EM TRAINING OF FINITE-STATE TRANSDUCERS AND ITS APPLICATION TO PRONUNCIATION MODELING

Han Shu and I. Lee Hetherington

Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA
{hshu,ilh}@sls.lcs.mit.edu

ABSTRACT

Recently, finite-state transducers (FSTs) have been shown to be useful for a number of applications in speech and language processing. FST operations such as composition, determinization, and minimization make manipulating FSTs very simple. In this paper, we present a method to learn weights for arbitrary FSTs using the EM algorithm. We show that this FST EM algorithm is able to learn pronunciation weights that improve the word error rate for a spontaneous speech recognition task.

1. INTRODUCTION

Recently, finite-state transducers (FSTs) have been shown to be useful for a number of applications in speech and language processing [1]. For example, the SUMMIT segment-based speech recognizer successfully utilizes FSTs to specify various constraints [2]. FST operations such as composition, determinization, and minimization make manipulating FSTs very simple. In this paper, we present a method to learn weights for arbitrary FSTs using the EM algorithm [3]. Our method is similar to that of [4]; however we do not explicitly make use of the "expectation semiring."

To test the FST EM algorithm, we apply the algorithm to the problem of learning pronunciation weights. With phonological rules and multiple phonemic pronunciations, the pronunciation graph for spontaneous speech can have high branching factors. Pronunciation weighting has been shown to be beneficial for segment-based speech recognition [5]. In [5] within-word pronunciation weights were ML estimated from training examples. In this paper, we experiment with learning various pronunciation weights on phonological rules and phonemic pronunciations via the proposed FST EM algorithm.

2. PROBABILISTIC INTERPRETATION OF FSTS

A weighted FST, T, assigns a weight, or score, to each complete path through it, where a path corresponds to a particular input and output label sequence (x, y). The interpretation of the weights depends on how they are manipulated algebraically, and the algebraic structure is a semiring.

This research was supported by DARPA under contract N66001-99-1-8904 monitored through Naval Command, Control and Ocean Surveillance Center.

2.1. Weight Semirings

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ defines the set $\mathbb{K}$ containing the weights, the operators $\oplus$ and $\otimes$, with the identity elements $\overline{0}$ and $\overline{1}$ such that for all $a, b, c \in \mathbb{K}$, $a \oplus \overline{0} = a$, $\overline{0} \oplus a = a$, $a \otimes \overline{1} = a$, $\overline{1} \otimes a = a$, $a \otimes \overline{0} = \overline{0}$, $\overline{0} \otimes a = \overline{0}$, $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$, and $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ [1]. When manipulating weighted transducers, the $\otimes$ and $\oplus$ operators are used to combine weights in series and parallel, respectively.

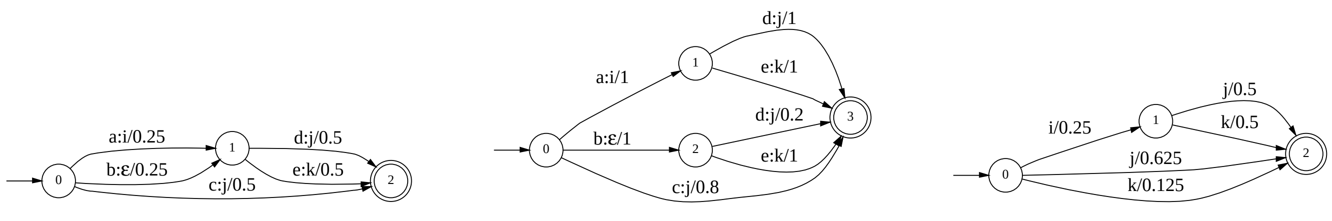
Two semirings commonly used within speech systems include the real semiring $(\mathbb{R}, +, \times, 0, 1)$ and the tropical semiring $(\mathbb{R}_+ \cup \infty, \min, +, \infty, 0)$. The real semiring, abbreviated here $(+, \times)$, can be used to represent probabilities directly, where we take the product of probabilities in series and the sum of probabilities in parallel. The tropical semiring, abbreviated here $(\min, +)$, can be used to represent negative log ($-\log$) probabilities where we take the sum of $-\log$ probabilities in series and the minimum, or most probable, $-\log$ probability in parallel. The $(\min, +)$ semiring corresponds to how scores are typically manipulated in a traditional Viterbi dynamic programming search.

