EMNLP Paper Review 2

SlotRefine: A Fast Non-Autoregressive Model for Joint Intent Detection and Slot Filling

Di Wu, Liang Ding, Fan Lu, Jian Xie

- https://arxiv.org/pdf/2010.02693.pdf
- https://github.com/moore3930/SlotRefine

https://arxiv.org/pdf/2010.02693.pdf

Concept

- Propose non-autoregressive model named SlotRefine for joint intent detection and slot filling
- Previous works heavily rely on autoregressive approaches
- Argue that identifying token dependencies among slot chunk is enough, and it is unnecessary to model the entire sequence dependency in autoregressive fashion
- Two-pass Refine Mechanism
    - the model generates a draft in the first pass and tries to find the beginning of each slot chunk.
    - by propagating the utterance again with the predicted "B- tags", the model is forced to learn how many identical "I-tags" follow them.

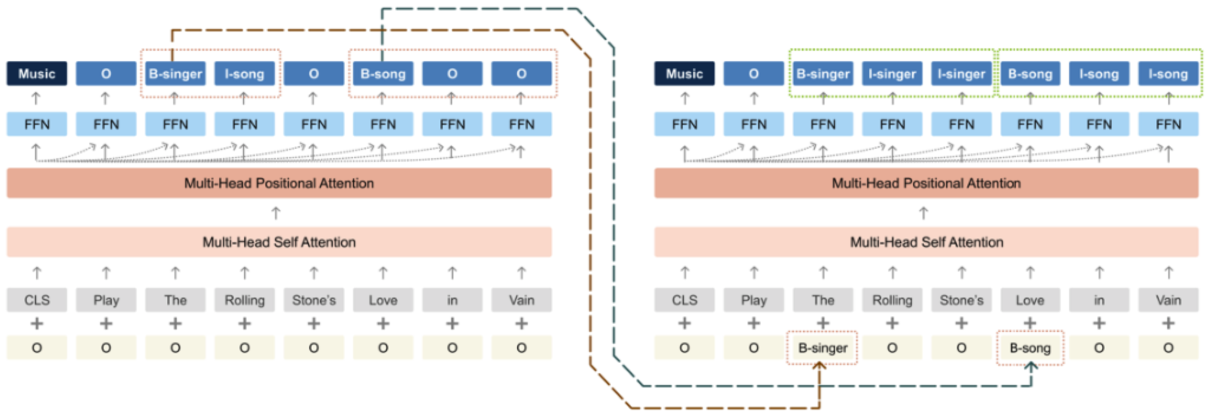
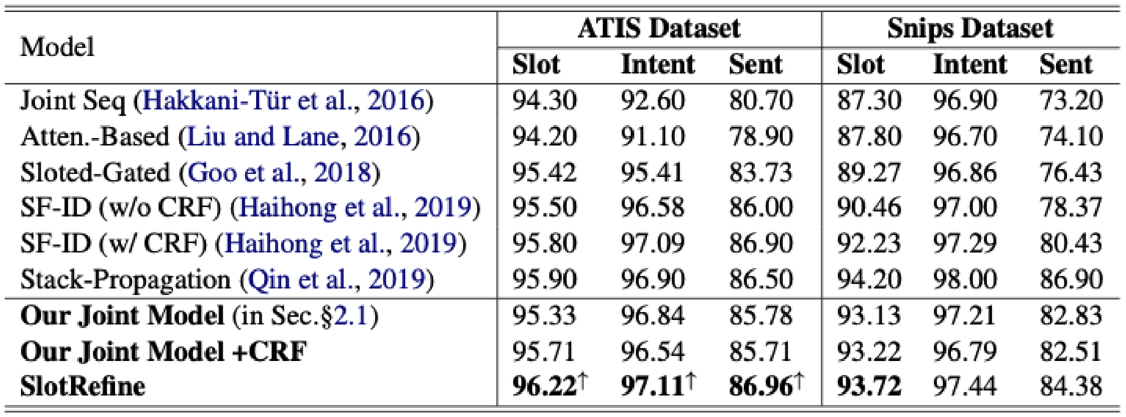


Figure 1: Illustration of SlotRefine, where the left and right part indicate the first and second iteration process respectively. In the first pass, wrong slot tagging results are predicted, as shown in the pink dotted box in the figure, and the "*B-tags*" (beginning tag of a slot) are feeded as additional information with utterance for second iteration. The slot results in the green dotted box are refined results by second pass. Note that the initial tag embedding "O" added to each inputting position is designed for the two-pass mechanism(Sec.§2.2).

