(\alpha A(x))}{\sum_{y \in \Sigma^*} \exp(\alpha A(y))} \quad (14)$$

where the scaling factor $\alpha \in [0, \infty)$ flattens the distribution when $\alpha < 1$ and sharpens it when $\alpha > 1$. We then applied the methods described in Section 5 to the lattice X using as hypothesis set H the unweighted lattice obtained from X.

The following parameters for the n-gram factors were used:

$$\theta_0 = \frac{-1}{T} \text{ and } \theta_n = \frac{1}{4Tpr^{n-1}} \text{ for } n \geq 1. \quad (15)$$

Experiments were conducted on two language pairs Arabic-English (aren) and Chinese-English (zhen) and for a variety of datasets from the NIST Open Machine Translation (OpenMT) Evaluation.[4] The values of α, p and r used for each pair are given in Table 3. We used the IBM implementation of the BLEU score (Papineni et al., 2001).

[4]http://www.nist.gov/speech/tests/mt

	α	p	r
aren	0.2	0.85	0.72
zhen	0.1	0.80	0.62

Table 3: Parameters used for performing MBR.

We implemented the following methods using the OpenFst library (Allauzen et al., 2007):

- *exact*: uses the similarity measure K_{LB} based on the linearized log-BLEU, implemented as described in (Tromble et al., 2008);
- *approx*: uses the approximation to K_{LB} from (Kumar et al., 2009) and described in the appendix;
- *ngram*: uses the similarity measure K_{NG} implemented using the algorithm of Section 4.2;
- *ngram1*: uses the similarity measure K^1_{NG} also implemented using the algorithm of Section 4.2.

The results from Tables 1-2 show that *ngram1* performs as well as *exact* on all datasets[5] while being two orders of magnitude faster than *exact* and overall more than 3 times faster than *approx*.

7 Conclusion

We showed that for broad families of transducers T and thus rational kernels, the expected similar-

[5]We consider BLEU score differences of less than 0.4% not significant (Koehn, 2004).

ity maximization problem can be solved efficiently. This opens up the option of seeking the most appropriate rational kernel or transducer T for the specific task considered. In particular, the kernel K used in our machine translation applications might not be optimal. One may well imagine for example that some n-grams should be further emphasized and others de-emphasized in the definition of the similarity. This can be easily accommodated in the framework of rational kernels by modifying the transition weights of T. But, ideally, one would wish to select those weights in an optimal fashion. As mentioned earlier, we leave this question to future work. However, we can offer a brief look at how one could tackle this question. One method for determining an optimal kernel for the expected similarity maximization problem consists of solving a problem similar to that of learning kernels in classification or regression. Let $X_1, \ldots, X_m$ be m lattices with $\mathrm{Ref}(X_1), \ldots, \mathrm{Ref}(X_m)$ the associated references and let $\widehat{x}(K, X_i)$ be the solution of the expected similarity maximization for lattice X_i when using kernel K. Then, the kernel learning optimization problem can be formulated as follows:

$$\min_{K \in \mathcal{K}} \frac{1}{m} \sum_{i=1}^{m} L(\widehat{x}(K, X_i), \mathrm{Ref}(X_i))$$
$$\text{s. t. } K = T \circ T^{-1} \wedge \mathrm{Tr}[K] \leq C,$$

where $\mathcal{K}$ is a convex family of rational kernels and $\mathrm{Tr}[K]$ denotes the trace of the kernel matrix. In particular, we could choose $\mathcal{K}$ as a family of linear combinations of base rational kernels. Techniques and ideas similar to those discussed by Cortes et al. (2008) for learning sequence kernels could be directly relevant to this problem.

A Appendix

We describe here the approximation of the K_{LB} similarity measure from Kumar et al. (2009). We assume in this section that the lattice X is deterministic in order to simplify the notations. The posterior probability of n-gram w in the lattice X can be formulated as:

$$p(w|X) = \sum_{x \in \Sigma^*} 1_x(w) P(x|s) = \sum_{x \in \Sigma^*} 1_x(w) X(x) \tag{16}$$

where s denotes the source sentence. When using the similarity measure K_{LB} defined Equation (10), Equation (3) can then be reformulated as:

$$\widehat{x} = \operatorname*{argmax}_{x' \in H} \theta_0 |x'| + \sum_{w} \theta_{|w|} c_{x'}(w) p(w|X). \tag{17}$$

The key idea behind this new approximation algorithm is to rewrite the n-gram posterior probability (Equation 16) as follows:

$$p(w|X) = \sum_{x \in \Sigma^*} \sum_{e \in E_X} f(e, w, \pi_x) X(x) \tag{18}$$

where E_X is the set of transitions of X, π_x is the unique accepting path labeled by x in X and $f(e, w, \pi)$ is a score assigned to transition e on path π containing n-gram w:

$$f(e, w, \pi) = \begin{cases} 1 & \text{if } w \in e, p(e|X) > p(e'|X), \\ & \text{and } e' \text{ precedes } e \text{ on } \pi \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

In other words, for each path π, we count the transition that contributes n-gram w and has the highest transition posterior probability relative to its predecessors on the path π; there is exactly one such transition on each lattice path π.

