BIRDTurk：将 BIRD 文本到 SQL 数据集改编为土耳其语

Burak Aktaş[1], Mehmet Can Baytekin[1], Süha Kağan Köse[1], Ömer İlbilgi[1],Elif Özge Yılmaz[2], Çağrı Toraman[2], Bilge Kaan Görür[1]

[1]Roketsan Inc.，土耳其人工智能技术部门[2]中东技术大学，计算机工程系，土耳其 burak.aktas@roketsan.com.tr，can.baytekin@roketsan.com.tr kagan.kose@roketsan.com.tr，omer.ilbilgi@roketsan.com.tr

yilmaz.ozge_01@metu.edu.tr, ctoraman@metu.edu.tr, kaan.gorur@roketsan.com.tr

arXiv:2602.03633v1 [cs.CL] 3 Feb 2026

摘要

文本到 SQL 系统在英语基准测试中表现出色，但其在形态丰富的低资源语言中的行为仍未得到充分探索。我们推出 *BIRDTurk*，这是首个针对土耳其语改编的 BIRD 基准。该基准通过受控的翻译流程构建，在将模式标识符改编为土耳其语的同时，严格保留 SQL 查询和数据库的逻辑结构与执行语义。翻译质量通过中心极限定理确定样本量进行验证，以确保 95% 的置信度；经人工评估的样本准确率达到 98.15%。使用*BIRDTurk*，我们评估了基于推理的提示、智能体多阶段推理和监督微调。结果表明，土耳其语会导致稳定的性能下降，这既源于语言结构差异，也源于其在大语言模型预训练中的代表性不足；而智能体推理则展现出更强的跨语言鲁棒性。对于标准多语言基线而言，监督微调仍颇具挑战，但在现代指令微调模型上能够有效扩展。*BIRDTurk* 提供了一个受控测试平台，可在真实数据库条件下开展跨语言文本到 SQL 评估。我们发布了训练集和开发集划分，以支持未来的研究。[1]

1 引言

数据库自然语言接口旨在让非专业用户能够使用日常语言查询结构化数据，从而使数据访问更加普及。这一愿景推动了文本到 SQL领域的显著进展，并得到大规模基准测试的支持，例如 WikiSQL（Zhong 等人，2017年）、Spider（Yu 等人，2018），以及较新的 BIRD（Li 等人，2023）和 Spider 2.0（Lei 等人，2025）。

[1]我们的数据集和源代码可在以下地址获取: https://github.com/metunlp/birdturk

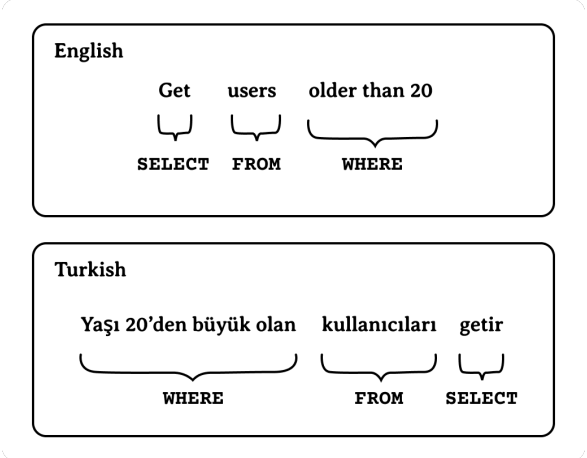

图 1：英语和土耳其语查询之间的结构差异。英语与 SQL 呈线性对应，而土耳其语将逻辑分散在后缀中，并改变词序（主宾谓语序），这使槽位对齐更加复杂。

尽管取得了这些进展，当前基准测试仍然 overwhelmingly 以英语为中心（Min 等人，2019；Tuan Nguyen 等人，2020）。对于土耳其语这类形态丰富、句法差异显著的语言而言，这一局限尤其重要。大多数最先进的文本到 SQL 模型都隐含地依赖英语与 SQL 之间紧密的句法对齐，因为两者都遵循主谓宾（SVO）语序。这种共享结构使输入词元与 SQL 结构之间能够形成相对线性的对应关系（Qin 等人，2022）。相比之下，土耳其语具有黏着形态和主语-宾语-谓语（SOV）语序（Oflazer，1994；Eryiğit 等人，2008；Umutlu 等人，2025），这打破了这种直接对齐，并使语义解析更加复杂（Dou 等人，2023）。

