ASE 2023

InfeRE: Step-by-Step Regex Generation via Chain of Inference

Shuai Zhang, Xiaodong Gu, Yuting Chen, Beijun Shen

School of Electronic Information and Electrical Engineering,

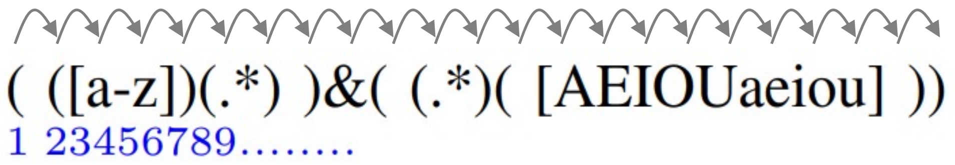
Shanghai Jiao Tong University, Shanghai, China

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Motivation

Q: lines starting with a lower-case letter and ending with vowel

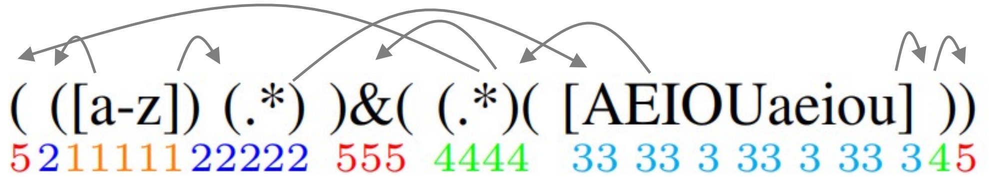
Generation order by autoregressive LM:



Intermediate steps

Step 1	lowercase	[a-z]
Step 2	start with	[a-z](.*)
Step 3	vowel	[AEIOUaeiou]
Step 4	end with	(.*)[AEIOUaeiou]
Step 5	and	(([a-z])(.*))&((.*)([AEIOUaeiou]))

The real order of text-matching processes:



Regex Generation Example

Generation order by autoregressive LM:

Q: lines starting with a lower-case letter and ending with vowel

$\Longrightarrow$ (

The real order of text-matching processes:

Q: lines starting with a lower-case letter and ending with vowel

$\Longrightarrow$

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

$\Rightarrow$

Generation order by autoregressive LM:

$\curvearrowright$

((

Q: lines starting with a lower-case letter and ending with vowel

$\Rightarrow$

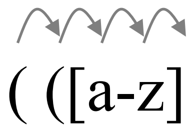
The real order of text-matching processes:

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

⇒

Generation order by autoregressive LM:



(([a-z]

Q: lines starting with a lower-case letter and ending with vowel

⇒

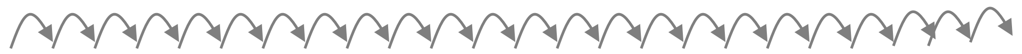
The real order of text-matching processes:

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

⇨

Generation order by autoregressive LM:



(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Q: lines starting with a lower-case letter and ending with vowel

⇨

The real order of text-matching processes:

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

$\Longrightarrow$

Generation order by autoregressive LM:

(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Q: lines starting with a lower-case letter and ending with vowel

$\Longrightarrow$

The real order of text-matching processes:

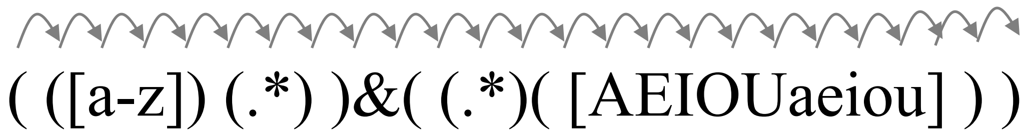
[a-z]

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

⇨

Generation order by autoregressive LM:

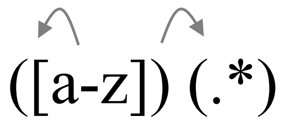


(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Q: lines <span style="color:red">starting with</span> a lower-case letter and ending with vowel

⇨

The real order of text-matching processes:



([a-z]) (.*)

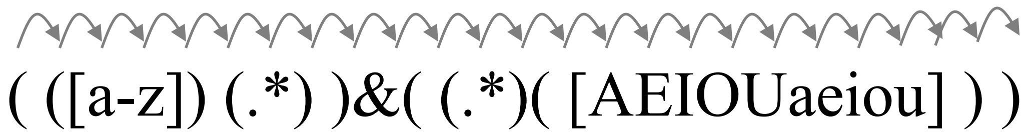
SHANGHAI JIAO TONG UNIVERSITY

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

⇒

Generation order by autoregressive LM:

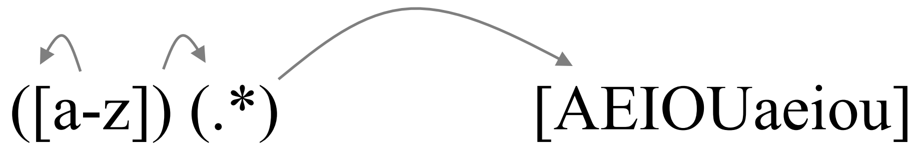


(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Q: lines starting with a lower-case letter and ending with vowel

⇒

The real order of text-matching processes:



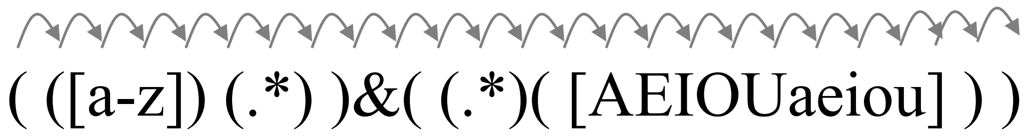
([a-z]) (.*) [AEIOUaeiou]

