A Finite-state transducers

A.1 Rational Relations

A **relation** is a set of pairs—in this paper, a subset of $\Sigma^* \times \Delta^*$, so it relates strings over an "input" alphabet Σ to strings over an "output" alphabet Δ.

A **weighted relation** is a function R that maps any string pair $(\mathbf{x}, \mathbf{y})$ to a **weight** in $\mathbb{R}_{\geq 0}$.

We say that the relation R is **rational** if R can be defined by some weighted finite-state transducer (FST) $\mathcal{T}$. As formalized in Appendix A.3, this means that $R(\mathbf{x}, \mathbf{y})$ is the total weight of all accepting paths in $\mathcal{T}$ that are labeled with $(\mathbf{x}, \mathbf{y})$ (which is 0 if there are no such accepting paths). The weight of each accepting path in $\mathcal{T}$ is given by the product of its arc weights, which fall in $\mathbb{R}_{>0}$.

The set of pairs $\text{support}(R) \triangleq \{(\mathbf{x}, \mathbf{y}) : R(\mathbf{x}, \mathbf{y}) > 0\}$ is then said to be a **regular relation** because it is recognized by the unweighted FST obtained by dropping the weights from $\mathcal{T}$. In this paper, we are interested in defining *non-rational* weighting functions R with this same regular support set.

A.2 Finite-state transducers

We briefly review finite-state transducers (FSTs). Formally, an FST is a tuple $\mathcal{T}_0 = (\Sigma, \Delta, Q, A_0, I, F)$ where

- Σ is a finite input alphabet
- Δ is a finite output alphabet
- Q is a finite set of states
- $A_0 \subseteq Q \times Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\})$ is the set of weighted arcs
- $I \subseteq Q$ is the set of initial states (conventionally $|I| = 1$)
- $F \subseteq Q$ is the set of final states

Let $\mathbf{a} = a_1 \ldots a_T$ (for $T \geq 0$) be an accepting path in $\mathcal{T}_0$, that is, each $a_i = (q_{i-1}, q_i, \sigma_i, \delta_i) \in A_0$ and $q_0 \in I, q_T \in F$. We say that the input and output strings of $\mathbf{a}$ are $\sigma_1 \cdots \sigma_T$ and $\delta_1 \cdots \delta_T$.

A.3 Real-valued weighted FSTs

Weighted FSTs (WFSTs) are defined very similarly to FSTs. A WFST is formally defined as a 6-tuple, just like an (unweighted) FST: $\mathcal{T} = (\Sigma, \Delta, Q, A, I, F)$, with arcs carrying weights: $A \subseteq Q \times Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R}$. Compared to FST arcs in Appendix A.2, a WFST arc each $a_i = (q_{i-1}, q_i, \sigma_i, \delta_i, \kappa_i) \in A$ has weight κ_i. We also define the weight of $\mathbf{a}$ to be $w(\mathbf{a}) \triangleq \bigotimes_{i=1}^{T} \kappa_i \in \mathbb{R}$.

The weight of the entire WFST $\mathcal{T}$ is defined as the total weight (under $\oplus$) of all accepting paths:

$$\mathcal{T}[\,] \triangleq \bigoplus_{\mathbf{a}} w(\mathbf{a}) \in \mathbb{R} \tag{14}$$

More interestingly, the weight $\mathcal{T}[\mathbf{x}, \mathbf{y}]$ of a string pair $\mathbf{x} \in \Sigma^*, \mathbf{y} \in \Delta^*$ is given by similarly summing $w(\mathbf{a})$ over just the accepting paths $\mathbf{a}$ whose input string is $\mathbf{x}$ and output string is $\mathbf{y}$.

B Training Procedure Details

We use stochastic gradient descent (SGD) to train $G_{\boldsymbol{\theta}}$. For each example, we compute the gradient using normalized importance sampling over an ensemble of 512 particles (paths), the maximum that we could compute in parallel. By using a large ensemble, we reduce both the bias (from normalized importance sampling) and the variance of the gradient estimate; we found that smaller ensembles did not work as well. Thus, we used only one example per minibatch.

