CASE ANALYSIS

Table I illustrates a case study of resolving query ambiguity with CLEAR on the BIRD dataset. Specifically, regarding scope ambiguity, the parser naturally considers "each" to "each superhero individually", thus stating `SELECT superhero_name` and `GROUP BY superhero_name`. Regarding attachment ambiguity, the parser thinks that "atoms with iodine and sulfur type elements" refers to "atoms with iodine and atoms with sulfur type elements" due to the unclear conjunction relationship, thereby constructing a condition clause of `T2.element IN ('i', 's')`. However, with CLEAR to detect and clarify user intent, correct predictions can be generated.

TABLE I: Case study of query ambiguity with CLEAR.

Scope Ambiguity	
Question:	*Calculate the average height **for each superhero**.*
Gold SQL:	SELECT CAST(SUM(height_cm) AS REAL) / COUNT(id) FROM superhero
MAC-SQL: **(Incorrect)**	SELECT superhero_name, AVG(height_cm) AS average_height FROM superhero WHERE height_cm IS NOT NULL GROUP BY superhero_name
MAC-SQL: **+ CLEAR (correct)**	φ: {for each superhero: [for each superhero individually, common to all superheroes]} ψ: {for each superhero: common to all superheroes} $\tilde{Q}$: Calculate the average height common to all superheros. **SQL**: SELECT SUM(height_cm) / COUNT(*) AS average_height FROM superhero WHERE height_cm IS NOT NULL
Attachment Ambiguity	
Question: Evidence:	*How many **atoms with iodine and sulfur type elements** are there in single bond molecules?* With iodine element refer to element = 'i'; with sulfur element refers to element = 's'; single type bond refers to bond_type = '-'.
Gold SQL:	SELECT COUNT(DISTINCT CASE WHEN T1.element = 'i' THEN T1.atom_id ELSE NULL END) AS iodine_nums , COUNT(DISTINCT CASE WHEN T1.element = 's' THEN T1.atom_id ELSE NULL END) AS sulfur_nums FROM atom AS T1 INNER JOIN connected AS T2 ON T1.atom_id = T2.atom_id INNER JOIN bond AS T3 ON T2.bond_id = T3.bond_id WHERE T3.bond_type = '-'
MAC-SQL: **(Incorrect)**	SELECT COUNT(T2.atom_id) FROM atom AS T2 WHERE T2.molecule_id IN (SELECT DISTINCT T1.molecule_id FROM bond AS T1 WHERE T1.bond_type = '-') AND T2.element IN ('i', 's')
MAC-SQL: **+ CLEAR (correct)**	φ: {atoms with iodine and sulfur type elements: [atoms with iodine and atoms with sulfur type elements, sulfur type elements and atoms with iodine]} ψ: {atoms with iodine and sulfur type elements: sulfur type elements and atoms with iodine} $\tilde{Q}$: How many sulfur type elements and atoms with iodine are there in single bond molecules? **SQL**: SELECT (SELECT COUNT(DISTINCT T1.'atom_id') FROM atom AS T1 INNER JOIN bond AS T2 ON T1.'molecule_id' = T2.'molecule_id' WHERE T1.'element' = 's' AND T2.'bond_type' = '-') AS sulfur_count, (SELECT COUNT(DISTINCT T1.'atom_id') FROM atom AS T1 INNER JOIN bond AS T2 ON T1.'molecule_id' = T2.'molecule_id' WHERE T1.'element' = 'i' AND T2.'bond_type' = '-') AS iodine_count

Figure 1 shows a database schema that can easily involve match ambiguity when answering user queries with SQL. Specifically, certain columns with similar names in different tables record the similar value, making parsers and SQL writers confused. However, when they are used interchangeably, the query results are not always the same. Therefore, CLEAR can play a significant role in this scenario, bringing about improvements in accuracy and reliability through disambiguation.

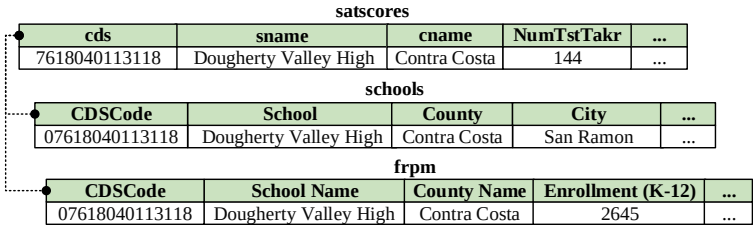


Fig. 1: Example of schema on BIRD.

