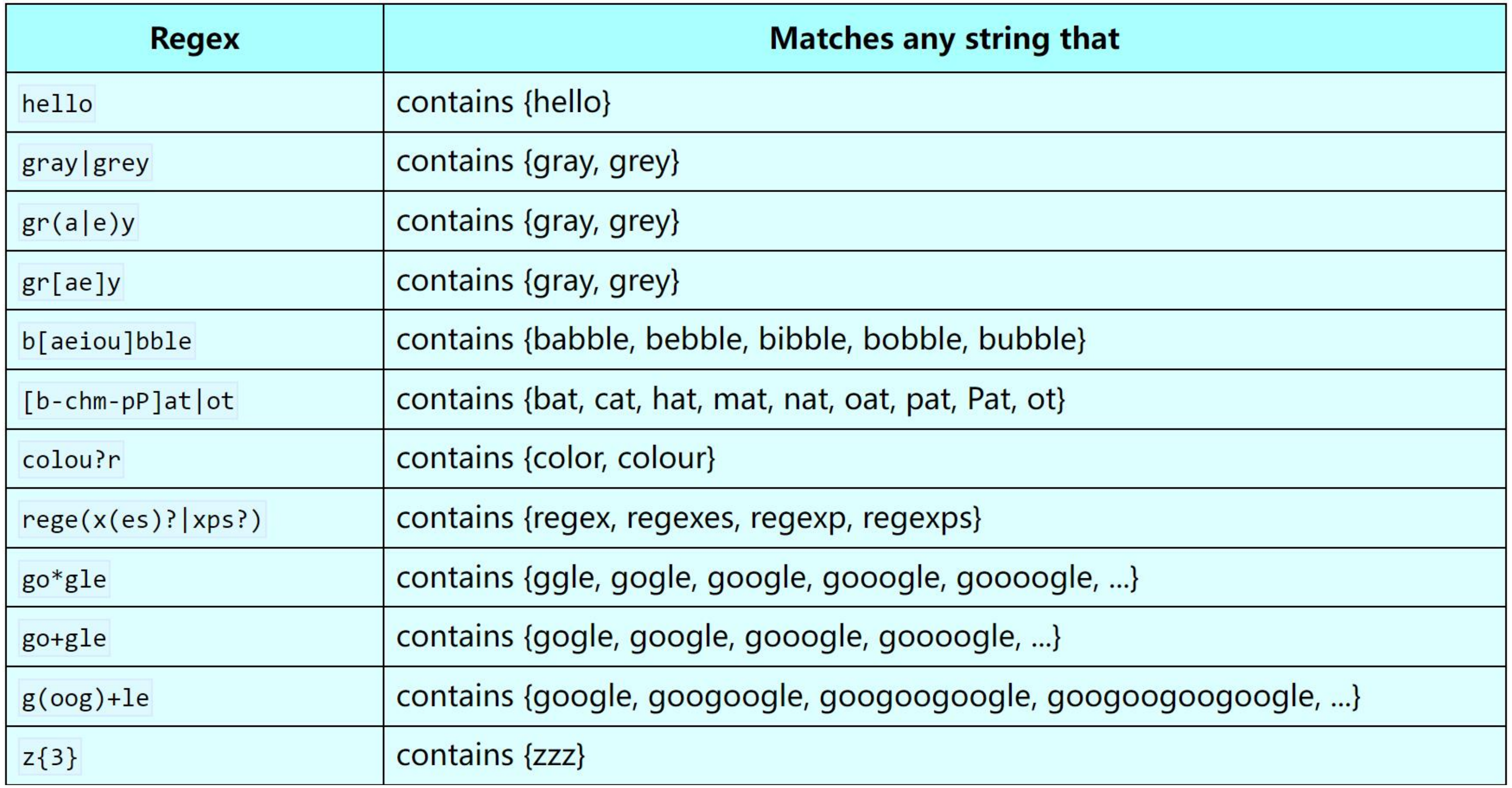
Regular Expressions in Neural Network

汪杰

[ACL18] Marrying Up Regular Expressions with Neural Networks: A Case Study for Spoken Language Understanding

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, Dongyan Zhao

正则表达式 Regular Expressions （RE）



Regex	Matches any string that
hello	contains {hello}
gray|grey	contains {gray, grey}
gr(a|e)y	contains {gray, grey}
gr[ae]y	contains {gray, grey}
b[aeiou]bble	contains {babble, bebble, bibble, bobble, bubble}
[b-chm-pP]at|ot	contains {bat, cat, hat, mat, nat, oat, pat, Pat, ot}
colou?r	contains {color, colour}
rege(x(es)?|xps?)	contains {regex, regexes, regexp, regexps}
go*gle	contains {ggle, gogle, google, gooogle, goooogle, ...}
go+gle	contains {gogle, google, gooogle, goooogle, ...}
g(oog)+le	contains {google, googoogle, googoogoogle, googoogoogoogle, ...}
z{3}	contains {zzz}

