Cold-Start and Interpretability: Turning Regular Expressions into Trainable Recurrent Neural Networks

Chengyue Jiang , Yinggong Zhao, Shanbo Chu , Libin Shen , Kewei Tu *

School of Information Science and Technology, ShanghaiTech University

Leyan Technologies

上海科技大学 ShanghaiTech University

乐言科技 Leyan Technologies

Symbolic rules vs. neural networks

Neural Network (NN)

- have good performance after trained on sufficient training data
- hard to interpret the results.

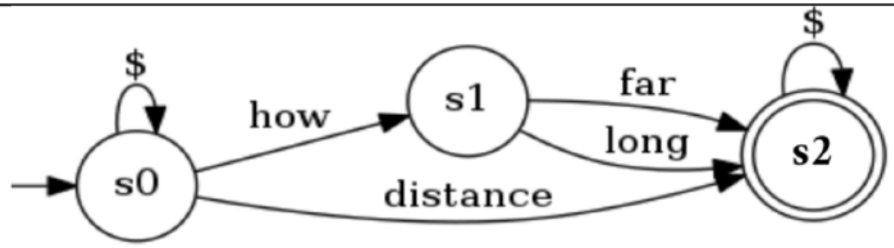
Rule based systems

- Highly interpretable, support fine-grained human inspection and manipulation, no training needed.
- Can not learn from data. Sometimes it’s hard to write rules.

Symbolic rules vs. neural networks

- Regular expressions (RE) are one of the most representative and useful forms of symbolic rules.
- RE are widely used for solving tasks such as pattern matching and intent classification.
- We aim to combine the advantages of NN and rules, by directly turning a RE-based system into a NN.

Background – Regular Expressions and FA

<table>
<tr><td>Label</td><td>[distance]</td></tr>
<tr><td>RE</td><td>$*(how (far | long) | distance) $*</td></tr>
<tr><td>Matched Text</td><td>⟨BOS⟩ tell me how far is oakland airport from downtown ⟨EOS⟩</td></tr>
<tr><td>FA</td><td>
</td></tr>
</table>

RE matches string $\boldsymbol{x}$

⇕

In the corresponding automaton, there exists at least a path from the start state to one of the final states after reading $\boldsymbol{x}$

