Efficient Determinization of Tagged Word Lattices using Categorial and Lexicographic Semirings

Izhak Shafran, Richard Sproat, Mahsa Yarmohammadi and Brian Roark
{zak,rws,mahsa,roark}@cslu.ogi.edu

***Abstract*—Speech and language processing systems routinely face the need to apply finite state operations (e.g., POS tagging) on results from intermediate stages (e.g., ASR output) that are naturally represented in a compact lattice form. Currently, such needs are met by converting the lattices into linear sequences (n-best scoring sequences) before and after applying the finite state operations. In this paper, we eliminate the need for this unnecessary conversion by addressing the problem of picking *only* the single-best scoring output labels for *every* input sequence. For this purpose, we define a *categorial* semiring that allows determinzation over strings and incorporate it into a $\langle\text{Tropical}, \text{Categorial}\rangle$ *lexicographic* semiring. Through examples and empirical evaluations we show how determinization in this lexicographic semiring produces the desired output. The proposed solution is general in nature and can be applied to multi-tape weighted transducers that arise in many applications.**

I. Introduction and Motivation

Automatic speech recognition and statistical machine translation systems often employ a cascade of coarse-to-fine stages, each stage refining the output of the previous stage with more complex models. The output of the intermediate stages are represented as n-best scoring sequences, where n is carefully chosen for each stage — a higher value of n reduces the risk of pruning the correct output and a lower value improves the efficiency of the system. The intermediate n-best scoring sequences in most cases have a large number of overlapping subsequences, making it efficient to represent them using lattices or weighted finite state transducers (WFST). Moreover, the WFST representation allows easy manipulation of the intermediate sequences through efficient and modular WFST operations of intersection, shortest path, determinization, and minimization, operations that are now routinely applied to process intermediate sequences.

Among the operations that are routinely needed to process intermediate results, consider a task such as that of estimating the part-of-speech (POS) of words in an ASR lattice or WFST ($\mathcal{W}$) using a WFST-based tagger ($\mathcal{P}$). All possible candidate POS-tags for all the word sequences in the lattice can be easily obtained by composing the two transducers, $\mathcal{W} \circ \mathcal{P}$. However, computing **only** the best scoring analysis for **every** word sequence in $\mathcal{W}$ is non-trivial. Picking the n-best scoring paths of $\mathcal{W} \circ \mathcal{P}$ does not solve the problem since the output may contain multiple analysis of same word sequence and all analyses of certain word sequences may be discarded as in the case of *fine mead* in Figure I. Currently, the solution requires enumerating all the unique word sequences in $\mathcal{W}$, computing the best analysis for each sequence and then converting the resulting sequences back into a WFST. This problem arises not only in POS tagging but also while processing WFSTs with any finite state tagging model such as named entity detectors or morphological analyzers. This problem also surfaces in spoken term detection or in discriminative training, when one needs to identify or extract the most appropriate time boundaries, acoustic model scores, language model scores, pronunciation scores, and pronunciations for each unique word sequence in an ASR lattice.

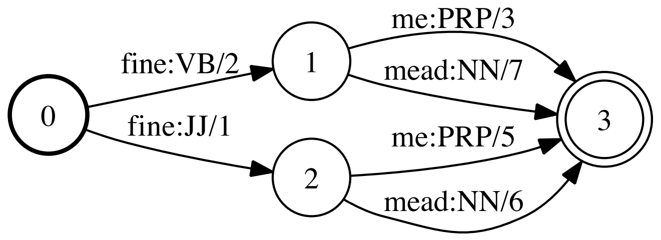

Fig. 1. In this simple illustrative POS-tagged WFST, 2-best scoring path discards all analyses of *fine mead*.

In this paper, we define two new semirings to efficiently solve the above described problem of computing the shortest transduction path for all unique input sequences. In Section II, we briefly describe the relevant terminology and notations from the finite state machine literature necessary for developing our solution. Then, we define the new semirings, illustrate their function with examples and sketch a proof of correctness in Section III. A few pre- and post-processing steps are necessary to present the lattices in a readable form, which are described in Section IV. Through empirical evaluation, we demonstrate the correctness of our solution in Section VI and then conclude.