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

⟹

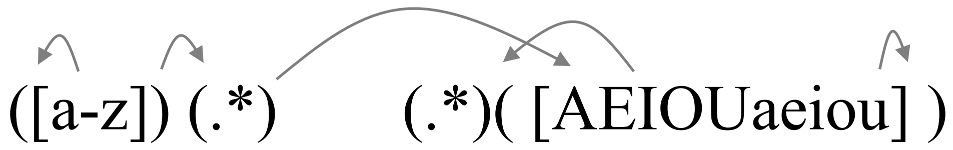
Generation order by autoregressive LM:


(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Q: lines starting with a lower-case letter and ending with vowel

⟹

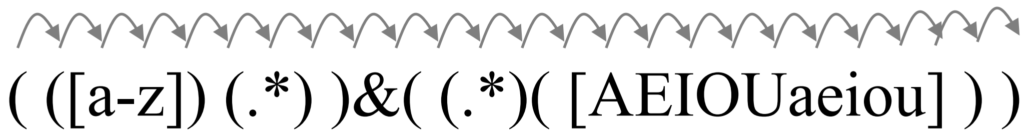
The real order of text-matching processes:


([a-z]) (.*)   (.*)([AEIOUaeiou])

Regex Generation Example

Q: lines starting with a lower-case letter and ending with vowel

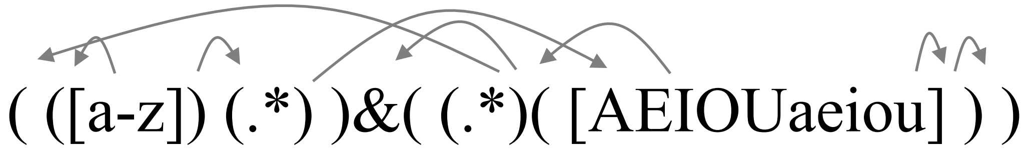
Generation order by autoregressive LM:



(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Q: lines starting with a lower-case letter and ending with vowel

The real order of text-matching processes:

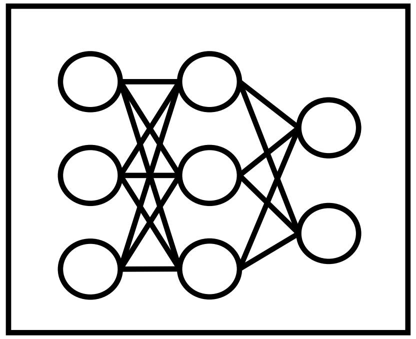


(([a-z]) (.*))&((.*)([AEIOUaeiou]))

Chain of Thought

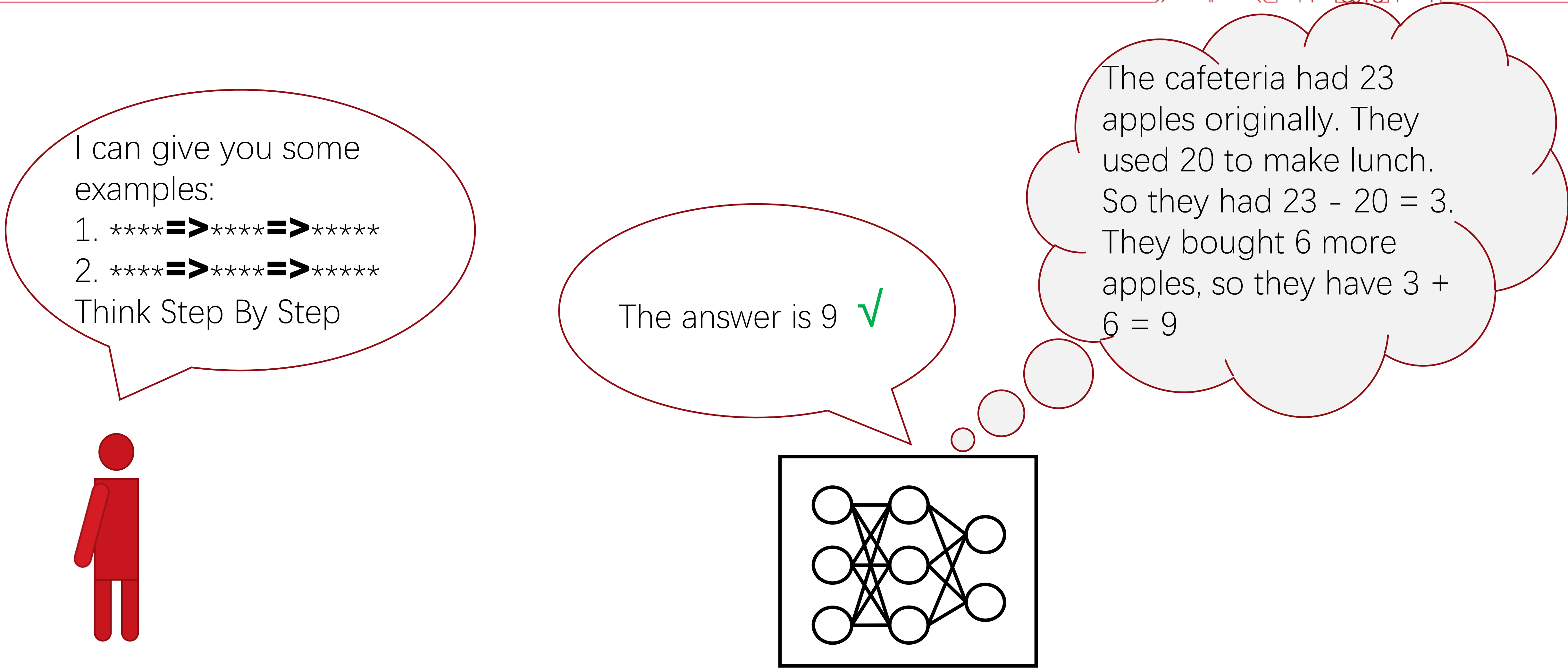
The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

