Extracurricular Materials

Neuralizing Regular Expressions

Papers

- Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu, "Cold-start and Interpretability: Turning Regular Expressions into Trainable Recurrent Neural Networks", EMNLP 2020.
- Chengyue Jiang, Zijian Jin, and Kewei Tu, "Neuralizing Regular Expressions for Slot Filling", EMNLP 2021.

Regular Expressions (RE)

- One of the most representative and useful forms of symbolic rules
- Widely used in practice: word tokenization, text classification, slot filling, etc.

Label	[*distance*]
RE	*$*(how (far | long) | distance) $**
Matched Text	⟨BOS⟩ tell me **how far** is oakland air-port from downtown ⟨EOS⟩

Regular Expressions (RE)

- Pros
  - Highly interpretable
  - Support fine-grained diagnosis and manipulation
    - Easy to add/delete/revise rules to quickly adapt to changes in task specification
  - No need for training
    - Hence no need for data annotation, less computational cost
    - Good for cold-start scenarios
- Cons
  - Rely on human experts to write
  - Often: high precision but low recall
  - Cannot evolve by training on labeled data when available
    - Underperform neural approaches in rich-resource scenarios

Our Idea

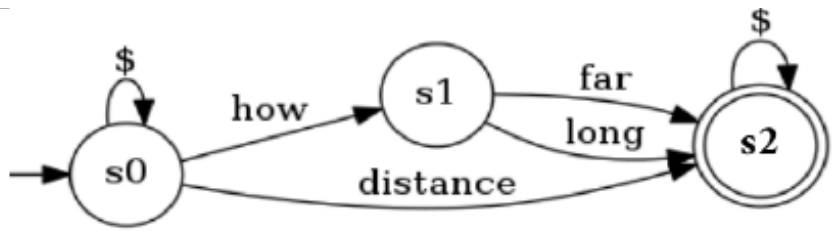
- Convert a RE to a new form of recurrent neural networks
  - Roughly equivalent to RE
    - ✓ Can still be used in cold-start scenarios
  - Trainable on labeled data
    - ✓ Can outperform REs and compete with neural approaches in rich-resource scenarios
  - Can be converted back to RE
    - ✓ Possibility of fine-grained manipulation

Step 1. RE to Finite Automaton (FA)

- Any RE can be converted into a FA that expresses the same language

Label	[*distance*]
RE	*$*(how (far | long) | distance) $**

FA



FA parameters

- Binary transition tensor:

$$T \in \mathbb{R}^{V \times K \times K}$$

- Binary start vector:

$$\alpha_0 \in \mathbb{R}^{K}$$

- Binary final vector:

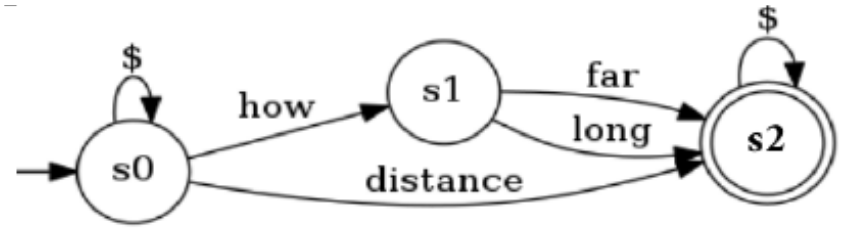
$$\alpha_\infty \in \mathbb{R}^{K}$$

V: vocabulary size

K: state number

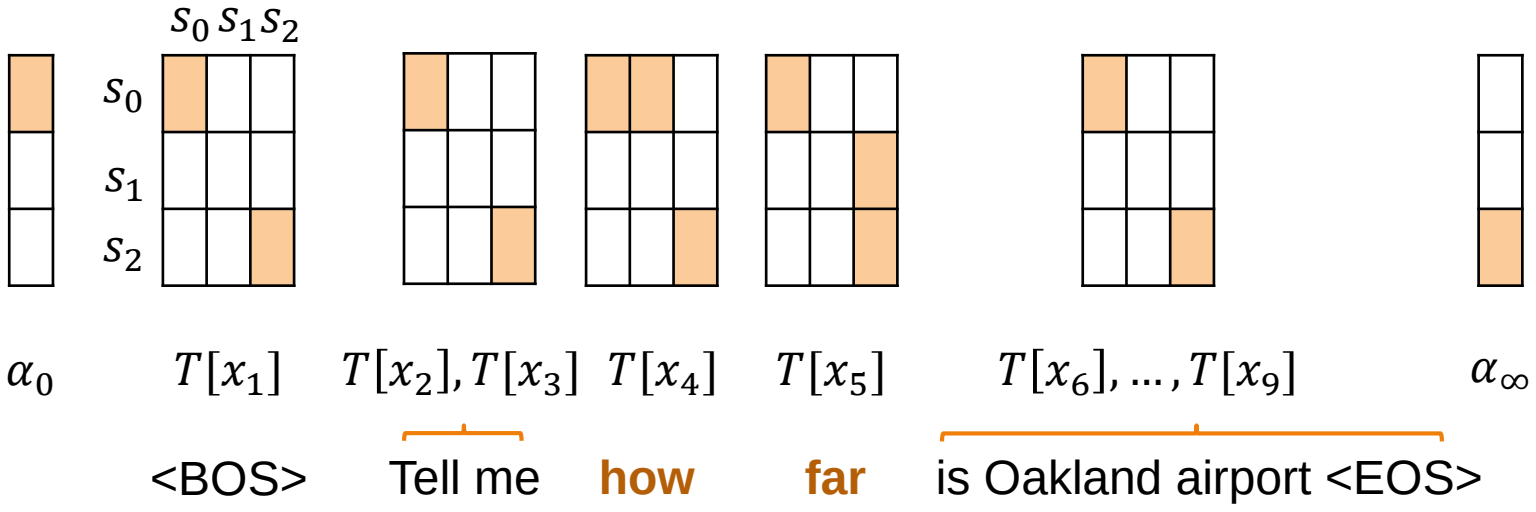
Step 2. FA as Recurrent Neural Network (RNN)

- Score of a FA accepting a sentence can be calculated using the forward algorithm



Forward score:

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



Step 2. FA as Recurrent Neural Network (RNN)

- The computation can be rewritten into a recurrent form

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

$$\boldsymbol{h_0} = \boldsymbol{\alpha}_0^T$$

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} \cdot \boldsymbol{T}[x_t],\ 1 \le t \le N \quad \text{(recurrent step)}$$

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

Step 3. Decomposing the Parameter Tensor

- Goal: reduce the computational complexity to match that of traditional RNN

Tensor Rank Decomposition

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

(word embedding) (state embeddings)

- Now the recurrent step becomes:

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

Step 4. Integrating Pretrained Word Embedding

- Goal: bringing external lexical knowledge into our model
- Method:
  - Approximate $\boldsymbol{E}_{\mathcal{R}}$ with $\boldsymbol{E}_w\boldsymbol{G}$

initialized with $\boldsymbol{E}_w^{\dagger}\boldsymbol{E}_{\mathcal{R}}$
$\boldsymbol{E}_w^{\dagger}$ is the pseudo-inverse of $\boldsymbol{E}_w$

external word embedding

  - Interpolate $\boldsymbol{E}_{\mathcal{R}}$ and $\boldsymbol{E}_w\boldsymbol{G}$
  - The recurrent step becomes:

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

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

FA-RNN

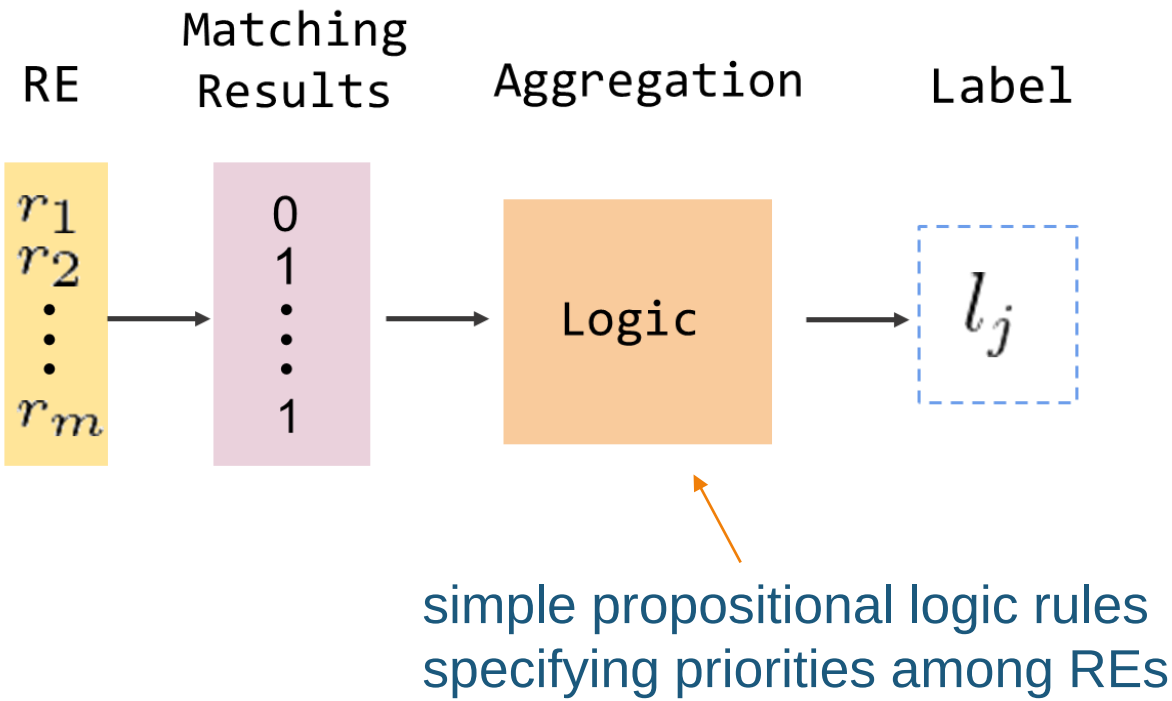
FA-RNN Extensions

- Gated extension
  - Add update gate and reset gate like in GRU
  - Initialize parameters to make the gates inactive initially

- Combine two FA-RNNs of opposite directions
  - Create a left-to-right FA-RNN from the RE
  - Create a right-to-left FA-RNN from the reversed RE
  - Output the average score of the two FA-RNNs