II. Background and Notations

Adopting the notation often used in the literature [1], a **semiring** is 4-tuple $(K, \oplus, \otimes, \bar{0}, \bar{1})$ with a nonempty set K on which two binary operations are defined, namely the semiring plus $\oplus$ and semiring times $\otimes$, such that:

1) $(K, \oplus)$ is a commutative monoid with identity $\bar{0}$;
2) $(K, \otimes)$ is a monoid with identity $\bar{1}$;
3) $\otimes$ distributes over $\oplus$; and
4) $\bar{0} \otimes k = k \otimes \bar{0} = \bar{0} \quad \forall k \in K$.

Typically, $\bar{1} \neq \bar{0}$ is assumed to avoid trivial semirings. The *tropical semiring* is an example of a well-known semiring and is defined as $(\Re \cup \{\infty\}, min, +, \infty, 0)$.

A **weighted finite-state transducer** T over a semiring $(K, +, *, 0, 1)$ is an 8-tuple $(\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where Σ and Δ are the finite input and output alphabets respectively, Q is a finite set of states of which I and F are initial and final subsets of states respectively, E is a finite set of transitions between pairs of states with an input and an output alphabet as well as a semiring weight $E \subseteq Q \times (\Sigma \cup \epsilon) \times (\Delta \cup \epsilon) \times K \times Q$, ϵ is an empty element in the alphabet, and λ and ρ are semiring weights associated with initial and final states respectively. A weighted finite-state acceptor can be regarded as a special case where either the input or the output alphabet is an empty set.

A weighted finite-state automaton (WFSA) or transducer is **deterministic** if it has a unique initial state and no two transitions leaving the same state have the same input label. A **generic determinization algorithm** can transform a weighted finite-state acceptor or transducer into its deterministic form if such a form exists. For details on the algorithm and conditions for determinization, see Section 6.2 in [1]. The condition most relevant for our purpose is that the algorithm works with any *weakly divisible* semiring. Briefly, a semiring $(K, \oplus, \otimes, \bar{0}, \bar{1})$ is said to be *divisible* if all non-$\bar{0}$ elements admit an inverse, that is, $(K - \bar{0})$ is a group. A semiring is *weakly divisible* if for any x and y in K such that $x \oplus y \neq \bar{0}$ there exists at least one z such that $(x \oplus y) \otimes z = x$. The $\otimes$ is *cancellative* if z is unique and can be written as $z = (x + y)^{-1}x$. The non-unique case is not relevant here.

III. The $\langle T, C \rangle$ Lexicographic Semiring

The problem of computing the single-best scoring transduction paths for all unique input sequences calls for some form of weighted determinization of an acceptor with input label sequences. For example, in Figure I, we need to determinize such that only the two paths — [*fine:VB\1 me:PRP\2*] and [*fine:JJ\1 mead:NN\5*]— are preserved. The outline of our solution is to convert the given weighted transducer into an equivalent weighted automaton that accepts the same input sequences. The original output symbols, however, are encoded as weights along with the original arc weights using a new semiring. This semiring is defined in a way that determinization only preserves the lowest scoring output symbols. After determinization, the result is transformed back into WFST in the original semiring. Note that the standard determinization for weighted transducers, such as the the one implemented in OpenFst [2], does not suffice for this: indeed, if a given word sequence has more than one possible tag sequence, and the transducer is thus *non-functional* — precisely the case we care most about, the transducer cannot be determinized.

Consider a semiring with a pair of weights, encoding the original arc weights (e.g., the probability of the arc) and the output arc symbols (e.g., the POS tag). Determinization in the transformed WFSA should clearly have the property that for any given input sequence, the single-best scoring output sequence is preserved and as such the *tropical* semiring is a natural choice for the first weight.