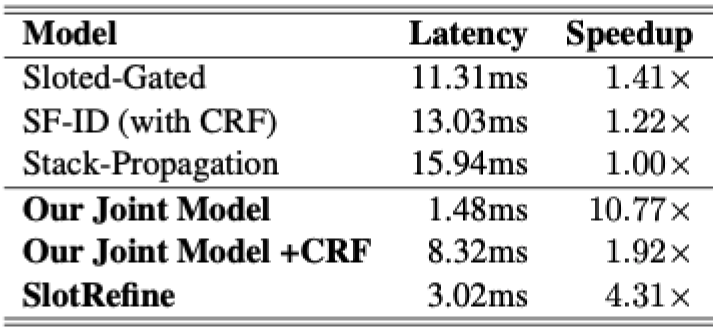
Experiment

- ATIS (Airline Travel Information Systems,Tur et al. (2010))
- Snips (collected by Snips personal voice assistant,Coucke et al. (2018))
- sentence accuracy: slots and intent are both correctly-predicted



Model	ATIS Dataset			Snips Dataset		
	Slot	**Intent**	**Sent**	**Slot**	**Intent**	**Sent**
Joint Seq (Hakkani-Tür et al., 2016)	94.30	92.60	80.70	87.30	96.90	73.20
Atten.-Based (Liu and Lane, 2016)	94.20	91.10	78.90	87.80	96.70	74.10
Sloted-Gated (Goo et al., 2018)	95.42	95.41	83.73	89.27	96.86	76.43
SF-ID (w/o CRF) (Haihong et al., 2019)	95.50	96.58	86.00	90.46	97.00	78.37
SF-ID (w/ CRF) (Haihong et al., 2019)	95.80	97.09	86.90	92.23	97.29	80.43
Stack-Propagation (Qin et al., 2019)	95.90	96.90	86.50	94.20	98.00	86.90
Our Joint Model (in Sec.§2.1)	95.33	96.84	85.78	93.13	97.21	82.83
Our Joint Model +CRF	95.71	96.54	85.71	93.22	96.79	82.51
SlotRefine	**96.22**$^{\uparrow}$	**97.11**$^{\uparrow}$	**86.96**$^{\uparrow}$	**93.72**	97.44	84.38

Table 1: Performance comparison on ATIS and Snips datasets. "↑"indicates significant difference ($p < 0.05$) with previous works. Model name written in bold refer to ours.



Model	Latency	Speedup
Sloted-Gated	11.31ms	1.41×
SF-ID (with CRF)	13.03ms	1.22×
Stack-Propagation	15.94ms	1.00×
Our Joint Model	1.48ms	10.77×
Our Joint Model +CRF	8.32ms	1.92×
SlotRefine	3.02ms	4.31×

Table 2: "Latency" is the average time to decode an utterance without minibatching. "Speedup" is compared against existing SOTA model (Haihong et al., 2019).

IGSQL: Database Schema Interaction Graph Based Neural Model for Context-Dependent Text-to-SQL Generation

Yitao Cai and Xiaojun Wan

- https://arxiv.org/pdf/2011.05744.pdf
- https://github.com/headacheboy/IGSQL

https://arxiv.org/pdf/2011.05744.pdf

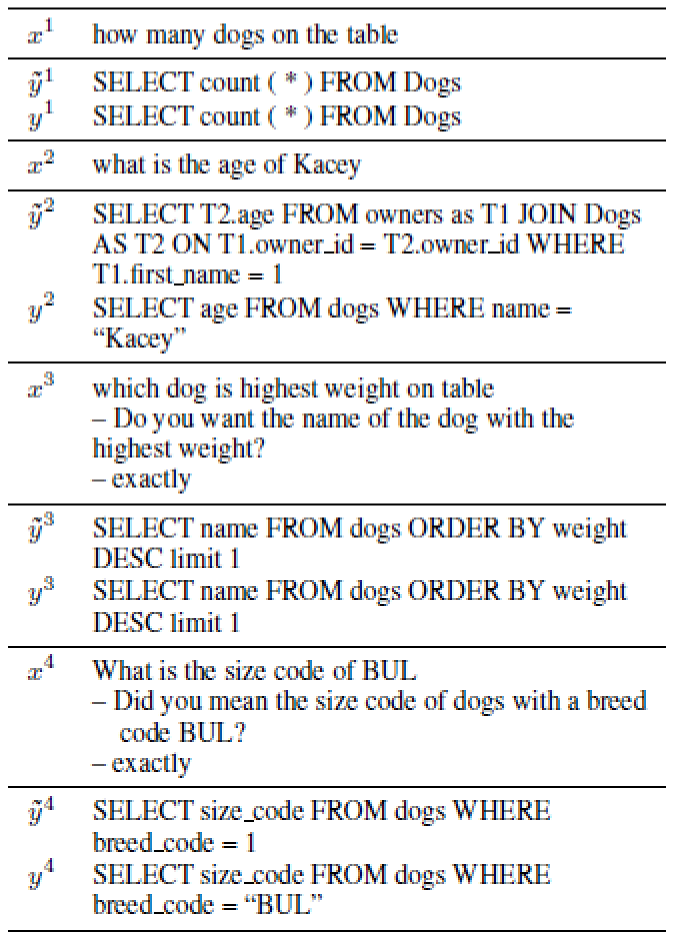
Abstract

- Context-dependent text-to-SQL task
- Previous models on context-dependent text-to-SQL task only concentrate on utilizing historical user inputs.
- ***a database schema interaction graph encoder to utilize historical information of database schema items***

- ***In decoding phase, we introduce a gate mechanism to weigh the importance of different vocabularies and then make the prediction of SQL tokens.***

Example

- EditSQL views “Kacey” as the name of a dog owner.
- However, since the context of the interaction is about dogs, “Kacey” should be the name of a dog.

