Appendices

A Proof of Proposition 11

Proof. Let’s consider a single-layer QRNN with 2-window convolutions:
\[ f_t = \sigma(V_f v_{t-1} + W_f v_t + b_f) , \]
\[ u_t = (1 - f_t) \odot g(V_u v_{t-1} + W_u v_t + b_u) , \]
\[ c_t = f_t \odot c_{t-1} + u_t . \]

A similar analysis applies to T-GRUs and T-LSTMs directly, and it should be straightforward
to generalize the discussion to QRNNs with larger convolution windows.

Let \( \Sigma \) denote the alphabet, and let \( x = x_1 x_2 \ldots x_n \in \Sigma^+ \) be a nonempty input string.
Consider a WFSA over the real semiring with \( 2|\Sigma| + 1 \) states, where \( q_0 \) is the initial state with \( \lambda(q_0) = 1 ; |\Sigma| \) of them are final states \( Q_2 = \{ q_{a,1} \in \Sigma \} \), with \( \rho(q_{a,1}) = 1 \), and the remaining \( |\Sigma| \) states are denoted by \( Q_1 = \{ q_{a} \in \Sigma \} \).

The transition weights \( \tau \) are constructed by

\[
\tau(q_0, p_a, \alpha) = 1 , \quad \forall p_a \in Q_1 ; \\
\tau(p_a, p_b, \beta) = 1 , \quad \forall p_a, p_b \in Q_1 ; \\
\tau(p_a, q_b, \beta) = \mu_a(\beta) , \quad \forall p_a \in Q_1 , \forall q_b \in Q_2 ; \\
\tau(q_a, q_b, \beta) = \phi_a(\beta) , \quad \forall q_a, q_b \in Q_2 .
\]

\( \tau = 0 \) otherwise. Then one dimension of the recurrent updates of a 2-window QRNN is recovered by parameterizing the weight functions as

\[
\mu_{x_{i-1}}(x_t) = [u_t]_i , \quad \phi_{x_{i-1}}(x_t) = [f_t]_i . \quad (22)
\]

The recurrent computation of a 2-window QRNN of hidden size \( d \) can then be recovered by collecting \( d \) such WFSAs.

B Proof of Proposition 12

Proof. We present the construction of WFSAs for a single layer \( n \)-gram RCNNs of hidden size \( d \).

Let’s assume a given input sequence \( x \in \Sigma^+ \), with \( |x| > n \), since otherwise one only needs include paddings, just as in a RCNN. Consider a WFSA with \( n + 1 \) states \( Q = \{ q_i \}_{i=0}^n \) over the real semiring. Use \( q_0 \) as the initial state with \( \lambda(q_0) = 1 \), and \( q_n \) as the final state with \( \rho(q_n) = 1 \). The transition weight function is defined by

\[
\tau(q_i, q_j, \alpha) = \begin{cases} 
1, & i = j = 0 , \\
\phi(\alpha), & j = i > 0 , \\
\mu_j(\alpha), & j = i + 1 , \\
0, & \text{otherwise}.
\end{cases}
\]

Let \( z_t^{(j)} \) denote the total score of all paths landing in state \( q_j \) just after consuming \( x_t \). Let \( z_t^{(j)} = 0, j = 0, \ldots, n \). By the forward algorithm

\[
z_t^{(0)} = 1 , \\
z_t^{(j)} = z_{t-1}^{(j)} \phi(x_t) + z_{t-1}^{(j-1)} \mu_j(x_t) , \quad j \geq 1 .
\]

Applying similar parametrization to that in \( 4.2 \), \( z_t^{(n)} \) recovers one dimension of the recurrence. Collecting \( d \) such WFSAs we recover the recurrence of a single layer \( n \)-gram RCNNs, with \( \lambda_t \) being a constant, or depending only on \( x_t \).

C Proof of Proposition 13

Proof. Closely following the 2-dimensional case in \( 4.3 \), let’s discuss a single layer ISAN of hidden size \( d \).

Consider a WFSA over the real semiring with \( 2d \) states. Let \( d \) of them, denoted by \( Q_2 = \{ q_i \}_{i=d+1}^{2d} \) be the initial states, with \( \lambda(q_i) = 1 , i = 1, \ldots, d \). Denote the other half \( \{ q_i \}_{i=1}^d \) by \( Q_1 \). Define transition weight \( \tau \) by:

\[
\tau(q_i, q_j, \alpha) = \begin{cases} 
1, & i = j = d , \\
\eta_j(\alpha), & i = j + d , \\
\mu_{j,i}(\alpha), & q_i, q_j \in Q_1 , \\
0, & \text{otherwise}.
\end{cases}
\]

\( \forall \alpha \in \Sigma \). Using \( q_i \in Q_1 \) as the final state with \( \rho(q_i) = 1 \), and denote the resulting WFSA by \( G_i \). By Forward algorithm, \( G_i \) recovers the \( i \)-th dimension of the single layer ISAN by letting \( [W_{x_i}]_{i,j} = \mu_{i,j}(x_i) \), and \( [b_{x_i}]_i = \eta_i(x_i) \); the \( d \)-dimensional recurrent computation is recovered by a set of WFSAs \( \{ G_i \}_{i=1}^d \) constructed similarly.

D Compared Models

This section formally describes the models compared in the experiments (\( 6.1 \)).

