Transducer Disambiguation with Sparse Topological Features

Gonzalo Iglesias[†] **Adrià de Gispert**[†‡] **Bill Byrne**[†‡]

[†]SDL Research, Cambridge, U.K.
{giglesias|agispert|bbyrne}@sdl.com

[‡]Department of Engineering, University of Cambridge, U.K.

Abstract

We describe a simple and efficient algorithm to disambiguate non-functional weighted finite state transducers (WFSTs), i.e. to generate a new WFST that contains a unique, best-scoring path for each hypothesis in the input labels along with the best output labels. The algorithm uses topological features combined with a tropical sparse tuple vector semiring. We empirically show that our algorithm is more efficient than previous work in a PoS-tagging disambiguation task. We use our method to rescore very large translation lattices with a bilingual neural network language model, obtaining gains in line with the literature.

1 Introduction

Weighted finite-state transducers (WFSTs), or *lattices*, are used in speech and language processing to compactly represent and manipulate a large number of strings. Applying a finite-state operation (eg. PoS tagging) to a lattice via composition produces a WFST that maps input (eg. words) onto output strings (eg. PoS tags) and preserves the arc-level alignment between each input and output symbol (eg. each arc is labeled with a word-tag pair and has a weight). Typically, the result of such operation is a WFST that is *ambiguous* because it contains multiple paths with the same input string, and *non-functional* because it contains multiple output strings for a given input string (Mohri, 2009).

Disambiguating such WFSTs is the task of creating a WFST that encodes only the best-scoring path of each input string, while still maintaining the arc-level mapping between input and output symbols. This is a non-trivial task[1], and so far only one algorithm has been described (Shafran et al., 2011); the main steps are:

(a) Map the WFST into an equivalent weighted finite-state automata (WFSA) using weights that contain *both* the WFST weight *and* output symbols (using a special semiring)

(b) Apply WFSA determinization under this semiring to ensure that only one unique path per input string survives

(c) Expand the result back to an WFST that preserves arc-level alignments

We present a new disambiguation algorithm that can efficiently accomplish this. In Section 2 we describe how the *tropical sparse tuple vector* semiring can keep track of individual arcs in the original WFST as topological features during the mapping step (**a**). This allows us to describe in Section 3 an efficient expansion algorithm for step (**c**). We show in Section 4 empirical evidence that our algorithm is more efficient than Shafran et al. (2011) in their same PoS-tagging task. We also show how our method can be applied in rescoring translation lattices under a bilingual neural-network model (Devlin et al., 2014), obtaining BLEU score gains consistent with the literature. Section 5 reviews related work and concludes.

2 Semiring Definitions

A WFST $T = (\Sigma, \Delta, Q, I, F, E, \rho)$ over a semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ has input and output alphabets Σ and Δ, a set of states Q, the initial state $I \in Q$, a set of final states $F \subset Q$, a set of transitions (edges) $E \subset (Q \times \Sigma \times \Delta \times \mathbb{K} \times Q)$, and a final state function $\rho : F \to \mathbb{K}$. We focus on extensions to the tropical semiring $(\mathbb{R} \pm \infty, \min, +, \infty, 0)$.

[1]Unless one enumerates all the possible input strings in the lattice, searches for the best output string for each input string, and converts the resulting sequences back into a WFST, which is clearly inefficient.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2275–2280,
Lisbon, Portugal, 17-21 September 2015.

Let $e = (p[e], i[e], o[e], w[e], n[e])$ be an edge in E. A path $\pi = e_1 ... e_n$ is a sequence of edges such that $n[e_j] = p[e_{j+1}], 1 \leq j < n$. $w[\pi] = \bigotimes_{e_j \in \pi} w[e_j]$; $p[\pi] = p[e_1], n[\pi] = n[e_n]$. A path is accepting if $p[\pi] = I$ and $n[\pi] \in F$. The weight associated by T to a set of paths Π is $T(\Pi) = \bigoplus_{\pi \in \Pi} w[\pi] \otimes \rho(n[\pi])$.