例如，考虑查询“`SELECT *FROM Users WHERE age > 20`”。如图1所示，英语短语“获取年龄大于 *20* 岁的用户”与 SQL 逻辑依次对齐。相反，在土耳其语译文“*Yaşı 20' denbüyük kullanıcıları getir*”中，对应于 `SELECT` (“*getir*”)的动作出现在句末。

此外，> 运算符的逻辑在形态上分布于离格后缀“*-den*”和形容词“*büyük*”之中。如图1所示，这种结构性不匹配给多语言模型的意图识别和槽位对齐机制带来了显著挑战（Dou 等，2023；Kanburo˘glu 和 Tek，2024）。

尽管 Tur2SQL（Kanburo˘glu andTek，2023）和 TURSpider（Kanburo˘gluand Tek，2024）等项目提供了有价值的初步资源，但它们并未体现现代企业环境的规模、模式复杂性或“脏数据”特征。现有土耳其语数据集大多遵循较早的基准范式，缺乏评估大语言模型处理现实世界数据库歧义能力所需的推理深度，而 BIRD（Li 等人，2023）正是引入了这类歧义。

为弥补这一空白，我们推出了 *BIRDTurk*，这是首个 BIRD 基准（Li 等人，2023）的土耳其语改编版。BIRDTurk 通过受控的翻译流程构建，在翻译自然语言问题并将模式标识符系统地本地化为土耳其语的同时，保留了原始 SQL 查询和数据库的逻辑结构与执行语义。为确保可扩展性和可靠性，我们采用基于中心极限定理（CLT）的统计验证框架，为翻译质量提供明确的置信区间。

我们的贡献可概括如下：

- 我们推出了*BIRDTurk*，这是首个基于 BIRD 改编的土耳其语文本到 SQL 数据集。
- 我们提出了一种以统计学为基础、基于 CLT 的框架，可高效验证大规模数据集翻译。
- 我们通过系统性实验建立了基线结果，涵盖基于推理的提示、智能体推理和监督微调。

2 相关工作

2.1 文本到 SQL 基准测试的发展

文本到 SQL 基准的发展轨迹反映了从受限语义解析到真实世界数据库落地的范式转变。早期数据集如 ATIS 和 GeoQuery（Price，1990年；Zelle 和 Mooney，1996年）仅限于单一领域，从而导致过拟合和记忆化问题（Finegan-Dollak 等人，2018）。尽管 WikiSQL（Zhong 等人，2017年）引入了规模，但它将任务过度简化为单表操作。Spider（Yu 等人，2018）通过引入复杂的 SQL 结构（例如嵌套、连接）和未见过的模式解决了这些局限，从而确立了事实上的跨领域结构泛化标准。然而，这些基准测试在“干净”的模式上运行，主要测试模型将自然语言词元映射到 SQL 语法的能力，而非对数据库内容进行推理。

近期的基准测试已超越句法对齐，转向在“脏”数据上进行以执行为中心的评估。BIRD（Li 等人，2023）通过引入海量含噪数据库（33.4 GB），代表了这一飞跃；这些数据库要求模型运用外部知识（例如领域术语、数值计算）进行推理，而非仅仅进行简单的模式链接。这一转变凸显了人类表现（92.96%）与最先进的大语言模型之间的显著差距，表明以内容为依据进行推理至关重要。沿着这一发展方向，Spider 2.0（Lei 等人，2025）进一步重新定义了这项任务，采用具备智能体特征的“代码智能体”范式，要求模型调试查询并在企业工作流中导航，从而将文本到 SQL 定位为一项多轮软件工程挑战，而非单轮翻译任务。

2.2 低资源语言面临的挑战

非英语数据集匮乏，促使人们对 Spider 1.0 进行了大量改编，从仅翻译问题的中文版本（Min 等人，2019），到全面进行模式本地化的越南语版本（Tuan Nguyen 等人，2020）、俄语版本（Bakshandaeva 等人，2022）和阿拉伯语版本（Almohaimeed 等人，2024）。尽管这些基准测试奠定了基础，但它们主要是在理想化的模式上测试句法对齐，往往避开了企业环境中固有的“脏数据”挑战。