2.2. Composition and Probabilities

Consider a cascade of FSTs $W_{X,Z} = T_{X|Y} \circ U_{Y|Z} \circ V_Z$. Define $W(x, z)$ to be the $\oplus$ sum over the weights of all paths through $W_{X,Z}$ with input sequence x and output sequence z, and define $T(x|y)$, $U(y|z)$, and $V(z)$ to be analogous $\oplus$ sums for the FSTs $T_{X|Y}$ and $U_{Y|Z}$ and finite-state acceptor (FSA) V_Z, respectively. From the definition of weighted composition we have:

$$W(x, z) = \bigoplus_y T(x|y) \otimes U(y|z) \otimes V(z)\ . \qquad (1)$$

If $W_{X,Z}$, $T_{X|Y}$, $U_{Y|Z}$, and V_Z represent probabilities in the real semiring $(+, \times)$, then $W(x|z) = P(x|z)$, $T(x|y) = P(x|y)$, $U(y|z) = P(y|z)$, and $V(z) = P(z)$. With the conditional independence assumption $P(x|y, z) = P(x|y)$,



(a) $T_{X,Y}$ (b) $T_{X|Y}$ (c) T_Y

Fig. 1. Example FSTs in the $(+, \times)$ semiring: (a) $T_{X,Y}$ representing joint probability $P(x, y)$, (b) $T_{X|Y}$ representing conditional probability $P(x|y)$, and (c) T_Y representing marginal probability $P(y)$.

Equation 1 becomes the familiar chain rule:

$$P(x, z) = \sum_y P(x|y)P(y|z)P(z) \ .$$

For the tropical semiring $(\min, +)$ with $-\log$ probabilities, Equation 1 yields the following approximation:

$$\log P(x, z) \approx \max_y \left[\log P(x|y) + \log P(y|z) + \log P(z)\right] \ .$$

This is analogous to the approximation made by a traditional Viterbi dynamic programming decoder when it considers best paths rather than summing over all paths.

It is important to note that for a cascade of FSTs to chain together to represent a probability such as $P(w, z)$ above, the intermediate FSTs must represent conditional probabilities as do $T_{X|Y}$ and $U_{Y|Z}$ in this example.

2.3. Joint and Conditional Probabilities

Given a conditional probability FST $T_{X|Y}$ and marginal probability FST T_Y, we can compute the joint probability FST as $T_{X,Y} = T_{X|Y} \circ T_Y$ as shown in Figure 1. As we will see in Section 3.1, we will need the ability to convert a joint probability into a conditional probability as in the familiar relation $P(x|y) = P(x, y)/P(y)$. The finite-state equivalent of this relation is

$$T_{X|Y} = T_{X,Y} \circ [\det(T_Y)]^{-1} \ , \qquad (2)$$

where $T_Y = \text{project}_Y(T_{X,Y})$ is the FSA representing the marginal distribution for y in which x labels have been discarded, $\det(\cdot)$ is determinization and $[\cdot]^{-1}$ replaces every non-$\overline{0}$ transition and final weight w by its reciprocal $\overline{1}\otimes^{-1}w$. For the $(+, \times)$ semiring this reciprocal is $1/w$, and for the $(\min, +)$ semiring it is $-w$.

The determinization (and included ϵ removal) is important so that the marginal is removed correctly. Recall that we defined $P(y)$ to be the $\oplus$ sum over the weights of all paths y through T_Y. With $\det(T_Y)$, there is at most one path for any given y, with the determinization having performed the necessary $\oplus$ sum. It is important to note that it is not always possible to determinize a cyclic weighted FSA [1], and thus it is not always possible to compute a conditional FST from a joint FST. However, we have yet to run into this situation in practice when training various pronunciation weights.