- 优点：简洁、可解释、可调的、不需要太多数据来生成
- 缺点：泛化性能差

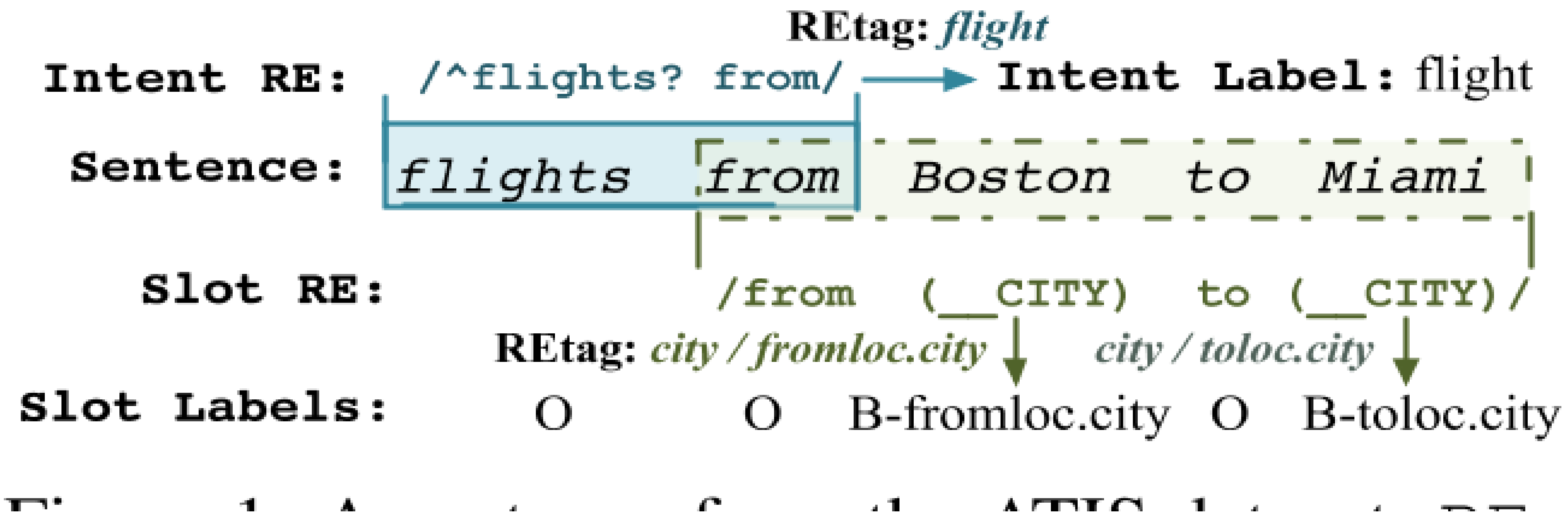
本文贡献

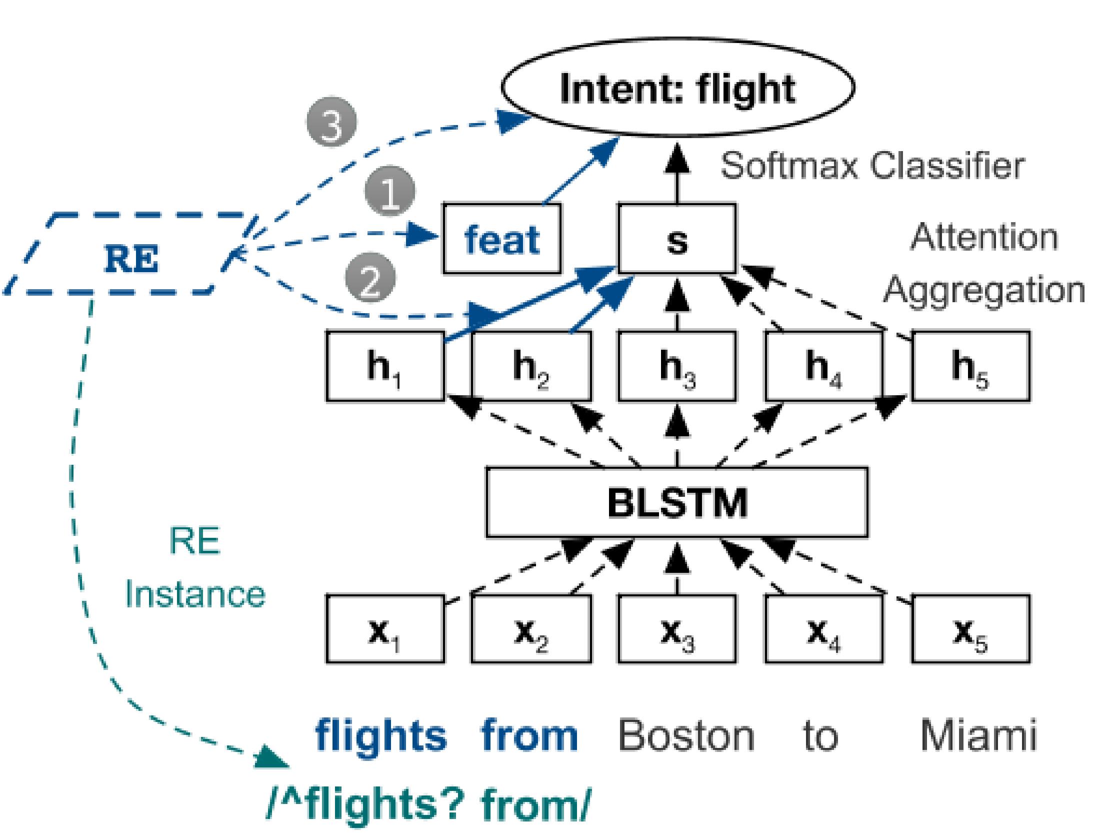
将神经网络（NN）与（人工设计的）正则表达式（RE）结合起来，用于提升监督学习性能，特别是 few-shot 情况下的性能。