在土耳其语领域，Tur2SQL（Kanburo˘gluand Tek，2023）和 TURSpider（Kanburo˘glu andTek，2024）是重要的里程碑。然而，它们也凸显了一个关键瓶颈：以英语为中心的大语言模型经常难以处理土耳其语的黏着语形态，后缀导致的分词不匹配会引发模式幻觉（Kanburo˘gluandTek，2024）。此外，由于沿用 Spider 的结构，现有土耳其语数据集并未评估现代数据库应用所需的基于内容的推理能力，而这正是 BIRDTurk 旨在填补的空白。此外，近期的土耳其语大语言模型基准测试工作，例如

TurkBench（Toraman 等人，2026）也纳入了对指令遵循能力的评估；然而，对面向 SQL 的推理仅有少量涉及，而且该基准测试并不针对本任务所需的、面向关系型数据库的结构化内容推理。

3 数据集构建

我们构建 *BIRDTurk* 的方式是将 BIRD 公开可用的训练集和开发集划分翻译为土耳其语。我们的目标是使该基准测试在各语言间功能上完全相同：底层数据库和 SQL 语义必须保持不变，同时对自然语言接口进行本地化。遵循先前的跨语言 Text-to-SQL 基准测试（Min 等人，2019；Tuan Nguyen 等人，2020；Bakshandaeva 等人，2022；Almohaimeed 等人，2024），我们优先考虑语义等价性、模式忠实度和执行一致性。由于基于大语言模型的翻译可能出现偏离——尤其是对于土耳其语这类形态丰富的语言（Kanburo˘glu 和 Tek，2024）——我们采用以模式为基础的流程，以约束可能破坏可执行性的修改。

我们采用仅模式本地化：我们翻译数据库/表/列标识符，但不翻译数据库单元格值（例如，WHERE 子句中带引号的字符串字面量）。这样可避免引入单独的值链接问题，并简化执行级比较。

3.1 模式映射

在翻译问题之前，我们为每个数据库建立确定性的模式映射 $\mathcal{M}$，定义原始英语标识符（表/列）与其土耳其语对应项之间的一一对应关系。预先固定这一词汇，可将问题翻译、证据本地化和 SQL 改写限制在封闭的标识符集合内，从而避免出现词汇表外术语。

我们直接从 SQLite 文件中提取模式元数据（表名、列属性、外键），并在可用时纳入数据库描述，以消除歧义。我们使用 `gemini-2.5-flash` (Deep-Mind，2025)翻译标识符，并施加严格限制：仅使用 ASCII 蛇形_命名标识符，并对反复出现的子术语进行标准化（例如，`movie_popularity` →`film_populerligi`、`first_name` → `ilk_isim`）。完整的模式映射提示见附录A.1。

我们还处理了少数标识符冲突情况，即不同的英文标识符映射为相同的土耳其语形式。虽然在我们的数据集中仅涉及两列，但我们仍确保数据库内名称唯一，并以确定性方式解决这些冲突（例如，采用稳定后缀规则），这也有助于缓解非英语模式环境中的模式幻觉（Kanburo˘glu andTek，2024；Douetal.，2023）。

3.2 翻译与本地化流程

我们将本地化视为一个约束满足问题：翻译后的土耳其语文本必须同时保留原始 SQL 语义，并符合 $\mathcal{M}$所定义的约束。为此，我们设计了一个由三个阶段组成的翻译与本地化流程，系统地将 BIRD 基准测试转换为其土耳其语对应版本 BIRDTurk。该流程确保与源查询在语义上保持忠实，同时在每个阶段落实语言和结构的一致性。图2展示了这一翻译流程的总体工作流程。

3.2.1 证据标准化与模式对齐

BIRD 中的证据字段将自然语言与模式引用交织在一起；不一致之处可能破坏意图与可执行 SQL 之间的联系。翻译之前，我们使用 $\mathcal{M}$ 对证据中的所有反引号标识符进行确定性重写，并确保满足一项不变条件：每个反引号括起的片段都必须与本地化 SQL 中使用的土耳其语蛇形命名标识符完全一致。

3.2.2 问题与证据的联合翻译

随后，我们在同一上下文窗口中，使用 `gemini-2.5-flash` （提示词见附录 A.2）联合翻译问题和与模式对齐的证据。模型必须逐字保留反引号括起的片段，同时将周围文本本地化；还必须保留语义常量（数字、日期）和比较逻辑（例如最高/最低/前 k 名）。为确保风格统一，我们遵循一套固定的指令（Tuan Nguyen 等人，2020）（例如，“列出” →“Listeleyiniz”，“有多少” → “有多少.....？ ”）。