Representing FA

Vocabulary size: V

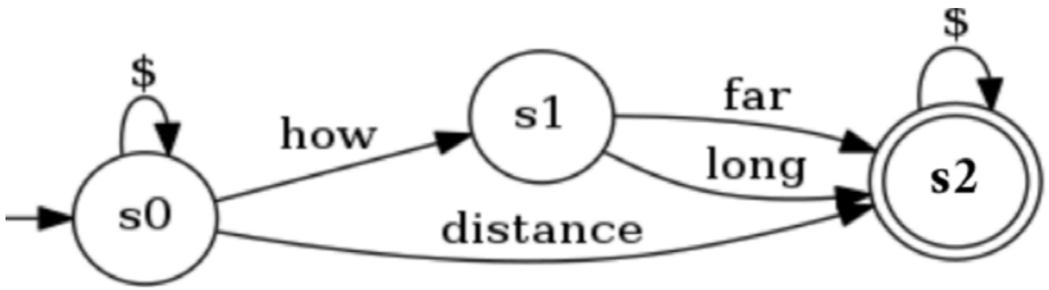
State size: K

One-hot transition tensor: $T \in \mathbb{R}^{V \times K \times K}$

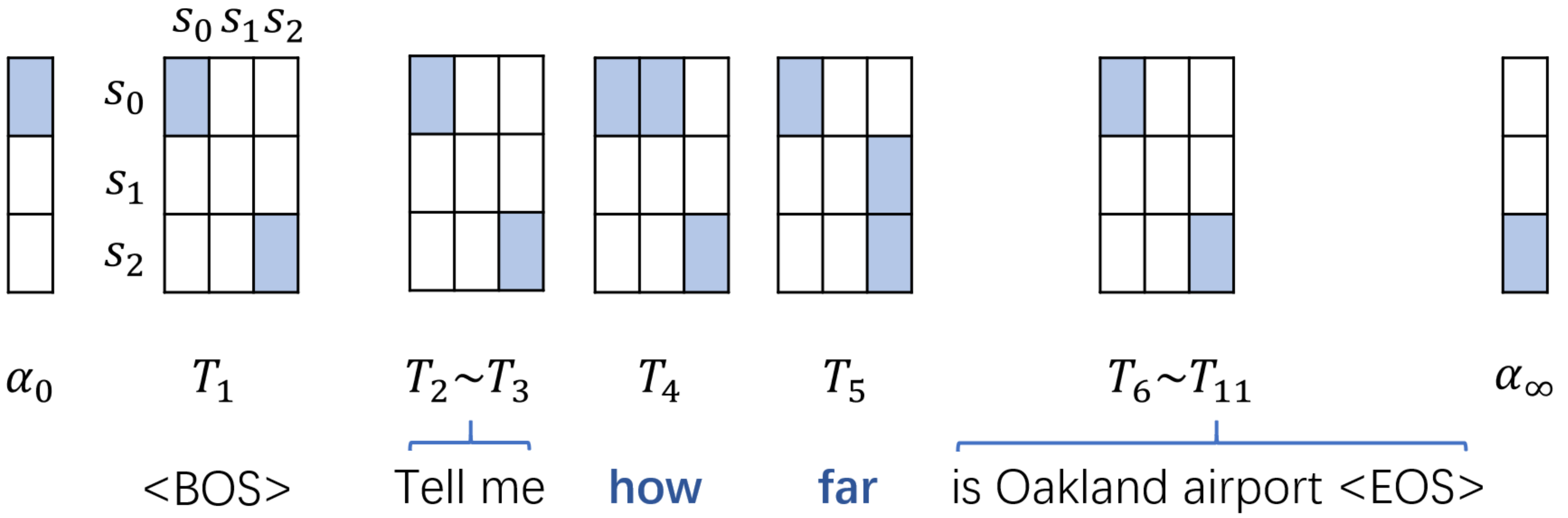
One-hot start vector: $\alpha_0 \in \mathbb{R}^K$

One-hot final vector: $\alpha_\infty \in \mathbb{R}^K$

Running a FA – Forward Algorithm



$$\boldsymbol{\alpha}_0^T \cdot \left(\prod_{i=1}^{N} \boldsymbol{T}[x_i] \right) \cdot \boldsymbol{\alpha}_\infty$$



RUN FA – Forward Algorithm

$$\boldsymbol{\alpha}_0^T \cdot \left(\prod_{i=1}^{N} \boldsymbol{T}[x_i] \right) \cdot \boldsymbol{\alpha}_\infty$$

$$\downarrow$$

$$\boldsymbol{h}_0 = \boldsymbol{\alpha}_0^T$$

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} \cdot \boldsymbol{T}[x_t],\ 1 \leq t \leq N$$

Pick the transition matrix

$$\mathcal{B}_{\text{forward}}(\mathcal{A}, \boldsymbol{x}) = \boldsymbol{h}_N \cdot \boldsymbol{\alpha}_\infty$$

Meaning of $\boldsymbol{h_t}[\boldsymbol{i}]$:

After reading $\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_t}$, the total number of paths from start state to state $\boldsymbol{i}$.

Meaning of $\boldsymbol{h_N}[\boldsymbol{j}],\ \boldsymbol{s_j} \in \boldsymbol{S_\infty}$

The number of paths from start state to each final states.

