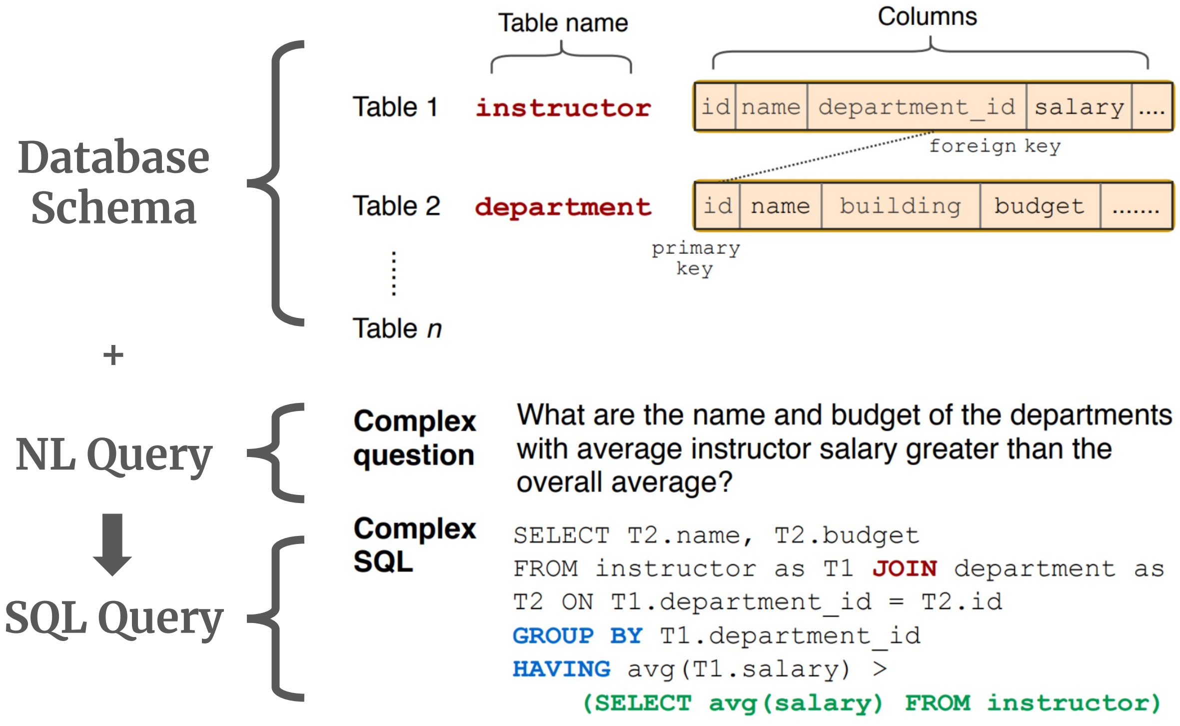
MetaSQL: A Generate-then-Rank Framework for Natural Language to SQL Translation

Yuankai Fan, Zhenying He, Tonghui Ren, Can Huang,
Yinan Jing, Kai Zhang, X.Sean Wang
Fudan University

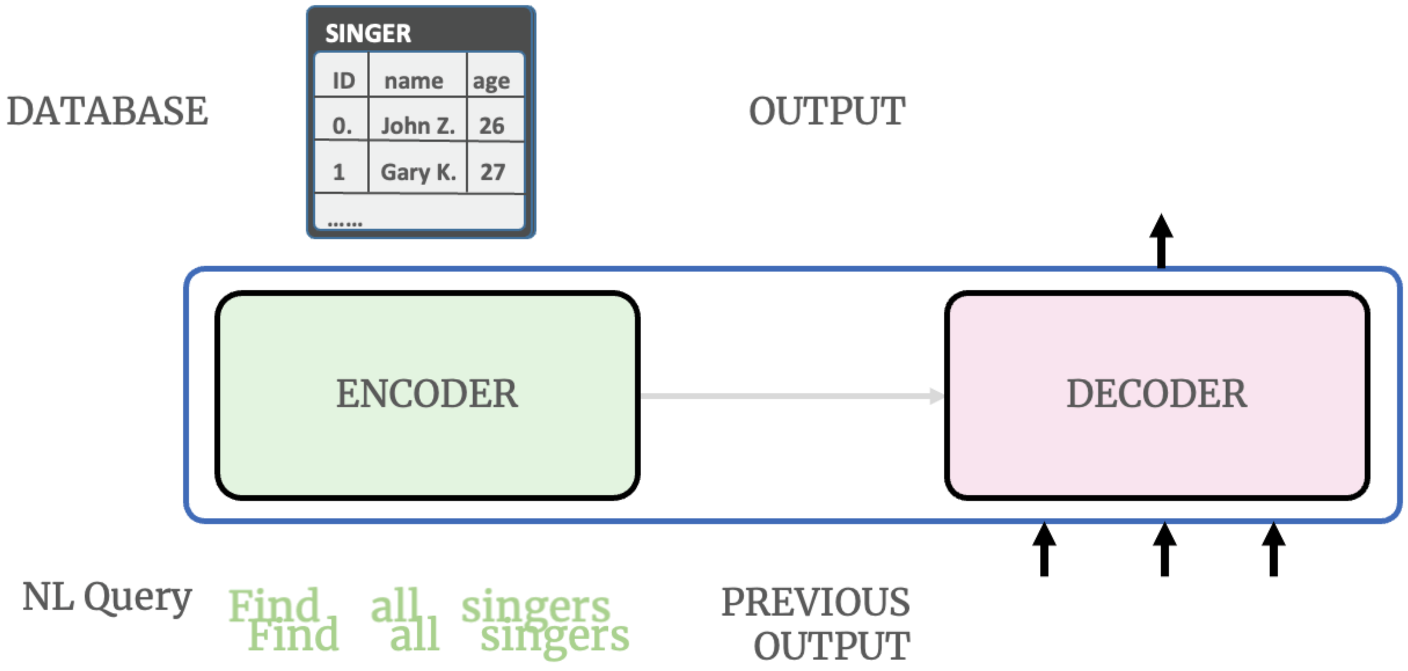
What is NL2SQL?