- 输入层：RE 作为输入特征
- 网络模块层面：RE 指导 NN 的注意力机制
- 输出层：通过可学习的方式结合 RE 与 NN 的输出

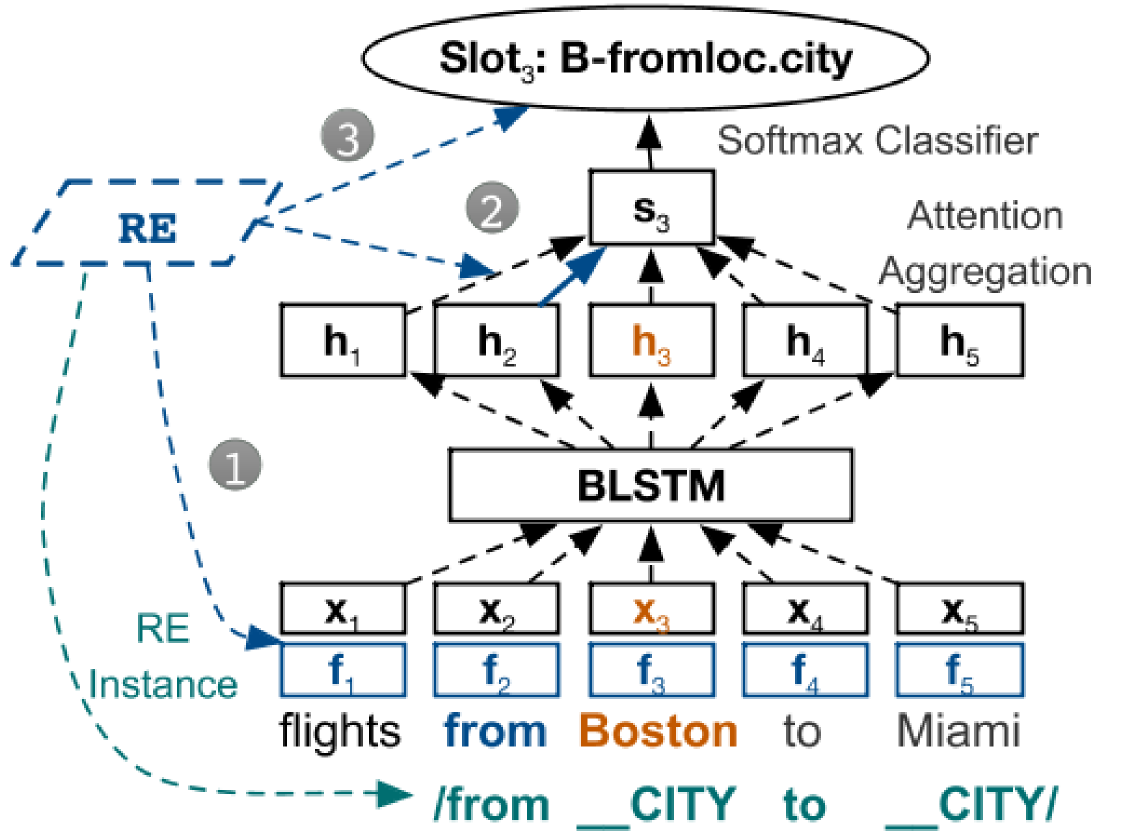
任务

- Intent Detection（句子分类）：RE 指导注意力机制
- Slot Filling（序列标注）：REtags 作为特征


REtag: *flight*
Intent RE: /^flights? from/
Intent Label: flight
Sentence: flights from Boston to Miami
Slot RE: /from (__CITY) to (__CITY)/
REtag: *city / fromloc.city*
city / toloc.city
Slot Labels: O O B-fromloc.city O B-toloc.city



(a) Intent Detection


Slot$_3$: B-fromloc.city
Softmax Classifier
Attention Aggregation
RE
s_3
h_1 h_2 h_3 h_4 h_5
BLSTM
x_1 x_2 x_3 x_4 x_5
f_1 f_2 f_3 f_4 f_5
RE Instance
flights from Boston to Miami
/from __CITY to __CITY/

(b) Slot Filling (predicting slot label for *Boston*)

Using REs at the Input Level

- Intent Detection:
  - 取成功匹配的 RE 的 REtag 的 embedding，作为 Softmax 分类器的输入。
- Slot Filling:
  - RE 的输出结果是 word-level tag，取 tag embedding 拼在单词的 embedding 后面。

Using REs at the Network Module Level

注意力的表示:

Base model: $\mathbf{s} = \sum_i \alpha_i \mathbf{h}_i, \quad \alpha_i = \frac{\exp\left(\mathbf{h}_i^\top \mathbf{W}\mathbf{c}\right)}{\sum_i \exp\left(\mathbf{h}_i^\top \mathbf{W}\mathbf{c}\right)}$

Intent Detection: $\mathbf{s}_k = \sum_i \alpha_{ki} \mathbf{h}_i, \quad \alpha_{ki} = \frac{\exp\left(\mathbf{h}_i^\top \mathbf{W}_a \mathbf{c}_k\right)}{\sum_i \exp\left(\mathbf{h}_i^\top \mathbf{W}_a \mathbf{c}_k\right)}$

Slot Filling: $\mathbf{s}_{pi} = \sum_j \alpha_{pij} \mathbf{h}_j, \quad \alpha_{pij} = \frac{\exp\left(\mathbf{h}_j^\top \mathbf{W}_{sp} \mathbf{h}_i\right)}{\sum_j \exp\left(\mathbf{h}_j^\top \mathbf{W}_{sp} \mathbf{h}_i\right)}$

Using REs at the Network Module Level

Intent Detection

输出类别为 k 的概率：

$$p_k = \frac{\exp\left(\operatorname{logit}_k\right)}{\sum_k \exp\left(\operatorname{logit}_k\right)}, \quad \operatorname{logit}_k = \mathbf{w}_k \mathbf{s}_k + b_k$$

引入 positive attentions 和 negative attentions

$$\operatorname{logit}_k = \operatorname{logit}_{pk} - \operatorname{logit}_{nk}$$