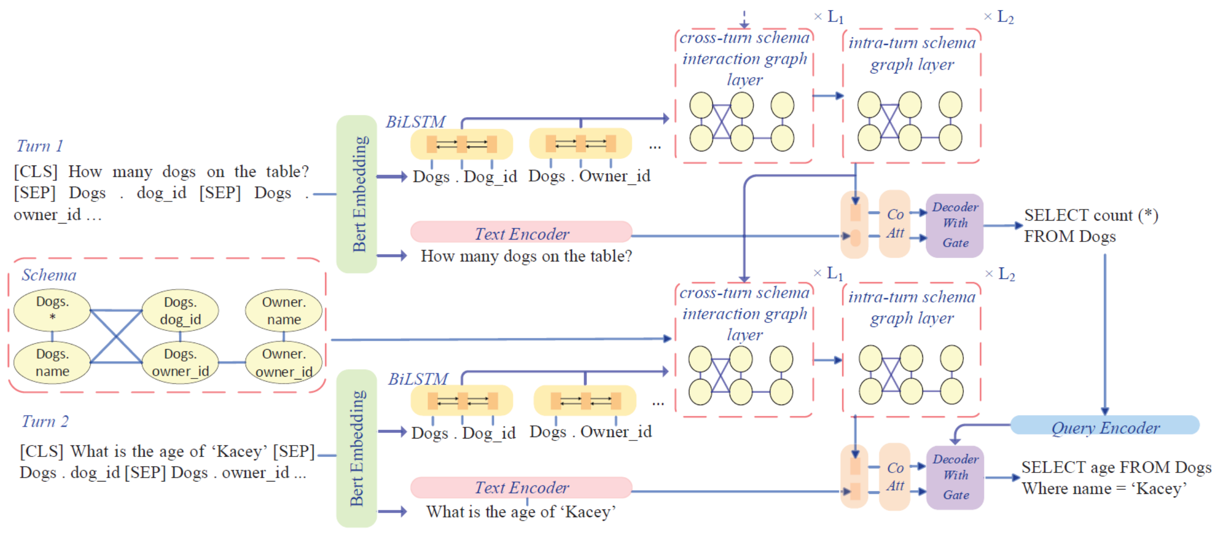


x^1	how many dogs on the table
$\tilde{y}^1$	SELECT count (*) FROM Dogs
y^1	SELECT count (*) FROM Dogs
x^2	what is the age of Kacey
$\tilde{y}^2$	SELECT T2.age FROM owners as T1 JOIN Dogs AS T2 ON T1.owner_id = T2.owner_id WHERE T1.first_name = 1
y^2	SELECT age FROM dogs WHERE name = “Kacey”
x^3	which dog is highest weight on table – Do you want the name of the dog with the highest weight? – exactly
$\tilde{y}^3$	SELECT name FROM dogs ORDER BY weight DESC limit 1
y^3	SELECT name FROM dogs ORDER BY weight DESC limit 1
x^4	What is the size code of BUL – Did you mean the size code of dogs with a breed code BUL? – exactly
$\tilde{y}^4$	SELECT size_code FROM dogs WHERE breed_code = 1
y^4	SELECT size_code FROM dogs WHERE breed_code = “BUL”

Table 1: An example interaction. x^i is the input sequence in i-th turn and y^i is the corresponding ground truth query. $\tilde{y}^i$ means that query is predicted by a model, which is EditSQL here.

Concept

- Encoder-decoder framework with attention mechanism
- Using BERT Embedding for user inputs and database schema items
- Database Schema Interaction Graph Encoder
  - For two database schema items appearing in two adjacent turns, short distance of items in the graph can reveal the context consistency.
  - For example, the distance between Dogs.* and correct item Dogs.name is 1. Distance between Dogs.* and wrong item owners.name is 3.