Spider example

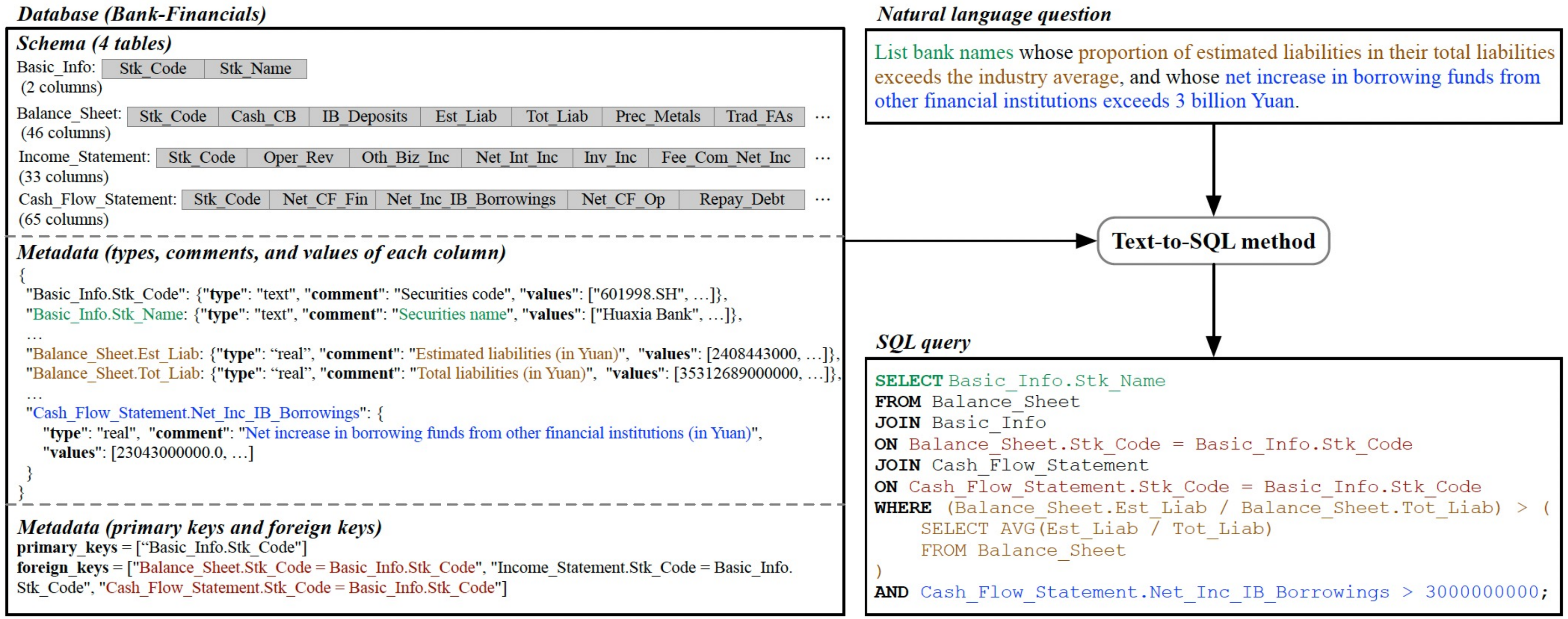
Mainstream Approach – Seq2Seq

- Based on **Sequence-to-sequence** framework
- Based on **pre-trained language models**



Mainstream Approach – LLM

- Large language models
  - **Out-of-the-box** or **fine-tuned** LLMs
- Use **In-context learning** with **thoughtfully crafted prompts**

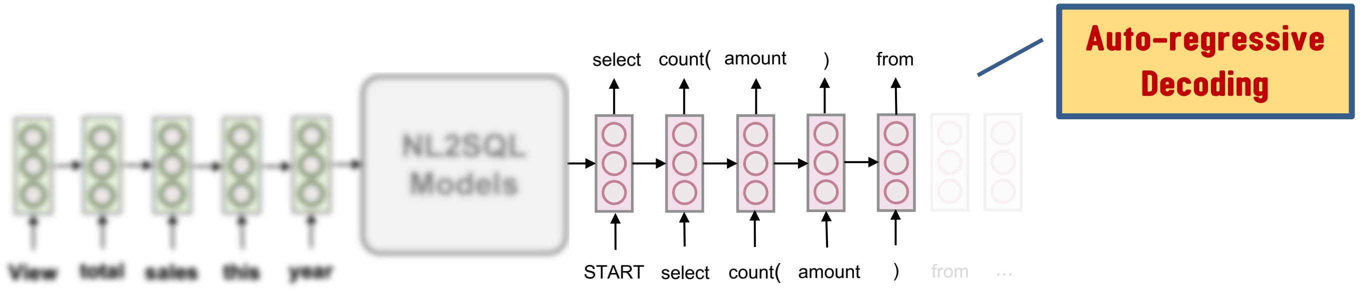


https://arxiv.org/abs/2402.16347

starcoder

Mainstream Approaches

- Mainstream approaches (either Seq2seq models or LLMs) primarily employ auto-regressive decoding to generate unique SQL queries



Existing Problem

- Problem: Auto-regressive decoding results in sub-optimal outputs
  - Lack of output diversity: beam search **tends to exhibit repetitiveness**

countryCode	language	isOfficial	percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7
ABW	Spanish	F	7.4
AFG	Balochi	F	0.9
AFG	Dari	T	32.1
AFG	Pashto	T	52.4
AFG	Turkmenian	F	
AFG	Uzbek	F	
BMU	English	T	
...			

code	name	continent	population
ABW	Aruba	North America	103000
AFG	Afghanistan	Asia	22720000
AIA	Anguilla	North America	8000
BMU	Bermuda	North America	65000
CHE	Switzerland	Europe	7160400
CMR	Cameroon	Africa	15085000
COL	Columbia	South America	42321000

NL Query: *What are the country codes for countries that do not speak English?*

SQL (Gold):
```
SELECT countrycode FROM CountryLanguage EXCEPT
SELECT countrycode FROM CountryLanguage WHERE language='English'
```

Beam search outputs from LGESQL model [11]

Top-1 SQL: `SELECT countrycode FROM CountryLanguage WHERE language!='value'`
Top-2 SQL: `SELECT code FROM CountryLanguage JOIN Country WHERE language!='value'`
Top-3 SQL: `SELECT countrycode FROM CountryLanguage WHERE language<='value'`
Top-4 SQL: `SELECT code FROM CountryLanguage JOIN Country WHERE surfacearea!='value'`
Top-5 SQL: `SELECT code FROM CountryLanguage JOIN Country WHERE countrycode!='value'`