3.2.3 基于 AST 的 SQL 本地化

为保留执行行为，我们避免神经 SQL 生成，改用确定性的、感知结构的重写方法。我们将每条 SQL 查询解析为抽象语法树（AST）（Aho 等人，

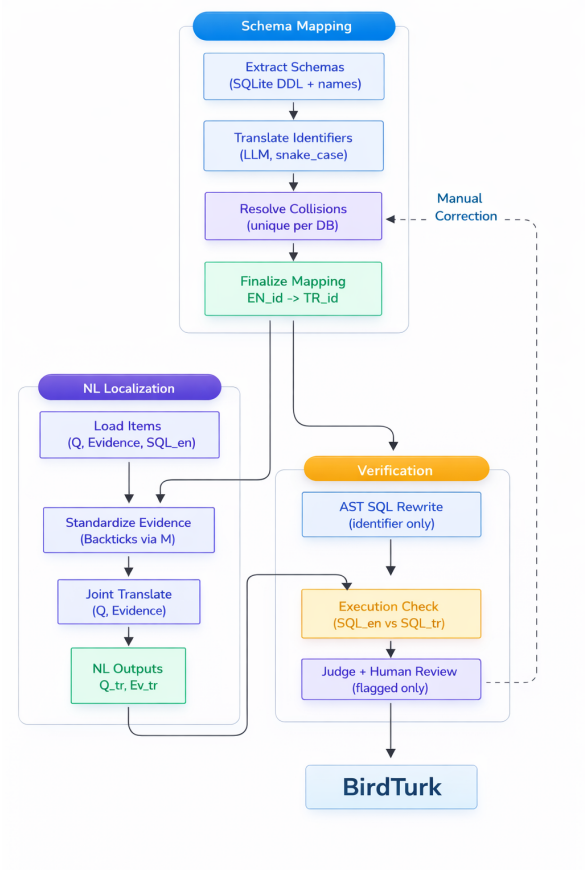

图 2：旨在将 BIRD SQL 基准转换为 BIRDTurk 的翻译与本地化流程，同时保留 SQL 意图并强制执行土耳其语语言约束。

2006），并且仅重写标识符节点（表/列），通过 $\mathcal{M}$；所有其他元素（关键字、运算符、函数、字面量）均保持不变。这避免了正则表达式/字符串替换的陷阱（例如别名冲突、部分匹配，如`table.column`，或意外修改字面量），并保留语法有效性和语义等价性。作为额外保障，我们使用基于评分标准的大语言模型评审，核验被标记实例的文本与 SQL 的对齐情况（提示见附录A.3）。

3.3 质量控制与统计验证

为确保可靠性，我们将自动化完整性检查与基于统计的人工评估相结合。

3.3.1 自动化一致性检查

人工审核前，所有示例都会经过自动验证，以筛除结构错误和执行错误。

执行等价性（主要信号）。我们采用执行正确性作为主要有效性标准。对于每个实例，我们都会在底层数据库上执行原始英语 SQL 和生成的土耳其语 SQL。我们会验证二者的结果集是否完全相同（$R_{en} = R_{tr}$），并在指定时严格确保行顺序一致。此步骤可确保本地化过程保留原始查询的可执行语义。

结构完整性与模式约束。我们通过算法验证土耳其语 SQL 中引用的所有模式标识符（表、列）均严格映射到本地化模式。此外，我们将自然语言证据中的反引号标识符与本地化模式进行交叉核对，以防止出现幻觉列。未通过这些检查的示例会被自动标记以供修正。

3.3.2 统计语义验证

自动检查无法全面评估语言流畅度和语义忠实度。为大规模评估翻译质量，我们采用基于中心极限定理（CLT）的概率抽样（Feller，1968）。

我们将翻译正确性建模为伯努利随机变量，并估算在*95%*置信水平（$Z = 1.96$）下，误差范围为 $\pm 3\%$（$E = 0.03$）所需的样本量。假设方差最大（$p = 0.5$），无限总体的样本量为：

$$n_0 = \frac{Z^2 \cdot p(1-p)}{E^2} \approx \frac{1.96^2 \cdot 0.25}{0.0009} \approx 1{,}068 \quad (1)$$

令 N 表示BIRDTurk中的样本总数，计算方式为训练集和开发集划分之和（$N = 10{,}962$）。应用有限总体修正（FPC）后，得到：

$$n = \frac{n_0}{1 + \frac{n_0 - 1}{N}} \approx \frac{1{,}068}{1 + \frac{1{,}067}{10{,}962}} \approx 974 \quad (2)$$