Choosing the semiring for the second weight however is non-trivial. The purpose of this weight is to save the output symbols of the single-best scoring path during determinization of the transformed WFSA. Note that this weight will also be subjected to operations of the determinization algorithm and hence the semiring chosen to represent this weight needs to be *weakly divisible*. For a string weight, this can be achieved by recording the division so that a subsequent $\otimes$ operation with the appropriate (inverse) string is *cancellative*. A natural model for this is *categorial grammar* [3]. In categorial grammar, there are a set of primitive categories, such as *N*, *V*, *NP*, as well as a set of complex types constructed out of left (\) or right (/) division operators. An expression $X \backslash Y$ denotes a category that, when combined with an X on its left, makes a Y. For example, a verb phrase in English could be represented as a *NP**S*, since when combined with an *NP* on the left, it makes an *S*. Similarly a determiner is *NP*/*N*, since it combines with an *N* on the right to make an *NP*.

The Categorial Semiring (Definition): We define the *left-categorial* semiring $(\Sigma^*, \oplus, \otimes, \infty_s, \epsilon)$ over strings Σ^* with ∞_s and ϵ as special infinity and null string symbols respectively. The $\otimes$ operation accumulates the symbols along a path using standard string catenation. The $\oplus$ operation simply involves a string comparison between the string representations of (possibly accumulated versions of) the output symbols or tags using lexicographic less-than ($<_L$). The $\oslash$ operation records the left-division in the same sense as categorial grammar:

$$\begin{aligned} w_1 \oplus w_2 &= \begin{cases} w_1 & \text{if } w_1 <_L w_2 \\ w_2 & \text{otherwise} \end{cases} \\ w_1 \otimes w_2 &= w_1 \cdot w_2 \\ w_1 \oslash w_2 &= w_2 \backslash w_1 \end{aligned}$$

Obviously, a *right-categorial* semiring can also be defined in a similar fashion with a right-division operator.

One difference between the categorial semiring and standard categorial grammar is that in the categorial semiring division may involve complex categorial weights that are themselves concatenated. For example one may need to left-divide a category *NN* by a complex category that itself involves a division and a multiplication. Denoting catenation here by '_', we might thus produce a category such as ⟨*VB**JJ_NN*⟩*NN*. We assume division has precedence over times (concatenation), so in order to represent this complex category, the disambiguating brackets ⟨⟩ are needed. The interpretation of this category is something that, when combined with the category *VB**JJ_NN* on the left, makes an *NN*.

Having chosen the semirings for the first and second weights in the transformed WFSA, we now need to define a joint semiring over both the weights and specify its operation.

The $\langle T, C \rangle$ Lexicographic Semiring (Definition): We define the $\langle T, C \rangle$ *Lexicographic Semiring* $(\langle \Re \cup \{\infty\}, \Sigma^* \rangle, \oplus, \otimes, \bar{0}, \bar{1})$ over a tuple of tropical and left-categorial weights, inheriting their corresponding identity elements. The $\oplus$ and $\otimes$ operator of this lexicographic semiring is defined in terms of the standard arithmetic less-than $<$ and lexicographic less-than $<_L$ as:

$$\begin{aligned}\langle w_1, w_2\rangle \oplus \langle w_3, w_4\rangle &= \begin{cases}\langle w_1, w_2\rangle \text{ if } w_1 < w_3; & \text{else}\\ \langle w_3, w_4\rangle \text{ if } w_1 > w_3; & \text{else}\\ \langle w_1, w_2\rangle \text{ if } w_2 <_L w_4; & \text{else}\\ \langle w_3, w_4\rangle & \end{cases}\\ \langle w_1, w_2\rangle \otimes \langle w_3, w_4\rangle &= \langle w_1 + w_3, w_2 \cdot w_4\rangle\end{aligned}$$

Note that for a lexicographic semiring to be valid, each of the underlying semirings must observe the *path property* [4], [5]. The path property of a semiring K is defined in terms of the *natural order* on K such that: $a <_K b$ iff $a \oplus b = a$. For example, the tropical semiring respects the path property. In the case of lexicographic semiring, it must always return one of the input weights, (w_1, w_2) or (w_3, w_4). Thus the result of $w_1 \oplus w_3$ must be either w_1 or w_3, and not some other value; and similarly for w_2 and w_4.