× L1
× L2
cross-turn schema interaction graph layer
intra-turn schema graph layer
Turn 1
[CLS] How many dogs on the table? [SEP] Dogs . dog_id [SEP] Dogs . owner_id ...
Bert Embedding
BiLSTM
Dogs . Dog_id Dogs . Owner_id ...
Text Encoder
How many dogs on the table?
Co Att
Decoder With Gate
SELECT count (*) FROM Dogs
Schema
Dogs. *
Dogs. dog_id
Owner. name
Dogs. name
Dogs. owner_id
Owner. owner_id
Turn 2
[CLS] What is the age of 'Kacey' [SEP] Dogs . dog_id [SEP] Dogs . owner_id ...
What is the age of 'Kacey'
Query Encoder
SELECT age FROM Dogs Where name = 'Kacey'

Experimental Result

Method	SParC Dev		SParC Test		CoSQL Dev		CoSQL Test	
	Ques	Int	Ques	Int	Ques	Int	Ques	Int
CD S2S	21.9	8.1	23.2	7.5	13.8	2.1	13.9	2.6
SyntaxSQL-con	18.5	4.3	20.2	5.2	15.1	2.7	14.1	2.2
EditSQL*	47.2	29.5	47.9	25.3	39.9	12.3	40.8	13.7
IGSQL*	**50.7**	**32.5**	**51.2**	**29.5**	**44.1**	**15.8**	**42.5**	**15.0**

Table 3: Results of models in SParC and CoSQL datasets. Ques means question match accuracy. Int means interaction match accuracy. * means that results are enhanced by BERT embedding.

TernaryBERT: Distillation-aware Ultra-low Bit BERT

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, Qun Liu

- https://arxiv.org/pdf/2009.12812.pdf

https://arxiv.org/pdf/2009.12812.pdf

Abstract

- Propose TernaryBERT, which ternarizes the weights in a fine-tuned BERT model.
- quantization(2bit- {−1, 0, +1}) + knowledge distillation
- quantization compresses a neural network by using lower bits for weight values without changing the model architecture
- 8-bit quantization is successfully applied to Transformer-based models with comparable performance as the full-precision baseline. However, quantizing these models to ultra low bits (e.g., 1 or 2 bits) can be much more challenging due to significant reduction in model capacity.

Approach

1. Quantization

   a. ternarize the weights wt in the student BERT model to wˆt.

2. Distillation

   a. forward pass with the ternarized model

   b. gradient of the distillation loss

c. use the full-precision weight for parameter update (important to keep the full-precision weight during training)

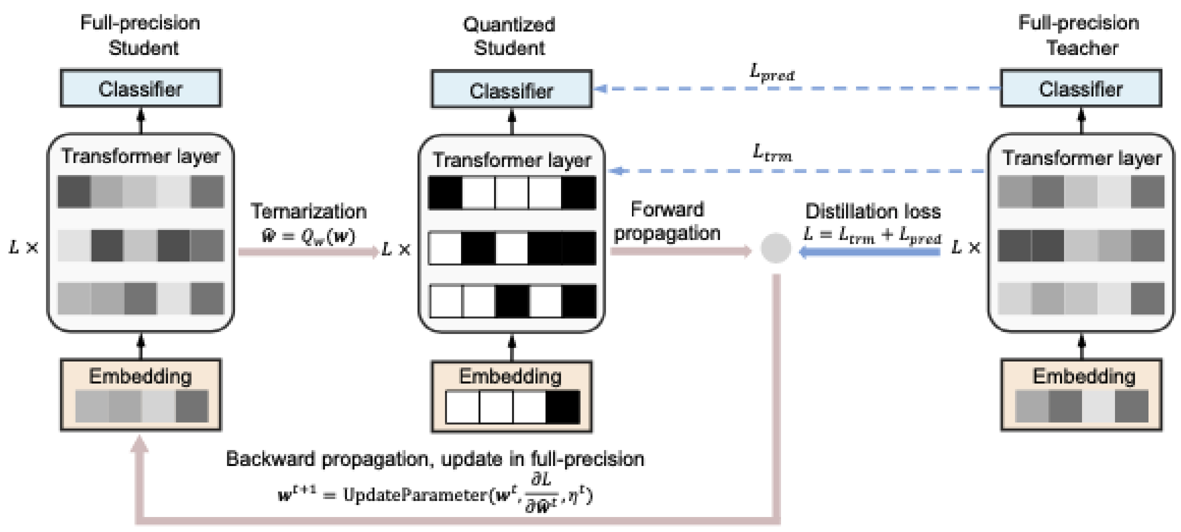


Figure 2: Depiction of the proposed distillation-aware ternarization of BERT model.

Experiment

- Experiments on the GLUE benchmark and SQuAD show that our proposed TernaryBERT outperforms the other BERT quantization methods, and even achieves comparable performance as the full- precision model while being 14.9x smaller.