Figure 1 shows various FSTs representing joint, conditional, and marginal probabilities. Note that the topology of the conditional FST $T_{X|Y}$ in (b) is different from the topology of the joint FST $T_{X,Y}$ in (a). In general the topology of a given joint distribution FST differs from the topology of its corresponding conditional distribution FST. Furthermore, some FST topologies are not able to support arbitrary conditional probability distributions due to ϵ outputs. For this reason, we chose to train a joint distribution FST using EM and afterwards convert it to a conditional distribution FST using Equation 2.

3. EM WEIGHT TRAINING

We can use the EM algorithm [3] to train a joint probability model for a FST $T_{X,Y}$. For a given joint input/output sequence pair (x_i, y_i), multiple paths through $T_{X,Y}$ may be permitted. We initialize the weights of $T_{X,Y}$ such that for each state, all leaving transitions are equally likely (and for these purposes exiting at a final state counts as a leaving transition). We use the $(+, \times)$ semiring during the EM training, and if desired convert trained weights to the $(\min, +)$ semiring after training. Finally, if we require a conditional probability model, we convert the joint FST to its corresponding conditional FST using the method of Section 2.3.

3.1. Isolated Training

For the expectation step of each EM iteration, for each input/output sequence pair (x_i, y_i) in the training corpus, we compute the expected number of times each transition in $T_{X,Y}$ is traversed as follows:

1. Compute $T_i = x_i \circ T_{X,Y} \circ y_i$, essentially the part of $T_{X,Y}$ supporting input sequence x and output sequence y. T_i may contain more than one path.
2. Normalize the weights in T_i such that probabilities of all paths sum to 1.
3. Update the expected transition counts for $T_{X,Y}$ that correspond to transitions in T_i.

For the maximization step, we convert the transition and state final counts to a joint probability distribution by normalizing counts so that the total weights of all transitions (and state finality) leaving each state is 1. To allow the trained joint distribution to generalize to unseen input/output sequences that it accepts, we typically apply a floor to all counts so that transitions are not assigned zero probabilities.

3.2. Training Within Cascade

We have outlined how to train an isolated joint FST from example pairs of direct input and output sequences. It is also possible to train a FST in the middle of a cascade such as $S_{W|X} \circ T_{X|Y} \circ U_{Y|Z}$. In this case, we may wish to train $T_{X|Y}$ from example sequence pairs (w_i, z_i). A straightforward way to accomplish this is to compute the weighted FSAs $x_i = \text{project}_X(w_i \circ S_{W|X})$ and $y_i = \text{project}_Y(U_{Y|Z} \circ z_i)$. These FSAs represent all possible input and output sequences (x, y) for $T_{X|Y}$ compatible with the given (w_i, z_i). Then, x_i and y_i FSAs can be used as in Section 3.1.

4. FSTS IN SUMMIT

In the SUMMIT segment-based speech recognition system [2], various constraints such as context dependency, phonological rules, lexicon, and language model, are combined using a cascade of FSTs, $A \circ U = A \circ (C \circ P \circ L \circ G)$. A is the acoustic likelihood of observations y given a sequence of context-dependent phones, C maps context-independent phones to context-dependent phones or diphones, P represents the phonological rules mapping phonemic sequences to phonetic sequences [6], L converts lexical items (i.e., words) to phonemic pronunciations, and FSA G is a language models which assigns probabilities to word sequences w.

In the typical formulation, the goal of recognition is to find the most likely sequence of words w^* given the acoustic observations y; that is:

$$w^* = \arg\max_w P(y|w) = \arg\max_w P(y, w), \tag{3}$$

where w ranges over all possible word sequences.

Using the analysis of Sec. 2.2 and appropriate conditional independence assumptions, the composition of a cascade of weighted FSTs, $A \circ C \circ P \circ L \circ G$, yields the joint probability of the observations y and word sequence w, $P(y, w)$. The recognition problem of Equation 3 is converted to the equivalent problem of searching for the best path in $A \circ U$.

5. EXPERIMENTS AND RESULTS

For the recognition task, we chose to use the JUPITER conversational weather information system as the experiment domain [2]. The training set consisted of 116,867 utterances, totaling 105 hours of speech. The acoustic model employed 1,573 clustered boundary diphone models using mixtures of diagonal Gaussians, with a total of 35,359 component Gaussians. The test set contained 1,711 utterances with 9,659 words, totaling 1.6 hours of speech. The decoding dictionary consisted of 2,014 unique words, covering the test set. Bigram and trigram class-based language models were used in the first pass and the second pass respectively. Both models were trained with 236 hand-crafted class definitions and transcriptions of the training set and 2,661 utterances containing out of training vocabulary words.

