Efficient Path Counting Transducers for Minimum Bayes-Risk Decoding of Statistical Machine Translation Lattices

Graeme Blackwood, Adrià de Gispert, William Byrne
Machine Intelligence Laboratory
Cambridge University Engineering Department
Trumpington Street, CB2 1PZ, U.K.
{gwb24|ad465|wjb31}@cam.ac.uk

Abstract
This paper presents an efficient implementation of linearised lattice minimum Bayes-risk decoding using weighted finite state transducer (WFST) operations\footnote{We omit an introduction to WFSTs for space reasons. See Mohri et al. (2008) for details of the general purpose WFST operations used in this paper.}. The LMBR decision rule in Tromble et al. (2008) has the form

$$\hat{E} = \argmax_{E' \in \mathcal{E}} \left\{ \theta_0|E'| + \sum_{u \in \mathcal{N}} \theta_u \#_u(E') p(u|\mathcal{E}) \right\}$$

(1)

where \( \mathcal{E} \) is a lattice of translation hypotheses, \( \mathcal{N} \) is the set of all \( n \)-grams in the lattice (typically, \( n = 1 \ldots 4 \)), and the parameters \( \theta \) are constants estimated on held-out data. The quantity \( p(u|\mathcal{E}) \) we refer to as the path posterior probability of the \( n \)-gram \( u \). This particular posterior is defined as

$$p(u|\mathcal{E}) = p(\mathcal{E}_u|\mathcal{E}) = \sum_{E \in \mathcal{E}_u} P(E|F),$$

(2)

where \( \mathcal{E}_u = \{ E \in \mathcal{E} : \#_u(E) > 0 \} \) is the subset of lattice paths containing the \( n \)-gram \( u \) at least once. It is the efficient computation of these path posterior \( n \)-gram probabilities that is the primary focus of this paper. We will show how general purpose WFST algorithms can be employed to efficiently compute \( p(u|\mathcal{E}) \) for all \( u \in \mathcal{N} \).

Tromble et al. (2008) use Equation (1) as an approximation to the general form of statistical machine translation MBR decoder (Kumar and Byrne, 2004):

$$\hat{E} = \arg \min_{E' \in \mathcal{E}} \sum_{E \in \mathcal{E}} L(E, E') P(E|F)$$

(3)

The approximation replaces the sum over all paths in the lattice by a sum over lattice \( n \)-grams. Even though a lattice may have many \( n \)-grams, it is possible to extract and enumerate them exactly whereas this is often impossible for individual paths. Therefore, while the Tromble et al. (2008) linearisation of the gain function in the decision rule is an approximation, Equation (1) can be computed exactly even over very large lattices. The challenge is to do so efficiently.

If the quantity \( p(u|\mathcal{E}) \) had the form of a conditional expected count

$$c(u|\mathcal{E}) = \sum_{E \in \mathcal{E}} \#_u(E) P(E|F),$$

(4)

it could be computed efficiently using counting transducers (Allauzen et al., 2003). The statistic \( c(u|\mathcal{E}) \) counts the number of times an \( n \)-gram occurs on each path, accumulating the weighted count over all paths. By contrast, what is needed by the approximation in Equation (1) is to identify all paths containing an \( n \)-gram and accumulate their probabilities. The accumulation of probabilities at the path level, rather than the \( n \)-gram level, makes the exact computation of \( p(u|\mathcal{E}) \) hard.

Tromble et al. (2008) address this problem by building a separate word sequence acceptor for each \( n \)-gram in \( \mathcal{N} \) and intersecting this acceptor
with the lattice to discard all paths that do not contain the \( n \)-gram; they then sum the probabilities of all paths in the filtered lattice. We refer to this as the sequential method, since \( p(u|E) \) is calculated separately for each \( u \) in sequence.

Allauzen et al. (2010) introduce a transducer for simultaneous calculation of \( p(u|E) \) for all unigrams \( u \in \mathcal{N}_1 \) in a lattice. This transducer is effective for finding path posterior probabilities of unigrams because there are relatively few unique unigrams in the lattice. As we will show, however, it is less efficient for higher-order \( n \)-grams.