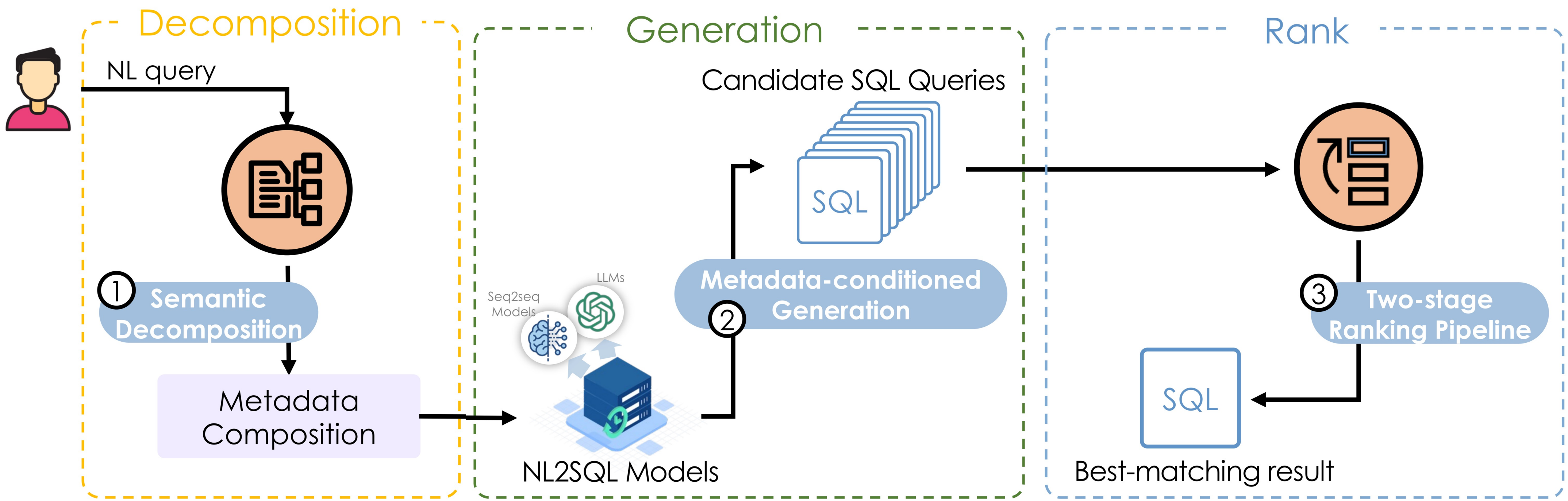
  - Lack of global context awareness: with the incremental nature of sequential generation, it may encounter local optima as **only partial context is considered**

Can We Have a Unified Framework?

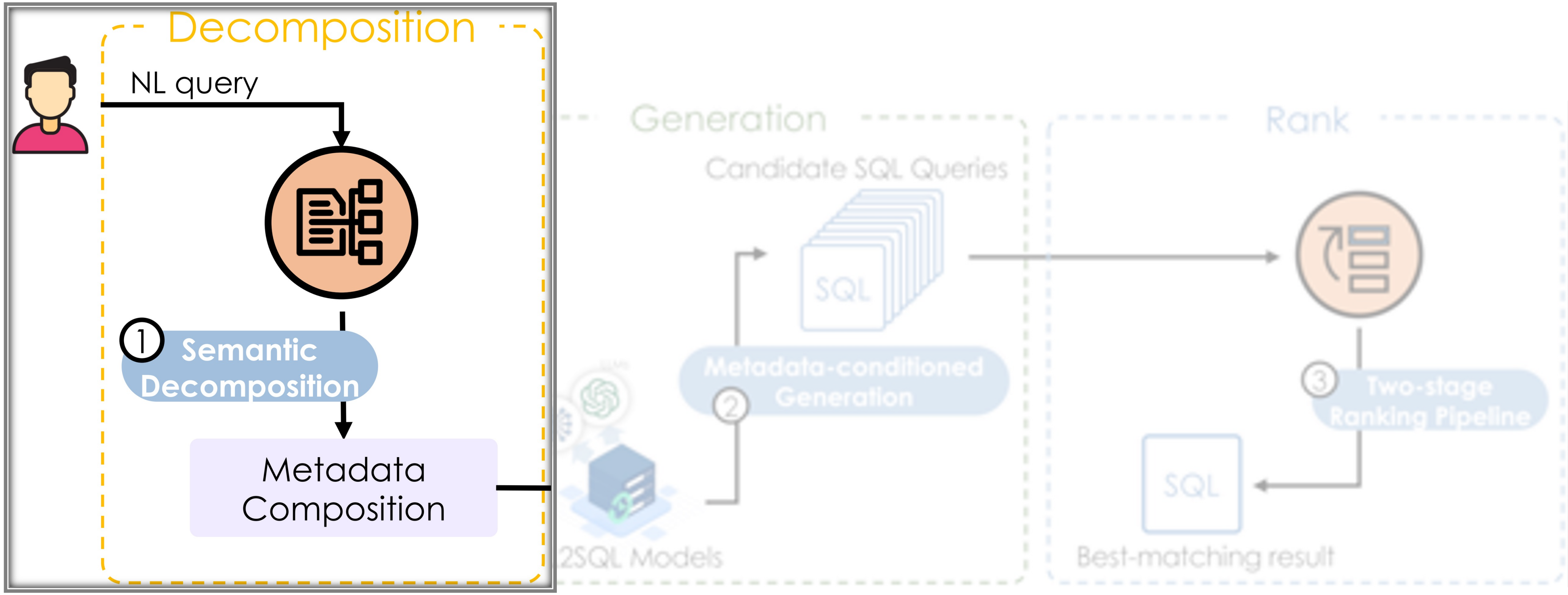
- Can we develop a unified framework for NL2SQL models, to improve their **vanilla** auto-regressive decoding?

What is MetaSQL?

- MetaSQL: NL2SQL with Metadata
  ① Semantics Decomposition
  ② Metadata-conditioned Generation
  ③ Learning-to-rank

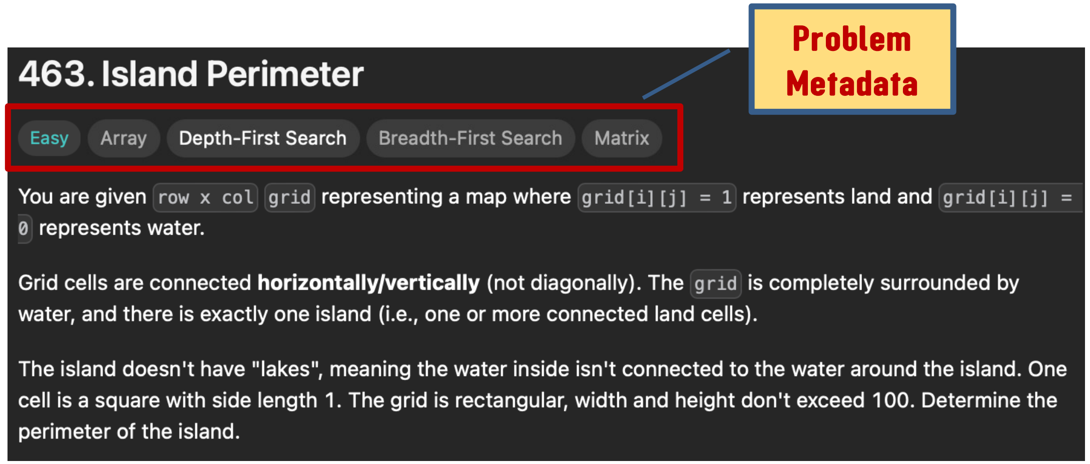


Decomposition



Decomposition

- Inspiration from competitive-level coding
  - Solve problem **with problem tags**
- Decompose the meaning of NL into **a set of query metadata**
  - Opetator Tag (e.g., *WHERE*)
  - Hardness Value (e.g., 200)
  - Correctness Indicator (e.g., *correct*)
- Frame the NL-to-metadata as a **multi-label classification problem**