2.1 Tropical Sparse Vector Semiring

Let $\bar{f}[e_i] = f_i \in \mathbb{R}^N$ be the unweighted feature vector associated with edge e_i, and let $\bar{\alpha} \in \mathbb{R}^N$ be a global feature weight vector. The tropical weight is then found as $w_i = w[e_i] = \bar{\alpha} \cdot \bar{f}_i = \sum_k \alpha_k f_{i,k}$.

Given a fixed $\bar{\alpha}$, we define the operators for the *tropical vector* semiring as: $\bar{f}_i \oplus_\alpha \bar{f}_j = \min(\bar{\alpha} \cdot \bar{f}_i, \bar{\alpha} \cdot \bar{f}_j)$ and $\bar{f}_i \otimes_\alpha \bar{f}_j = \sum_k \alpha_k (f_{i,k} + f_{j,k})$. The tropical weights are maintained correctly by the vector semiring as $\bar{f}_i \oplus_\alpha \bar{f}_j = w_i \oplus w_j$ and $\bar{f}_i \otimes_\alpha \bar{f}_j = w_i \otimes w_j$. Finally, we define the element-wise times operator as: $\bar{f}_i * \bar{f}_j = \bar{f}_m$, where $f_{m,k} = f_{i,k} + f_{j,k}, \forall k$. It follows that $w_i \otimes w_j = \alpha \cdot (\bar{f}_i * \bar{f}_j)$.

When dealing with high-dimensional feature vectors which have few non-zero elements, it is convenient in practice (for computational efficiency) to use a sparse representation for vectors: $\bar{f} = [(i, f_i), i : f_i \neq 0]$. That is, $\bar{f}$ is comprised of a sparse set of tuples (i, f_i), where i is a feature index and f_i is its value; e.g. $[(2, f_2)]$ is short for $[0, f_2, 0, 0]$ if $N = 4$.

The semiring that operates on sparse feature vectors, which we call *tropical sparse tuple vector* semiring[2], uses conceptually identical operators as the non-sparse version defined above, so it also maintains the tropical weights w correctly.

3 A Disambiguation Algorithm

We now describe how we use the semiring described in Section 2 for steps (**a**) and (**b**), and describe an expansion algorithm for step (**c**) that efficiently converts the output of determinization into an unambiguous WFST with arc-level alignments.

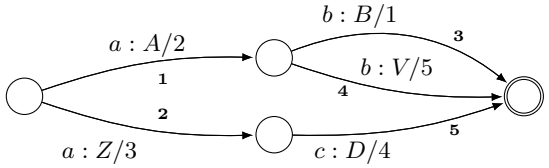
WFSA with Sparse Topological Features

Let T be a tropical-weight WFST with K edges. T is topologically sorted so that if an edge e_k preceeds an edge $e_{k'}$ on a path, then $k < k'$. We now use tropical sparse tuple vector weights to create a WFSA A that maintains (in its weights) pointers to specific edges in T. These 'pointers' are sparse topological features.

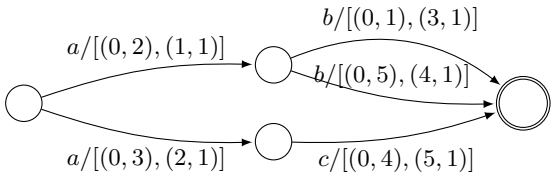
[2]We implement this semiring as an extension to the sparse tuple weights of the OpenFst library (Allauzen et al., 2007).

For each edge $e_k = (p_k, i_k, o_k, w_k, n_k)$ of T, we create an edge $e'_k = (p_k, i_k, i_k, \bar{f}_k, n_k)$ in A, where $\bar{f}_k = [w_k, 0, \ldots, 0, 1, 0, \ldots, 0]$; the 1 is in the k^{th} position. In other words, $f_{k,0}$ is the tropical weight of the k^{th} edge in T and $f_{k,k} = 1$ indicates that this tropical weight belongs to edge k in T. In sparse notation, $\bar{f}_k = [(0, w_k), (k, 1)]$.

For example, this non-deterministic transducer T:



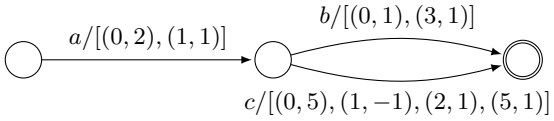
is mapped to acceptor A with topological features:



Given $\alpha = [1, 0, \ldots, 0]$, operations on A yield the same path weights as in the usual tropical semiring.