In the baseline JUPITER system, C, P, and L were unweighted FSTs, all weights are 0 in the $(\min, +)$ semiring (i.e., $\overline{1}$). The language model G was a weighted FSA, and the weights were imported from a separately trained n-gram. The baseline recognition word error rate (WER) was 9.4%. There is no need to assign weights to C because it represents a one-to-one mapping from phone sequences to their corresponding sequence of context-dependent labels. We suspected that training the weights of P and L would decrease the word error rate. The FST EM algorithm presented in this paper enabled learning weights for P and L without implementing custom training methods.

To learn the weights, we first computed the "reference phone labels" on the training set with baseline acoustic models and the baseline U with unweighted P and L. For simplicity the one-best "reference phone labels" were used instead a phone lattice. Together, the "reference phone labels" and the reference word transcription on the training set formed the example sequence pair (x_i, z_i) in Sec. 3.2 needed for EM training. For each experiment, we first trained a joint FST, then we computed the corresponding conditional FST using Equation 2. Because decoding in SUMMIT was done via the $(\min, +)$ semiring, we also evaluated Equation 2 with the same semiring, so that the resulting conditional FSTs would contain at least one input sequence with a path weight of 0 (i.e., $\overline{1}$) for any possible output sequence. It is important not to penalize output sequences with many alternative input sequences. We also tried to evaluate Equation 2 via $(+, \times)$ semiring, and the WER increased. For the purpose of fair comparison, we used the same beam pruning parameters for all the conditions.

The results of training various pronunciation weights were summarized in Table 1 and described in more detail in the following subsections.

5.1. Training Phonological Rules *P*

In SUMMIT, we currently use about 168 hand-crafted phonological rules that map 63 "phonemic" input symbols to 71 "phonetic" output symbols. The rules apply to both within-word and cross-word phoneme sequences. For example, one rule for flappable t is expressed by:

$$\{\text{VOWEL}\}\ t\ \{\text{VOWEL}\} \Rightarrow dx \mid tcl\ t.$$

This rule only applies to intervocalic ts. In this case, t can be mapped to a flap dx or t closure, tcl, followed by a t release [6]. The weights associated with this rule would model how often an intervocalic t is flapped.

P has about 800 states and 12,000 transitions. To speed up training, we de-composed P into a cascade of three FSTs, $P = S \circ R \circ I$, where both S and I represented deterministic mappings between input and output sequences [6]. Thus, learning weights on R alone is equivalent to learning on P in whole. R contained only 249 states and 884 transitions. After EM training R, and building a new U with $C \circ S \circ tr(R) \circ I \circ L \circ G$, where $tr(R)$ denotes the conditional probability FST R after EM training with data. This new recognizer with trained P, $tr(P)$, obtained a word error rate rate of 9.0%, a relative reduction of 4.3% from the baseline.

U	WER	Rel. Red.
$C \circ P \circ L \circ G$	9.4%	-
$C \circ tr(P) \circ L \circ G$	9.0%	4.3%
$C \circ P \circ tr(L) \circ G$	8.8%	6.4%
$C \circ tr(P) \circ tr(L) \circ G$	8.7%	7.4%
$C \circ tr(P \circ L) \circ G$	8.7%	7.4%

Table 1. Recognition results and relative reduction (Rel. Red.) in WER for various pronunciation weight training configurations.

5.2. Training Phonemic Pronunciations L

L represents the phonemic pronunciations of words. L has 5,542 states and 8,312 transitions. The training procedure learned relative frequencies of the different pronunciations which were used by the training data. The phonemic pronunciations in the training L were not shared between similar words, e.g., the paths for the word "rain" and the word "raining" are not shared. Thus, this learning process only trained word-dependent phonemic pronunciation weights. The new recognizer with $tr(L)$ achieved a WER of 8.8%, a relative reduction of 6.4% from the baseline.