Table II illustrates a case study of resolving match ambiguity with CLEAR on the BIRD dataset. Regarding attribute ambiguity (i.e. column, table, join and aggregate), parsers tend to align the question to the schema according to the relevant semantics of words, which may introduce errors. For vagueness ambiguity, parsers will get confused by vague descriptions, resulting in insufficient column names being selected. However, CLEAR can be aware of these problem and provide a clarification for the actual requirements.

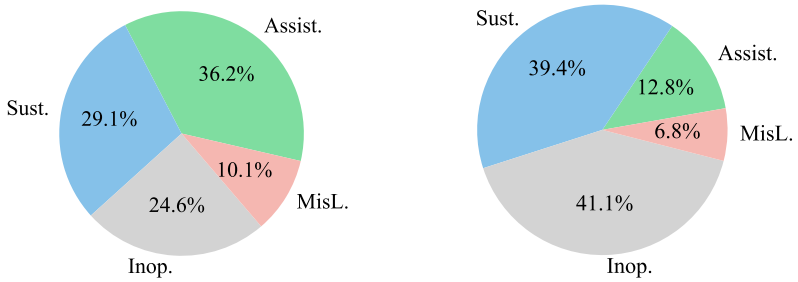
TABLE II: Case study of match ambiguity with CLEAR.

Column Ambiguity

Question:	*What is the **postal street address** for the school with the 6th highest Math average? Indicate the school's name.*
Schema:	satscores: AvgScrMath | schools: MailStreet, MailStrAbr, Street, School
Gold SQL:	SELECT T2.MailStreet, T2.School FROM satscores AS T1 INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode ORDER BY T1.AvgScrMath DESC LIMIT 5, 1
DAIL-SQL: **(Incorrect)**	SELECT schools.Street, schools.School FROM satscores INNER JOIN schools ON satscores.cds = schools.CDSCode ORDER BY satscores.AvgScrMath DESC LIMIT 1 OFFSET 5
DAIL-SQL: **+ CLEAR (correct)**	φ: postal street address: [schools: [MailStrAbr], schools: [MailStreet]] ψ: postal street address: schools: [MailStreet] $\tilde{Q}$: What is the postal street address (schools.MailStreet) for the school with the 6th highest Math average? Indicate the school 's name. **SQL**: SELECT T1.School, T1.MailStreet FROM schools AS T1 INNER JOIN satscores AS T2 ON T1.CDSCode = T2.cds ORDER BY T2.AvgScrMath DESC LIMIT 1 OFFSET 5
	Table Ambiguity
Question:	*Which **school** in **Contra Costa** has the highest number of test takers?*
Schema:	satscores: cds, sname, cname | schools: CDSCode, School, County
Gold SQL:	SELECT sname FROM satscores WHERE cname = 'Contra Costa' AND sname IS NOT NULL ORDER BY NumTstTakr DESC LIMIT 1
BIRD-SQL: **(Incorrect)**	SELECT T2.School FROM satscores AS T1 INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode WHERE T2.County = 'Contra Costa' AND T1.NumTstTakr IS NOT NULL ORDER BY T1.NumTstTakr DESC LIMIT 1
BIRD-SQL: **+ CLEAR (correct)**	φ: {school: [{schools: [School]}, {satscores: [sname]}], Contra Costa: [{schools: [County]}, {satscores: [cname]}]} ψ: {school: {satscores: [sname]}, Contra Costa: {satscores: [cname]}} $\tilde{Q}$: Which school (satscores.sname) in Contra Costa (satscores.cname) has the highest number of test takers? **SQL**: SELECT sname FROM satscores WHERE cname = 'Contra Costa' AND sname IS NOT NULL ORDER BY NumTstTakr DESC LIMIT 1
	Join Ambiguity
Question:	*Give the **code of sets** have expansion type of 'expansion'?*
Schema:	set_translations: id, setCode | cards: id, setCode | sets: id, code, type
Gold SQL:	SELECT T2.setCode FROM sets AS T1 INNER JOIN set_translations AS T2 ON T2.setCode = T1.code WHERE T1.type = 'expansion' GROUP BY T2.setCode
MAC-SQL: **(Incorrect)**	SELECT code FROM sets WHERE type = 'expansion'
MAC-SQL: **+ CLEAR (correct)**	φ: code: [set_translations: [setCode], cards: [setCode], sets: [code]] ψ: code: set_translations: [setCode] $\tilde{Q}$: Give the code (set_translations.setCode) of sets have expansion type of 'expansion'? **SQL**: SELECT setCode FROM set_translations WHERE id IN (SELECT id FROM sets WHERE type = 'expansion')
	Aggregate Ambiguity
Question:	*What is the highest **eligible free rate for K-12** students in the schools in Alameda County?*
Schema:	frpm: Free Meal Count (K-12), Enrollment (K-12), Percent (%) Eligible Free (K-12), County Name
Gold SQL:	SELECT "Free Meal Count (K-12)" / "Enrollment (K-12)" FROM frpm WHERE "County Name" = 'Alameda' ORDER BY (CAST('Free Meal Count (K-12)' AS REAL) / 'Enrollment (K-12)') DESC LIMIT 1
BIRD-SQL: **(Incorrect)**	SELECT MAX(frpm."Percent (%) Eligible Free (K-12)") FROM frpm INNER JOIN schools ON frpm.CDSCode = schools.CDSCode WHERE schools.County = 'Alameda'
BIRD-SQL: **+ CLEAR (correct)**	φ: {eligible free rate: [{frpm: [Percent (%) Eligible Free (K-12)]}, {frpm: [Percent (%) Eligible FRPM (K-12)]}, {frpm: [Enrollment (K-12), Free Meal Count (K-12)]}]} ψ: {eligible free rate: {frpm: [Enrollment (K-12), Free Meal Count (K-12)]}} $\tilde{Q}$: What is the highest eligible free rate (frpm.Enrollment (K-12) and frpm.Free Meal Count (K-12)) for K-12 students in the schools in Alameda County? **SQL**: SELECT MAX("Free Meal Count (K-12)" / "Enrollment (K-12)") AS highest_eligible_free_rate FROM frpm WHERE "County Name" = 'Alameda'
	Vagueness Ambiguity
Question:	*Under whose administration does the school with the highest number of test takers whose total SAT Scores are greater or equal to 1500 belong to? Indicate his or her **full name**. full name means first name, last name.*
Schema:	schools: AdmFName1, AdmLName1, AdmEmail1, AdmFName2, AdmLName2, AdmEmail2, AdmFName3, AdmLName3, AdmEmail3
Gold SQL:	SELECT T2.AdmFName1, T2.AdmLName1, T2.AdmFName2, T2.AdmLName2, T2.AdmFName3, T2.AdmLName3 FROM satscores AS T1 INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode ORDER BY T1.NumGE1500 DESC LIMIT 1
TA-SQL: **(Incorrect)**	SELECT schools.AdmFName1, schools.AdmLName1 FROM satscores INNER JOIN schools ON satscores.cds = schools.CDSCode WHERE satscores.NumGE1500 = (SELECT MAX(NumGE1500) FROM satscores);