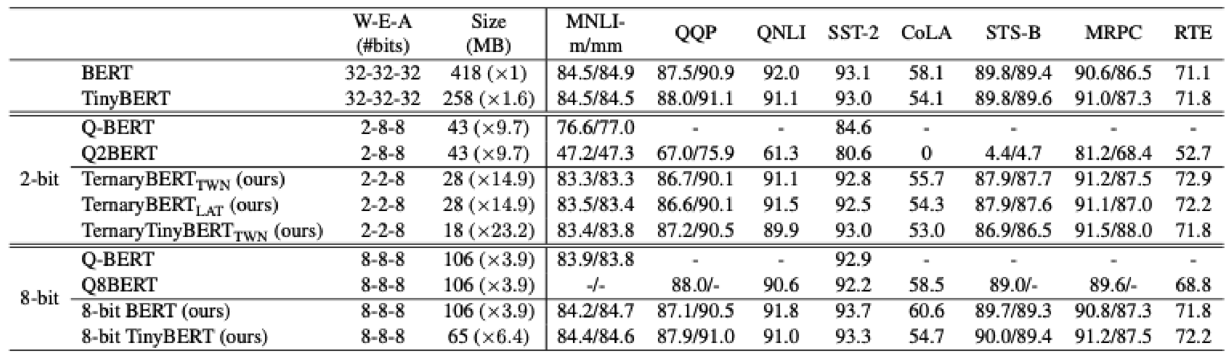


Table 1: Development set results of quantized BERT and TinyBERT on the GLUE benchmark. We abbreviate the number of bits for weights of Transformer layers, word embedding and activations as “W-E-A (#bits)”.

		W-E-A (#bits)	Size (MB)	MNLI-m/mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE
	BERT	32-32-32	418 (×1)	84.5/84.9	87.5/90.9	92.0	93.1	58.1	89.8/89.4	90.6/86.5	71.1
	TinyBERT	32-32-32	258 (×1.6)	84.5/84.5	88.0/91.1	91.1	93.0	54.1	89.8/89.6	91.0/87.3	71.8
2-bit	Q-BERT	2-8-8	43 (×9.7)	76.6/77.0	-	-	84.6	-	-	-	-
	Q2BERT	2-8-8	43 (×9.7)	47.2/47.3	67.0/75.9	61.3	80.6	0	4.4/4.7	81.2/68.4	52.7
	TernaryBERT$_{TWN}$ (ours)	2-2-8	28 (×14.9)	83.3/83.3	86.7/90.1	91.1	92.8	55.7	87.9/87.7	91.2/87.5	72.9
	TernaryBERT$_{LAT}$ (ours)	2-2-8	28 (×14.9)	83.5/83.4	86.6/90.1	91.5	92.5	54.3	87.9/87.6	91.1/87.0	72.2
	TernaryTinyBERT$_{TWN}$ (ours)	2-2-8	18 (×23.2)	83.4/83.8	87.2/90.5	89.9	93.0	53.0	86.9/86.5	91.5/88.0	71.8
8-bit	Q-BERT	8-8-8	106 (×3.9)	83.9/83.8	-	-	92.9	-	-	-	-
	Q8BERT	8-8-8	106 (×3.9)	-/-	88.0/-	90.6	92.2	58.5	89.0/-	89.6/-	68.8
	8-bit BERT (ours)	8-8-8	106 (×3.9)	84.2/84.7	87.1/90.5	91.8	93.7	60.6	89.7/89.3	90.8/87.3	71.8
	8-bit TinyBERT (ours)	8-8-8	65 (×6.4)	84.4/84.6	87.9/91.0	91.0	93.3	54.7	90.0/89.4	91.2/87.5	72.2

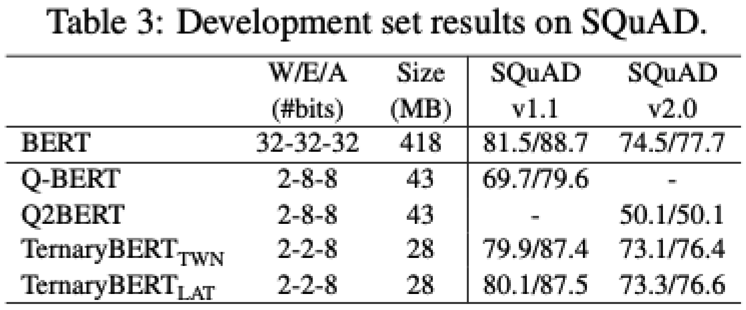


Table 3: Development set results on SQuAD.