A Sketch of a Proof of Correctness: The correctness of this lexicographic semiring for our problem can be proved by tracing the results of operation in a generic determinization algorithm (e.g., [1]). Instead, here we provide an intuition using the example in Figure I. The two input strings *fine me* and *fine mead* share the prefix *fine*. In the first case *fine* is a verb (VB), whereas in the second it is a adjective (JJ). When two outgoing arcs have same input symbols, the determinization algorithm chooses the arc with the lowest weight, $\langle 1, \mathit{JJ}\rangle$. For potential future use (residual), the other weight $\langle 2, \mathit{VB}\rangle$ is divided by the lowest weight $\langle 1, \mathit{JJ}\rangle$ and the result $\langle 1, \mathit{JJ}\backslash \mathit{VB}\rangle$ is saved. Note that the divide operation for the tropical semiring is arithmetic subtraction. When processing the next set of arcs, the determinization algorithm will encounter two paths for the input *fine mead*. The accumulated weight on the path through nodes 0-2-3 is straightforward and is $\langle 1, \mathit{JJ}\rangle \otimes \langle 6, \mathit{NN}\rangle = \langle 7, \mathit{JJ\_NN}\rangle$. The accumulated weight computed by the determinization algorithm through 0-1-3 consists of three components – the lowest weight for *fine*, the saved residual and the arc weight for *mead* from 1-3. Thus, the accumulated weight for 0-1-3 for *fine mead* is $\langle 1, \mathit{JJ}\rangle \otimes \langle 1, \mathit{JJ}\backslash \mathit{VB}\rangle \otimes \langle 7, \mathit{NN}\rangle$, which will reduce to $\langle 9, \mathit{VB\_NN}\rangle$. From the two possible paths that terminate at node 3 with input string *fine mead*, the determinization algorithm will pick one with the lowest accumulated weight, $\langle 7, \mathit{JJ\_NN}\rangle \oplus \langle 9, \mathit{VB\_NN}\rangle = \langle 7, \mathit{JJ\_NN}\rangle$, the expected result. Similarly, the determinization algorithm for the input *fine me* will result in picking the weight $\langle 5, \mathit{VB\_PRP}\rangle$. Thus, the determinization algorithm will produce the desired result for both input strings in Figure I and this can be shown to be true in general.

After determinization, the output symbols (tags) on the second weight may accumulate in certain paths, as in the example above. These weights need to be mapped back to associated input symbols (words). This mapping and the complete procedure for computing the single-best transduction paths for all unique input sequences for a given WFST (word lattice) using the $\langle T, C\rangle$ lexicographic semiring is described in the next section.

IV. Lattice Representation

Consider a lattice transducer where input labels are words (e.g., generated by a speech recognizer), output labels are tags (e.g. generated by a part-of-speech tagger), and weights in the tropical semiring represent negative log probabilities. In general for any given word sequence, there may be many paths in the lattice with that word sequence, with different costs corresponding to different ways of deriving that word sequence from the acoustic input, as well as different possible ways of tagging the input.

The procedure for removing the redundant paths is as follows. We convert the weighted transducer to an equivalent acceptor in the $\langle T, C\rangle$-lexicographic semiring as in Figure 2. This acceptor is determinized in the $\langle T, C\rangle$-lexicographic

```
CONVERT(L)
1  L′ ← new FST
2  for $s$ in States(L) do
3      add state $s'$ to L′
4      if $s$ is Start(L) then
           Start(L′) ← $s'$
5      if Final($s$, L) = $\infty$ then
           Final($s'$, L′) ← $\langle \infty, \infty_s \rangle$
6      else Final($s'$, L′) ← ⟨Final(s, L), $\epsilon$⟩
7      for arc in Arcs($s$, L) do
8          add arc′ to Arcs($s'$, L′)
9          in-label(arc′) ← in-label(arc)
10         next-state(arc′) ← next-state(arc)
11         weight(arc′) ← ⟨weight(arc), out-label(arc)⟩
12 return L′
```

Fig. 2. Pseudo code for converting POS-tagged word lattice into an equivalent $\langle T, C\rangle$ lexicographic acceptor.