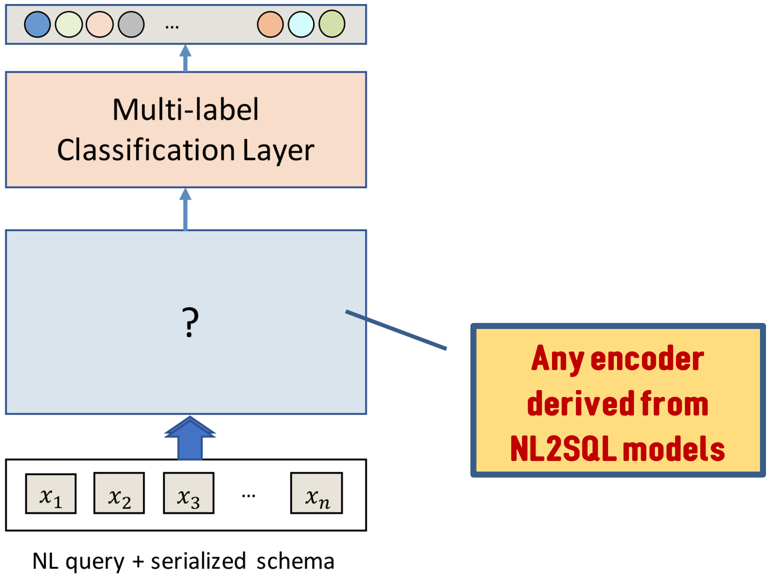
463. Island Perimeter

Easy Array Depth-First Search Breadth-First Search Matrix

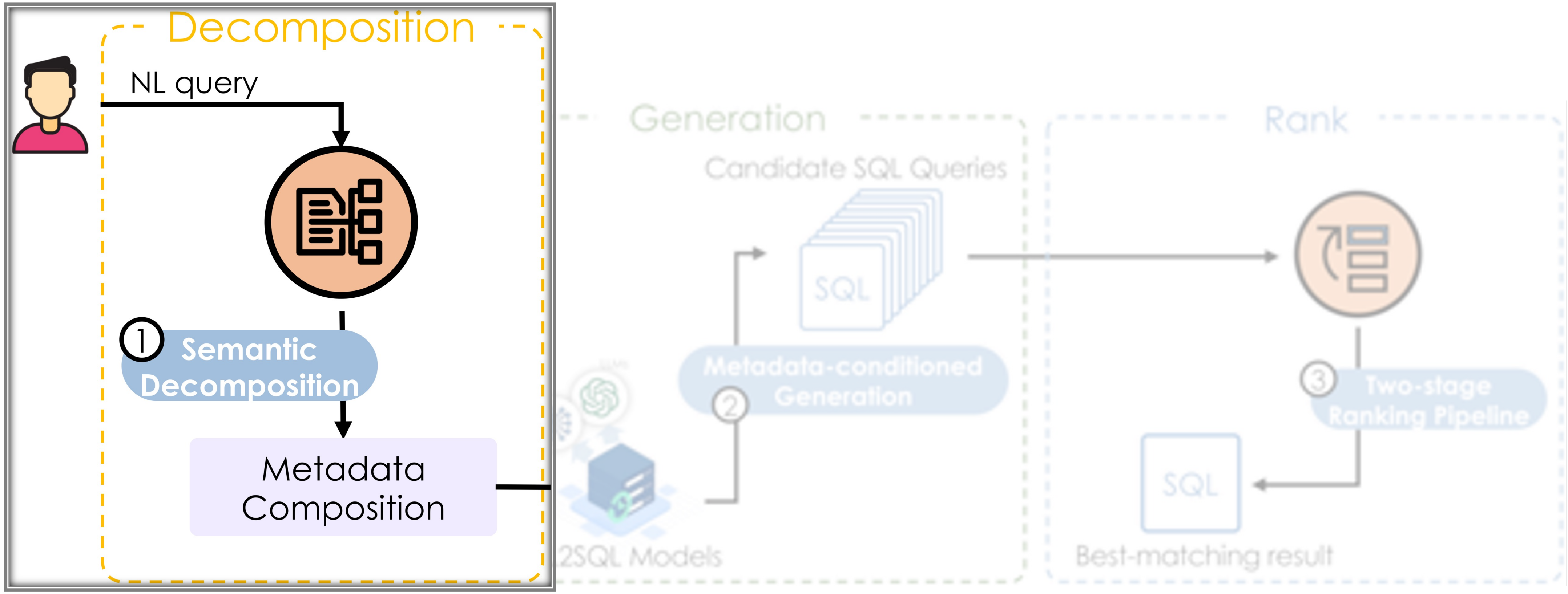
You are given `row x col` `grid` representing a map where `grid[i][j] = 1` represents land and `grid[i][j] = 0` represents water.

Grid cells are connected **horizontally/vertically** (not diagonally). The `grid` is completely surrounded by water, and there is exactly one island (i.e., one or more connected land cells).