We note that $f(e, w, \pi)$ relies on the full path π which means that it cannot be computed based on local statistics. We therefore approximate the quantity $f(e, w, \pi)$ with $f^*(e, w, X)$ that counts the transition e with n-gram w that has the highest arc posterior probability relative to predecessors in the entire lattice X. $f^*(e, w, X)$ can be computed locally, and the n-gram posterior probability based on f^* can be determined as follows:

$$\begin{aligned} p(w|\mathcal{G}) &= \sum_{x \in \Sigma^*} \sum_{e \in E_X} f^*(e, w, X) X(x) \\ &= \sum_{e \in E_x} 1_{w \in e} f^*(e, w, X) \sum_{x \in \Sigma^*} 1_{\pi_x}(e) X(x) \\ &= \sum_{e \in E_X} 1_{w \in e} f^*(e, w, X) P(e|X), \end{aligned} \tag{20}$$

where $P(e|X)$ is the posterior probability of a lattice transition $e \in E_X$. The algorithm to perform Lattice MBR is given in Algorithm 1. For each state t in the lattice, we maintain a quantity Score(w, t) for each n-gram w that lies on a path from the initial state to t. Score(w, t) is the highest posterior probability among all transitions on the paths that terminate on t and contain n-gram w. The forward pass requires computing the n-grams introduced by each transition; to do this, we propagate n-grams (up to maximum order -1) terminating on each state.

Algorithm 1 MBR Decoding on Lattices

```
 1: Sort the lattice states topologically.
 2: Compute backward probabilities of each state.
 3: Compute posterior prob. of each $n$-gram:
 4: for each transition $e$ do
 5:     Compute transition posterior probability $P(e|X)$.
 6:     Compute $n$-gram posterior probs. $P(w|X)$:
 7:     for each $n$-gram $w$ introduced by $e$ do
 8:         Propagate $n-1$ gram suffix to $h_e$.
 9:         if $p(e|X) > \text{Score}(w, T(e))$ then
10:             Update posterior probs. and scores:
                $p(w|X) \mathrel{+}= p(e|X) - \text{Score}(w, T(e))$.
                $\text{Score}(w, h_e) = p(e|X)$.
11:         else
12:             $\text{Score}(w, h_e) = \text{Score}(w, T(e))$.
13:         end if
14:     end for
15: end for
16: Assign scores to transitions (given by Equation 17).
17: Find best path in the lattice (Equation 17).
```

References

Alfred V. Aho and Margaret J. Corasick. 1975. Efficient String Matching: An Aid to Bibliographic Search. *Communications of the ACM*, 18(6):333–340.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: a general and efficient weighted finite-state transducer library. In *CIAA 2007*, volume 4783 of *LNCS*, pages 11–23. Springer. http://www.openfst.org.

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

Peter J. Bickel and Kjell A. Doksum. 2001. *Mathematical Statistics, vol. I.* Prentice Hall.

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

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. 2008. Learning sequence kernels. In *Proceedings of MLSP 2008*, October.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *Proceedings of ACL and IJCNLP*, pages 567–575.

Vaibhava Goel and William J. Byrne. 2000. Minimum Bayes-risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3).

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *EMNLP*, Barcelona, Spain.

Shankar Kumar and William J. Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *HLT-NAACL*, Boston, MA, USA.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Association for Computational Linguistics and IJCNLP*.

Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. 2002. The Spectrum Kernel: A String Kernel for SVM Protein Classification. In *Pacific Symposium on Biocomputing*, pages 566–575.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proceedings of ACL and IJCNLP*, pages 593–601.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watskins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–44.

Mehryar Mohri and Richard Sproat. 1996. An Efficient Compiler for Weighted Rewrite Rules. In *Proceedings of ACL '96*, Santa Cruz, California.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 6, pages 213–254. Springer.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical mchine translation. *Computational Linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176 (W0109-022), IBM Research Division.

Arto Salomaa and Matti Soittola. 1978. *Automata-Theoretic Aspects of Formal Power Series*. Springer.

Roy W. Tromble, Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 620–629.