因此，我们手动评估了随机抽取的*974*个示例；其中*956*个被判定为正确，观测到的翻译准确率为*98.15%*。根据CLT，这意味着在95%的置信水平下，数据集层面的准确率误差范围为 $\pm 3\%$。

算法 1 概述了我们的三阶段、以模式为基础的构建流程，包括模式映射、问题与证据的联合本地化，以及经过验证的执行一致的 SQL 重写。

算法 1 BIRDTURK 构建流程概览。要求：SQLite 数据库 $\mathcal{D}$；英语条目 $\mathcal{X}$，包含 (db_id, $question$, $evidence$, sql_{en}) 确保：土耳其语条目 $\mathcal{X}_{tr}$，包含 (db_id_{tr}, $question_{tr}$, $evidence_{tr}$, sql_{tr}) 1: 阶段 1：模式映射 2: 对所有 $db \in \mathcal{D}$ 执行 3: $S_{db} \leftarrow$ 提取模式元数据（数据定义语言 +标识符列表） 4: $M_{db} \leftarrow$ 大语言模型将 S_{db} 中的标识符翻译为土耳其语 ASCII 蛇形_大小写 5: M_{db} ←解决标识符冲突（在 db内唯一） 6: 结束循环 7: 阶段 2：自然语言本地化 8: 对所有 $x \in \mathcal{X}$ 执行 9: $M \leftarrow M_{x.db\ id}$ 10: e^{std} ←通过 M重写 $x.evidence$ 中的反引号标识符 11: (q_{tr}, e_{tr}) ← 由大语言模型联合翻译 ($x.question$, e^{std})，并冻结反引号内容 12: 结束循环 13: 阶段 3：SQL 本地化与验证 14: 对所有 $x \in \mathcal{X}$ 执行 15: $M \leftarrow M_{x.db\ id}$ 16: $sql_{tr} \leftarrow$ 通过 M对 AST 进行重写 $x.sql_{en}$，替换标识符节点 17: 如果 EXECEQUAL(sql_{en}, sql_{tr} 在 db上) 为假，则 18: 标记 x以供审核 19: 结束条件 20: 如果 x 已标记，则 21: 基于评分标准的评审 + 并进行人工修正；如有需要，更新 M 22: 重新运行验证 23: 结束条件 24: 存储 (db_id_{tr}, q_{tr}, e_{tr}, sql_{tr}) 25: 结束循环

3.4 数据集统计

BIRD 是一个大规模基准测试，包含 12,751 个文本到 SQL 配对，分布于训练集/开发集/测试集，以及涵盖 37 个领域的 95 个数据库，总计 33.4 GB（Li 等人，2023）。每个数据库平均包含 7.3 张表和约 549,000 行；最大的数据库（“Donor”）为 4.5 GB。

表 1 展示了对 BIRDTurk 语言特征及其与原始 BIRD 基准在结构上的对齐情况进行的定量分析。比较结果揭示了系统性的语言变化这些变化在训练集和开发集划分中均保持一致，证实了跨语言适配过程的稳健性。

土耳其语的黏着特性压缩了表层词数（训练集：*-27.3%*，开发集：*-26.9%*），因为英语虚词被吸收到后缀中；与此同时，字符数仍大致相近（减少 4.3–5.1%），这表明在更高的信息密度下，内容得以保留（Eryi˘git 等人，2008）。

词汇多样性显著提高：TTR 增至两倍以上（训练集：*+122.2%*，开发集：*+100.8%*），词汇量也大幅增加（*+68.2%*，*+51.2%*）。这反映了土耳其语丰富的形态变化：同一词元会生成许多表层形式，从而加剧模型面临的稀疏性问题（Hakkani-Tür 等人，2002）。

SQL 长度基本保持不变（-0.5%），表明结构复杂度得以保留。相比之下，证据词元数量有所增加（训练集：*+9.3%*），这与“多语言分词税”一致（Ahia 等，2023）：以英语为中心的分词器往往会将土耳其语后缀过度拆分为多个子词单元（Rust 等，2021；Toraman 等，2023）。

4 实验

本节介绍用于验证 *BIRDTurk* 作为评估数据集和监督训练资源的实验设置。遵循原始 BIRD 基准测试的实验分类，我们在两种互补的文本到 SQL 范式下评估 BIRDTurk：仅推理提示（包括智能体推理流程）和监督微调。