5.3. Training P and L Separately

In the two previous subsections, we trained weights for P and L separately. We can use both of them simultaneously by constructing U with a FST cascade using both $tr(P)$ and $tr(L)$, $C \circ tr(P) \circ tr(L) \circ G$. The WER obtained using this new U was 8.7%, better than using either $tr(P)$ or $tr(L)$ alone, but only slightly.

5.4. Training $P \circ L$

Both P and L have relatively few branching points that need to be trained. To increase the number of parameters to be learned, we decided to train word-dependent pronunciation weights by composing P with L, (i.e. $P \circ L$). The resulting $P \circ L$ contains 14,428 states and 127,113 transitions, which was significantly bigger than the size of P or L. EM training to obtain the joint probability FST required only slightly more computation than training either P or L alone. The size of U with either $tr(P)$ or $tr(L)$ was similar to the baseline U. However, the U with $tr(P \circ L)$ had 50 times more transitions than the baseline U because the marginal distribution FSA increased in size dramatically. The marginal FSA which models word sequences had learned a complicated model with long range dependencies. After projection of the joint probability FST, 27,467 transitions out of 127,113 of the resulting FSA were ϵ transitions. The determinization (including ϵ removal) of the marginal distribution FSA dramatically increased its size to nearly 6 million transitions. The resulting U size actually increased to 20 million transitions. Clearly, the application of Equation 2 to compute the exact conditional may be computationally impractical, and an approximation may be necessary for larger FSTs. Despite the increased number of parameters in $P \circ L$, the WER achieved was the same 8.7% achieved by $tr(P) \circ tr(L)$.

6. CONCLUSION & FUTURE WORK

We have presented a method to train FSTs directly via the EM algorithm in this paper. The method operates on any generic FST, even those with ϵ transitions. Because some FST topologies are not able to support arbitrary conditional probability distributions due to ϵ outputs, we chose to train a joint probability FST first, then compute the corresponding conditional probability FST from the trained joint FST.

By learning pronunciation weights on P, L, and $P \circ L$ with the FST EM algorithm, we showed that WER can be reduced. To our knowledge, this is the first application of a FST training algorithm. In our experiments, weights on word-dependent phonemic pronunciations reduced WER more than weighting phonological rules. However, a trained pronunciation rules P has the advantage that it can provide pronunciation weights for unseen words. This property is desirable because it can provide some degree of vocabulary-independent pronunciation weighting. We plan to address this issue by training syllable-based pronunciation weights, also automatically learning pronunciation rules [7].

Since Equation 2 does not guarantee that the resulting conditional probability FST will be similar in size to the joint probability FST, there will be cases where the exact application of Equation 2 is impractical, e.g. $P \circ L$. To overcome this problem, we plan to try different methods to approximate the marginal FST. Recall that the "reference phone labels" are computed using U. We plan to experiment with iteratively computing new "reference phone labels" based on the U with trained pronunciation weights, then training the pronunciation weights would reduce word error rate further. This FST EM algorithm can have many applications other than pronunciation weight learning. We also plan to apply these results for the speech synthesis research activity in our group.

7. REFERENCES

[1] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, pp. 269–311, 1997.

[2] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. J. Hazen, and I. L. Hetherington, "JUPITER: A telephone-based conversational interface for weather information," *IEEE Trans. on Speech and Audio Processing*, vol. 8, no. 1, pp. 100–112, 2000.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, June 1977.

[4] J. Eisner, "Parameter estimation for probabilistic finite-state transducers," in *Proc. of the Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July 2002.

[5] N. Ström, I. L. Hetherington, T. J. Hazen, E. Sandness, and J. Glass, "Acoustic modeling improvements in a segment-based speech recognizer," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Snowbird, Dec. 1999, pp. 139–142.

[6] I. L. Hetherington, "An efficient implementation of phonological rules using finite-state transducers," in *Proc. Eurospeech*, Aalborg, Sept. 2001, pp. 1599–1602.

[7] S. Seneff, "The use of linguistic hierarchies in speech understanding," in *Proc. ICSLP*, Sydney, Aug. 1998.