The answer is 27 ✗

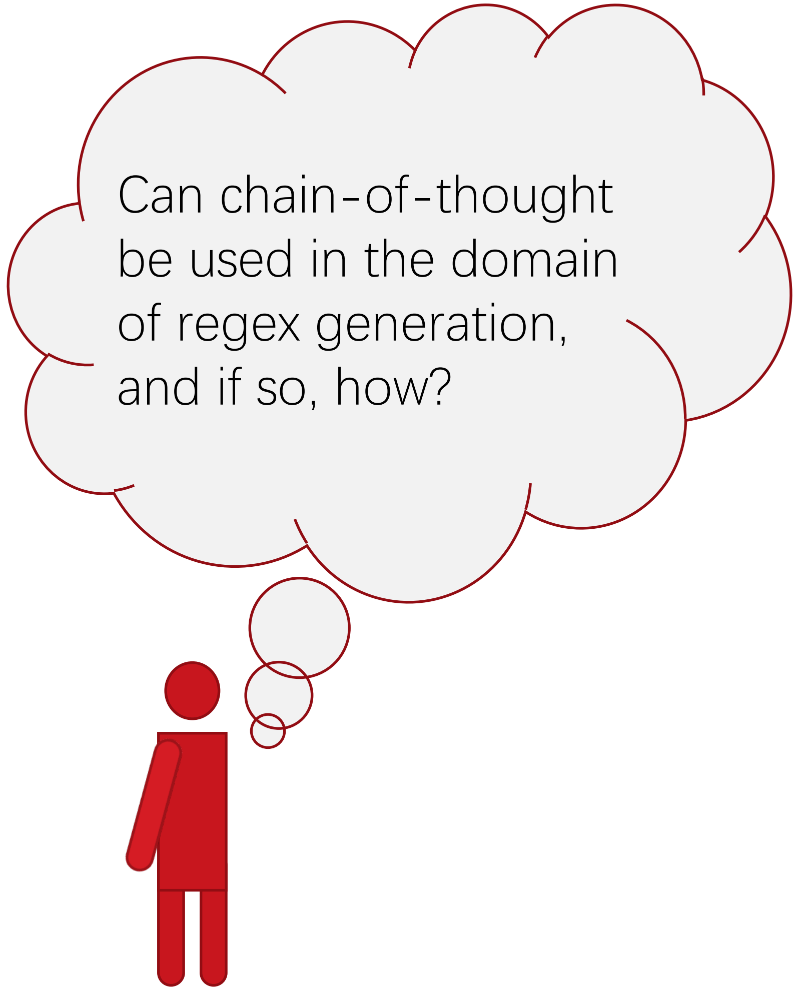


SHANGHAI JIAO TONG UNIVERSITY

Chain of Thought


I can give you some examples:
1. ******=>********=>*******
2. ******=>********=>*******
Think Step By Step
The answer is 9 √
The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9

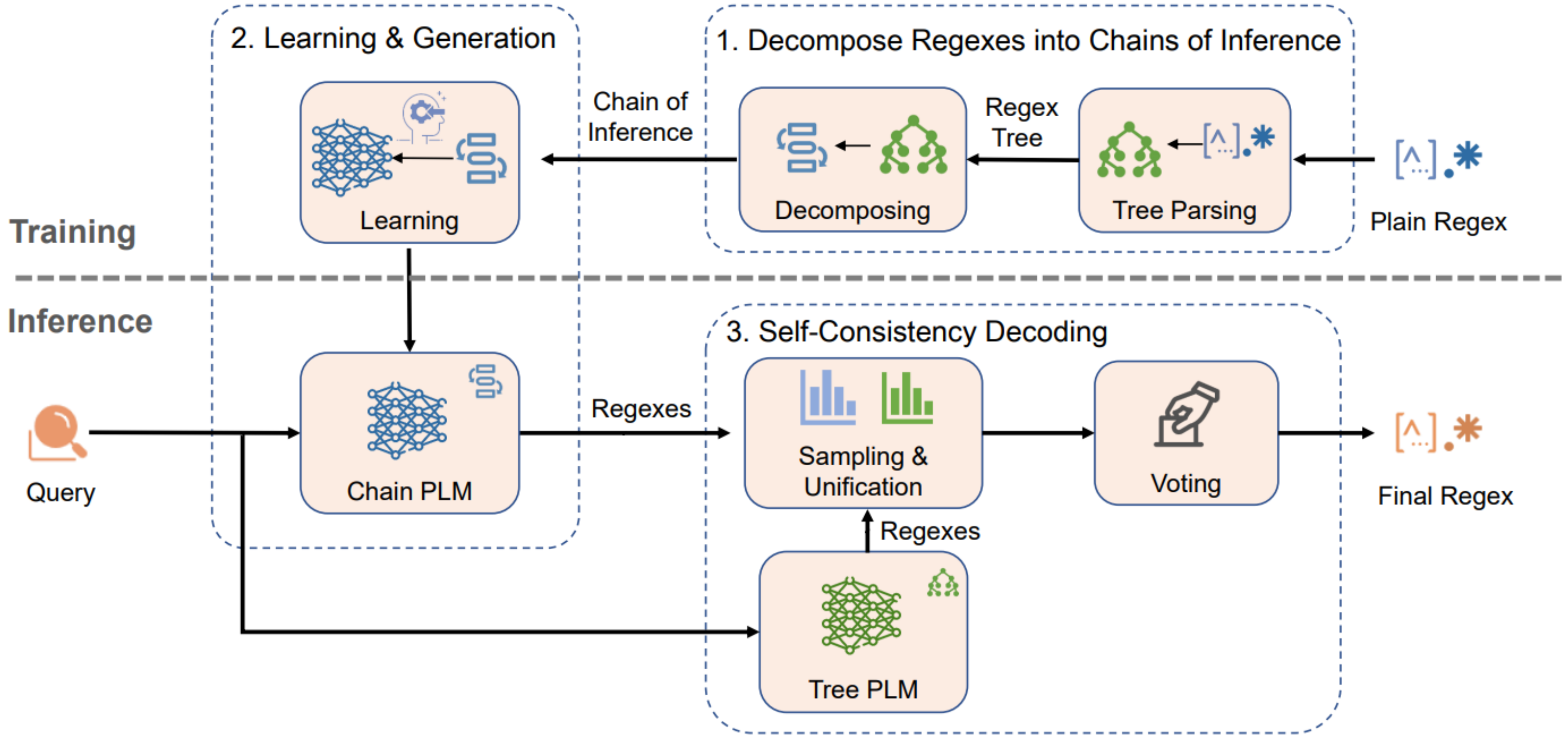
When Chain of Thought Met Regex Generation