4.1 基线方法

4.1.1 基于提示的推理

我们在纯推理设置下评估文本到 SQL 生成；在这种设置中，不进行特定任务训练或参数更新，所有适配均通过提示实现。

作为直接基线，我们采用零样本上下文学习（ICL）设置：模型接收数据库模式的文本描述以及自然语言问题，并一次性生成 SQL 查询（参见附录 A.4 和A.5 中的提示模板）。此设置用于评估模型在没有显式监督或结构指导的情况下，对土耳其语文本到 SQL 查询的泛化能力。

为考察结构化推理的效果，

表 1：BIRDTurk 训练集与开发集相对于原始英语 BIRD 的语言与结构统计。该比较凸显了显著的跨语言差异，例如因黏着而导致词数减少，以及词汇多样性增加。

统计项	训练集			开发集		
	BIRD（英语）	BIRDTurk（土耳其语）	变化（%）	BIRD（英语）	BIRDTurk（土耳其语）	变化（%）
问题总数	9,428	9,428	–	1,534	1,534	–
每个问题的平均词数	14.05	10.21	-27.3%	14.55	10.64	-26.9%
每个问题的平均字符数	79.81	75.75	-5.1%	82.76	79.18	-4.3%
每个问题的平均词元数	15.74	11.91	-24.3%	16.24	12.22	-24.8%
词汇多样性						
词汇量（不重复）	9,002	15,142	+68.2%	2,450	3,704	+51.2%
词形-词次比（TTR）	6.07%	13.49%	+122.2%	9.84%	19.76%	+100.8%
复杂性与完整性						
平均 SQL 词元数	31.03	30.89	-0.5%	31.46	31.26	-0.6%
平均证据词元数	21.32	23.31	+9.3%	21.49	22.04	+2.6%

我们还评估了 DIN-SQL（PourrezaandRafiei, 2023），这是一种智能体多阶段推理流程，可将 SQL 生成分解为多个引导步骤。该流程包含：

- **模式链接**以确定相关的表和列，
- **意图分类**以推断高级查询结构，
- **自我纠正** 利用执行反馈来优化生成的SQL。

为确保与土耳其语兼容，我们采用保留结构的策略翻译原始 DIN-SQL 提示词（附录 A.6）。

两种方法均使用 gemini-2.5-flash-lite 实现，从而确保性能差异源于推理策略，而非模型容量。所有推理实验均在相同配置下，分别针对英语 BIRD 和 BIRDTurk 进行。

4.1.2 监督微调

我们还在 BIRDTurk 上评估监督式文本到 SQL 学习，以检验翻译后的数据集是否支持参数更新训练。

遵循原始 BIRD 设置，我们对 mT5 系列的多语言模型（mT5-small、mT5-base、mT5-large）进行微调。尽管 mT5 提供了适用于土耳其语的多语言分词，但微调这些模型仅带来有限的性能提升，表明在此设置下学习有效的土耳其语文本到 SQL 映射存在挑战。

受这些局限性的推动，我们转向 *Qwen2.5-Coder* 系列，采用 0.5B、1.5B和3B 指令微调变体。我们选择这一特定架构，是基于近期实证发现中两个互补因素。首先，Qwen2.5-Coder 系列在开源权重模型的代码生成基准测试中取得了最先进的性能，这得益于一个规模达 5.5 万亿词元、并富含合成数据的大规模预训练语料库（Hui 等人, 2024）。其次，这对我们的跨语言研究重点至关重要：近期研究凸显了通义千问架构对土耳其语的结构优势。Hacıfazlıo˘glu 等人 （2024）表明，与 Llama-3 和 Gemma 等模型相比，通义千问分词器实现了更优的压缩率，有效缓解了土耳其语黏着词缀的过度切分。这种分词效率使模型能够以更少的词元保持土耳其语自然语言查询的语义完整性，从而增强土耳其语意图与 SQL 逻辑之间的对齐。

所有通义千问模型均使用相同的监督协议在 BIRDTurk 上进行微调与评估。尽管其绝对性能仍低于基于推理的流水线，但它们在所有指标上均持续优于 mT5，这证实配备高效分词器的现代指令微调模型能够为土耳其语文本到 SQL 任务提供更有效的学习信号。