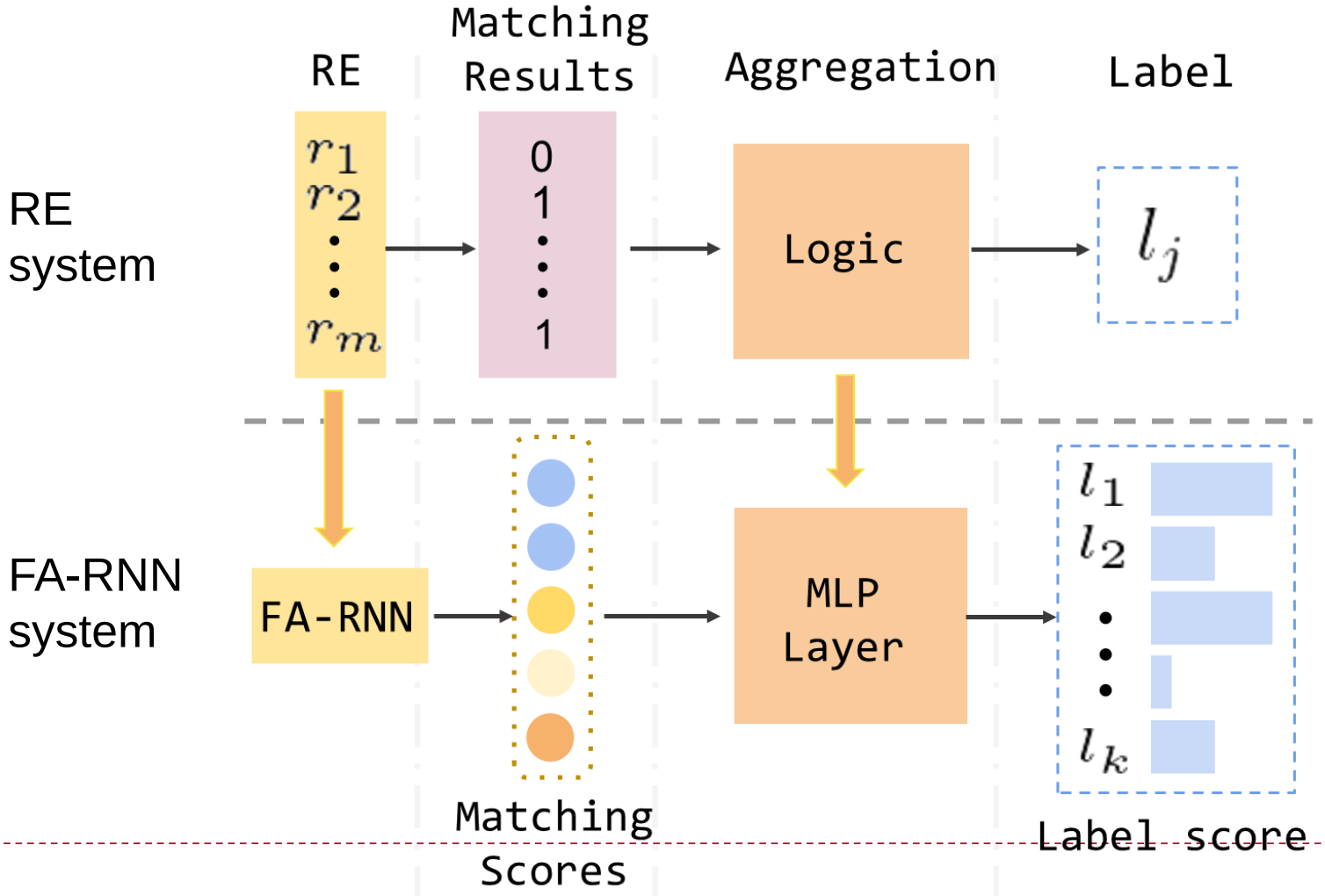
Text classification

- An RE system for text classification:
  - Aggregating results from multiple REs to form a prediction