WFSA determinization

We now apply the standard determinization algorithm to A, which yields A^D:



This now accepts only one best-scoring path for each input string, and the weights 'point' to the sequence of arcs traversed by the relevant path in T. In turn, this reveals the best output string for the given input string. For example, the path-level features associated with '$a\,b$' are $[(0, 3), (1, 1), (3, 1)]$, indicating a path $\pi = e_1 e_3$ with tropical weight 3 through T (and hence output string 'AB').

The topology of A^D is compact because multiple input strings may share arcs while still encoding different output strings in their weights. This is achieved by 'cancelling' topological features in subsequent arcs and 'replacing' them by new ones as one traverses the path. For example, the string '$a\,c$' initially has feature $(1, 1)$, but this gets cancelled later in the path by $(1, -1)$, and replaced by $[(2, 1), (5, 1)]$, indicating a path $\pi = e_2 e_5$ through T with output string 'ZD'.

EXPANDTFEA($A^r = (\Sigma, \Delta, Q, I, F, E)$)

```
1  I' ← (I,0)
2  ENQUEUE(S, I')
3  while |S| do
4    (q, f̄) ← HEAD(S)
5    DEQUEUE(S)
6    if q ∈ F then
7      F' ← F' ∪ {(q, f̄))}
8    for each e ∈ E(q) do
9      (w', t', f̄') ← POPTFEA(f̄, e)
10     q' ← (n[e], f̄')
11     e' ← ((q, f̄), i[e], i[e], [(0, w'), (t', 1)], q')
12     Q' ← Q' ∪ {q'}
13     E' ← E' ∪ {e'}
14     ENQUEUE(S, (n[e], f̄'))
15 return B^r = (Σ, Δ, Q', I', F', E')
```

Figure 1: Expansion and topological feature repositioning algorithm for step (**c**).

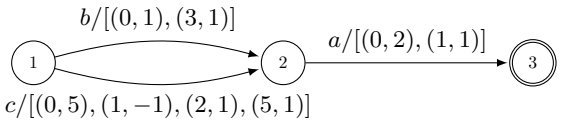
Expansion Algorithm

We now describe an expansion algorithm to convert A^D into an unambiguous WFST T' that maintains the arc-level input-output alignments of the original transducer T. In our example, T' should be identical to T except for edge **4**, which is removed.

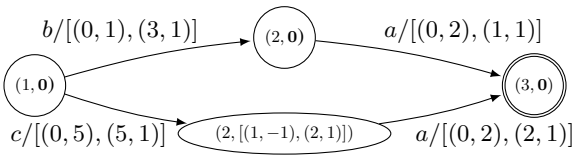
Due to the WFSA determinization algorithm, we observe empirically that the cancelling features in A^D *tend* to appear in a path *after* the feature itself. This allows us to define an algorithm that traverses A^D in reverse (from its final states to its initial state) and creates an equivalent acceptor with the topology of T'.

The algorithm is described in Figure 1. It performs a forward pass through A^r (the reverse of A^D). The intuition is that, for each arc, we create a new arc where we 'pop' the highest topological feature (as it will not be cancelled later) and its tropical weight. The new states encode the original state q and the residual features that have not been 'popped' yet. For each edge $E(q)$, the auxiliary POPTFEA($\bar{f}, e$) returns a $(w', t', \bar{f}')$ tuple, where w' is the tropical weight obtained as $\bar{f} \otimes_\alpha \bar{f}[e]$ (which is equivalent to $f_0 + f[e]_0$ given our $\bar{\alpha}$); t' is the index of the highest topological feature of $\bar{f} * \bar{f}[e]$; and $\bar{f}'$ is the vector of residual topological features after excluding f_0 and $f_{t'}$, that is, $\bar{f} * \bar{f}[e] * [(0, -w'), (t', -1)]$. For example, POPTFEA($[(0,5),(1,-1),(2,1),(6,1)]$) returns $(5, 6, [(1,-1),(2,1)])$; if w has only one topological feature, the residual is $\mathbf{0}$. The residual in all final states of B^r will be $\mathbf{0}$ (no topological features still to be popped).