RE 通过 loss 控制模型：

$$\operatorname{loss}_{att} = \sum_k \sum_i t_{ki} \log\left(\alpha_{ki}\right)$$

Using REs at the Network Module Level

Slot Filling

不能对每个类别都计算正负注意力，因为计算量太大（$2 \times L \times n^2$）。因此选择共享正负注意力。

输入 Softmax 分类器之前把词表示 h_i 和注意力拼接：

$$\mathbf{p}_i = \mathrm{softmax}((\mathbf{W}_p[\mathbf{s}_{pi}; \mathbf{h}_i] + \mathbf{b}_p) - (\mathbf{W}_n[\mathbf{s}_{ni}; \mathbf{h}_i] + \mathbf{b}_n))$$

Using REs at the Output Level

原 NN 的 $\mathrm{logit}'_{\mathrm{k}}$ 的基础上加上 RE 相关的权重。

$$\mathrm{logit}_k = \mathrm{logit}'_k + w_k z_k$$

- 没有给每个 RE 都加上参数（可学习的权重），因为通常只有很少几句句子能匹配某个 RE
- 在 logit 而不是最终概率上加权重，因为 logit 是无约束的。

实验

ATIS dataset：18 句子分类类别，127 序列标注类别

预训练的词表示：100d GloVe word vectors

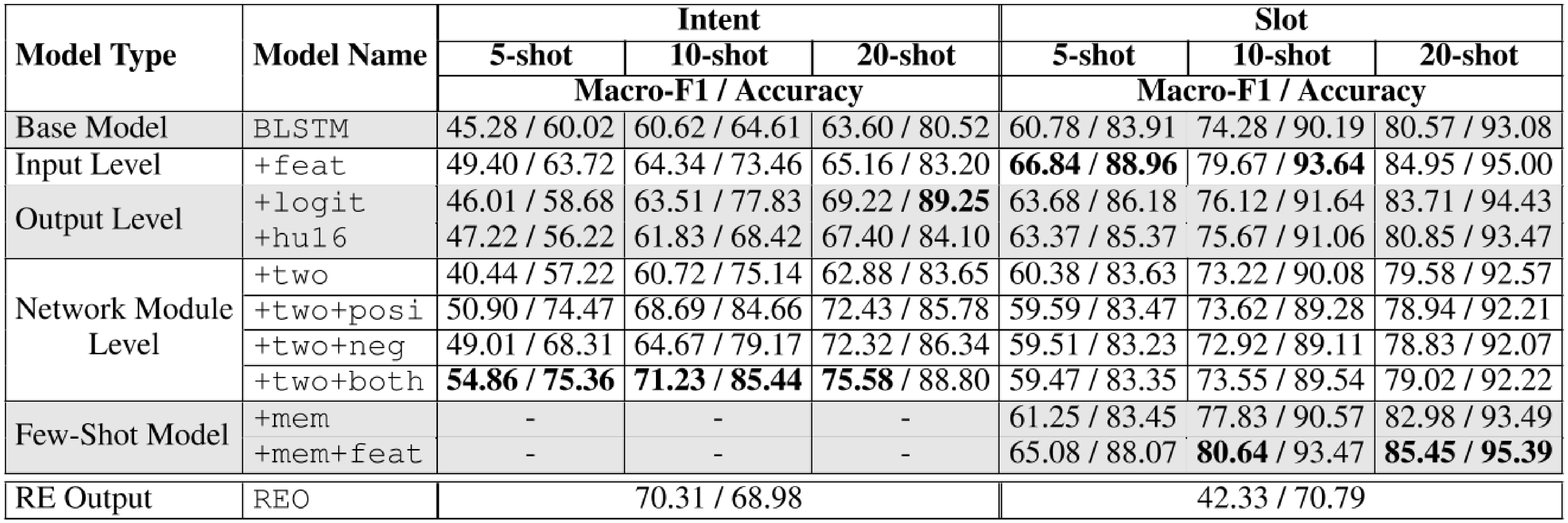
few-shot设置：

- full few-shot learning setting：每个类别 5、10、20 个训练样本
- partial few-shot learning setting：对最常见的 3 个类别给 300 个样本，其余类别 few-shot

RE 人工构造。

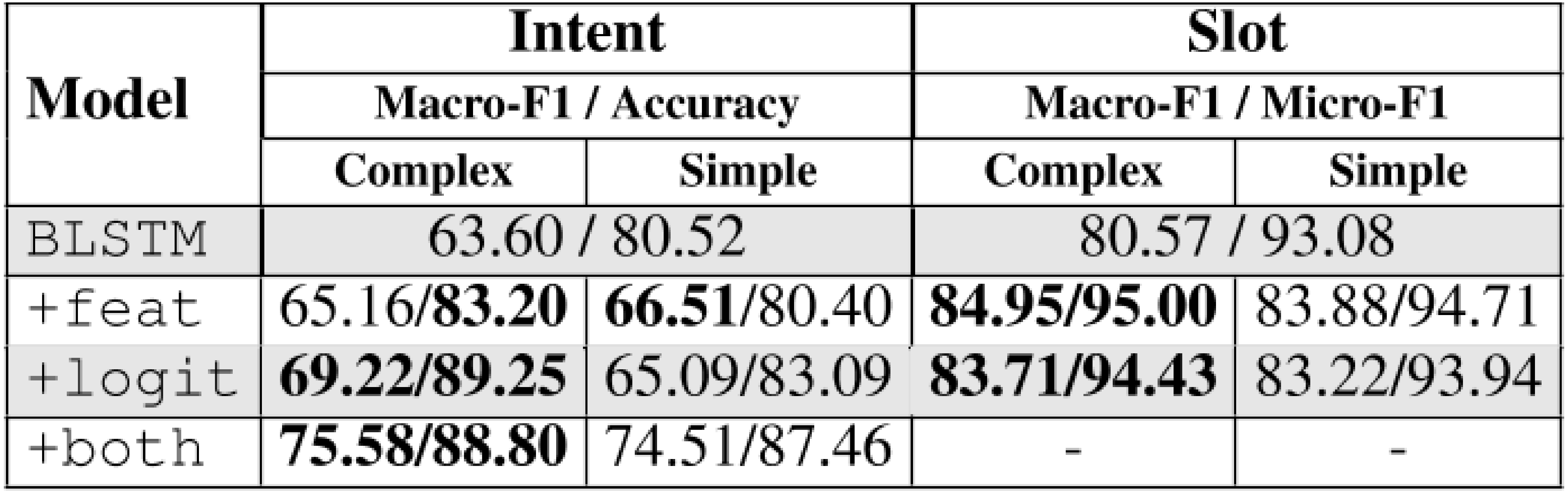
- 对照 20 个样本构造。当能够较好地匹配（resonable precision）这 20 个样本就认为构造完成。
- city 这类正则列表是从整个数据集中提取的