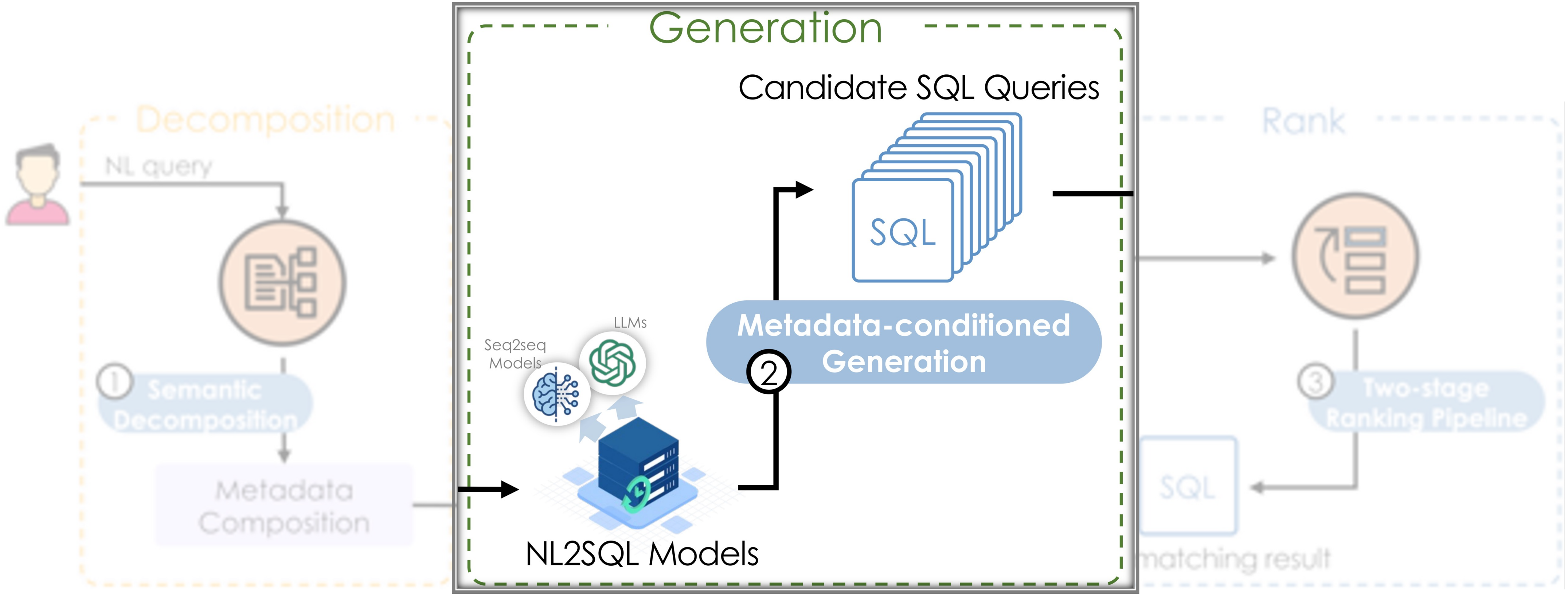
The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island. One cell is a square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.