为确保对基础模型与微调模型能力的比较公平，两者均采用标准的上下文学习提示策略进行推理。

4.2 评估设置与指标

我们在各自数据集的开发集划分 上评估所有模型，并报告三次独立运行的平均指标，以确保结果稳健。主要评估指标为执行准确率（*EX*）和有效效率分数（*VES*），遵循 BIRD 基准测试协议；同时报告完全匹配（*EM*），以便与先前研究进行比较。

完全匹配（EM）。完全匹配用于衡量预测的 SQL 查询经过规范化处理后，在结构上是否与真实 SQL 完全一致：

$$\mathrm{EM} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}_{\mathrm{SQL}}(Y_n, \hat{Y}_n),$$

其中：

$$\mathbb{I}_{\mathrm{SQL}}(Y, \hat{Y}) = \begin{cases} 1, & \text{if } Y \equiv \hat{Y} \text{ after normalization,} \\ 0, & \text{otherwise.} \end{cases}$$

执行准确率（EX）。执行准确率用于衡量预测 SQL 与真实 SQL 的执行结果是否完全一致：

$$\mathrm{EX} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}(V_n, \hat{V}_n),$$

其中：

$$\mathbb{I}(V, \hat{V}) = \begin{cases} 1, & \text{if } V = \hat{V}, \\ 0, & \text{otherwise.} \end{cases}$$

有效效率分数（VES）。有效效率分数在执行准确率的基础上，进一步考虑执行效率：

$$\mathrm{VES} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}(V_n, \hat{V}_n) \cdot R(Y_n, \hat{Y}_n),$$

其中：

$$R(Y_n, \hat{Y}_n) = \sqrt{\frac{E(Y_n)}{E(\hat{Y}_n)}}.$$

此处， $E(\cdot)$表示执行时间。平方根项可减轻执行时间方差和极端异常值的影响（Liet al., 2023）。

5 结果与讨论

5.1 BIRDTurk 上的总体性能

我们首先概述模型在土耳其语 BIRDTurk 数据集上的整体表现。

在所有实验设置下，BIRDTurk 都构成了一个具有挑战性的测试平台，既体现了土耳其语的语言特性，也反映了继承自 BIRD 的企业级文本到 SQL 任务的复杂性。

我们的实验呈现出三个一致的趋势。首先，对于较早期的多语言基线模型，在 BIRDTurk 上进行监督微调仍然颇具挑战；而较新的指令微调模型则表现出更明显且可扩展性更强的学习行为。其次，在相同的评估条件下，基于推理的方法的执行准确率远高于监督方法。第三，在基于推理的范式中，无论是英语还是土耳其语，智能体推理的表现始终优于直接提示。

我们将在下文详细分析这些观察结果。

5.2 BIRDTurk 上的监督微调分析

我们首先开展监督实验，直接评估 BIRDTurk 是否支持土耳其语中的参数更新学习。表2 报告了在同一 BIRD- Turk 开发集划分上评估的微调模型结果。指标包括执行准确率（EA）、有效效率分数（VES）和完全匹配（EM），数值越高表示性能越好（↑）。

模型	微调后的 ↑		
	EA	VES	EM
mT5-small	0.26	0.28	0.00
mT5-base	0.98	1.04	0.06
mT5-large	1.83	2.05	0.13
Qwen2.5-Coder-0.5B-Instruct	1.24	1.38	0.33
Qwen2.5-Coder-1.5B-Instruct	7.24	8.01	2.61
Qwen2.5-Coder-3B-Instruct	**15.38**	**17.12**	**4.82**

表2：土耳其语 BIRDTurk 开发集划分上的监督式文本到 SQL 性能。结果为模型经过监督微调后的表现。评估指标包括执行准确率（EA）、有效效率分数（VES）和完全匹配（EM），数值越高越好（↑）。

表 2中有几点值得讨论。我们排除了基础模型的结果，因为它们的表现都接近 0%，其中表现最佳的基础模型（Qwen2.5-Coder-3B-Instruct）执行准确率也仅为 2.22%。多语言基线模型在微调后仅有小幅提升，执行准确率仍低于 2%。相比之下，经过指令微调的模型则表现出明显且稳定的提升，其

性能随模型规模可靠地扩展 .