Can chain-of-thought be used in the domain of regex generation, and if so, how?

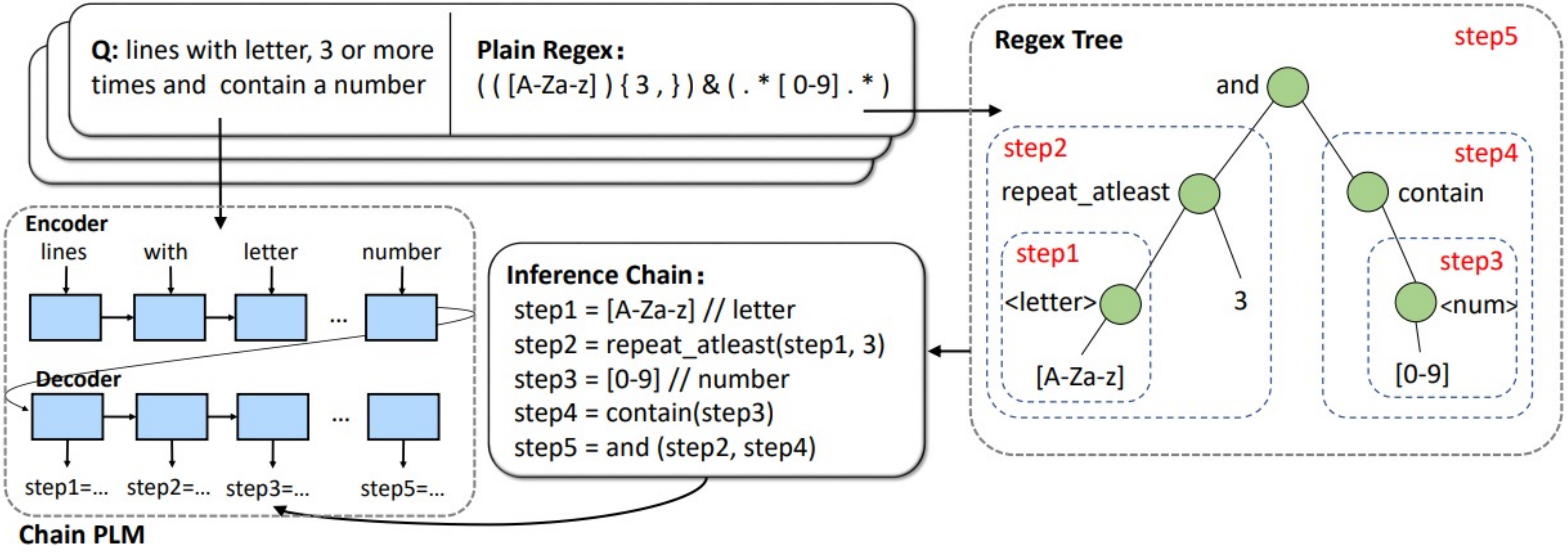
Main Challenges:

1. wide range of domains encompassed by existing LLM
2. considerable expense associated with the manual formulation of prompts