Allauzen et al. (2010) use exact statistics for the unigram path posterior probabilities in Equation (1), but use the conditional expected counts of Equation (4) for higher-order \( n \)-grams. Their hybrid MBR decoder has the form

\[
\hat{E} = \arg\max_{E' \in \mathcal{E}} \left\{ \theta_0 | E'| + \sum_{u \in \mathcal{N} : 1 \leq |u| \leq k} \theta_u \#_u (E') p(u|E) + \sum_{u \in \mathcal{N} : k < |u| \leq 4} \theta_u \#_u (E') c(u|E) \right\},
\]

where \( k \) determines the range of \( n \)-gram orders at which the path posterior probabilities \( p(u|E) \) of Equation (2) and conditional expected counts \( c(u|E) \) of Equation (4) are used to compute the expected gain. For \( k < 4 \), Equation (5) is thus an approximation to the approximation. In many cases it will be perfectly fine, depending on how closely \( p(u|E) \) and \( c(u|E) \) agree for higher-order \( n \)-grams. Experimentally, Allauzen et al. (2010) find this approximation works well at \( k = 1 \) for MBR decoding of statistical machine translation lattices. However, there may be scenarios in which \( p(u|E) \) and \( c(u|E) \) differ so that Equation (5) is no longer useful in place of the original Troumb et al. (2008) approximation.

In the following sections, we present an efficient method for simultaneous calculation of \( p(u|E) \) for \( n \)-grams of a fixed order. While other fast MBR approximations are possible (Kumar et al., 2009), we show how the exact path posterior probabilities can be calculated and applied in the implementation of Equation (1) for efficient MBR decoding over lattices.

2 \( N \)-gram Mapping Transducer

We make use of a trick to count higher-order \( n \)-grams. We build transducer \( \Phi_n \) to map word sequences to \( n \)-gram sequences of order \( n \). \( \Phi_n \) has a similar form to the WFST implementation of an \( n \)-gram language model (Allauzen et al., 2003). \( \Phi_n \) includes for each \( n \)-gram \( u = w_1^n \) arcs of the form:

The \( n \)-gram lattice of order \( n \) is called \( \mathcal{E}_n \) and is found by composing \( \mathcal{E} \circ \Phi_n \), projecting on the output, removing \( \epsilon \)-arcs, determinizing, and minimizing. The construction of \( \mathcal{E}_n \) is fast even for large lattices and is memory efficient. \( \mathcal{E}_n \) itself may have more states than \( \mathcal{E} \) due to the association of distinct \( n \)-gram histories with states. However, the counting transducer for unigrams is simpler than the corresponding counting transducer for higher-order \( n \)-grams. As a result, counting unigrams in \( \mathcal{E}_n \) is easier than counting \( n \)-grams in \( \mathcal{E} \).

3 Efficient Path Counting

Associated with each \( \mathcal{E}_n \), we have a transducer \( \Psi_n \), which can be used to calculate the path posterior probabilities \( p(u|E) \) for all \( u \in \mathcal{N}_n \). In Figures 1 and 2 we give two possible forms\(^2\) of \( \Psi_n \) that can be used to compute path posterior probabilities over \( n \)-grams \( u_{1,2} \in \mathcal{N}_n \) for some \( n \). No modification to the \( \rho \)-arc matching mechanism is required even in counting higher-order \( n \)-grams since all \( n \)-grams are represented as individual symbols after application of the mapping transducer \( \Phi_n \).

Transducer \( \Psi_n^L \) is used by Allauzen et al. (2010) to compute the exact unigram contribution to the conditional expected gain in Equation (5). For example, in counting paths that contain \( u_1 \), \( \Psi_n^L \) retains the first occurrence of \( u_1 \) and maps every other symbol to \( \epsilon \). This ensures that in any path containing a given \( u \), only the first \( u \) is counted, avoiding multiple counting of paths.

We introduce an alternative path counting transducer \( \Psi_n^R \) that effectively deletes all symbols except the last occurrence of \( u \) on any path by ensuring that any paths in composition which count earlier instances of \( u \) do not end in a final state. Multiple counting is avoided by counting only the last occurrence of each symbol \( u \) on a path.

We note that initial \( \epsilon \) arcs in \( \Psi_n^L \) effectively create \( |\mathcal{N}_n| \) copies of \( \mathcal{E}_n \) in composition while searching for the first occurrence of each \( u \). Com-

---

\(^2\)The special composition symbol \( \sigma \) matches any arc; \( \rho \) matches any arc other than those with an explicit transition. See the OpenFst documentation: http://openfst.org
posing with \( \Psi_n^R \) creates a single copy of \( \mathcal{E}_n \) while searching for the last occurrence of \( u \); we find this to be much more efficient for large \( N_n \).