Results



Model Type	Model Name	Intent			Slot		
		5-shot	10-shot	20-shot	5-shot	10-shot	20-shot
		Macro-F1 / Accuracy			Macro-F1 / Accuracy		
Base Model	BLSTM	45.28 / 60.02	60.62 / 64.61	63.60 / 80.52	60.78 / 83.91	74.28 / 90.19	80.57 / 93.08
Input Level	+feat	49.40 / 63.72	64.34 / 73.46	65.16 / 83.20	**66.84 / 88.96**	79.67 / **93.64**	84.95 / 95.00
Output Level	+logit	46.01 / 58.68	63.51 / 77.83	69.22 / **89.25**	63.68 / 86.18	76.12 / 91.64	83.71 / 94.43
	+hu16	47.22 / 56.22	61.83 / 68.42	67.40 / 84.10	63.37 / 85.37	75.67 / 91.06	80.85 / 93.47
Network Module Level	+two	40.44 / 57.22	60.72 / 75.14	62.88 / 83.65	60.38 / 83.63	73.22 / 90.08	79.58 / 92.57
	+two+posi	50.90 / 74.47	68.69 / 84.66	72.43 / 85.78	59.59 / 83.47	73.62 / 89.28	78.94 / 92.21
	+two+neg	49.01 / 68.31	64.67 / 79.17	72.32 / 86.34	59.51 / 83.23	72.92 / 89.11	78.83 / 92.07
	+two+both	**54.86 / 75.36**	**71.23 / 85.44**	**75.58** / 88.80	59.47 / 83.35	73.55 / 89.54	79.02 / 92.22
Few-Shot Model	+mem	-	-	-	61.25 / 83.45	77.83 / 90.57	82.98 / 93.49
	+mem+feat	-	-	-	65.08 / 88.07	**80.64** / 93.47	**85.45 / 95.39**
RE Output	REO	70.31 / 68.98			42.33 / 70.79		

Results



Model	Intent		Slot	
	Macro-F1 / Accuracy		Macro-F1 / Micro-F1	
	Complex	Simple	Complex	Simple
`BLSTM`	63.60 / 80.52		80.57 / 93.08	
`+feat`	65.16/**83.20**	**66.51**/80.40	**84.95/95.00**	83.88/94.71
`+logit`	**69.22/89.25**	65.09/83.09	**83.71/94.43**	83.22/93.94
`+both`	**75.58/88.80**	74.51/87.46	-	-