	W/E/A (#bits)	Size (MB)	SQuAD v1.1	SQuAD v2.0
BERT	32-32-32	418	81.5/88.7	74.5/77.7
Q-BERT	2-8-8	43	69.7/79.6	-
Q2BERT	2-8-8	43	-	50.1/50.1
TernaryBERT$_{TWN}$	2-2-8	28	79.9/87.4	73.1/76.4
TernaryBERT$_{LAT}$	2-2-8	28	80.1/87.5	73.3/76.6

Sketch-Driven Regular Expression Generation from Natural Language and Examples

Xi Ye, Qiaochu Chen, Xinyu Wang, Isil Dillig, Greg Durrett

- https://arxiv.org/pdf/2009.11264.pdf
- https://github.com/xiye17/SketchRegex/

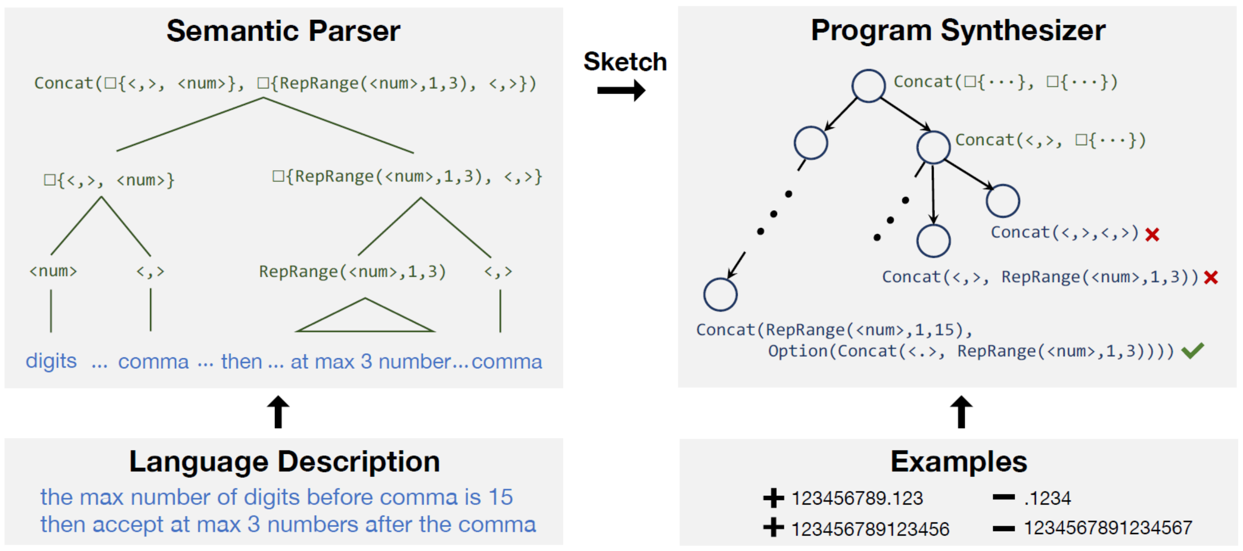
https://arxiv.org/pdf/1908.05848.pdf

Abstract

- Recent systems for converting NL to RE
  - typically deal with short, formulaic text and can only produce simple regexes.
- Real world regexes are complex, hard to describe with brief sentences
- a framework for regex synthesis in this setting where both natural language (NL) and examples(positive/negative) are available.