Path posterior probabilities are calculated over each \( \mathcal{E}_n \) by composing with \( \Psi_n \) in the log semiring, projecting on the output, removing \( \epsilon \)-arcs, determinizing, minimising, and pushing weights to the initial state (Allauzen et al., 2010). Using either \( \Psi_n^L \) or \( \Psi_n^R \), the resulting counts acceptor is \( \mathcal{X}_n \).

It has a compact form with one arc from the start state for each \( u_i \in N_n \):

\[
\begin{array}{c}
0 \\
\end{array} \begin{array}{c}
\xrightarrow{\text{u}_1:u_1} \epsilon \frac{1}{\log p(u_1|\mathcal{E})} \xrightarrow{\sigma} 3 \\
\xrightarrow{\text{u}_2:u_2} \epsilon \frac{1}{\log p(u_2|\mathcal{E})} \xrightarrow{\alpha} 4
\end{array}
\]

Figure 1: Path counting transducer \( \Psi_n^L \) matching first (left-most) occurrence of each \( u \in N_n \).

\[
\begin{array}{c}
0 \\
\xrightarrow{\text{u}_1:u_1} \epsilon \frac{1}{\log p(u_1|\mathcal{E})} \xrightarrow{\sigma} 1 \\
\xrightarrow{\text{u}_2:u_2} \epsilon \frac{1}{\log p(u_2|\mathcal{E})} \xrightarrow{\alpha} 2
\end{array}
\]

Figure 2: Path counting transducer \( \Psi_n^R \) matching last (right-most) occurrence of each \( u \in N_n \).

3.1 Efficient Path Posterior Calculation

Although \( \mathcal{X}_n \) has a convenient and elegant form, it can be difficult to build for large \( N_n \) because the composition \( \mathcal{E}_n \circ \Psi_n \) results in millions of states and arcs. The log semiring \( \epsilon \)-removal and determinization required to sum the probabilities of paths labelled with each \( u \) can be slow.

However, if we use the proposed \( \Psi_n^R \), then each path in \( \mathcal{E}_n \circ \Psi_n^R \) has only one non-\( \epsilon \) output label \( u \) and all paths leading to a given final state share the same \( u \). A modified forward algorithm can be used to calculate \( p(u|\mathcal{E}) \) without the costly \( \epsilon \)-removal and determinization. The modification simply requires keeping track of which symbol \( u \) is encountered along each path to a final state.

More than one final state may gather probabilities for the same \( u \); to compute \( p(u|\mathcal{E}) \) these probabilities are added. The forward algorithm requires that \( \mathcal{E}_n \circ \Psi_n^R \) be topologically sorted; although sorting can be slow, it is still quicker than log semiring \( \epsilon \)-removal and determinization.

The statistics gathered by the forward algorithm could also be gathered under the expectation semiring (Eisner, 2002) with suitably defined features. We take the view that the full complexity of that approach is not needed here, since only one symbol is introduced per path and per exit state.

Unlike \( \mathcal{E}_n \circ \Psi_n^R \), the composition \( \mathcal{E}_n \circ \Psi_n^L \) does not segregate paths by \( u \) such that there is a direct association between final states and symbols. The forward algorithm does not readily yield the per-symbol probabilities, although an arc weight vector indexed by symbols could be used to correctly aggregate the required statistics (Riley et al., 2009). For large \( N_n \) this would be memory intensive. The association between final states and symbols could also be found by label pushing, but we find this slow for large \( \mathcal{E}_n \circ \Psi_n \).

4 Efficient Decoder Implementation

In contrast to Equation (5), we use the exact values of \( p(u|\mathcal{E}) \) for all \( u \in N_n \) at orders \( n = 1 \ldots 4 \) to compute

\[
\hat{E} = \arg\min_{E \in \mathcal{E}} \left\{ \theta_0 |E| + \sum_{n=1}^4 g_n(E, E') \right\}, \quad (6)
\]

where \( g_n(E, E') = \sum_{u \in \mathcal{N}_n} \theta_n \#u(E') p(u|\mathcal{E}) \) using the exact path posterior probabilities at each order. We make acceptors \( \Omega_n \) such that \( \mathcal{E} \circ \Omega_n \) assigns order \( n \) partial gain \( g_n(E, E') \) to all paths \( E \in \mathcal{E} \). \( \Omega_n \) is derived from \( \Phi_n \) directly by assigning arc weight \( \theta_n \times p(u|\mathcal{E}) \) to arcs with output label \( u \) and then projecting on the input labels. For each \( n \)-gram \( u = u^n \) in \( N_n \) arcs of \( \Omega_n \) have the form:

\[
\begin{array}{c}
\overrightarrow{w_1} \xrightarrow{\text{w}_n/\theta_0 \times p(u|\mathcal{E})} \overrightarrow{w_2}
\end{array}
\]

To apply \( \theta_0 \) we make a copy of \( \mathcal{E} \), called \( \mathcal{E}_0 \), with fixed weight \( \theta_0 \) on all arcs. The decoder is formed as the composition \( \mathcal{E}_0 \circ \Omega_1 \circ \Omega_2 \circ \Omega_3 \circ \Omega_4 \) and \( \hat{E} \) is extracted as the maximum cost string.

