AtomSQL: Interactive Disambiguation of NL-to-SQL via User-Guided Atom-Level Alignment

Aritra Mazumder, Parth Desai, Fuheng Zhao, Anna Fariha

Kahlert School of Computing, University of Utah

VLDB 2026 BOSTON
DEMONSTRATION

THE UNIVERSITY of UTAH

1 Natural Language to SQL Translation is Ambiguous

Housing coordinator

Show **two** **older** **students** who have **two pets** .

NL Atom

atom

T5-LM

```
SELECT StuID
FROM Has_Pet JOIN Student
GROUP BY StuID
HAVING COUNT(*) = 2
```

All exact-two-pet students

Llama 3

```
SELECT * FROM Student
WHERE Age > AVG(Age)
AND StuID IN (
  JOIN Has_Pet JOIN Pets
  HAVING COUNT(PetType) = 2)
```

SQL Atom

Older exact-two-pet students

SQLCoder

```
SELECT fname, lname
FROM student JOIN has_pet
HAVING COUNT(petid) ≥ 2
ORDER BY age DESC LIMIT 2
```

Two oldest students with ≥2 pets

The same question can have different SQL translations, which will give different answers.

2 Existing NL2SQL mappers don't show the source of ambiguity

Query-level confidence

Bhattacharjya et al., NeurIPS 2024; Somov & Tutubalina, AAAI 2025

1 `SELECT fname, lname FROM student JOIN has_pet HAVING COUNT(petid) ≥ 2 ORDER BY age DESC LIMIT 2` — 0.54 highest

2 `SELECT * FROM Student WHERE Age > (SELECT AVG(Age) FROM Student) AND StuID IN (SELECT StuID FROM Has_Pet GROUP BY StuID HAVING COUNT(*) = 2)` — 0.31

3 `SELECT StuID FROM Has_Pet JOIN Student GROUP BY StuID HAVING COUNT(*) = 2` — 0.22

"older"	"two pets"	"show two"
?	?	?

Query-level score doesn't capture atom-level ambiguity.

Clarification dialogue

Zhao et al., PVLDB 2024; Ding et al., SIGMOD 2026

Exactly two pets?

Yes.

"older"	"two pets"	"show two"
hidden	asked	hidden

Asking about every atom requires too much human interaction, causing fatigue.

3 AtomSQL reveals atom-level confidence across models; learns from feedback

NL query Show **two** **older** **students** with **two pets** .

A Model execution layer

T5-LM
```
SELECT has_pet.StuID
FROM Has_pet JOIN Student ON StuID
GROUP BY has_pet.StuID
HAVING COUNT(*) = 2
```

Llama 3
```
SELECT * FROM Student
WHERE Age > (SELECT AVG(Age) FROM Student)
AND LName IN (SELECT LName FROM Student JOIN Has_Pet JOIN Pets
GROUP BY LName HAVING COUNT(PetType) = 2)
```

SQLCoder
```
SELECT s.fname, s.lname, COUNT(h.petid)
FROM student s JOIN has_pet h ON stuid
GROUP BY s.fname, s.lname
HAVING COUNT(h.petid) ≥ 2
ORDER BY s.age DESC LIMIT 2
```

B Atom decomposer

NL atoms	SQL atoms
students	Student ⋈ Has_Pet
older	filter | sort | none
two pets	= 2 | ≥ 2
show two	LIMIT 2 | none

C Agreement and confidence engine

	T	L	S
join	●	●	●
filter	○	●	○
sort	○	○	●
none	●	○	○
= 2	●	●	○
≥ 2	○	○	●
LIMIT 2	○	○	●
none	●	●	○

weights

D User preference store

T5-LM	33%
Llama 3	33%
SQLCoder	33%

Equal at first; shifts with feedback

E Query assembler

```
SELECT fname, lname
FROM student JOIN has_pet ON StuID
GROUP BY fname, lname
HAVING COUNT(*) = 2
```

THE LOOP: USER CHOICE UPDATES THE PREFERENCE STORE

Inspect the result

```
FROM student JOIN has_pet
HAVING COUNT(*) = 2
```

Ambiguous atoms highlighted for inspection

The user chooses the NL-to-SQL interpretation they prefer.

Alternative atoms

older

- Llama 3 — `WHERE Age > AVG(Age)` — Prefer this atom
- SQLCoder — `ORDER BY Age DESC` — Prefer this atom
- T5-LM in best — none — Prefer this atom

two pets

- T5-LM chosen — `HAVING COUNT(*) = 2` — Prefer this atom
- Llama 3 — `HAVING COUNT(PetType) = 2` — Prefer this atom
- SQLCoder — `HAVING COUNT(petid) ≥ 2` — Prefer this atom

show two

- SQLCoder — `LIMIT 2` — Prefer this atom
- T5-LM / Llama 3 in best — none — Prefer this atom

Preference store updates

T5-LM	↑ higher
Llama 3	–
SQLCoder	↓ lower

References
[1] D Bhattacharjya et al. Consistency-based Black-box Uncertainty Quantification for Text-to-SQL. NeurIPS 2024
[2] O Somov and E Tutubalina. Confidence Estimation for Error Detection in Text-to-SQL Systems. AAAI 2025
[3] F Zhao, S Deep, F Psallidas, A Floratou, D Agrawal, and A E Abbadi. Sphinteract: Resolving Ambiguities in NL2SQL Through User Interaction. PVLDB 2024
[4] Z Ding, Y Lin, T Zeng, R Zhu, B Ding, and J Zhou. AmbiSQL: Interactive Ambiguity Detection and Resolution for Text-to-SQL. SIGMOD 2026

Paper; Code; Demo
github.com/aritra741/AtomSQL