FA-RNN (I) Reducing model parameter size using Tensor Decomposition

Tensor Rank Decomposition

$$\boldsymbol{T} \in \mathbb{R}^{V \times K \times K} \longrightarrow \boldsymbol{E}_{\mathcal{R}} \in \mathbb{R}^{V \times r}, \boldsymbol{D}_1 \in \mathbb{R}^{K \times r}, \boldsymbol{D}_2 \in \mathbb{R}^{K \times r}$$

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} \cdot \boldsymbol{T}[x_t] \longrightarrow \begin{aligned} \boldsymbol{v_t} &= \boldsymbol{E_R}(\boldsymbol{x_t}) \\ \boldsymbol{a} &= (\boldsymbol{h}_{t-1} \cdot \boldsymbol{D}_1) \circ \boldsymbol{v}_t \\ \boldsymbol{h}_t &= \boldsymbol{a} \cdot \boldsymbol{D}_2^T \end{aligned}$$

FA-RNN (II) Integrating Word Vectors to Inject Word Information.

$\boldsymbol{E}_w \in \mathbb{R}^{V \times D}$ Word Embedding Matrix

$\boldsymbol{u}_t \in \mathbb{R}^D$ Word Vector

$\beta \in [0, 1]$ Balancing Constant

$\boldsymbol{G} \in \mathbb{R}^{D \times r}$ Projection Matrix from D (embedding dim) to r (rank)

$\boldsymbol{G} = \boldsymbol{E}_w^{\dagger} \boldsymbol{E}_{\mathcal{R}}$

$\boldsymbol{u}_t \boldsymbol{G} \rightarrow \boldsymbol{v}_t$

$\mathbf{G}$ is initialized based on the intuition that we want the projected vectors to approximate $\boldsymbol{v}_t$

$$\boldsymbol{a} = (\boldsymbol{h}_{t-1} \cdot \boldsymbol{D}_1) \circ \boldsymbol{v}_t$$
$$\boldsymbol{h}_t = \boldsymbol{a} \cdot \boldsymbol{D}_2^T$$

$\Rightarrow$

$$\boldsymbol{z}_t = \beta \boldsymbol{v}_t + (1 - \beta) \boldsymbol{u}_t \boldsymbol{G}$$
$$\boldsymbol{a} = (\boldsymbol{h}_{t-1} \cdot \boldsymbol{D}_1) \circ \boldsymbol{z}_t$$
$$\boldsymbol{h}_t = \boldsymbol{a} \cdot \boldsymbol{D}_2^T$$

FA-RNN (III) Gated Variants

Add forget gate and reset gate like GRU, initialize them to $\mathbf{1}$

$$\boldsymbol{z}_t = \beta \boldsymbol{v}_t + (1-\beta)\boldsymbol{u}_t \boldsymbol{G}$$
$$\boldsymbol{a} = (\boldsymbol{h_{t-1}} \cdot \boldsymbol{D}_1) \circ \boldsymbol{z}_t$$
$$\boldsymbol{h_t} = \boldsymbol{a} \cdot \boldsymbol{D}_2^T$$

$$\boldsymbol{z}_t = \beta \boldsymbol{v}_t + (1-\beta)\boldsymbol{u}_t \boldsymbol{G}$$
$$\boldsymbol{f}_t = \boldsymbol{\sigma}(\boldsymbol{W}_f \boldsymbol{z}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f)$$
$$\boldsymbol{r}_t = \boldsymbol{\sigma}(\boldsymbol{W}_r \boldsymbol{z}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r)$$
$$\hat{\boldsymbol{h}}_{t-1} = (1-\boldsymbol{r}_t) \circ \boldsymbol{h}_0 + \boldsymbol{r_t} \circ \boldsymbol{h_{t-1}}$$
$$\boldsymbol{a} = (\hat{\boldsymbol{h}}_{\boldsymbol{t-1}} \cdot \boldsymbol{D}_1) \circ \boldsymbol{z}_t$$
$$\hat{\boldsymbol{h}}_t = \boldsymbol{a} \cdot \boldsymbol{D}_2^T$$
$$\boldsymbol{h}_t = (1-\boldsymbol{f}_t) \circ \boldsymbol{h}_{t-1} + \boldsymbol{f}_t \circ \hat{\boldsymbol{h}}_t$$

