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 transducers. We introduce transducers to efficiently count lattice paths containing n-grams and use these to gather the required statistics. We show that these procedures can be implemented exactly through simple transformations of word sequences to sequences of n-grams. This yields a novel implementation of lattice minimum Bayes-risk decoding which is fast and exact even for very large lattices.

1 Introduction

This paper focuses on an exact implementation of the linearised form of lattice minimum Bayes-risk (LMBR) decoding using general purpose weighted finite state transducer (WFST) operations[1]. The LMBR decision rule in Tromble et al. (2008) has the form

$$\hat{E} = \operatorname*{argmax}_{E' \in \mathcal{E}} \left\{ \theta_0 |E'| + \sum_{u \in \mathcal{N}} \theta_u \#_u(E') p(u|\mathcal{E}) \right\} \tag{1}$$

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 θ 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

$$p(u|\mathcal{E}) = p(\mathcal{E}_u|\mathcal{E}) = \sum_{E \in \mathcal{E}_u} P(E|F), \tag{2}$$

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):

$$\hat{E} = \operatorname*{argmin}_{E' \in \mathcal{E}} \sum_{E \in \mathcal{E}} L(E, E') P(E|F) \tag{3}$$

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

$$c(u|\mathcal{E}) = \sum_{E \in \mathcal{E}} \#_u(E) P(E|F), \tag{4}$$

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) approach this problem by building a separate word sequence acceptor for each n-gram in $\mathcal{N}$ and intersecting this acceptor

[1]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.

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 $p(u|\mathcal{E})$ is calculated separately for each u in sequence.

Allauzen et al. (2010) introduce a transducer for simultaneous calculation of $p(u|\mathcal{E})$ 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

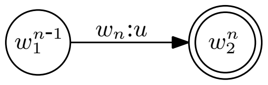
$$\begin{aligned}\hat{E} &= \underset{E' \in \mathcal{E}}{\operatorname{argmax}} \Big\{ \theta_0 |E'| \\ &+ \sum_{u \in \mathcal{N}: 1 \leq |u| \leq k} \theta_u \#_u(E') p(u|\mathcal{E}) \\ &+ \sum_{u \in \mathcal{N}: k < |u| \leq 4} \theta_u \#_u(E') c(u|\mathcal{E}) \Big\}, \quad (5)\end{aligned}$$

where k determines the range of n-gram orders at which the path posterior probabilities $p(u|\mathcal{E})$ of Equation (2) and conditional expected counts $c(u|\mathcal{E})$ 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 $p(u|\mathcal{E})$ and $c(u|\mathcal{E})$ 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 $p(u|\mathcal{E})$ and $c(u|\mathcal{E})$ differ so that Equation (5) is no longer useful in place of the original Tromble et al. (2008) approximation.

In the following sections, we present an efficient method for simultaneous calculation of $p(u|\mathcal{E})$ 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 Φ_n to map word sequences to n-gram sequences of order n. Φ_n has a similar form to the WFST implementation of an n-gram language model (Allauzen et al., 2003). Φ_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 ϵ-arcs, determinizing, and minimising. 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 Ψ_n which can be used to calculate the path posterior probabilities $p(u|\mathcal{E})$ for all $u \in \mathcal{N}_n$. In Figures 1 and 2 we give two possible forms[2] of Ψ_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 ρ-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 Φ_n.

Transducer Ψ_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, Ψ_n^L retains the *first* occurrence of u_1 and maps every other symbol to ϵ. 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 Ψ_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 $\epsilon{:}\epsilon$ arcs in Ψ_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 σ matches any arc; ρ matches any arc other than those with an explicit transition. See the OpenFst documentation: http://openfst.org

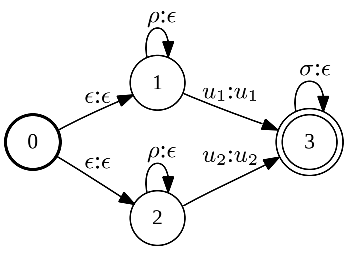


Figure 1: Path counting transducer Ψ_n^L matching first (left-most) occurrence of each $u \in \mathcal{N}_n$.

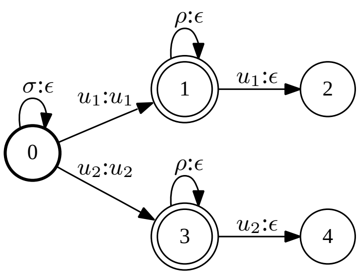
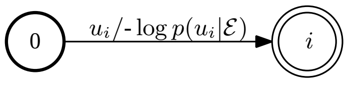


Figure 2: Path counting transducer Ψ_n^R matching last (right-most) occurrence of each $u \in \mathcal{N}_n$.

posing with Ψ_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 $\mathcal{N}_n$.