\( \text{RRNN}(\mathbb{B}) \). \( \text{RRNN}(\mathbb{B}) \) is derived from \( \mathbb{B} \) (\( 4.1 \)).

\[
f_t = \sigma(W_f v_t + b_f) , \quad (23a)
\]
\[
u_t = (1 - f_t) \odot W_u v_t , \quad (23b)
\]
\[
c_t = f_t \odot c_{t-1} + u_t , \quad (23c)
\]
\[
o_t = \sigma(W_o v_t + b_o) , \quad (23d)
\]
\[
h_t = \tanh(o_t \odot c_t) . \quad (23e)
\]
\textbf{RRNN(2)ₘₖₜ.} Also derived from \textit{B}, but uses the max-plus semiring (§5.2).

\begin{align*}
    \mathbf{f}_t &= \log \sigma(\mathbf{W}_f \mathbf{v}_t + \mathbf{b}_f), \\
    \mathbf{u}_t &= \mathbf{W}_u \mathbf{v}_t, \\
    \mathbf{c}_t &= \max\{\mathbf{f}_t + \mathbf{c}_{t-1}, \mathbf{u}_t\}, \\
    \mathbf{o}_t &= \log \sigma(\mathbf{W}_o \mathbf{v}_t + \mathbf{b}_o), \\
    \mathbf{h}_t &= \tanh(\mathbf{o}_t + \mathbf{c}_t).
\end{align*}

\textbf{RRNN(ℂ).} \textbf{RRNN(ℂ)} is derived from \textit{ℂ} (§4.2):

\begin{align*}
    f_t^{(j)} &= \sigma(\mathbf{W}_f^{(j)} \mathbf{v}_t + \mathbf{b}_f^{(j)}), \quad j = 1, 2, \\
    u_t^{(j)} &= (1 - f_t^{(j)}) \circ \mathbf{W}_u^{(j)} \mathbf{v}_t, \quad j = 1, 2, \\
    c_t^{(1)} &= c_{t-1}^{(1)} \circ f_t^{(1)} + u_t^{(1)}, \\
    c_t^{(2)} &= c_{t-1}^{(2)} \circ f_t^{(2)} + c_t^{(1)} \circ u_t^{(2)}, \\
    o_t &= \sigma(\mathbf{W}_o \mathbf{v}_t + \mathbf{b}_o), \\
    h_t &= \tanh(o_t + c_t).
\end{align*}

\textbf{RRNN(ℍ).} Derived from \textit{ℍ} (§5.1).

\begin{align*}
    f_t^{(j)} &= \sigma(\mathbf{W}_f^{(j)} \mathbf{v}_t + \mathbf{b}_f^{(j)}), \quad j = 1, 2, \\
    u_t^{(j)} &= (1 - f_t^{(j)}) \circ \mathbf{W}_u^{(j)} \mathbf{v}_t, \quad j = 1, 2, \\
    p_t^{(j)} &= \sigma(\mathbf{b}_p^{(j)}), \quad j = 1, 2, \\
    \mathbf{r} &= \sigma(\mathbf{b}_r), \\
    c_t^{(1)} &= c_{t-1}^{(1)} \circ f_t^{(1)} + u_t^{(1)}, \\
    c_t^{(2)} &= c_{t-1}^{(2)} \circ f_t^{(2)} + (c_t^{(1)} + \mathbf{r}) \circ u_t^{(2)}, \\
    c_t &= p_t^{(1)} \circ c_t^{(1)} + p_t^{(2)} \circ c_t^{(2)}, \\
    o_t &= \sigma(\mathbf{W}_o \mathbf{v}_t + \mathbf{b}_o), \\
    h_t &= \tanh(o_t + c_t).
\end{align*}

The output gates (Equations 23d, 24d, 25e, and 26h) are optional. They are only used in language modeling experiments, where we empirically find that they improve performance.

E Experimental Setup

E.1 Implementation Details


E.2 Language Modeling

For hyperparameters, we do not deviate much from the language modeling experiments in Lei et al. (2017b). We change the hidden sizes for all compared models based on the trainable parameter budget, and adjust the dropout probabilities accordingly to keep the number of remaining hidden units is roughly the same in expectation. Besides, we observe that RRNN(ℂ) and RRNN(ℍ) fail to converge when optimized with the SGD algorithm using 1.0 initial learning rate. And thus we use 0.5 for both models. Other hyperparameters are kept the same as Lei et al. (2017b).

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Type} & \textbf{Values} \\
\hline
Hidden size & [100, 300] \\
Vertical dropout & [0.0, 0.5] \\
Recurrent dropout & [0.0, 0.5] \\
Embedding dropout & [0.0, 0.5] \\
Learning Rate & $[10^{-2}, 10^{-4}]$ \\
$\ell_2$ regularization & $[10^{-5}, 10^{-7}]$ \\
Gradient Clipping & [1.0, 5.0] \\
\hline
\end{tabular}
\caption{The hyperparameters explored using random search algorithm in the text classification experiments.}
\end{table}

E.3 Text classification

We train our models using Adam (Kingma and Ba, 2015) with a batch size of 16 (for Amazon) or 64 (for the smaller datasets). Initial learning rate and $\ell_2$ regularization are hyperparameters. We use 300-dimensional GloVe 840B embeddings (Pennington et al., 2014) normalized to unit length and fixed, replacing unknown words with a special UNK token. Two layer RNNs are used in all cases. For regularization, we use three types of dropout: a recurrent variational dropout, vertical dropout and a dropout on the embedding layer.

We tune the hyperparameters of our model on the development set by running 20 epochs of random search. We then take the best development results. The hyperparameters values explored are summarized in Table 6. We train all models for 500 epochs, stopping early if development accuracy does not improve for 30 epochs. During training, we halve the learning rate if development accuracy does not improve for 10 epochs.