Decomposition



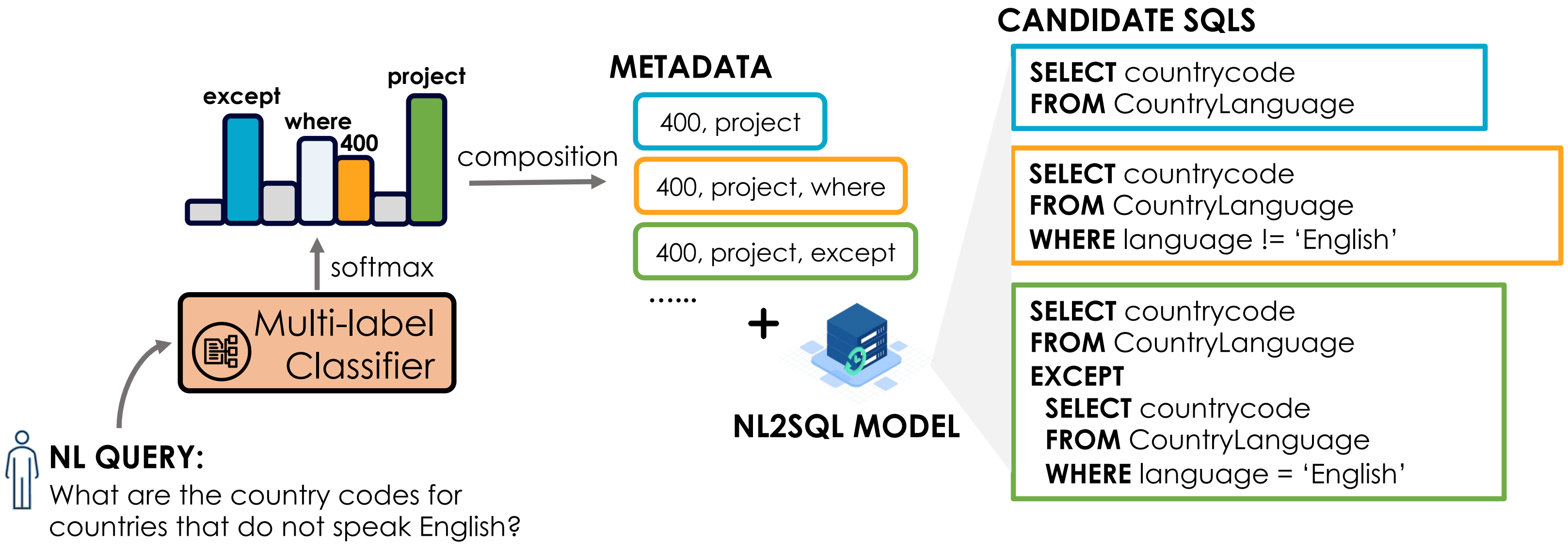
Generation



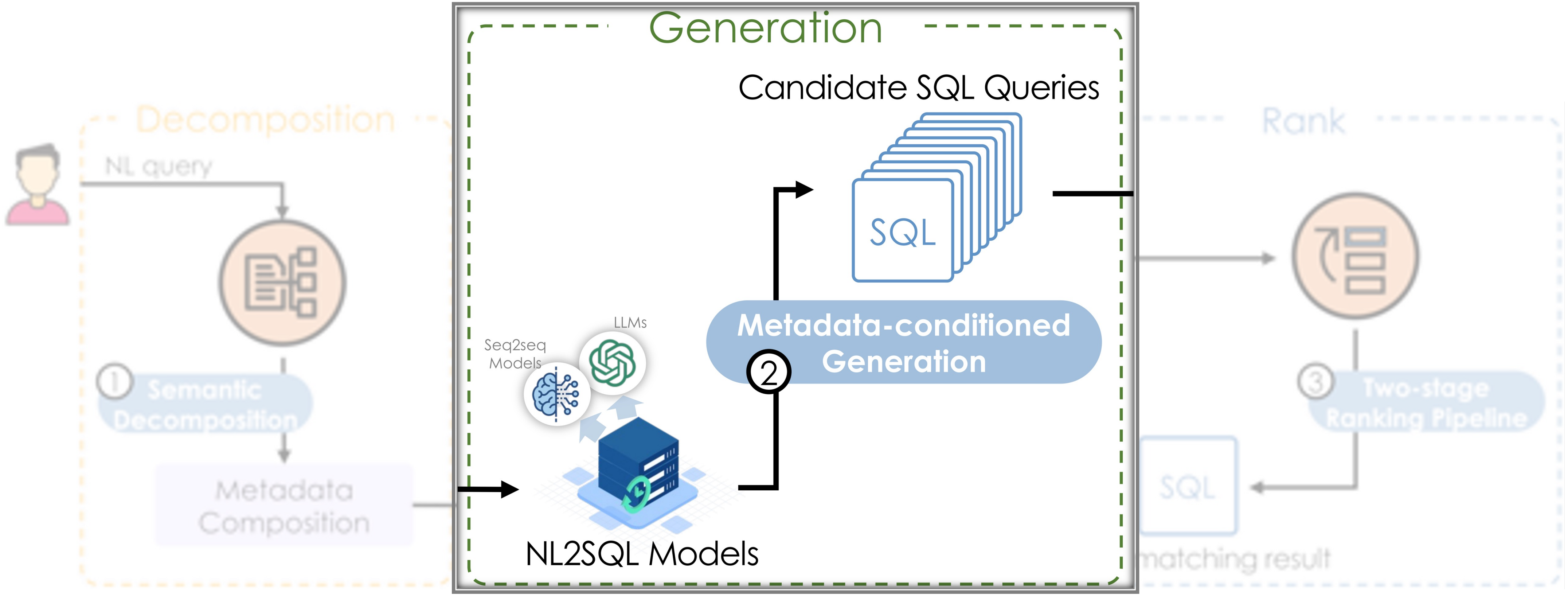
Generation
Candidate SQL Queries
SQL
Metadata-conditioned Generation
2
LLMs
Seq2seq Models
NL2SQL Models
Decomposition
NL query
1 Semantic Decomposition
Metadata Composition
Rank
3 Two-stage Ranking Pipeline
SQL
matching result

Generation

- Valinna NL2SQL -> Metadata-conditioned NL2SQL
- Metadata as language prompt
  - For Seq2seq models, re-train by having metadata as extra inputs
  - For LLMs, use in-context learning with metadata infomraiton

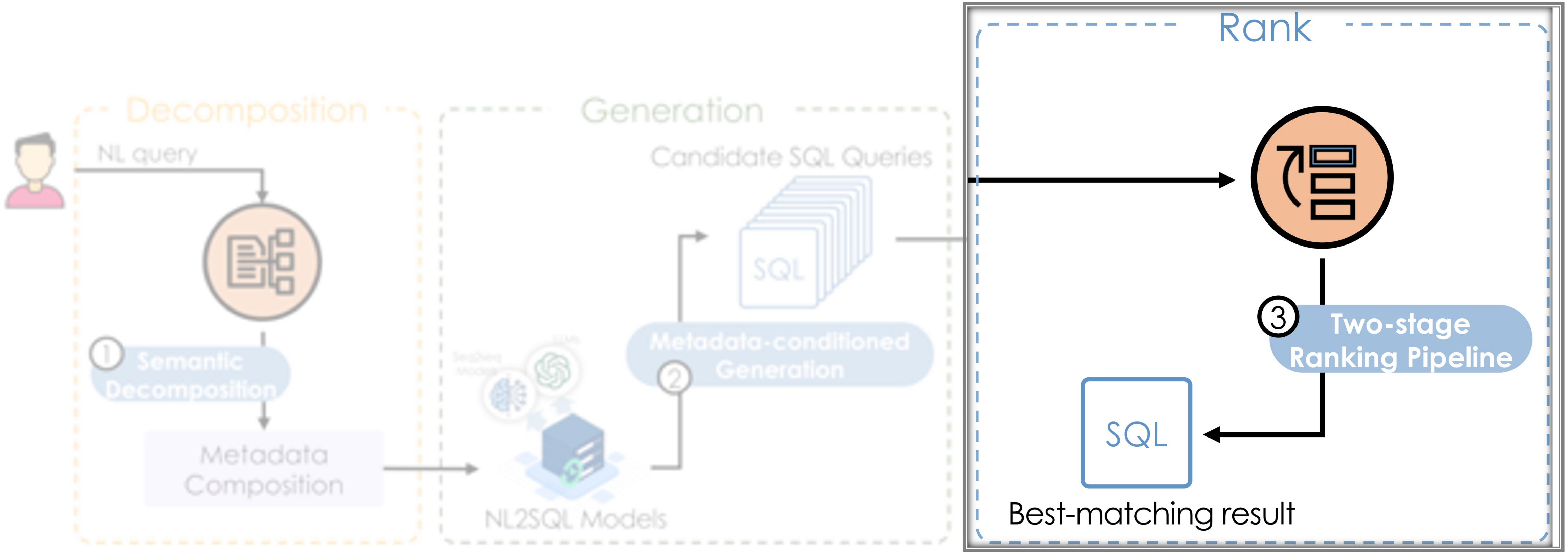


Generation



Decomposition
NL query
① Semantic Decomposition
Metadata Composition
Generation
Candidate SQL Queries
SQL
Metadata-conditioned Generation
②
LLMs
Seq2seq Models
NL2SQL Models
Rank
③ Two-stage Ranking Pipeline
SQL
matching result

Ranking


Decomposition
NL query
① Semantic Decomposition
Metadata Composition
Generation
Candidate SQL Queries
SQL
② Metadata-conditioned Generation
NL2SQL Models
Rank
③ Two-stage Ranking Pipeline
SQL
Best-matching result

Ranking

- Rank based on *semantic similarity* with NL and SQL
- **Two-stage ranking pipeline**
  1) First-stage ranking for fast filtering
     - *Dual-tower architecture with similarity function*
  2) Second-stage ranking for top-1 selection

Ranking

- Two-stage Ranking pipeline
  - First-stage ranking for fast filtering
  - **Second-stage ranking** for top-1 selection

Finer-grained mismatch

NL Query	*Find the last name of the student who has a cat that is age 3.*	Similarity Score
Mismatched SQL Queries	`SELECT student.lname FROM student JOIN has_pet JOIN pets WHERE pets.pet_age=3 AND pets.pettype='cat'`	0.76
	`SELECT student.lname FROM student JOIN has_pet JOIN pets WHERE pets.pettype='cat' AND pets.pet_age=3`	0.82
	`SELECT student.lname, pets.pettype FROM student JOIN has_pet JOIN pets WHERE pets.pet_age=3 AND pets.pettype='cat'`	0.73
Matched SQL Query	`SELECT student.lname FROM student JOIN has_pet JOIN pets WHERE pets.pettype='cat' AND pets.pet_age=3`	0.72