FA-RNN (IV) Bidirectional Variants

We reverse the RE

free $* (phone | phones) $*

$* (phone | phones) $* free

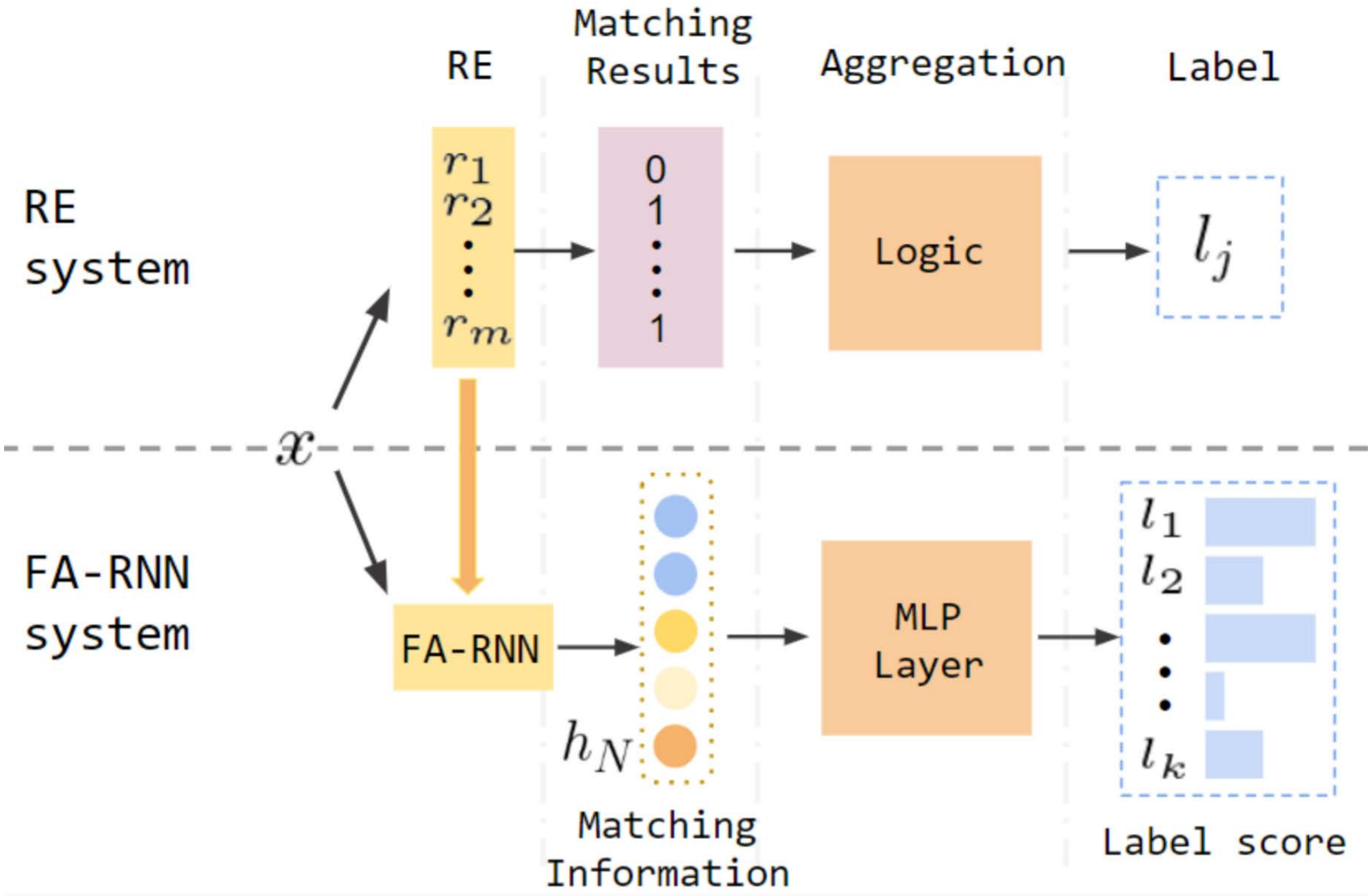
Convert to FA-RNN and feed in the reversed input to obtain

$$\overleftarrow{\boldsymbol{h_t}}$$

Averaging the forward and backward hidden states.