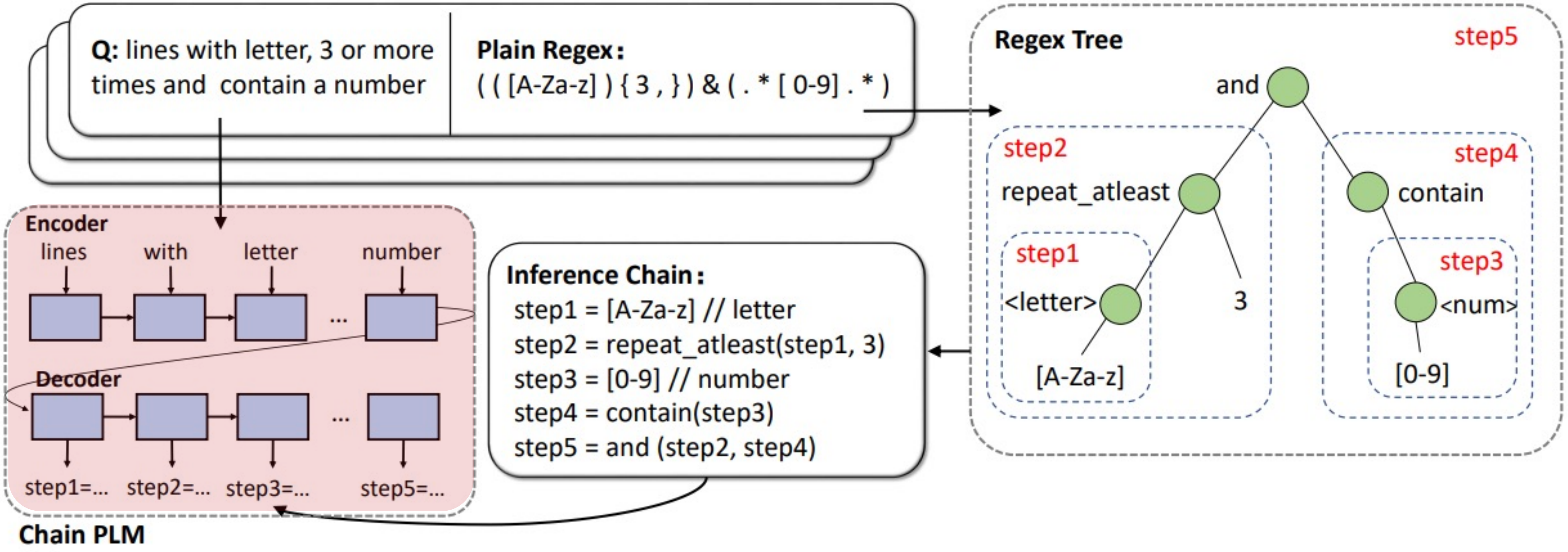
Overview of InfeRE



1. Creating Chains of Inference

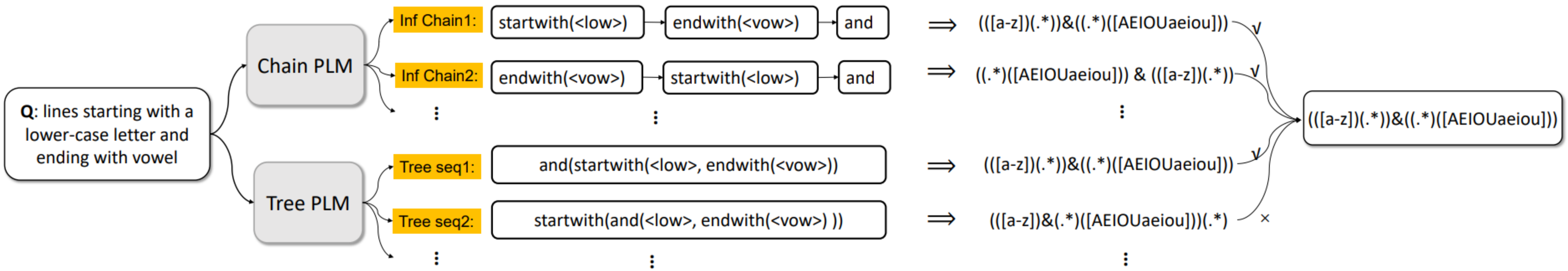


2. Learning & Generation



Inference Chain:
step1 = [A-Za-z] // letter
step2 = repeat_atleast(step1, 3)
step3 = [0-9] // number
step4 = contain(step3)
step5 = and (step2, step4)

3. Self-consistency Decoding


Q: lines starting with a lower-case letter and ending with vowel
Chain PLM
Tree PLM
Inf Chain1:
startwith(<low>)
endwith(<vow>)
and
(([a-z])(.*))&((.*)([AEIOUaeiou]))
√
Inf Chain2:
endwith(<vow>)
startwith(<low>)
and
((.*)([AEIOUaeiou])) & (([a-z])(.*))
√
Tree seq1:
and(startwith(<low>, endwith(<vow>))
(([a-z])(.*))&((.*)([AEIOUaeiou]))
√
Tree seq2:
startwith(and(<low>, endwith(<vow>)))
(([a-z])&(.*)([AEIOUaeiou]))(.*)
×
(([a-z])(.*))&((.*)([AEIOUaeiou]))

Evaluation

RQ1: How effective is InfeRE in generating regexes from natural language descriptions?

RQ2: What is the effect of chain of inference?

RQ3: What is the effect of self-consistency decoding and how does the number of output samples affect the performance of self-consistency decoding?

RQ4: What is the impact of data size on the performance of regex generation?

Experiment Setup

- **Datasets**

Dataset	Train(Fine-tune)	Valid	Test
NL-RX-Turk	6,500	1,000	2,500
KB13	618	206	206