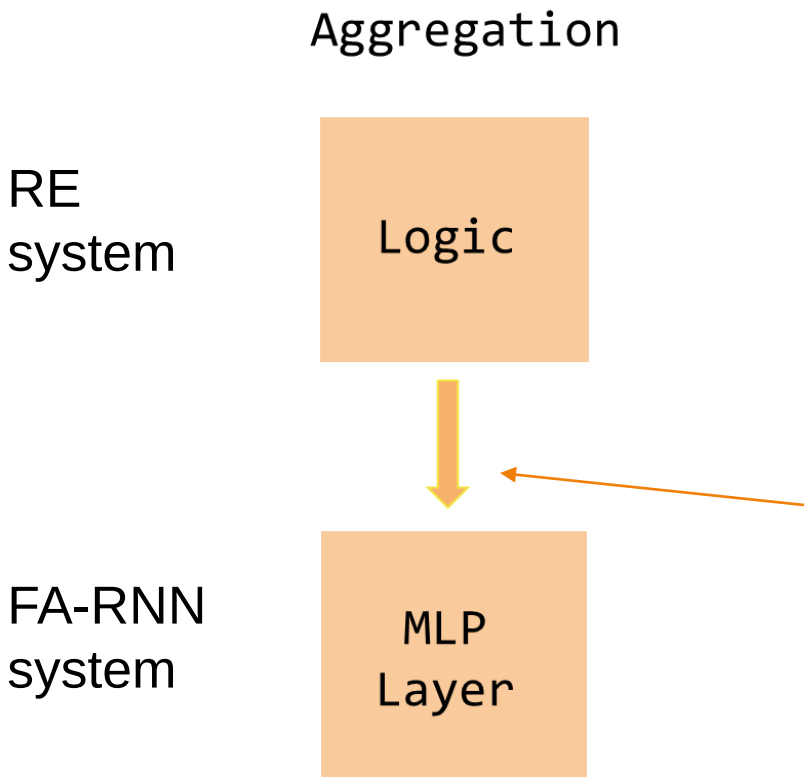
Text classification

- From a RE system to a FA-RNN system



Text classification

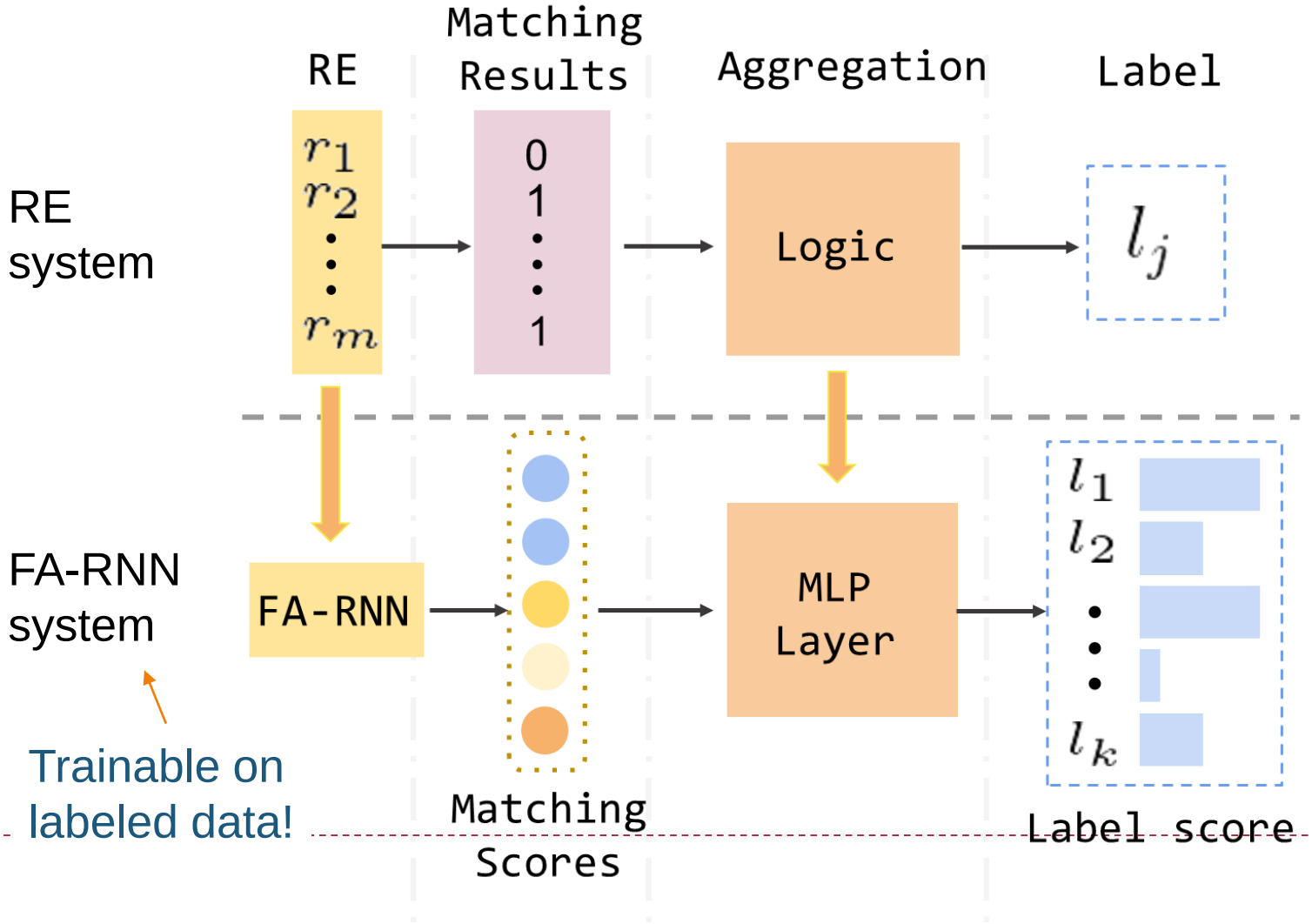
- From a RE system to a FA-RNN system



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

Text classification

- From a RE system to a FA-RNN system



Experiments

- Three intent classification datasets:
  - ATIS, QC (TREC-6), SMS
- Baselines
  - Bi-RNN/GRU/LSTM, CNN, DAN
  - RE-enhanced NN (+i, +o, +io) [Luo et al., 2016]
  - Knowledge Distillation (+kd, +pr) [Hinton et al,.2015; Hu et al,. 2016]

Experiments – Zero-Shot

	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+*pr*	9.94	17.70	53.00
BiGRU+*kd*	9.94	17.70	53.00

Experiments – Low-Resource and Full Training

	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

Summary

- Turning symbolic systems (RegExp) to neural networks
  - Combining strengths of symbolic rules and neural networks
  - Excels in zero-shot and low-resource scenarios
  - Competitive in rich-resource scenarios