- Problem: Current ranking primarily rely on sentence-level supervision to distinguish matched and mismatched candidates
  - The semantic mismatch usually happens in finer grain, *i.e.*, phrase level

Ranking

- Enhanced Second-stage ranking model
  - Incorporate multi-grained signals (sentence-level and phrase-level)
  - Construct multi-scale loss
    - NL-to-SQL Global Loss

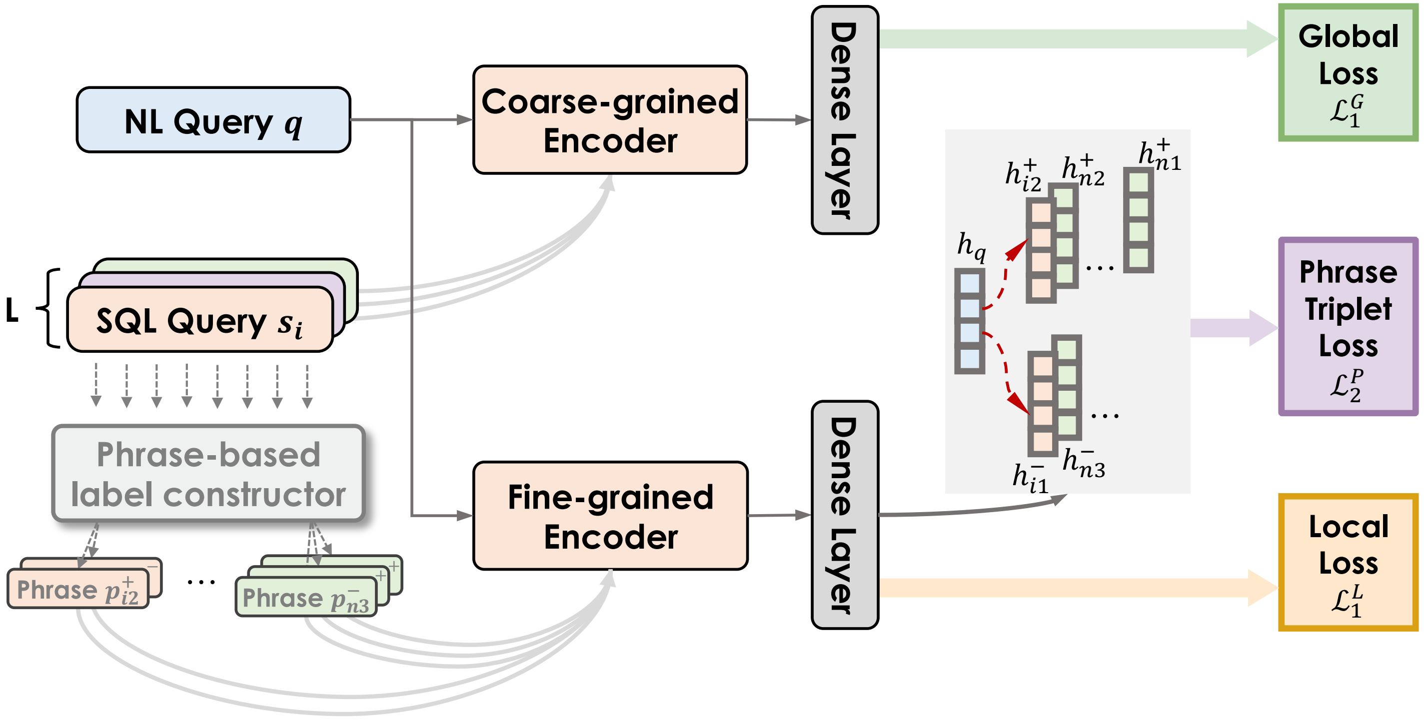
$$\mathcal{L}_0^G = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i^G - y_i)^2$$

    - NL-to-Phrase Local Loss

$$\mathcal{L}_1^L = \frac{1}{N}\sum_{i=1}^{N}(\sum_{k=1}^{K}\hat{y}_{i,k}^L - y_i)^2$$

    - Phrase Triplet Loss

$$\mathcal{L}_3^P = TriL_\alpha(h_{q_i}, h_{s_i}^+, h_{s_i}^-)$$



Evaluation

- Benchmarks
  - ✓ **Spider**
  - ✓ **ScienceBenchmark**

- Metrics
  - ✓ Translation Accuracy *(syntactic equivalence)*
  - ✓ Execution Accuracy
  - ✓ Translation Precision/MRR *(for ranking evaluation)*

Evaluation

- Overall results on Spider and ScienceBenchmark datasets

NLIDB Models	SPIDER$_{Dev}$ EM%	SPIDER$_{Dev}$ EX%	SPIDER$_{Test}$ EM%	SPIDER$_{Test}$ EX%	SCIENCEBENCHMARK[1] EM%(ONCOMX)	SCIENCEBENCHMARK[1] EM%(CORDIS)	SCIENCEBENCHMARK[1] EM%(SDSS)
BRIDGE [36]	68.7	68.0	65.0	64.3	16.5	23.0	5.0
BRIDGE+METASQL	70.5(↑1.8)	69.2(↑1.2)	-	-	**18.6**(↑2.1)	**25.0**(↑2.0)	**7.0**(↑2.0)
GAP [9]	71.8	34.9	69.7	-	33.0	20.0	5.0
GAP+METASQL	73.4(↑1.6)	37.2(↑2.3)	-	-	**35.0**(↑2.0)	20.0	**6.0**(↑1.0)
LGESQL [11]	75.1	36.3	72.0	34.2	41.7	24.0	4.0
LGESQL+METASQL	**77.4**(↑2.3)	**42.0**(↑5.7)	**72.3**(↑0.3)	**55.7**(↑21.5)	**42.7**(↑1.0)	**28.0**(↑4.0)	**12.0**((↑8.0)
RESDSQL$_{LARGE}$ [12]	75.8	80.1	-	-	42.7	29.0	4.0
RESDSQL$_{LARGE}$+METASQL	76.9(↑1.1)	81.5(↑1.4)	-	-	**49.7**(↑7.0)	**33.0**(↑4.0)	**10.0**(↑6.0)
CHATGPT	51.5	65.3	-	-	51.2	40.0	11.0
CHATGPT+METASQL	**65.1**(↑13.6)	**74.2**(↑8.9)	-	-	**53.2**(↑2.0)	**42.0**(↑2.0)	**16.0**(↑5.0)
GPT-4	54.3	67.4	-	-	65.7	42.0	15.0
GPT-4+METASQL	**69.6**(↑15.3)	**76.8**(↑9.4)	-	-	**68.6**(↑2.9)	42.0	**17.6**(↑2.6)