semiring, to yield a lattice where each distinct sequence of input-labels (words) corresponds to a single path. The resulting deterministic acceptor is converted back to a transducer (L') in the original tropical semiring, keeping the input labels the same, but assigning the first weight as arc costs and possibly complex tags in the second weight as arc output labels. The conversion is performed by an algorithm that is essentially the inverse of the algorithm in Figure 2. Since the output tags may be complex categorial tags, one further step is needed to complete the process, namely to build a *mapper* FST (M) that converts sequences of complex tags back to sequences of simple tags. The algorithm for constructing this mapper is given in Figure 3, and an illustration can be found in Figure 6. Finally, the determinized transducer is composed with the mapper, ($L' \circ M$), to yield the desired result. Note, crucially, that the mapper will in general change the topology of the determinized acceptor, splitting states as needed. This can be seen by comparing Figures 5 and 7 below.

A simple example will serve to illustrate the algorithms. Consider the toy lattice in Figure 4. Note that there are four paths through the lattice, with two possible tags for the word sequence *Time flies like an arrow*. We would like to derive a

```
BUILDMAPPER(L)

for $s$ in States(L) do
    for arc in Arcs($s$, L) do
        SYMBOLS $\leftarrow$ output-label(arc)
$M \leftarrow$ new FST
Start($M$) $\leftarrow$ 0
Final($M$) $\leftarrow$ 0
for $l$ in SYMBOLS do
    if IsSimple($l$) then AddArc(M, 0, Arc(0, $l$, $l$))
    else MAKEPATH(M, $l$)
return $M$
```

```
MAKEPATH(M, l)

Split $l$ into vector $inputs$ on \, but treat any string in $l$ enclosed in $\langle\rangle$ as a single unit.
Split the final element of $inputs$ on _ into vector $outputs$
Replace the final element of $inputs$ with $l$
Right-pad $outputs$ with $\epsilon$ labels so that $|inputs| = |outputs|$
Right-pad $inputs$ with $\epsilon$ labels so that $|inputs| = |outputs|$
Create a path in $M$ starting at state 0 and ending in 0 where the $i$th arc has labels $inputs_i : outputs_i$
return
```

Fig. 3. Pseudo code for construction of mapper transducer. The function *IsSimple* returns true in case the tag is a simple tag, not a complex categorial tag.

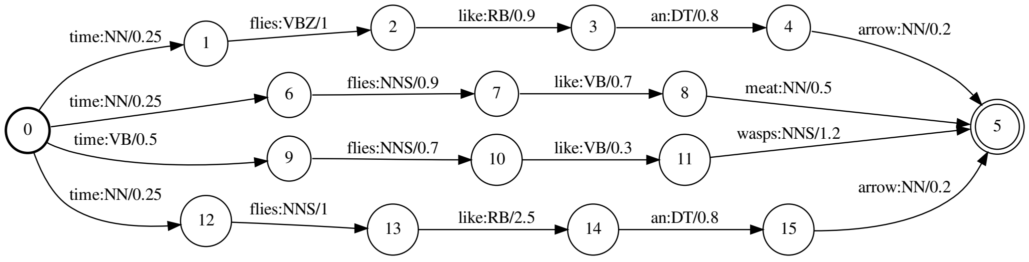

Fig. 4. Sample input lattice.

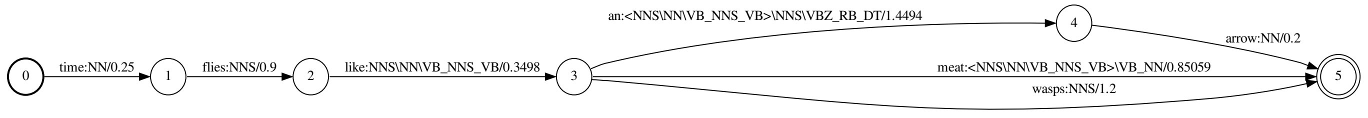
time:NN/0.25
flies:NNS/0.9
like:NNS\NN\VB_NNS_VB/0.3498
an:<NNS\NN\VB_NNS_VB>\NNS\VBZ_RB_DT/1.4494
arrow:NN/0.2
meat:<NNS\NN\VB_NNS_VB>\VB_NN/0.85059
wasps:NNS/1.2