[ACL20] Benchmarking Multimodal Regex Synthesis with Complex Structures

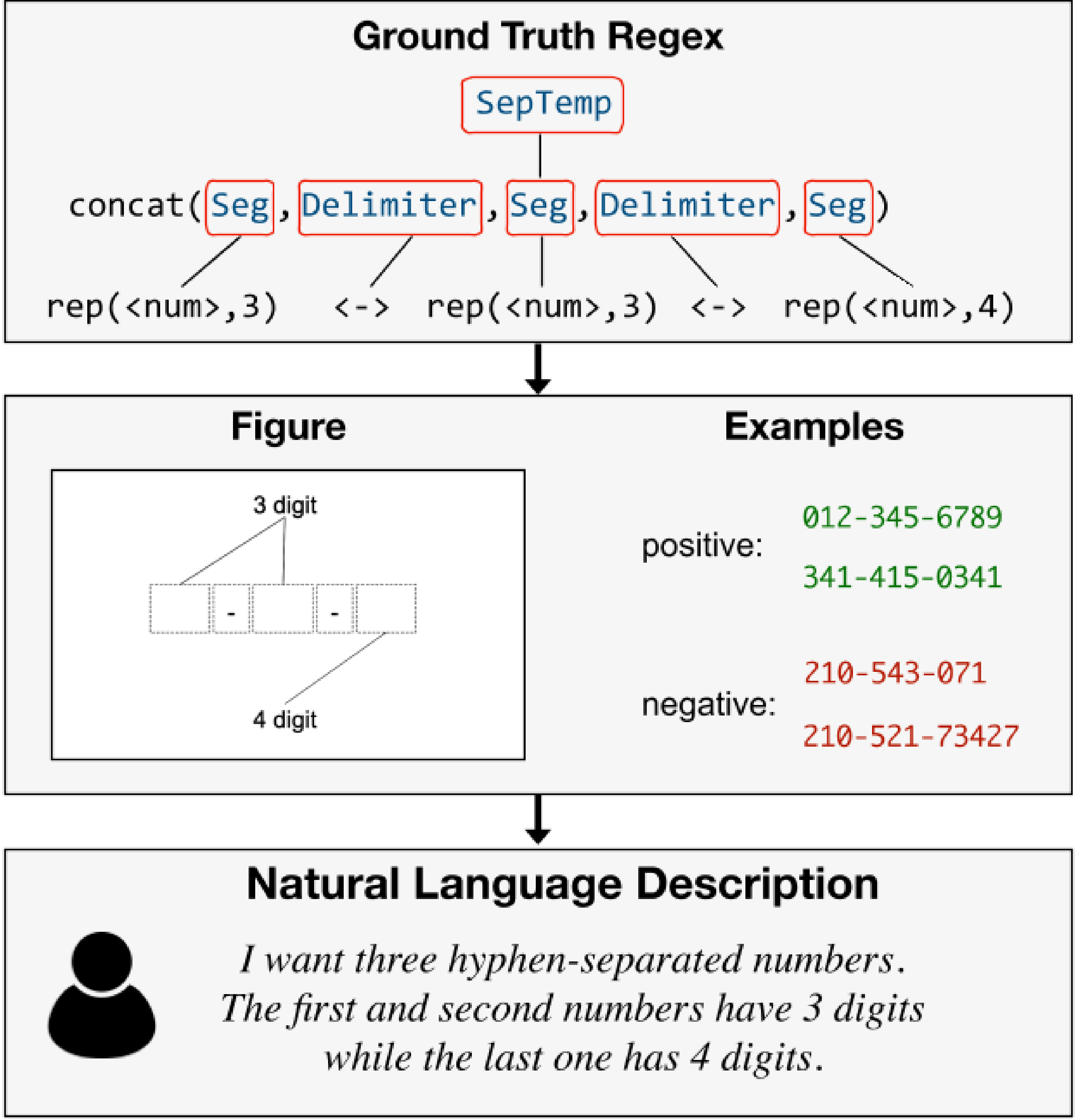
Xi Ye, Qiaochu Chen, Isil Dillig, Greg Durrett

本文贡献

提供了新的数据集：Structured Regex，是 RE 和自然语言之间的对应关系。

数据集生成流程

1. 根据 Stack Overflow 的用户提问人工制定模板，表示用户关心的 RE 中的一些常见结构
2. 对模板随机采样得到 RE （structured probabilistic grammar），然后采样 RE 得到一些正负样本
3. 将 RE 转化为描述性的图像
4. 人工标注任务：根据 RE 的描述性图像以及对应的正负样本，写出 RE 的自然语言描述。



```
concat(Seg,Delimiter,Seg,Delimiter,Seg)
rep(<num>,3)   <->   rep(<num>,3)   <->   rep(<num>,4)
```