Graphically, in our running example A^r is:



and the output is B^r:



Reversing B^r yields an acceptor B (still in the sparse tuple vector semiring) which has the same topology as our goal T' and can be trivially mapped to T' in linear time: each arc takes the tropical weight via $\bar{\alpha}$ and has only one topological feature which points to the arc in T containing the required output symbol.

Two-pass Expansion

As mentioned earlier, our algorithm relies on 'cancelling' topological features appearing after the feature they cancel in a given path. In general, consider T a weighted transducer and A it's equivalent automaton with sparse topological features, as described here. A^p is the result of applying standard WFST operations, such as determinization, minimization or shortest path. Assume as a final step that the weights have been pushed towards the final states. It is worth noting the property that: two topological features in a path accepted by A will never get reordered in A^p, although they can appear together on the same edge, as shown in our running example. Indeed, if A^p contains only one single path, all the topological features would appear on the final state.

Let us define a function $d_A(e)$ as the minimum number of edges on any path in A from the start state to $n[e]$ through edge e.

Consider all edges e_i in A and e^p in A^p, with $f[e_i]_i = 1$ and $f[e^p]_i \neq 0$, i.e. we are interested in the topological feature contribution on e^p due to the edge e_i in A. If $d_A(e_i) \leq d_{A^p}(e^p)$ is always satisfied, then EXPANDTFEA will yield the correct answer because the residual at each state, together with the the weight of the current edge, contains all the necessary information to pop the next correct topological feature.

However, many deterministic WFSAs will not exhibit this behaviour (eg. minimised WFSAs), even after pushing the weights towards the final states. For example see this acceptor A:

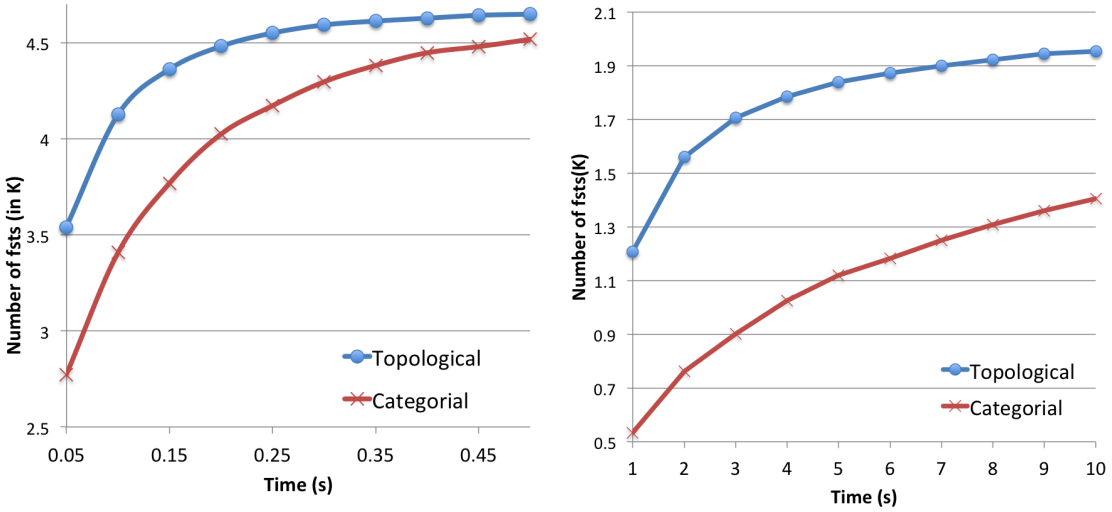
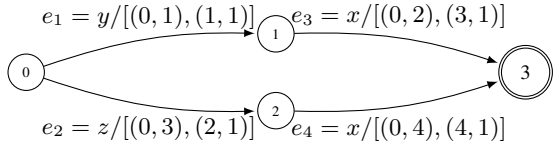
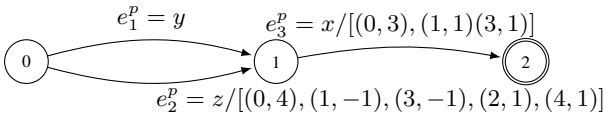


Figure 2: Number of succesfully disambiguated transducers over time for PoS tagged lattices (left) and HiFST lattices (right).