5 Lattice Generation for LMBR

Lattice MBR decoding performance and efficiency is evaluated in the context of the NIST
Table 1: BLEU scores for Arabic→English maximum likelihood translation (ML), MBR decoding using the hybrid decision rule of Equation (5) at $0 \leq k \leq 3$, and regular linearised lattice MBR (LMBR).

<table>
<thead>
<tr>
<th></th>
<th>mt0205tune</th>
<th>mt0205test</th>
<th>mt08nw</th>
<th>mt08ng</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>54.2</td>
<td>53.8</td>
<td>51.4</td>
<td>36.3</td>
</tr>
<tr>
<td>$k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>52.6</td>
<td>52.3</td>
<td>49.8</td>
<td>34.5</td>
</tr>
<tr>
<td>1</td>
<td>54.8</td>
<td>54.4</td>
<td>52.2</td>
<td>36.6</td>
</tr>
<tr>
<td>2</td>
<td>54.9</td>
<td>54.5</td>
<td>52.4</td>
<td>36.8</td>
</tr>
<tr>
<td>3</td>
<td>54.9</td>
<td>54.5</td>
<td>52.4</td>
<td>36.8</td>
</tr>
<tr>
<td>LMBR</td>
<td>55.0</td>
<td>54.6</td>
<td>52.4</td>
<td>36.8</td>
</tr>
</tbody>
</table>

Table 2: Time in seconds required for path posterior $n$-gram probability calculation and LMBR decoding using sequential method and left-most ($\Psi^L_n$) or right-most ($\Psi^R_n$) counting transducer implementations.

<table>
<thead>
<tr>
<th></th>
<th>mt0205tune</th>
<th>mt0205test</th>
<th>mt08nw</th>
<th>mt08ng</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postiors</td>
<td>sequential</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Psi^L_n$</td>
<td>3160</td>
<td>3306</td>
<td>2090</td>
<td>3791</td>
</tr>
<tr>
<td>$\Psi^R_n$</td>
<td>6880</td>
<td>7387</td>
<td>4201</td>
<td>8796</td>
</tr>
<tr>
<td>Decoding</td>
<td>sequential</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Psi_n$</td>
<td>4340</td>
<td>4530</td>
<td>2225</td>
<td>4104</td>
</tr>
<tr>
<td>Total</td>
<td>sequential</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Psi^L_n$</td>
<td>7711</td>
<td>8065</td>
<td>4437</td>
<td>8085</td>
</tr>
<tr>
<td>$\Psi^R_n$</td>
<td>7458</td>
<td>8075</td>
<td>4495</td>
<td>9199</td>
</tr>
<tr>
<td></td>
<td>2321</td>
<td>2348</td>
<td>1468</td>
<td>3149</td>
</tr>
</tbody>
</table>

Arabic→English machine translation task. The development set mt0205tune is formed from the odd numbered sentences of the NIST MT02–MT05 testsets; the even numbered sentences form the validation set mt0205test. Performance on NIST MT08 newswire (mt08nw) and newsgroup (mt08ng) data is also reported.

First-pass translation is performed using HiFST (Iglesias et al., 2009), a hierarchical phrase-based decoder. Word alignments are generated using MTTK (Deng and Byrne, 2008) over 150M words of parallel text for the constrained NIST MT08 Arabic→English track. In decoding, a Shallow-1 grammar with a single level of rule nesting is used and no pruning is performed in generating first-pass lattices (Iglesias et al., 2009).

The first-pass language model is a modified Kneser-Ney (Kneser and Ney, 1995) 4-gram estimated over the English parallel text and an 881M word subset of the GigaWord Third Edition (Graff et al., 2007). Prior to LMBR, the lattices are rescored with large stupid-backoff 5-gram language models (Brants et al., 2007) estimated over more than 6 billion words of English text.