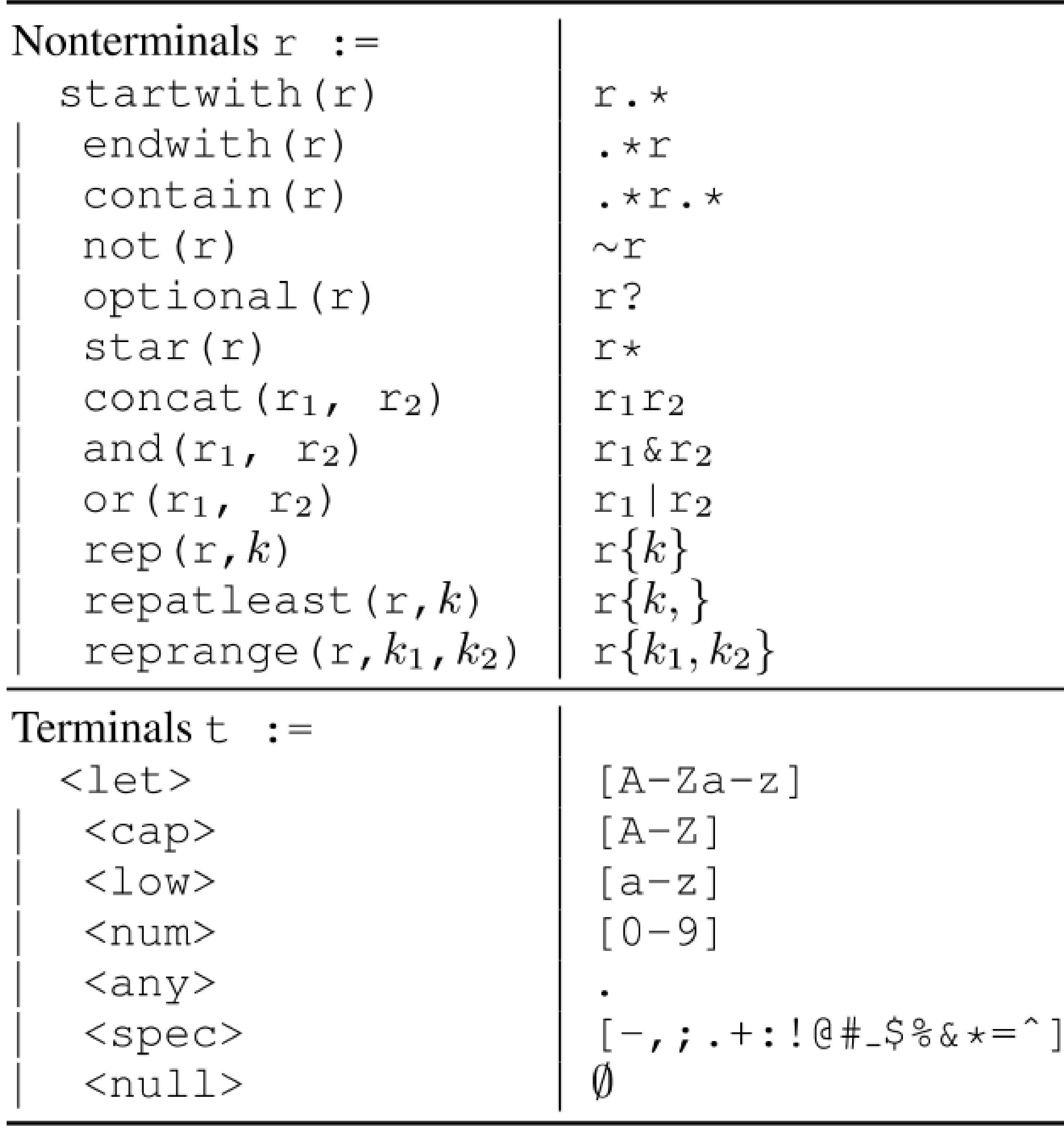
positive: 012-345-6789, 341-415-0341

negative: 210-543-071, 210-521-73427

I want three hyphen-separated numbers. The first and second numbers have 3 digits while the last one has 4 digits.

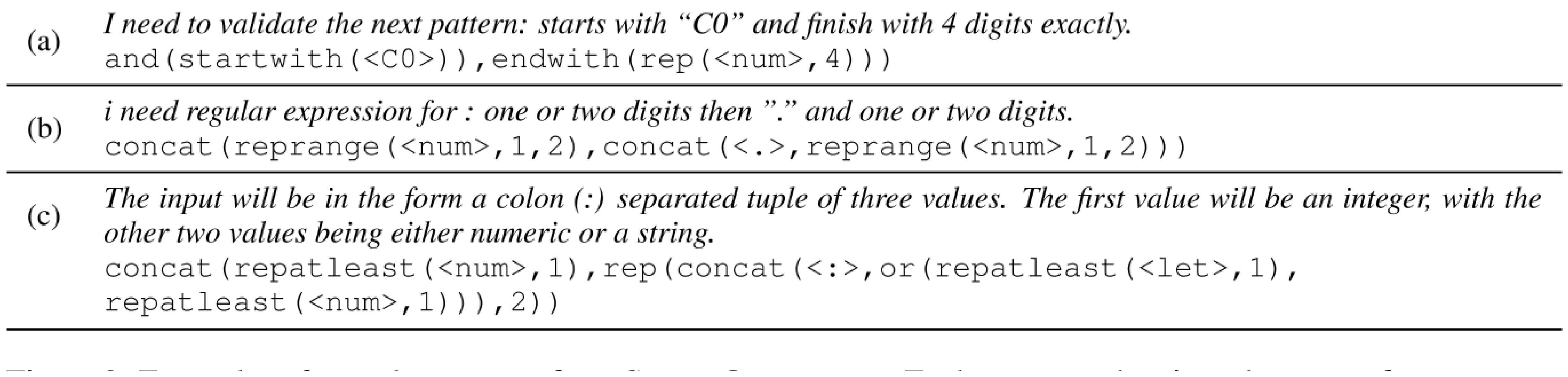
Domain Specific Language (DSL)

Our regex DSL and the corresponding constructions in standard regular language. Our regex DSL is as expressive as and can be easily translated to standard regex syntax.



Nonterminals r :=		
	startwith(r)	r.*
$\mid$	endwith(r)	.*r
$\mid$	contain(r)	.*r.*
$\mid$	not(r)	~r
$\mid$	optional(r)	r?
$\mid$	star(r)	r*
$\mid$	concat(r_1, r_2)	$r_1 r_2$
$\mid$	and(r_1, r_2)	r_1&r_2
$\mid$	or(r_1, r_2)	$r_1 \mid r_2$
$\mid$	rep(r,k)	r{k}
$\mid$	repatleast(r,k)	r{k,}
$\mid$	reprange(r,k_1,k_2)	r{k_1,k_2}
Terminals t :=		
	<let>	[A-Za-z]
$\mid$	<cap>	[A-Z]
$\mid$	<low>	[a-z]
$\mid$	<num>	[0-9]
$\mid$	<any>	.
$\mid$	<spec>	[-,;.+:!@#_$%&*=^]
$\mid$	<null>	$\emptyset$

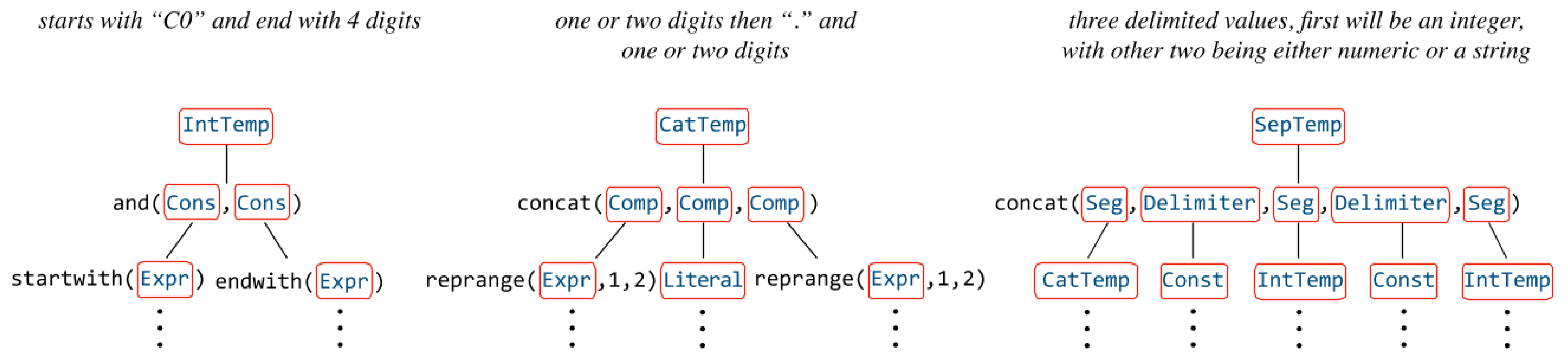
DSL 例子



(a)	*I need to validate the next pattern: starts with "C0" and finish with 4 digits exactly.* `and(startwith(<C0>)),endwith(rep(<num>,4)))`
(b)	*i need regular expression for : one or two digits then "." and one or two digits.* `concat(reprange(<num>,1,2),concat(<.>,reprange(<num>,1,2)))`