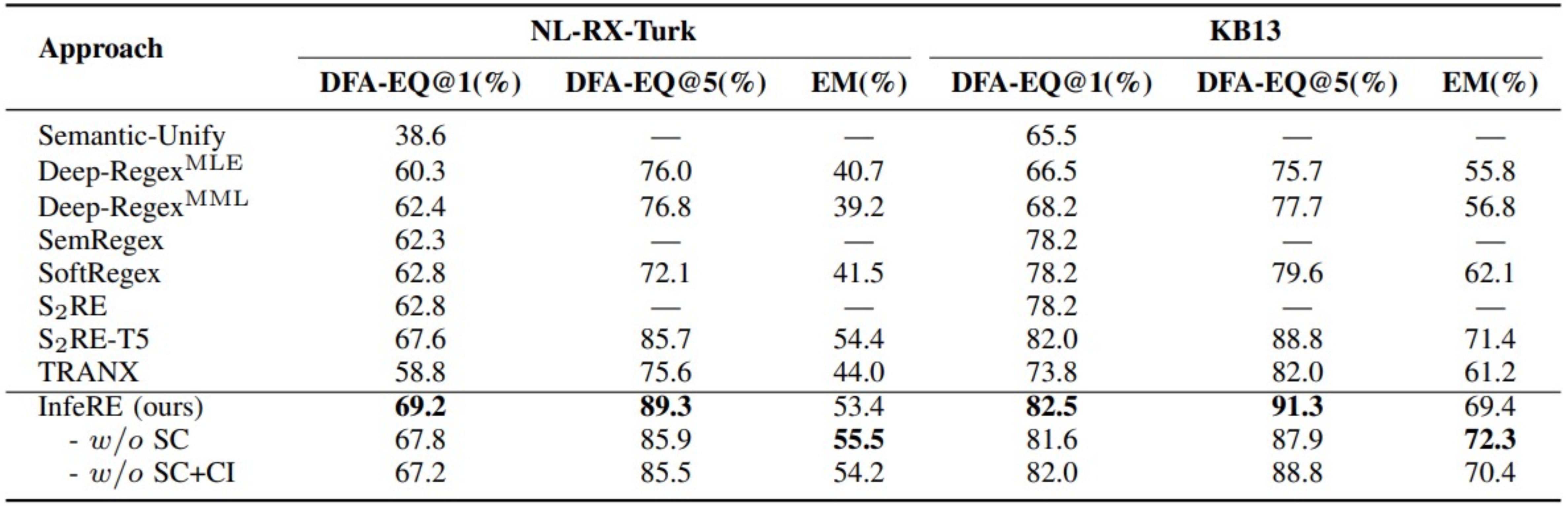
- **Metrics**
  - **EM** measures the ratio of regexes that exactly match ground—truth regexes
  - **DFA-EQ** measures the ratio of regexes that semantically equivalence by comparing their DFAs.

Comparison Methods

1. Semantic-Unify
2. Deep-Regex
3. SemRegex
4. SoftRegex
5. S_2RE
6. S_2RE-T5
7. TRANX

Experimental Results

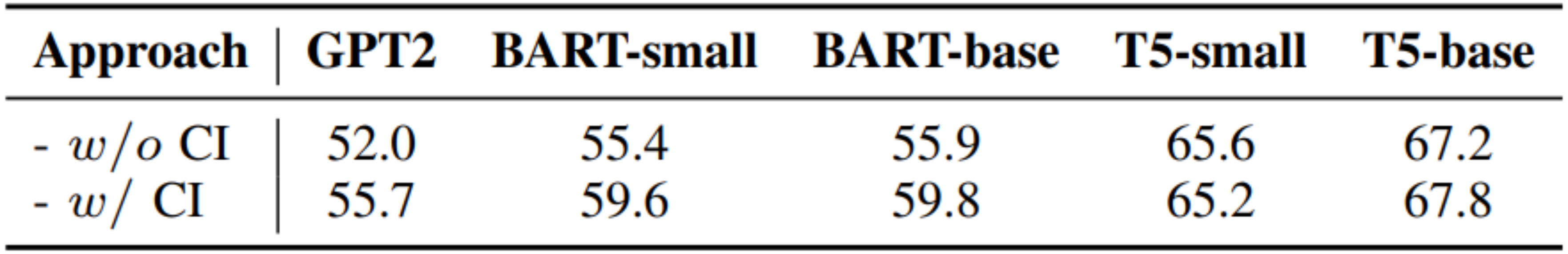
- Effectiveness of InfeRE in Regex Generation (RQ1)



Approach	NL-RX-Turk			KB13		
	DFA-EQ@1(%)	DFA-EQ@5(%)	EM(%)	DFA-EQ@1(%)	DFA-EQ@5(%)	EM(%)
Semantic-Unify	38.6	—	—	65.5	—	—
Deep-Regex^{MLE}	60.3	76.0	40.7	66.5	75.7	55.8
Deep-Regex^{MML}	62.4	76.8	39.2	68.2	77.7	56.8
SemRegex	62.3	—	—	78.2	—	—
SoftRegex	62.8	72.1	41.5	78.2	79.6	62.1
S_2RE	62.8	—	—	78.2	—	—
S_2RE-T5	67.6	85.7	54.4	82.0	88.8	71.4
TRANX	58.8	75.6	44.0	73.8	82.0	61.2
InfeRE (ours)	**69.2**	**89.3**	53.4	**82.5**	**91.3**	69.4
- *w/o* SC	67.8	85.9	**55.5**	81.6	87.9	**72.3**
- *w/o* SC+CI	67.2	85.5	54.2	82.0	88.8	70.4

Experimental Results

- Effect of Chain of Inference (RQ2)