TA-SQL: **+ CLEAR (correct)**	φ: {full name: [{schools: [AdmFName1, AdmLName1]}, {schools: [AdmFName2, AdmLName2]}, {schools: [AdmFName3, AdmLName3]}]} ψ: {full name: {schools: [AdmFName1, AdmLName1, AdmFName2, AdmLName2, AdmFName3, AdmLName3]}} $\tilde{Q}$: Under whose administration does the school with the highest number of test takers whose total SAT Scores are greater or equal to 1500 belong to? Indicate his or her full name (schools.AdmFName1 and schools.AdmLName1 and schools.AdmFName2 and schools.AdmLName2 and schools.AdmFName3 and schools.AdmLName3). **SQL**: SELECT schools.AdmFName1, schools.AdmLName1, schools.AdmFName2, schools.AdmLName2, schools. AdmFName3, schools.AdmLName3 FROM schools INNER JOIN satscores ON schools.CDSCode = satscores.cds WHERE satscores.NumGE1500 = (SELECT MAX(NumGE1500) FROM satscores)

We also conduct a statistic analysis for the influence that CLEAR brings on the parser. We quantify CLEAR's impact on AmbiQT across four aspects: "assistance" (Assist.) for invalid SQL predictions rendered valid, "sustainable" (Sust.) for valid predictions maintained, "inoperative" (Inop.) for invalid predictions unchanged, and "misleading" (MisL.) for valid predictions adversely affected. Figure 2 shows the statistics of disambiguation results where instances with no ambiguity detected by CLEAR are excluded, highlighting the positive assistance of CLEAR for parsers. Nonetheless, despite successfully resolving ambiguity, some predictions remain misleading and inoperative, which suggests that the robustness and other challenges in NL2SQL necessitate further refinement to bolster its performance.



(a) GPT-4o on CLAMBSQL. (b) MAC-SQL on BIRD.

Fig. 2: The influence of CLEAR on parsers.