The $n$-gram factors $\theta_0, \ldots, \theta_4$ are set according to Tromble et al. (2008) using unigram precision $p = 0.85$ and average recall ratio $r = 0.74$. Our translation decoder and MBR procedures are implemented using OpenFst (Allauzen et al., 2007).

6 LMBR Speed and Performance

Lattice MBR decoding performance is shown in Table 1. Compared to the maximum likelihood translation hypotheses (row ML), LMBR gives gains of +0.8 to +1.0 BLEU for newswire data and +0.5 BLEU for newsgroup data (row LMBR).

The other rows of Table 1 show the performance of LMBR decoding using the hybrid decision rule of Equation (5) for $0 \leq k \leq 3$. When the conditional expected counts $c(u|\mathcal{E})$ are used at all orders (i.e., $k = 0$), the hybrid decoder BLEU scores are considerably lower than even the ML scores. This poor performance is because there are many unigrams $u$ for which $c(u|\mathcal{E})$ is much greater than $p(u|\mathcal{E})$. The consensus translation maximising the conditional expected gain is then dominated by unigram matches, significantly degrading LMBR decoding performance. Table 1 shows that for these lattices the hybrid decision rule is an accurate approximation to Equation (1) only when $k \geq 2$ and the exact contribution to the gain function is computed using the path posterior probabilities at orders $n = 1$ and $n = 2$. 

3http://www.itl.nist.gov/iad/mig/tests/mt
We now analyse the efficiency of lattice MBR decoding using the exact path posterior probabilities of Equation (2) at all orders. We note that the sequential method and both simultaneous implementations using path counting transducers $\Psi_n^L$ and $\Psi_n^R$ yield the same hypotheses (allowing for numerical accuracy); they differ only in speed and memory usage.

**Posteriors Efficiency** Computation times for the steps in LMBR are given in Table 2. In calculating path posterior $n$-gram probabilities $p(u|\mathcal{E})$, we find that the use of $\Psi_n^L$ is more than twice as slow as the sequential method. This is due to the difficulty of counting higher-order $n$-grams in large lattices. $\Psi_n^L$ is effective for counting unigrams, however, since there are far fewer of them. Using $\Psi_n^R$ is almost twice as fast as the sequential method. This speed difference is due to the simple forward algorithm. We also observe that for higher-order $n$, the composition $\mathcal{E}_n \circ \Psi_n^R$ requires less memory and produces a smaller machine than $\mathcal{E}_n \circ \Psi_n^L$. It is easier to count paths by the final occurrence of a symbol than by the first.

**Decoding Efficiency** Decoding times are significantly faster using $\Omega_n$ than the sequential method: average decoding time is around 0.1 seconds per sentence. The total time required for lattice MBR is dominated by the calculation of the path posterior $n$-gram probabilities, and this is a function of the number of $n$-grams in the lattice $|\mathcal{N}|$. For each sentence in mt0205tune, Figure 3 plots the total LMBR time for the sequential method (marked ‘o’) and for probabilities computed using $\Psi_n^R$ (marked ‘+’). This compares the two techniques on a sentence-by-sentence basis. As $|\mathcal{N}|$ grows, the simultaneous path counting transducer is found to be much more efficient.

Figure 3: Total time in seconds versus $|\mathcal{N}|$.

**7 Conclusion**

We have described an efficient and exact implementation of the linear approximation to LMBR using general WFST operations. A simple transducer was used to map words to sequences of $n$-grams in order to simplify the extraction of higher-order statistics. We presented a counting transducer $\Psi_n^R$ that extracts the statistics required for all $n$-grams of order $n$ in a single composition and allows path posterior probabilities to be computed efficiently using a modified forward procedure.

We take the view that even approximate search criteria should be implemented exactly where possible, so that it is clear exactly what the system is doing. For machine translation lattices, conflating the values of $p(u|\mathcal{E})$ and $c(u|\mathcal{E})$ for higher-order $n$-grams might not be a serious problem, but in other scenarios – especially where symbol sequences are repeated multiple times on the same path – it may be a poor approximation.

We note that since much of the time in calculation is spent dealing with $\epsilon$-arcs that are ultimately removed, an optimised composition algorithm that skips over such redundant structure may lead to further improvements in time efficiency.

**Acknowledgments**

This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

**References**


Cyril Allauzen, Shankar Kumar, Wolfgang Macherey, Mehryar Mohri, and Michael Riley. 2010. Expected
sequence similarity maximization. In Human Language Technologies 2010: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, California, June.