And A^p is a minimised version of A:



As $d_A(e_3) = d_A(e_4) = 2$ and $d_{A^p}(e_2^p) = 1$, the distance test fails for both topological features $(3,-1)(4,1)$. Running B^r=EXPANDTFEA($(A^p)^r$) will not cancel feature $(3,1)$ along the path 'z x' and will pop $(4,1)$ instead, storing the remaining none-$\mathbf{0}$ residual in a final state of B^r.

As mentioned before, two topological features along the same path in A will not reorder in A^p. In this example, as $(4,1)$ appears in edge e_2^p, feature $(2,1)$ must also appear in this edge (or on an earlier edge, in a more complicated machine). In general, any remaining topological features along the path back to the start state of A^p will all be popped *after* their correct edges in B^r. All edges in B^r pass the distance test compared to A^r, the reversed form of A: for all edges e_i with $f[e_i]_i = 1$ in A^r and e^q in B^r such that $f[e^q]_i \neq 0$, $d_{A^r}(e_i) \leq d_{B^r}(e^q)$. Edges in these machines are now reverse sorted, i.e. if an edge e_k precedes an edge $e_{k'}$ on a path, then $k' < k$.

We can perform a second pass with the same algorithm over B, with the only minor modification that t' is now the index of the *lowest* topological feature of $\bar{f} * \bar{f}[e]$. This expands the acceptor correctly. Because correct expansions yield $\mathbf{0}$ residuals on the final states, the algorithm can be trivially modified to trigger the second pass automatically if the residual on any final state is not $\mathbf{0}$.

4 Experiments

We evaluate our algorithm, henceforth called *topological*, in two ways: we empirically contrast disambiguation times against previous work, and then apply it to rescore translation lattices with bilingual neural network models.

4.1 PoS Transducer Disambiguation

We apply our algorithm to the 4,664 NIST English CTS RT Dev04 set PoS tagged lattices used by Sproat el al. (2014); these were generated with a speech recognizer similar to (Soltau et al., 2005) and tagged with a WFST-based HMM tagger. The average number of states is 493. We contrast with the *lexicographic tropical categorial semiring* implementation of Shafran et al. (2011), henceforth referred to as the *categorial* method.

Figure 2 (left) shows the number of disambiguated WFSTs as processing time increases. The topological algorithm proves much faster (and we observe no memory footprint differences). In 50ms it disambiguates 3540 transducers, as opposed to the 2771 completed by the categorial procedure; the slowest WFST to disambiguate takes 230 seconds in the categorial procedure and 60 seconds in our method. Using sparse topological features with our semiring disambiguates all WF-

STs faster in 99.8% of cases.

4.2 Neural Network Bilingual Rescoring

We use the disambiguation algorithm to apply the bilingual neural network language model (BiLM) of Devlin et al. (2014) to the output lattices of the CUED OpenMT12 Arabic-English hierarchical phrase-based translation system[3] using HiFST (de Gispert et al., 2010). We use a development set *mt0205tune* (2075 sentences) and a validation set *mt0205test* (2040 sentences) from the NIST MT02 through MT05 evaluation sets.

The edges in these WFSTs are of the form $t{:}i/w$, where t is the target word, i is the source sentence position t aligns to, and w contains the translation and language model score. HiFST outputs these WFSTs by using a standard hiero grammar (Chiang, 2007) augmented with target side heuristic alignments or *affiliations* to the source (Devlin et al., 2014).

In a rule over source and target words

$$X \rightarrow < s_1\, X\, s_2\, s_3,\, t_1\, X\, t_2 > / 2, 1$$

the feature '$2, 1$' indicates that the target word t_1 is aligned to source word s_2 and that t_2 aligns to s_1. As rules are applied in translation, this information can be used to link target words to absolute positions within the source sentence.

Allowing for admissible pruning, all possible affiliation sequences under the grammar for every translation are available in the WFSTs; disambiguation keeps the best affiliation sequence for each translation hypothesis, which allows the rescoring of very large lattices with the BiLM model.