$$(\overrightarrow{\boldsymbol{h_t}} + \overleftarrow{\boldsymbol{h_t}})/2$$

RE-System => FA-RNN system (I)



Propositional Logic:
AND/OR/NOT

Logic	Soft Logic
$\neg A$	$1 - a$
$A \vee B$	$\min(1, a + b)$
$A \wedge B$	$\max(0, a + b - 1)$

Use soft logic to construct MLP layer

RE-System => FA-RNN system (II) Training

- Feed logits into CrossEntropy loss function and optimize with Adam optimizer.
- We use fixed $\boldsymbol{E_R}$, so FA-RNN has comparable model parameters to traditional RNNs.

Experiments (I) Datasets

- 3 intent classification datasets: ATIS, QC(TREC-6), SMS.
- Different settings: zero-shot/low-resource/rich-resource

	#Train	#Dev	#Test	$\|\mathcal{L}\|$	$\|\mathcal{R}\|$	K	%Acc
ATIS	3982	996	893	26	27	107	87.0
	*$ * flights | flight | ((go | get | fly) from $ * to $ *) $ * → FLIGHT*						
QC	4965	500	500	6	68	94	64.4
	*$ * what $? does $+ (stand? for) $* → ABBREVIATION*						
SMS	4502	500	500	2	36	52	93.2
	$ free $ * (phone | phones) $* → SPAM*						

Experiments (II) Baselines: NNs and Rule Enhanced NNs

- Bi-(RNN/GRU/LSTM)/CNN/DAN + Linear + CE
- Enhancement by RE parsed results. (+i, +o, +io) [Luo et al., 2016]
- Knowledge Distillation. (+pr, +kd)

  [Hu et al,. 2016; Hinton et al,.2015]

	ATIS	QC	SMS
RE system	87.01	64.40	93.20
FA-RNN	86.53	61.95	93.00
FA-GRU	86.81	62.90	93.20
BiFA-RNN	88.10	62.90	93.00
BiFA-GRU	88.63	62.90	93.20
BiGRU+i	1.34	18.75	11.90
BiGRU+o	30.74	27.50	30.40
BiGRU+io	38.69	25.70	73.25
BiGRU+i+u	86.42	64.85	92.75
BiGRU+o+u	83.03	64.95	93.05
BiGRU+io+u	86.14	64.75	92.70

Results (I)
Zero-shot

Results (II)
Low-resource and full dataset

	ATIS (26-class)			QC (6-class)			SMS (2-class)		
	1%	10%	100%	1%	10%	100%	1%	10%	100%
FA-RNN	90.43	90.79	96.52	**67.75**	79.6	91.3	93.1	96.75	98.8
FA-GRU	88.94	**90.85**	96.61	66.2	80.7	91.85	**94.25**	**96.8**	**99.2**
BiFA-RNN	89.31	**90.85**	96.72	57.65	81.5	91.55	91.7	96.7	99
BiFA-GRU	**90.62**	90.26	96.64	64.15	**82.8**	**92.4**	93.9	96.75	98.8
CNN	71.61	86.09	94.74	50.9	74.9	89.25	89.85	95.9	98.8
DAN	71.02	83.68	90.4	47.25	65.4	77.8	89.9	93.7	98.6
RNN	70.91	75.17	91.55	22.4	67.9	85	85.1	89.85	97.75
LSTM	69.37	78.14	95.72	40.45	75.75	90	86.2	95.75	97.85
GRU	70.72	88.52	96.3	42.35	79.75	91.2	86.15	95.55	98.05
BiRNN	70.72	79.98	93.39	49.35	75.95	87.35	86.75	94.9	97.8
BiLSTM	70.77	87.12	96.25	55.95	76.75	90.95	92.15	95.8	97.7
BiGRU	70.69	88.35	96.75	62.7	80.05	91.5	89.6	95.95	98.4
BiGRU +i	82.84	90.01	96.56	66.3	80.25	92	90.95	96.75	98.55
BiGRU +o	80.21	89.22	96.33	60.15	80.2	91.7	90.6	95.95	98.4
BiGRU +io	82.61	89.95	95.46	65.05	79.65	90.7	93.85	96.75	98.25
BiGRU +pr	72.4	88.89	96.5	61.6	80.45	91.85	90.9	96.05	98.45
BiGRU +kd	73.38	88.86	**96.75**	62.65	80.3	91.25	87.65	96	98.55