Fig. 5. Lattice after conversion to the $\langle T, C\rangle$ semiring, and determinization.

lattice that has just three paths, retaining a single best scoring path for the latter word sequence. The result of converting the lattice in Figure 4 to the $\langle T, C\rangle$ semiring, followed by determinization, and conversion back to the tropical semiring, is shown in Figure 5. Note now that there are three paths, as desired, and that the tags on several of the paths are complex categorial tags. To understand the semantics of the categorial weights, consider the path that contains the words *time flies like meat*, which has the categorial sequence *NN*, *NNS*, *NNS*$\backslash$*NN*$\backslash$*VB_NNS_VB*, $\langle$*NNS*$\backslash$*NN*$\backslash$*VB_NNS_VB*$\rangle\backslash$*VB_NN*. Notionally, the cancellation, working from right to left, first reduces *NNS*$\backslash$*NN*$\backslash$*VB_NNS_VB* with $\langle$*NNS*$\backslash$*NN*$\backslash$*VB_NNS_VB*$\rangle\backslash$*VB_NN*, yielding *VB_NN*. This then is concatenated with the first two categories to yield the sequence *NN_NNS_VB_NN*. The actual cancellation is performed by the mapper transducer in Figure 6; the cancellation just described can be seen in the path that exits state 0, passes through state 5, and returns to state 0.

The $\langle T, C\rangle$-semiring can be generalized to compute the single-best path in multi-tape weighted transducers. For ex-

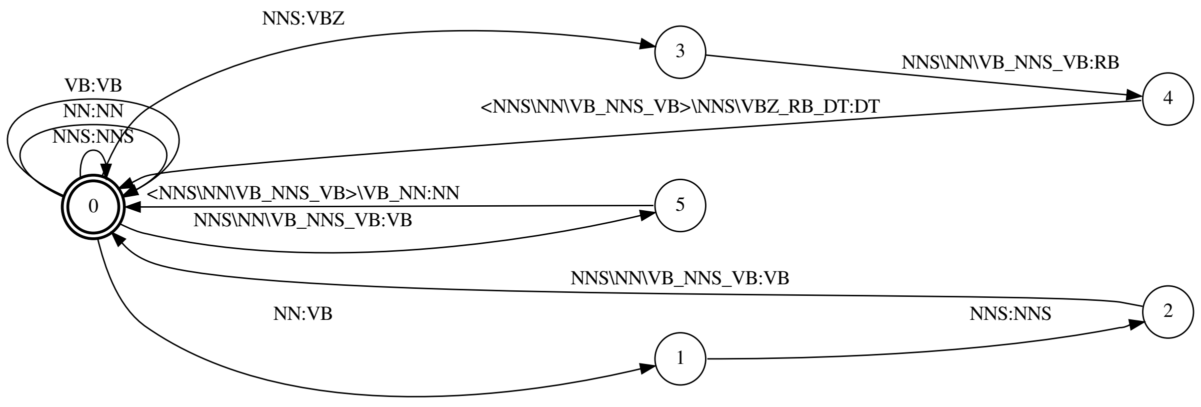

Fig. 6. After conversion of the $\langle T, C\rangle$ lattice back to the tropical, this mapper will convert the lattice to its final form.

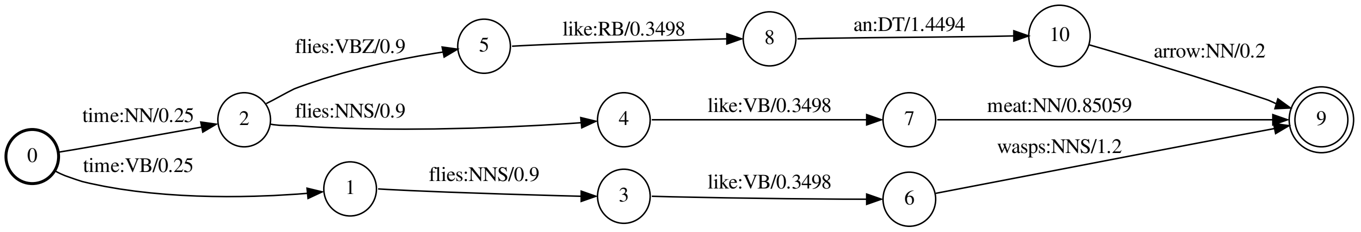

Fig. 7. Final output lattice with the desired three paths.