We train the 'clamped' proposal distribution $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathbf{x} \circ \mathcal{T} \circ \mathbf{y})$ differently from the 'free' ones $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathbf{x} \circ \mathcal{T})$ and $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathcal{T} \circ \mathbf{y})$. The clamped distribution is trained alternately with $G_{\boldsymbol{\theta}}$, as listed in Algorithm 2. We evaluate on the development dataset at the end of each epoch using the **Reranking External** method described in §4.3. When the EM accuracy stops improving, we fix the parameters of $G_{\boldsymbol{\theta}}$ and start training $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathbf{x} \circ \mathcal{T})$ and $q_{\boldsymbol{\phi}}(\mathbf{a} \mid \mathcal{T} \circ \mathbf{y})$ on the inclusive KL divergence objective function, using methods described in (Lin and Eisner, 2018). We then initialize the free distributions' RNNs using those of the clamped distributions. We train the free proposal distributions for 30 epochs, and evaluate on the development dataset at the end of each epoch. Results from the best epochs are reported in this paper.

C Further Analyses

C.1 Does feeding alignments into the decoder help?

In particular, we attribute our models' outperforming Neuralized IBM Model 1 to the fact that a complete history of past alignments is remembered in the RNN state. (Wu et al., 2018) noted that in character transduction tasks, past alignment information seemed to barely affect decoding decisions made

Input / Output	Paths	$\hat{P}(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$
/mɑɹʃ/ marche	ϵ:m m:ɑ a:ɹ r:ʃ c:ϵ h:ϵ e:ϵ	96.5%
	ϵ:m m:ɑ a:ɹ r:ϵ ϵ:ʃ c:ϵ h:ϵ e:ϵ	2.5%
	ϵ:m m:ɑ a:ϵ ϵ:ɹ r:ʃ c:ϵ h:ϵ e:ϵ	1.0%
/ɔnslɔt/ onslaught	ϵ:ɔ o:n n:ϵ ϵ:s s:l l:ɔ a:ϵ u:ϵ g:ϵ h:t t:ϵ	76.3%
	ϵ:ɔ o:n n:s s:l l:ɔ a:ϵ u:ϵ g:ϵ h:t t:ϵ	21.4%
	ϵ:ɔ o:n n:ϵ ϵ:s s:l l:ɔ a:ϵ u:ϵ g:ϵ h:ϵ ϵ:t t:ϵ	1.5%
/wɪlɪŋhəm/ Willing-ham	ϵ:w W:ɪ i:l l:ϵ l:ϵ ϵ:ɪ i:ŋ n:ϵ g:ϵ ϵ:h h:ə a:ϵ ϵ:m m:ϵ	40.1%
	ϵ:w W:ɪ i:l l:ϵ l:ɪ i:ŋ n:ϵ g:ϵ ϵ:h h:ə a:ϵ ϵ:m m:ϵ	36.6%
	ϵ:w W:ɪ i:l l:ϵ l:ɪ i:ŋ n:ϵ g:h h:ə a:ϵ ϵ:m m:ϵ	7.4%
/gezɪ/ ghezzi	ϵ:g g:ϵ h:e e:z z:ϵ ɪ:z i:ϵ	98.8%
	ϵ:g g:e h:ϵ e:z z:ɪ z:ϵ i:ϵ	1.2%

Table 5: Most probable paths from $\mathbf{x} \circ \mathcal{T} \circ \mathbf{y}$ under the approximate posterior distribution.

afterwards. However, we empirically find that there is performance gain by explicitly modeling past alignments. This also shows up in our preliminary experiments with non-input-feeding seq2seq models, which resulted in about 1% of lowered accuracy and about 0.1 longer edit distance.

C.2 Interpretability of learned NFST paths

The model is not required to learn transduction rules that conform to our linguistic knowledge. However, we expect that a well-performing one would tend to pick up rules that resemble what we know. To verify this, we obtain samples (listed in Table 4) from $\hat{p}(\mathbf{a} \mid \mathbf{x}, \mathbf{y})$ using the importance sampling algorithm described in §3.3. We find that our NFST model has learned to align phonemes and graphemes, generating them alternately. It has no problem picking up obvious pairs in the English orthography (e.g. (ʃ, c h), and (ŋ, n g)). We also find evidence that the model has picked up how context affects alignment: for example, the model has learned that the bigram 'gh' is pronounced differently in different contexts: in 'onslau**gh**t,' it is aligned with ɔ in the sequence 'augh;' in 'Willin**gh**am,' it spans over two phonemes ŋ h; and in '**gh**ezzi,' it is aligned with the phoneme g. We also find that our NFST has no problem learning phoneme-grapheme alignments that span over two arcs, which is beyond the capability of ordinary WFSTs.