Analyze (I) Ablation

FA-RNN	ATIS	QC	SMS
-F	**96.52**	**91.30**	98.80
-V	95.66	88.20	97.85
-F-O	94.51	87.80	**99.20**
-F-Rand	92.16	80.60	95.40
-V-Rand	91.26	78.60	97.00
-F-Rand$\boldsymbol{E}_w$	94.17	84.40	97.00
-Train$\boldsymbol{E}_{\mathcal{R}}$	96.41	89.20	99.00

Analyze (II) Integrating Word Embedding

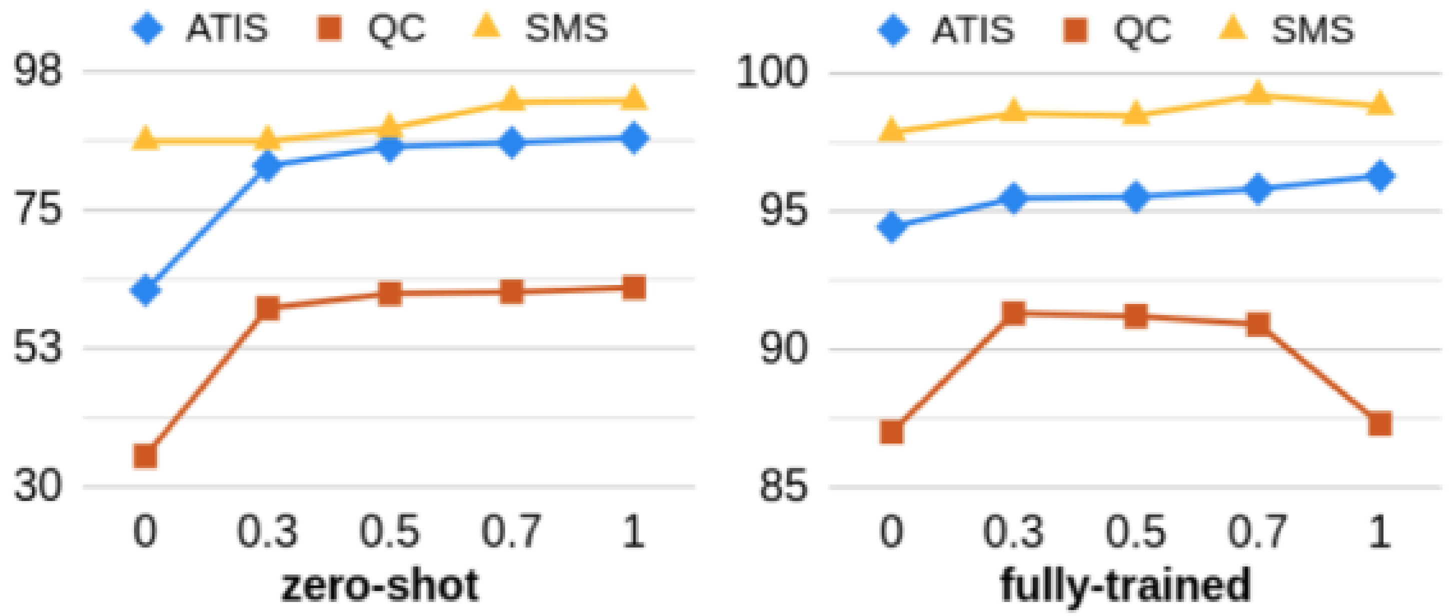


Figure 3: Performance of FA-RNN with different β

Interpretability (I) Convert FA-RNN back to WFA

Model parameters after training

$$\Theta_{\text{RE}} = \left\langle \hat{\boldsymbol{E}}_{\mathcal{R}}, \hat{\boldsymbol{D}}_1, \hat{\boldsymbol{D}}_2, \hat{\boldsymbol{G}} \right\rangle \quad \boldsymbol{E}_w$$

Recover the WFA tensor from Model parameters

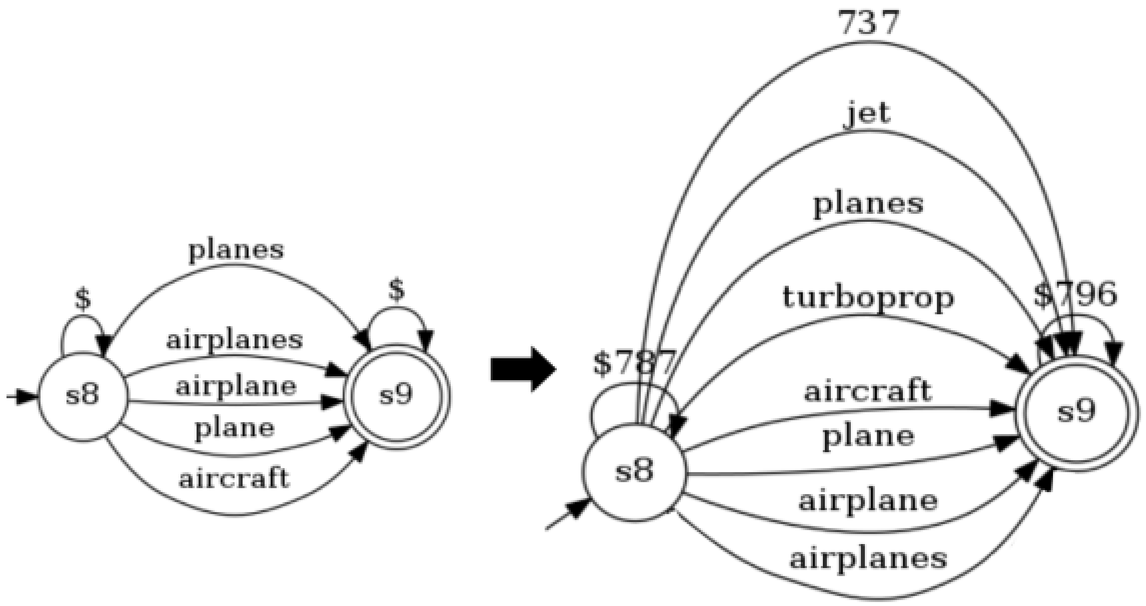
$$\hat{\boldsymbol{E}}_{w\mathcal{R}} = \beta \cdot \hat{\boldsymbol{E}}_{\mathcal{R}} + (1-\beta) \cdot \boldsymbol{E}_w \hat{\boldsymbol{G}}$$

$$\hat{\boldsymbol{T}}_{(1)} = (\hat{\boldsymbol{D}}_2 \odot \hat{\boldsymbol{D}}_1) \hat{\boldsymbol{E}}_{w\mathcal{R}}^T$$

Interpretability (III) Convert FA-RNN back to RE

We threshold the WFA tensor to obtain an NFA, and convert the NFA to RE.

Extracted RE vs original RE
ATIS +0.45%
QC +9.2%
SMS -1.2%



Conclusion

- We propose FA-RNN.
- It can be initialized from REs and learn from data
- It outperforms previous neural classification approaches in zero-shot and low-resource scenarios and is competitive in rich-resource scenarios
- It is also interpretable and can be converted back into REs