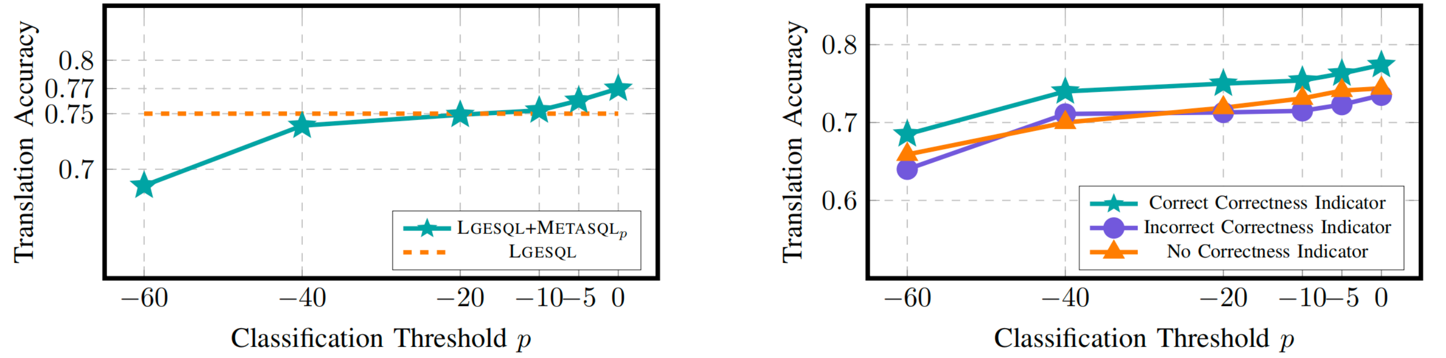
NL2SQL Models	Easy	Medium	Hard	Extra Hard	Overall
BRIDGE	91.1	73.3	54.0	39.2	68.7
BRIDGE+METASQL	89.1(↓2.0)	**75.3**(↑2.0)	**58.0**(↑4.0)	42.8(↑3.6)	70.5
GAP	91.5	74.2	64.4	44.2	71.8
GAP+METASQL	91.1(↓0.4)	**78.0**(↑3.8)	**64.9**(↑0.5)	43.4(↓0.8)	73.4
LGESQL	91.9	77.4	65.5	53.0	75.1
LGESQL+METASQL	94.0(↑2.1)	**81.4**(↑4.0)	**70.1**(↑4.6)	49.4(↓3.6)	77.4
RESDSQL$_{LARGE}$	90.3	82.7	62.6	47.0	75.8
RESDSQL$_{LARGE}$+METASQL	92.5(↑2.2)	**83.9**(↑1.2)	**64.1**(↑1.5)	48.2(↑1.2)	76.9
CHATPGT	84.7	51.3	39.7	15.1	51.5
CHATPGT+METASQL	89.0(↑3.3)	**70.6**(↑19.3)	**55.2**(↑15.5)	24.4(↑9.3)	65.1
GPT-4	82.2	56.3	51.3	14.6	54.3
GPT-4+METASQL	91.1(↑8.9)	**74.7**(↑18.4)	**64.1**(↑12.8)	36.1(↑21.5)	69.6

NL2SQL Models	MRR	Precision@1	Precision@3	Precision@5
BRIDGE+METASQL	73.8	70.5	76.7	78.6
GAP+METASQL	76.4	73.4	79.9	**81.0**
LGESQL+METASQL	78.2	76.8	79.6	**80.9**
RESDSQL$_{LARGE}$+METASQL	78.8	77.2	80.6	**80.1**
CHATGPT+METASQL	52.6	51.5	64.3	64.5
GPT-4+METASQL	69.6	69.6	72.5	72.5

MetaSQL can consistently improve model performance over six baseline models

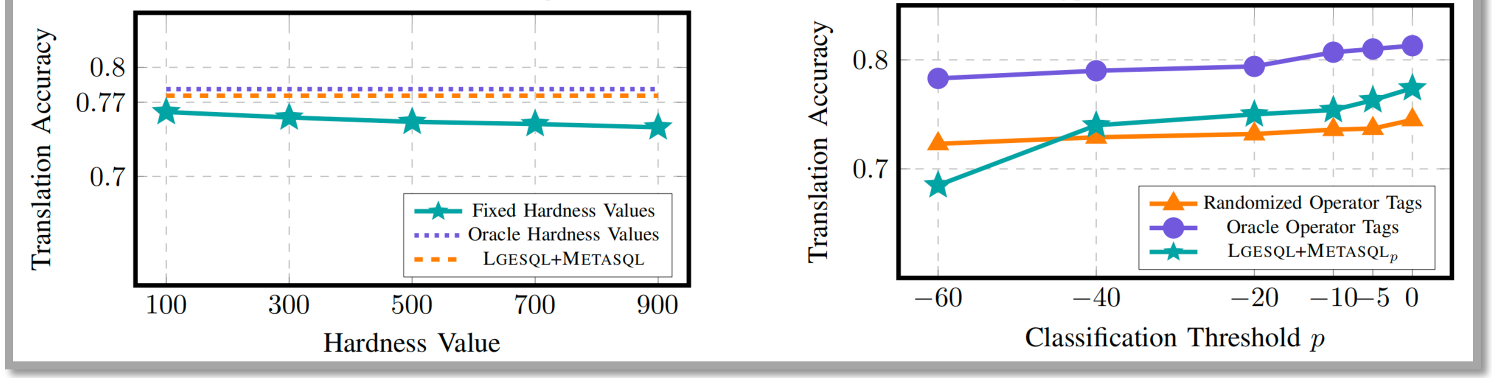
Evaluation

- Results on metadata sensitivity analysis



(a) Translation accuracy of METASQL with different classification thresholds. The dotted line shows the results of original LGESQL.

(b) Translation accuracy of METASQL applied to LGESQL using different correct indicator-type of metadata information.



MetaSQL is relatively not sensitive to hardness values, while exhibits greater sensitivity to operator tags and correctness indicators

Conclusion

- Metadata bring good hints 👍 for NL2SQL
- MetaSQL: A generate-then-rank framework for NL2SQL
  - **Coarse-to-fine > End-to-end!**
- GAR: A generate-then-rank approach for NL2SQL

Yuankai Fan <fanyuankai@fudan.edu.cn>

https://github.com/Kaimary/MetaSQL

Yuankai Fan, Zhenying He, Tonghui Ren, Dianjun Guo, Lin Chen, Ruisi Zhu, Guanduo Chen, Yinan Jing, Kai Zhang, X.Sean Wang. GAR: A Generate-and-Rank Approach for Natural Language to SQL Translation, ICDE '23, 11—122.