ample, by encoding the arc likelihoods, the phone sequence, the clustered allophone sequence, acoustic state sequence, and acoustic segmental duration associated with word sequence as a $\langle T, C, C, C, T\rangle$ lexicographic semiring and determinizing the resulting automaton, we can extract the tags corresponding to the single-best word sequence. Thus, our method is much more flexible and powerful than algorithms developed specifically for determinizing POS-tagged word lattices as in [6] or approximations specific to applications as in [7].

V. Sketch of Argument for Efficiency

Consider an input lattice with V_{lat} states, E_{lat} arcs, and maximum out-degree of D_{lat}. We first tag the lattice by composing with a POS-tagger, which has complexity $\mathcal{O}(V_{lat}V_{pos}D_{lat}\log(D_{pos} + M_{pos}))$, where M_{pos} is the maximum number of times an (input) symbol repeats at some state in the tagger.[1] The result is then converted to the $\langle T, C\rangle$ semiring, which is linear in $V + E$. The resulting acceptor is then determinized, an operation that is exponential in the size of the machine. Construction of the mapper transducer is linear in the size of the categorial label set. Finally, composition of the result with the mapper transducer is $\mathcal{O}(V_{lat}V_{map}D_{lat}(\log D_{map} + M_{map}))$.

Now consider the complexity of the alternative algorithm that we sketched in the introduction. First one must extract each of the individual paths in the lattice, which is exponential in D_{lat}. Then each of these paths must be composed with the tagger, requiring $\mathcal{O}(V_{path}V_{pos}D_{path}\log D_{pos})$. The shortest path must then be computed for each — $\mathcal{O}(V_{path}\log V_{path} + E_{path})$. Finally the paths must be unioned back together, which is linear in the size of each path. The time (as well as storage) requirements of pulling out what can potentially be a very large number of paths will dominate this approach and render it far less efficient than the algorithm proposed here.

VI. Experiments

The proposed solution was empirically evaluated on 4,664 lattices from the NIST RT Dev04 test set. The lattices were generated using a state-of-the-art speech recognizer, similar to [8], trained on about 2000 hours of data, that performed at a word error rate of about 24%. The utterances were decoded in three stages using speaker independent models, vocal-tract length normalized models and speaker-adapted models. The three sets of models were similar in complexity with 8000

[1] http://www.openfst.org/twiki/bin/view/FST/ComposeDoc

clustered pentaphone states and 150K Gaussians with diagonal covariances.

The lattices from the recognizer were tagged using a weighted finite state tagger. The tagger was trained on the Switchboard portion of the Penn Treebank [9]. Treebank tokenization is different from the recognizer tokenization in some instances, such as for contractions ("don't" becomes "do n't") or possessives ("aaron's" becomes "aaron 's"). Further, many of the words in the recognizer vocabulary of 93k words are unobserved in tagger training, and are mapped to an OOV token "$\langle$unk$\rangle$". Words in the treebank not in the recognizer vocabulary are also mapped to "$\langle$unk$\rangle$", thus providing probability mass for that token in the tagger. A tokenization transducer $\mathcal{T}$ was created to map from recognizer vocabulary to tagger vocabulary.

A bitag HMM model is estimated and encoded in a tagging transducer $\mathcal{P}$. In this model, the transition probability is conditioned just on the previous word's tag. The transition probabilities are smoothed using Witten-Bell smoothing. For each word in the tagger input vocabulary, only POS-tags observed with each word are allowed for that word, i.e., emission probability is not smoothed and is zero for unobserved tag/word pairs. For a given word lattice $\mathcal{L}$, it is first composed with the tokenizer $\mathcal{T}$, then with the POS-tagger $\mathcal{P}$ to produce a transducer with original lattice word strings on the input side and tag strings on the output side.