This disambiguation task involves much bigger lattices than the POS-tagging task: the average number of states of the HiFST lattices is 38,200. Figure 2 (right) shows the number of *mt0205tune* disambiguated WFSTs over time compared to the categorial method. As with the PoS disambiguation task, the topological method is always much faster than the categorial one. After 10 seconds, our method has disambiguated 1953 lattices out of 2075, whereas the categorial method has only finished 1405. The slowest WFST to disambiguate takes 6700 seconds with the categorial procedure, which compares to 1000 seconds in our case.

The BiLM model is trained with NPLM (Vaswani et al., 2013) with a context of 3 source and 4 target words. Lattice rescoring with this model requires a special variation of the standard WFST composition which looks at both input and output labels on a transducer arc; we use KenLM (Heafield, 2011) to retrieve neural network scores for on-the-fly composition. We retune the parameters with Lattice MERT (Macherey et al., 2008) . Results are shown in Table 1. Acknowledging the task differences with respect to (Devlin et al., 2014), we find BLEU gains consistent with rescoring results reported in their Table 5.

[3] See http://www.nist.gov/itl/iad/mig/openmt12results.cfm.

system	mt0205tune	mt0205test
baseline	52.2	51.9
+BiLM	53.0	52.9

Table 1: Translation scores in lower-case BLEU.

5 Conclusions and Related Work

We have described a tagging disambiguation algorithm that supports non-functional WFSTs, which cannot be handled directly by neither WFST determinization (Mohri, 1997) nor WFST disambiguation (Mohri, 2012). We show it is faster than the implementation with a lexicographic-tropical-categorial semiring described by Shafran et al. (2011) and describe a use case in a practical rescoring task of an MT system with bilingual neural networks that yield 1.0 BLEU gain.

Povey et al. (2012) also use a special semiring that allows to map non-functional WFSTs into WFSAs by inserting the tag into a string weight. However, in contrast to our implementation and that of Shafran et al (2011), no expansion into an WFST with aligned input/output is described.

Lexicographic semirings, used for PoS tagging disambiguation (Shafran et al., 2011), have been also shown to be useful in other tasks (Sproat et al., 2014), such as optimized epsilon encoding for backoff language models (Roark et al., 2011), and hierarchical phrase-based decoding with Pushdown Automata (Allauzen et al., 2014).

The tools for disambiguation and WFST composition with bilingual models, along with a tutorial to replicate Section 4.2, are all available at `http://ucam-smt.github.io`.

Acknowledgments

We thank the authors of (Sproat et al., 2014) for generously sharing their PoS-tagging experiments.

References

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 CIAA*.

Cyril Allauzen, Bill Byrne, Adrià de Gispert, Gonzalo Iglesias, and Michael Riley. 2014. Pushdown Automata in Statistical Machine Translation. *Computational Linguistics*, 40(3):687–723.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational Linguistics*, 36(3):505–533.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of ACL*.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of EMNLP*.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation. In *Proceedings of EMNLP*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 213–254. Springer Berlin Heidelberg.

Mehryar Mohri. 2012. A Disambiguation Algorithm for Finite Automata and Functional Transducers. In *Proceedings of CIAA*, volume 7381, pages 265–277.

Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukas Burget, Arnab Ghoshal, Milos Janda, Martin Karafiat, Stefan Kombrink, Petr Motlıcek, Yanmin Qian, Korbinian Riedhammer, Karel Vesely, and Ngoc Thang Vu. 2012. Generating Exact Lattices in the WFST Framework. In *Proceedings of ICASSP*.

Brian Roark, Richard Sproat, and Izhak Shafran. 2011. Lexicographic Semirings for Exact Automata Encoding of Sequence Models. In *Proceedings of ACL-HLT*.

Izhak Shafran, Richard Sproat, Mahsa Yarmohammadi, and Brian Roark. 2011. Efficient Determinization of Tagged Word Lattices using Categorial and Lexicographic Semirings. In *Proceedings of ASRU*.

Hagen Soltau, Brian Kingsbury, Lidia Mangu, Daniel Povey, George Saon, and Geoffrey Zweig. 2005. The IBM 2004 Conversational Telephony System for Rich Transcription. In *Proceedings of ICASSP*.

Richard Sproat, Mahsa Yarmohammadi, Izhak Shafran, and Brian Roark. 2014. Applications of Lexicographic Semirings to Problems in Speech and Language Processing. *Computational Linguistics*, 40(4):733–761.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *Proceedings of EMNLP*.