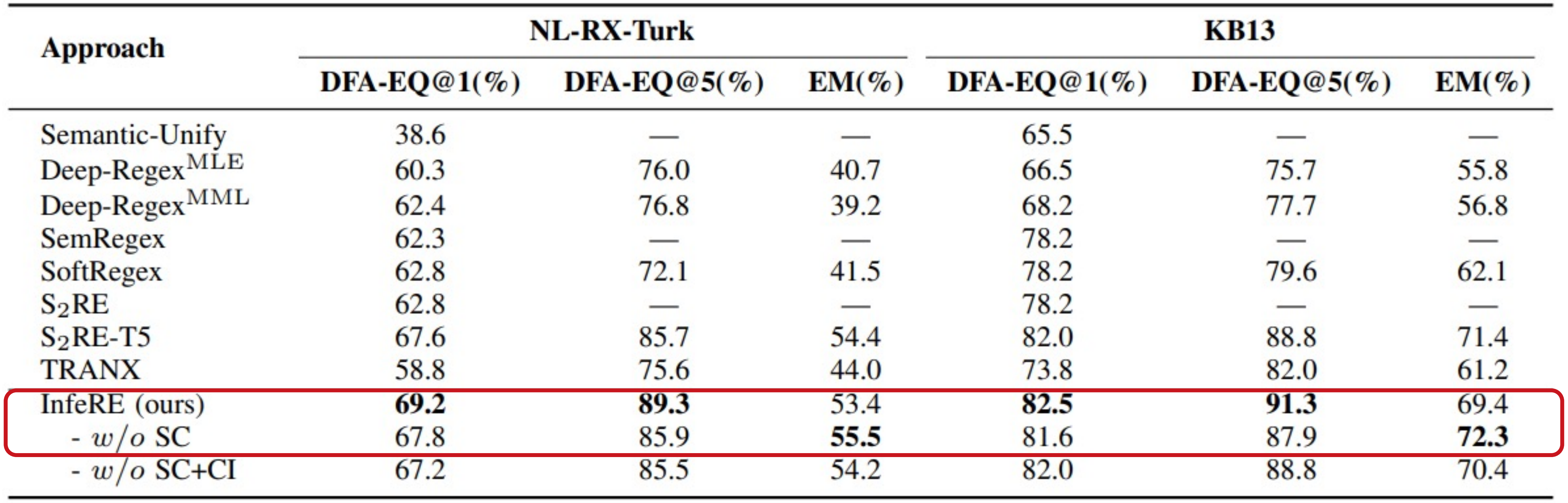


Approach	GPT2	BART-small	BART-base	T5-small	T5-base
- *w/o* CI	52.0	55.4	55.9	65.6	67.2
- *w/* CI	55.7	59.6	59.8	65.2	67.8

SHANGHAI JIAO TONG UNIVERSITY

Experimental Results

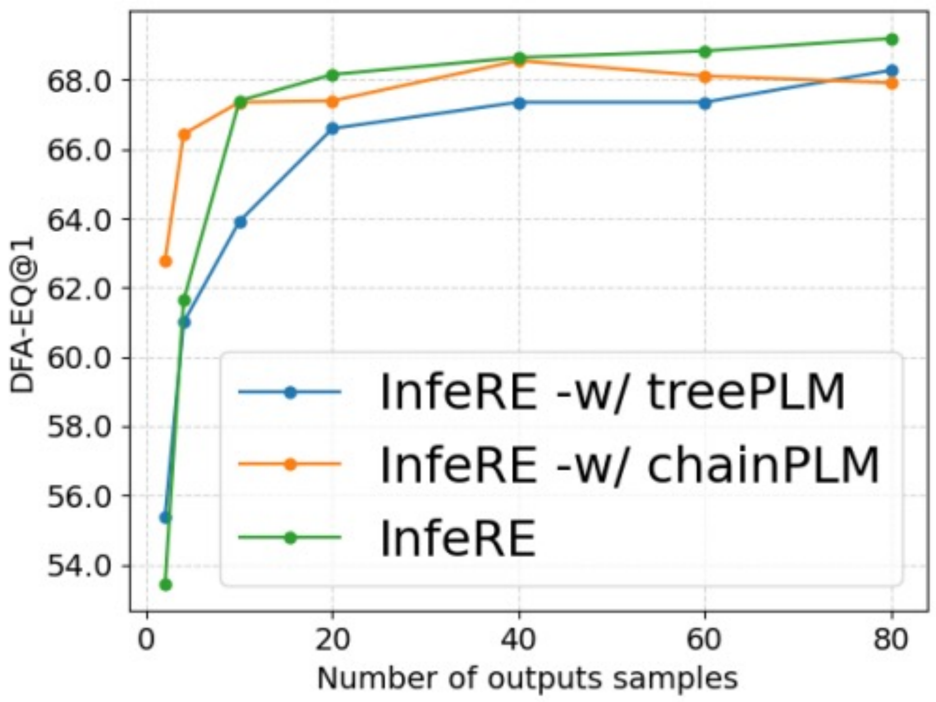
- Effect of Self-Consistency Decoding (RQ3)



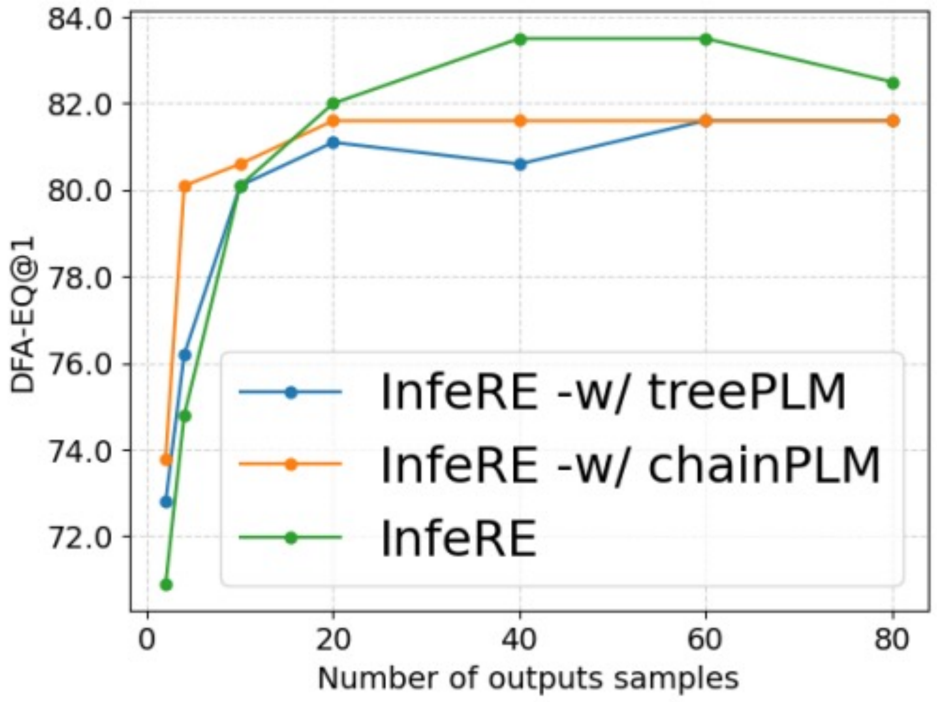
Approach	NL-RX-Turk			KB13		
	DFA-EQ@1(%)	DFA-EQ@5(%)	EM(%)	DFA-EQ@1(%)	DFA-EQ@5(%)	EM(%)
Semantic-Unify	38.6	—	—	65.5	—	—
Deep-Regex^{MLE}	60.3	76.0	40.7	66.5	75.7	55.8
Deep-Regex^{MML}	62.4	76.8	39.2	68.2	77.7	56.8
SemRegex	62.3	—	—	78.2	—	—
SoftRegex	62.8	72.1	41.5	78.2	79.6	62.1
S_2RE	62.8	—	—	78.2	—	—
S_2RE-T5	67.6	85.7	54.4	82.0	88.8	71.4
TRANX	58.8	75.6	44.0	73.8	82.0	61.2
InfeRE (ours)	**69.2**	**89.3**	53.4	**82.5**	**91.3**	69.4
- *w/o* SC	67.8	85.9	**55.5**	81.6	87.9	**72.3**
- *w/o* SC+CI	67.2	85.5	54.2	82.0	88.8	70.4