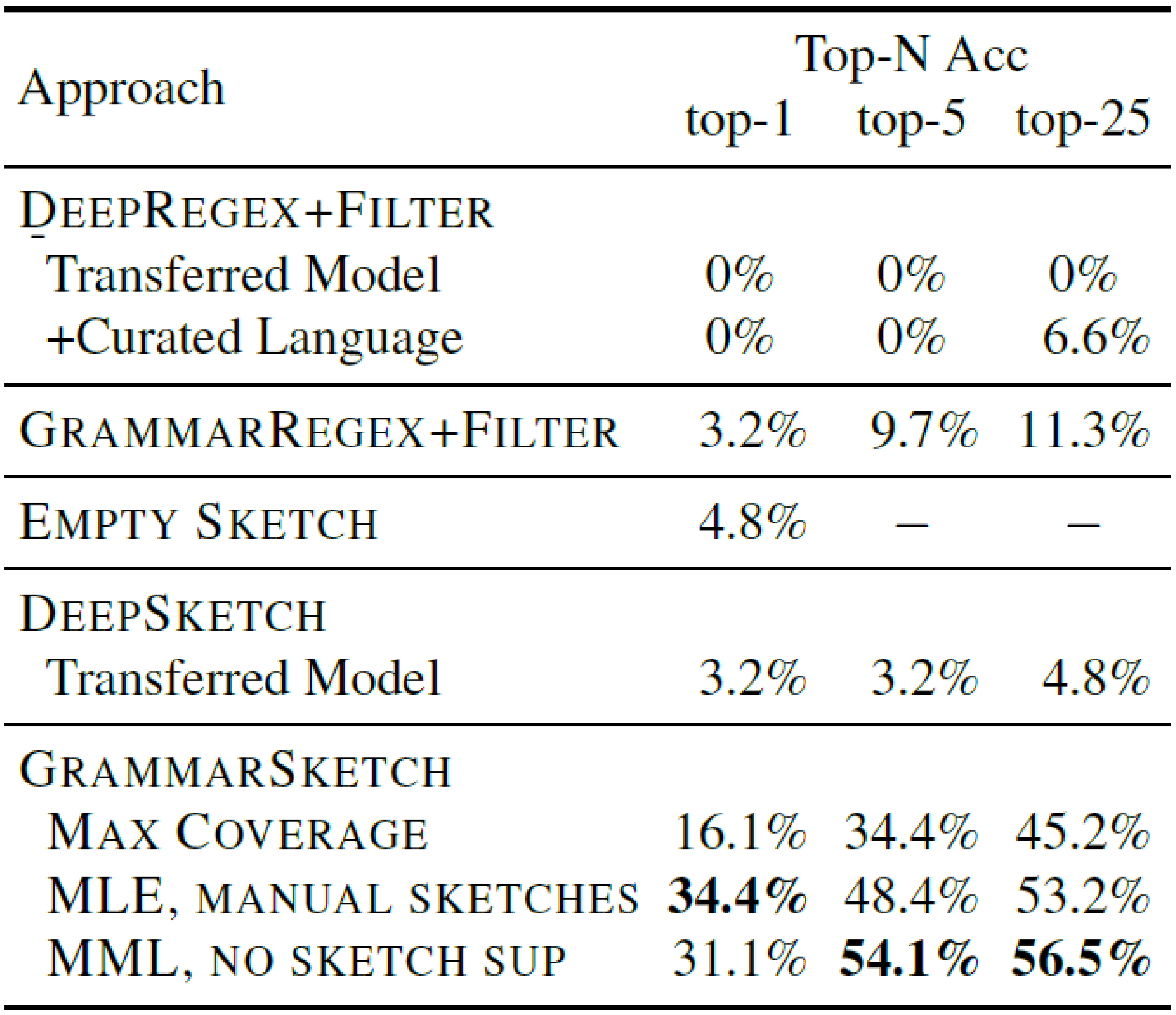
Concept

- https://stackoverflow.com/questions/19076566/regular-expression-that-validates-decimal-18-3
- Parse the description into an intermediate representation, called a sketch
  - neural network-based with a seq-to-seq model (Luong et al., 2015)
  - grammar-based with a semantic parser (Berant et al., 2013).
  - The purpose of the sketch is to capture useful components from the description as well as the high-level structure of the regex.
- use an off-the-shelf program synthesizer, mildly customized for our task, to producea regex consistent with both the sketch and the provided examples.