Path posterior probabilities are calculated over each $\mathcal{E}_n$ by composing with Ψ_n in the log semiring, projecting on the output, removing ϵ-arcs, determinizing, minimising, and pushing weights to the initial state (Allauzen et al., 2010). Using either Ψ_n^L or Ψ_n^R, the resulting counts acceptor is $\mathcal{X}_n$. It has a compact form with one arc from the start state for each $u_i \in \mathcal{N}_n$:



3.1 Efficient Path Posterior Calculation

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

However, if we use the proposed Ψ_n^R, then each path in $\mathcal{E}_n \circ \Psi_n^R$ has only one non-ϵ 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 ϵ-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 ϵ-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.

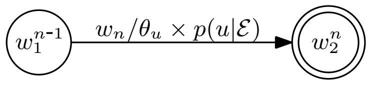
Unlike $\mathcal{E}_n \circ \Psi_n^R$, the composition $\mathcal{E}_n \circ \Psi_n^L$ does not segregate paths by u such that there is a direct association between final states and symbols. The forward algorithm does not readily yield the per-symbol probabilities, although an arc weight vector indexed by symbols could be used to correctly aggregate the required statistics (Riley et al., 2009). For large $\mathcal{N}_n$ 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 $u \in \mathcal{N}_n$ at orders $n = 1 \dots 4$ to compute

$$\hat{E} = \operatorname*{argmin}_{E' \in \mathcal{E}} \left\{ \theta_0 |E'| + \sum_{n=1}^{4} g_n(E, E') \right\}, \quad (6)$$

where $g_n(E, E') = \sum_{u \in \mathcal{N}_n} \theta_u \#_u(E') p(u|\mathcal{E})$ using the exact path posterior probabilities at each order. We make acceptors Ω_n such that $\mathcal{E} \circ \Omega_n$ assigns order n partial gain $g_n(E, E')$ to all paths $E \in \mathcal{E}$. Ω_n is derived from Φ_n directly by assigning arc weight $\theta_u \times p(u|\mathcal{E})$ to arcs with output label u and then projecting on the input labels. For each n-gram $u = w_1^n$ in $\mathcal{N}_n$ arcs of Ω_n have the form:



To apply θ_0 we make a copy of $\mathcal{E}$, called $\mathcal{E}_0$, with fixed weight θ_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

		mt0205tune	mt0205test	mt08nw	mt08ng
ML		54.2	53.8	51.4	36.3
k	0	52.6	52.3	49.8	34.5
	1	54.8	54.4	52.2	36.6
	2	54.9	54.5	52.4	36.8
	3	54.9	54.5	52.4	36.8
LMBR		55.0	54.6	52.4	36.8

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).

		mt0205tune	mt0205test	mt08nw	mt08ng
Posteriors	sequential	3160	3306	2090	3791
	Ψ_n^L	6880	7387	4201	8796
	Ψ_n^R	1746	1789	1182	2787
Decoding	sequential	4340	4530	2225	4104
	Ψ_n	284	319	118	197
Total	sequential	7711	8065	4437	8085
	Ψ_n^L	7458	8075	4495	9199
	Ψ_n^R	2321	2348	1468	3149

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

Arabic→English machine translation task[3]. 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).

[3]http://www.itl.nist.gov/iad/mig/tests/mt

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 $n = 2$.

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 Ψ_n^L and Ψ_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 Ψ_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. Ψ_n^L is effective for counting unigrams, however, since there are far fewer of them. Using Ψ_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 Ω_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 Ψ_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.

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 Ψ_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

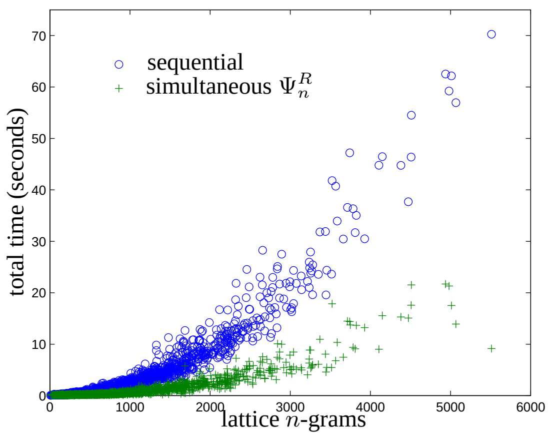


Figure 3: Total time in seconds versus $|\mathcal{N}|$.

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 ϵ-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, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 557–564.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of the 9th International Conference on Implementation and Application of Automata*, pages 11–23. Springer.

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.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867.

Yonggang Deng and William Byrne. 2008. HMM word and phrase alignment for statistical machine translation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):494–507.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8, Philadelphia, July.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2007. English Gigaword Third Edition.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 433–441, Boulder, Colorado, June. Association for Computational Linguistics.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing*, pages 181–184.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of Human Language Technologies: The 2004 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 169–176.

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 Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171, Suntec, Singapore, August. Association for Computational Linguistics.

M. Mohri, F.C.N. Pereira, and M. Riley. 2008. Speech recognition with weighted finite-state transducers. *Handbook on Speech Processing and Speech Communication*.

Michael Riley, Cyril Allauzen, and Martin Jansche. 2009. OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 9–10, Boulder, Colorado, May. Association for Computational Linguistics.

Roy Tromble, Shankar Kumar, Franz 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, Honolulu, Hawaii, October. Association for Computational Linguistics.