Experimental Results

- Effect of Self-Consistency Decoding (RQ3)



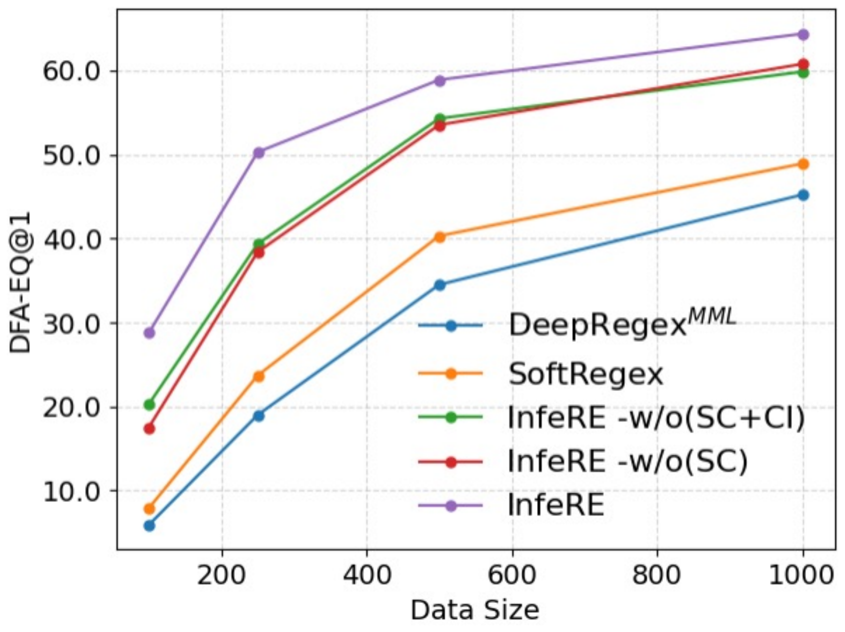
(a) NL-RX-Turk



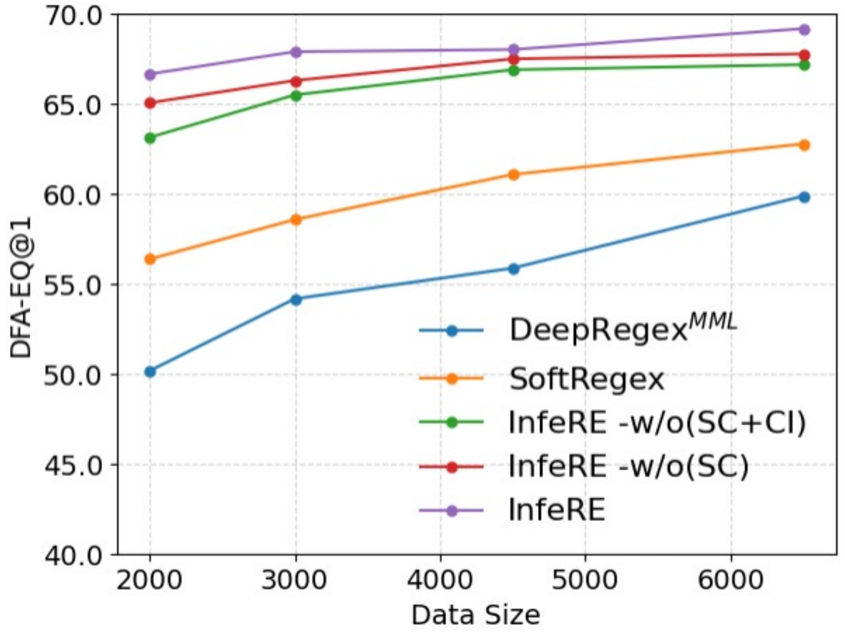
(b) KB13

Experimental Results

- Impact of Data Size (RQ4)



(a) Size from 100 to 1000



(b) Size from 2000 to 6500

Conclusion

InfeRE: Step-by-Step Regex Generation via Chain of Inference

- a novel paradigm of regex generation via chain of inference
- achieves significant improvement in regex generation

Future Work

- chain of inference in other forms
- more code intelligent tasks

Thank You!

Q&A