(c)	*The input will be in the form a colon (:) separated tuple of three values. The first value will be an integer, with the other two values being either numeric or a string.*
	`concat(repatleast(<num>,1),rep(concat(<:>,or(repatleast(<let>,1),`
	`repatleast(<num>,1))),2))`

Structured Grammar 模板

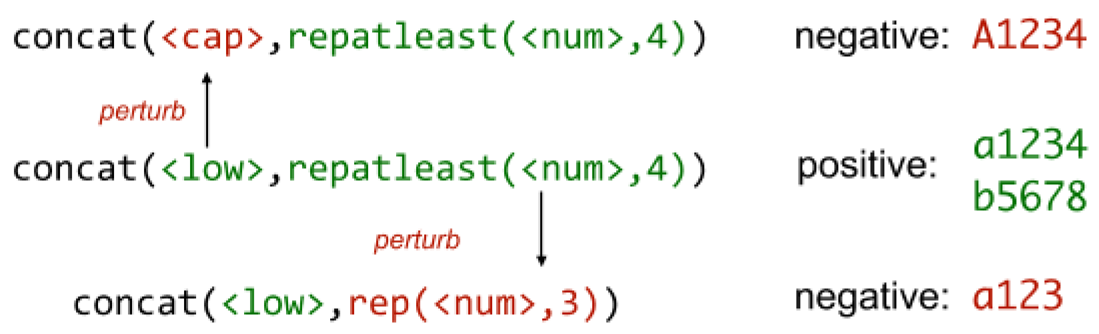

starts with "C0" and end with 4 digits
IntTemp
and(Cons, Cons)
startwith(Expr) endwith(Expr)
one or two digits then "." and one or two digits
CatTemp
concat(Comp, Comp, Comp)
reprange(Expr,1,2) Literal reprange(Expr,1,2)
three delimited values, first will be an integer, with other two being either numeric or a string
SepTemp
concat(Seg, Delimiter, Seg, Delimiter, Seg)
CatTemp Const IntTemp Const IntTemp

Cons	start[end]with(Expr) | not(start[end]with(Expr))	# must (not) start/end with
	contain(Expr) | not(contain(Expr))	# must (not) contain
	rep(<any>,k) | repatleast(<any>,k) |reprange(<any>,k,k)	# length constraints
	AdvStartwithCons | AdvEndwithCons	# adversative macro (e.g., start with capitals except A)
	CondContainCons	# conditional macro. (e.g. letter, if contained, must be after a digit)
Comp	Literal | or(Literal,Literal,...)	# literals like digits, letters, strings, or set of literals.
	rep(Expr,k) | repatleast(Expr,k) | reprange(Expr,k,k)	# e.g, 3 digits, 2 - 5 letter, etc.
	optional(Comp)	# components can be optional.

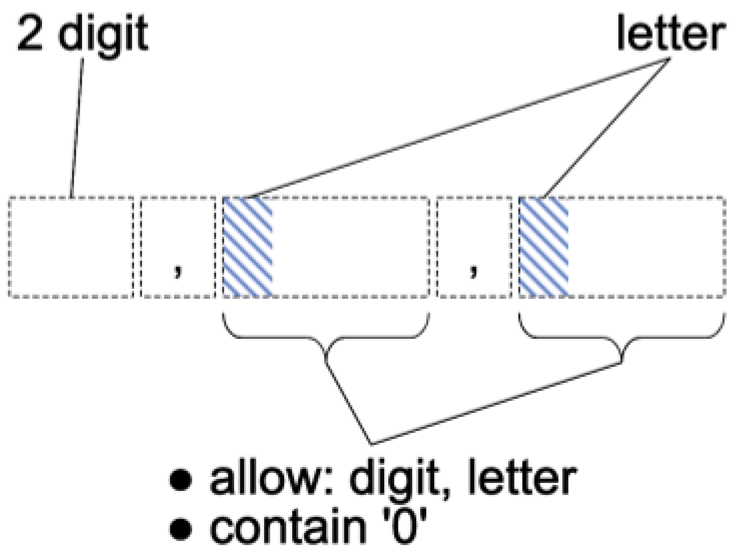
采样正负样本

- 正样本：随机遍历 RE 对应的 DFA。尽量走遍每个状态，以得到更好的区分度
- 负样本：扰动 RE，然后在新的（错误的）DFA 上采样正样本
    - 直接在原 DFA 上采样负样本的话，质量比较差（错误太明显）

负样本扰动示意图：


concat(<cap>,repatleast(<num>,4))
negative: A1234
perturb
concat(<low>,repatleast(<num>,4))
positive: a1234
b5678
perturb
concat(<low>,rep(<num>,3))
negative: a123

生成描述性图像



Three comma separated segments. The first segment is 2 digits. The other two consist of digits or letters but must start with a letter and contain "0".

[illegible]