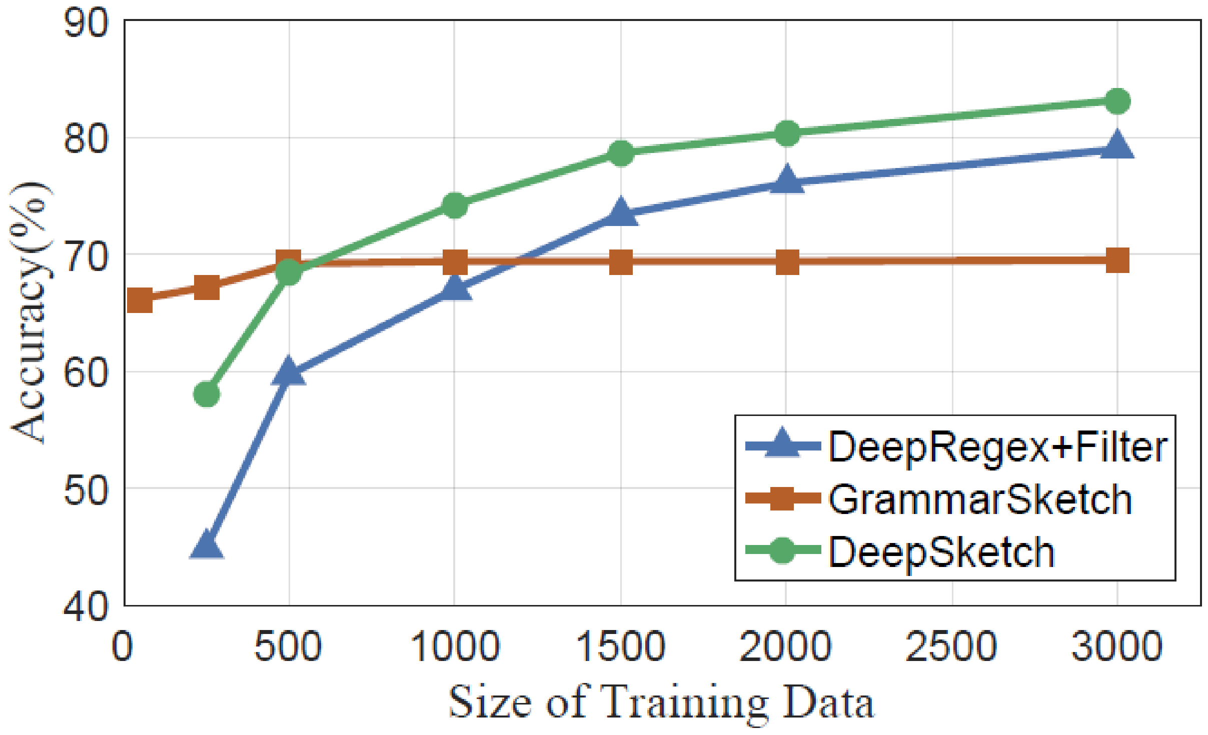

Semantic Parser
Concat(□{<,>, <num>}, □{RepRange(<num>,1,3), <,>})
□{<,>, <num>}
□{RepRange(<num>,1,3), <,>}
<num>
<,>
RepRange(<num>,1,3)
<,>
digits ... comma ... then ... at max 3 number...comma
Sketch
Program Synthesizer
Concat(□{···}, □{···})
Concat(<,>, □{···})
Concat(<,>,<,>) ✗
Concat(<,>, RepRange(<num>,1,3)) ✗
Concat(RepRange(<num>,1,15), Option(Concat(<.>, RepRange(<num>,1,3)))) ✓
Language Description
the max number of digits before comma is 15 then accept at max 3 numbers after the comma
Examples
+ 123456789.123
+ 123456789123456
− .1234
− 1234567891234567

Experimental

- STACKOVERFLOW 너무 어려워서 Top N 개중에 하나라도 잘되면 OK.



Approach	Top-N Acc		
	top-1	top-5	top-25
DEEPREGEX+FILTER			
Transferred Model	0%	0%	0%
+Curated Language	0%	0%	6.6%
GRAMMARREGEX+FILTER	3.2%	9.7%	11.3%
EMPTY SKETCH	4.8%	–	–
DEEPSKETCH			
Transferred Model	3.2%	3.2%	4.8%
GRAMMARSKETCH			
MAX COVERAGE	16.1%	34.4%	45.2%
MLE, MANUAL SKETCHES	**34.4%**	48.4%	53.2%
MML, NO SKETCH SUP	31.1%	**54.1%**	**56.5%**



On the Ability and Limitations of Transformers to Recognize Formal Languages

Satwik Bhattamishra, Kabir Ahuja, Navin Goyal

- https://arxiv.org/pdf/2009.11264.pdf

https://arxiv.org/pdf/2009.11264.pdf

Abstract

- Transformers have supplanted recurrent models in a large number of NLP tasks.

- the differences in their abilities to model different syntactic properties remain largely unknown.
    - LSTMs generalize very well on regular languages and have close connections with counter languages.
- Provide a construction of Transformers for a subclass of counter languages
- In experiments, we find that Transformers do well on this subclass, and their learned mechanism strongly correlates with our construction.

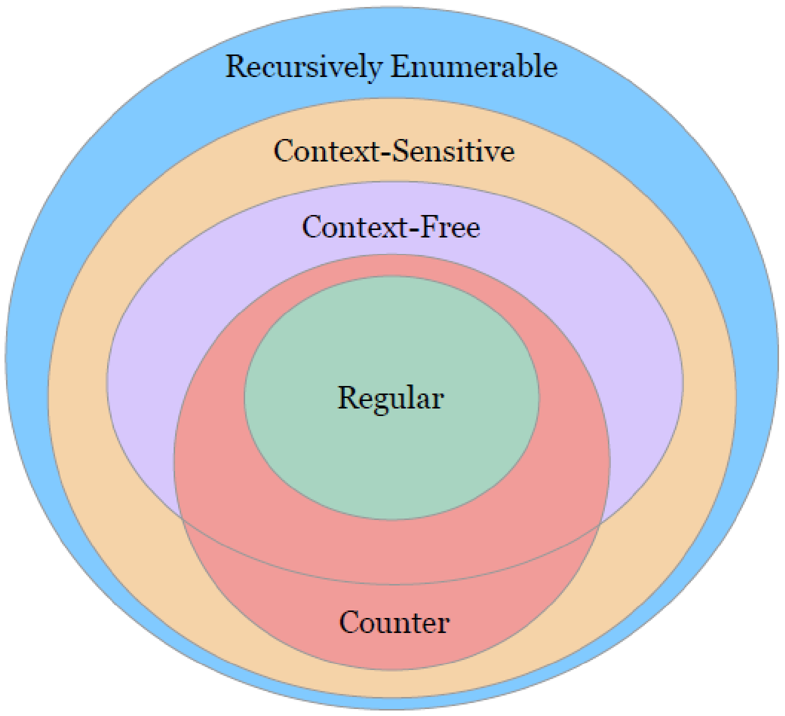


Figure 1: Counter languages form a strict superset of regular languages, and are a strict subset of context-sensitive languages. Counter and context-free languages have a nonempty intersection and neither set is contained in the other.