D Algorithms and Full Tables

Algorithm 1 Compute approximate gradient for updating $G_{\boldsymbol{\theta}}$

```
Require: $G_{\boldsymbol{\theta}} : \mathcal{A} \to \mathbb{R}$ is an NFST scoring function, $q$ is a distribution over paths, $M \in \mathbb{N}$ is the sample size
 1: function GET-GRADIENT($G_{\boldsymbol{\theta}}$, $M$, $q$)
 2:     for $m$ in $1 \ldots M$ do
 3:         $\mathbf{a}^{(m)} \sim q$
 4:         $\tilde{w}^{(m)} \leftarrow \frac{\exp G_{\boldsymbol{\theta}}(\mathbf{a}^{(m)})}{q(\mathbf{a})}$
 5:     end for
 6:     $\hat{Z} \leftarrow \sum_{m=1}^{M} \tilde{w}^{(m)}$
 7:     for $m$ in $1 \ldots M$ do
 8:         $w^{(m)} \leftarrow \frac{\tilde{w}^{(m)}}{\hat{Z}}$
 9:     end for
10:     return $-\sum_{m=1}^{M} w^{(m)} \nabla_{\boldsymbol{\theta}} G_{\boldsymbol{\theta}}(\mathbf{a}^{(m)})$
11: end function
```

Algorithm 2 Training procedure for $G_{\boldsymbol{\theta}}$.

```
Require: $(\mathcal{T}, G_{\boldsymbol{\theta}})$ is an NFST, $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1) \ldots (\mathbf{x}_{|\mathcal{D}|}, \mathbf{y}_{|\mathcal{D}|})\}$ is the training dataset, LR : $\mathbb{N} \to \mathbb{R}$ is a learning rate scheduler, $\boldsymbol{\theta}_0$ are the initial parameters of $G_{\boldsymbol{\theta}}$, $M$ is a given sample size, maxEpoch $\in \mathbb{N}$ is the number of epochs to train for
 1: procedure TRAIN($\mathcal{T}$, $G_{\boldsymbol{\theta}}$, $\mathcal{D}$, LR, $\boldsymbol{\theta}_0$, $M$, maxEpochs)
 2:     for epoch $\in [1 \ldots \mathrm{maxEpochs}]$ do
 3:         for $(\mathbf{x}_i, \mathbf{y}_i) \in \mathrm{shuffle}(\mathcal{D})$ do
 4:             $\mathcal{T}' \leftarrow \mathbf{x}_i \circ \mathcal{T} \circ \mathbf{y}_i$
 5:             Construct distribution $q(\cdot \mid \mathcal{T}')$ according to equation (8)
 6:             $\mathbf{u} \leftarrow$ GET-GRADIENT$(G_{\boldsymbol{\theta}}, M, q)$ (listed in Algorithm 1)
 7:             $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mathrm{LR}(\mathrm{epoch}) \times \mathbf{u}$
 8:             (Optional) update the parameters of $q(\cdot \mid \mathcal{T}')$.
 9:         end for
10:     end for
11: end procedure
```

	Exact Match						Edit Distance					
	Dev			Test			Dev			Test		
	P2G	G2P	Avg.	P2G	G2P	Avg.	P2G	G2P	Avg.	P2G	G2P	Avg.
AP	17.0	39.4	28.2	18.6	37.8	28.2	1.84	1.186	1.513	1.782	1.152	1.467
Reranking AP	21.2	44.2	32.7	22.6	41.0	31.8	1.604	1.034	1.319	1.624	1.04	1.332
Reranking External	22.2	44.4	33.3	24.2	41.2	32.7	1.606	0.988	1.297	1.59	1.006	1.298
Reranking AP + External	22.0	42.0	32.0	24.0	40.0	32.0	1.608	1.01	1.309	1.588	1.018	1.303

Table 6: Comparison of decoding methods on G2P and P2G. Exact match % accuracy (higher is better) and edit distance (lower is better).

	Exact Match						Edit Distance					
	Dev			Test			Dev			Test		
	P2G	G2P	Avg.	P2G	G2P	Avg.	P2G	G2P	Avg.	P2G	G2P	Avg.
-IPA -PHONE	21.2	42.4	31.8	21.2	37.4	29.3	1.708	1.052	1.38	1.618	1.128	1.373
+IPA -PHONE	20.4	42.2	31.3	20.2	38.2	29.2	1.692	1.042	1.367	1.714	1.148	1.431
+IPA +PHONE	21.2	44.2	32.7	22.6	41.0	31.8	1.604	1.034	1.319	1.624	1.04	1.332

Table 7: Exact match % accuracy (higher is better) and edit distance (lower is better) on G2P and P2G. The effectiveness of different FST designs.