This model was validated on a 2,000 sentence held-aside subset of the Switchboard treebank, and it achieved 91.5% tagging accuracy. This is several points lower than the kinds of accuracies that are typically reported for POS-tagging, although recent work on tagging conversational speech [10] reported accuracy of 92.4% for an HMM tagger on the task. That result is on a different validation set, but the result indicates that this is a harder tagging task than more typical text domains. Our system also likely suffers from using a single "$\langle$unk$\rangle$" category, which is quite coarse and does not capture informative suffix and prefix features that are common in such models for tagging OOVs. For the purposes of this paper, however, the model serves to demonstrate the utility of the semiring using a large model. A similar WFST topology can be used for discriminatively trained models using richer feature sets, which would achieve higher accuracy on the task.

The tagged lattices, obtained from composing ASR lattice with the POS tagger, were then converted to the $\langle T, C\rangle$-lexicographic semiring, determinized in this lexicographic semiring and then converted back using the mapper transducer as discussed in Section IV. As we argued in section V, this is far more efficient than the alternative of enumerating all the word sequences, tagging each individually, computing the shortest path, and then unioning the result back together.

The results of this operation were compared with the method of taking the 1,000 best paths through the original lattice, and removing any path where the path's word sequence had been seen in a lower-cost path. This generally resulted in a rank-ordered set of paths with $n < 1000$ members.

In all cases the n-best paths produced by the method proposed in this paper were identical to the n-best paths produced by the method just described. The only differences were due to minor floating-point number differences (expected due to weight-pushing in determinization), and cases where equivalent weighted paths were output in different orders.

VII. Conclusion

In this paper we introduced the *categorial* semiring, using which we created a novel $\langle T, C\rangle$ lexicographic semiring. Through illustrative examples and empirical evaluations, we show how this lexicographic semiring can be employed to extract the **single**-best scoring path for **every** input sequence in a weighted finite state transducer. The proposed solution is general in nature and can be extended to extract multiple tags corresponding to the best scoring path by tailoring the tuple weights in the lexicographic semiring, with tropical semiring for costs and categorial semiring for tags.

VIII. Acknowledgements

This research was supported in part by NSF Grants #IIS-0811745, #IIS-0905095 and #IIS-0964102, and DARPA Grant #HR0011-09-1-0041. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or DARPA. We also thank Cyril Allauzen for useful discussion.

References

[1] M. Mohri, "Weighted automata algorithms," in *Handbook of Weighted Automata*, ser. Monographs in Theoretical Computer Science, M. Droste, W. Kuich, and H. Vogler, Eds. Springer, 2009, pp. 213–254.

[2] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, ser. Lecture Notes in Computer Science, vol. 4783. Springer, 2007, pp. 11–23. [Online]. Available: http://www.openfst.org

[3] J. Lambek, "The mathematics of sentence structure," *American Mathematical Monthly*, vol. 65, no. 3, pp. 154–170, 1958.

[4] M. Mohri, "Semiring framework and algorithms for shortest-distance problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, no. 3, pp. 321–350, 2002.

[5] B. Roark, R. Sproat, and I. Shafran, "Lexicographic semirings for exact automata encoding of sequence models," in *Proceedings of ACL-HLT, 2011*. Portland, OR: Association for Computational Linguistics, 2011.

[6] E. Roche and Y. Schabes, "Deterministic part-of-speech tagging with finite-state transducers," *Computational Linguistics*, vol. 21, pp. 227–253, June 1995. [Online]. Available: http://portal.acm.org/citation.cfm?id=211190.211200

[7] M. Shugrina, "Formatting time-aligned ASR transcripts for readability," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. HLT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 198–206. [Online]. Available: http://portal.acm.org/citation.cfm?id=1857999.1858022

[8] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig, "The IBM 2004 conversational telephony system for rich transcription," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005.

[9] M. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[10] V. Eidelman, Z. Huang, and M. Harper, "Lessons learned in part-of-speech tagging of conversational speech," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2010, pp. 821–831.