这些结果表明，BIRDTurk 提供了可用的监督信号；然而，要在土耳其语中有效学习，依赖的是指令遵循和结构化生成能力更强的模型，而非仅仅依靠多语言覆盖。

5.3 基于推理的验证：英语—土耳其语对比

为确保观察到的监督学习行为并非数据集构建或评估噪声所致，我们接下来将在提示条件相同的情况下考察基于推理的表现。所有推理实验均使用相同的底层语言模型 gemini-2.5-flash-lite，以便在不同语言之间进行受控比较。

表3 比较了在原始英语 BIRD 基准和其土耳其语对应数据集 BIRDTurk 上的直接提示（上下文学习）与智能体推理（DIN-SQL）。

方法	BIRD（土耳其语）↑			BIRDTurk（土耳其语）↑		
	执行准确率	VES	完全匹配	执行准确率	VES	完全匹配
上下文学习	58.21	63.15	12.65	43.16	45.34	10.82
DIN-SQL	**60.89**	**64.66**	**17.60**	**49.02**	**51.10**	**11.41**

表3：英语 BIRD 与土耳其语 BIRDTurk 开发集划分上的基于推理和智能体性能比较。指标包括执行准确率（EA）、有效效率分数（VES）和完全匹配（EM），数值越高表示性能越好（↑）。

在两个数据集上，英语在绝对指标上始终优于土耳其语。然而，各方法的相对排序保持不变：DIN-SQL 在两种语言中均持续优于直接提示。

这种稳定性表明，BIRDTurk 保留了 BIRD 的核心结构与推理特征，而所观察到的性能差距主要归因于语言带来的难度，而非评估不一致。

5.4 智能体推理在不同语言中的影响

更仔细地考察表 3 可以发现，智能体推理在不同语言中所带来的相对增益存在不对称。虽然 DIN-SQL 在英语和土耳其语中均优于直接提示，但改进幅度在土耳其语设定下更为明显。

在英语中，智能体推理在所有评估指标上均带来适中但稳定的提升。

相比之下，在土耳其语中，DIN-SQL带来的相对提升更大，尤其体现在执行准确率和有效效率分数上。这一模式表明，当自然语言与 SQL 之间的表层对齐程度较弱时，显式的任务分解、模式接地和迭代纠正发挥着更为关键的作用。

一种合理的解释是，土耳其语的黏着形态以及主宾谓语序在词元和短语层面引入了额外的歧义。智能体流水线通过强制执行与模式结构和执行语义对齐的中间推理步骤，降低了对直接词元到 SQL 对应关系的依赖。

重要的是，这些趋势是在相同的推理配置下观察到的，表明智能体推理的收益主要源于语言因素，而非模型特定效应。

5.5 跨语言性能比较

英语与土耳其语之间剩余的性能差距可归因于两个互补因素：土耳其语固有的语言特性，以及大语言模型预训练中的表征不平衡。

从语言学角度看，土耳其语带来的结构挑战使直接的 Text-to-SQL 迁移更为复杂：（1）主语–宾语–动词（主宾谓语序）语序破坏了自然语言与 SQL 语法之间的线性对齐；（2）黏着型形态导致词汇稀疏性增加，因为语义信息分布在大量低频表层形式中；（3）后缀密集的词形加剧了词元切分碎片化，使同等语义内容对应更长的词元序列。

除这些固有挑战之外，当前的大语言模型主要以英语为中心的语料进行训练，使土耳其语在预训练数据中严重代表性不足。这种不平衡限制了模型对土耳其语语言模式的接触，进一步加剧了该语言形态复杂性所带来的困难。这些因素——结构差异与有限的预训练覆盖——共同构成了对跨语言泛化尤为困难的场景。

智能体推理通过强制进行模式 grounding 和中间决策，在一定程度上缓解了这些影响，从而减少对表层词元对齐以及预训练期间习得的特定语言先验的依赖。

6 结论

在这项工作中，我们推出了*BIRDTurk*，这是面向真实数据库设置而设计的 BIRD Text-to-SQL基准的土耳其语改编版。通过谨慎控制改编过程，在本地化模式标识符和自然语言问题的同时，确保不同语言之间的执行行为和数据库结构具有可比性，我们构建了一个受控的评估数据集，可用于直接、公平地进行跨语言比较。

我们采用基于统计学原理的中心极限定理框架验证了翻译质量，无需进行穷尽式人工标注，即可确保数据集级可靠性。借助 BIRDTurk，我们在土耳其语语言条件下，对基于推理的提示、智能体多阶段推理流水线以及监督微调方法进行了系统评估。

结果表明，在纯推理设置下，土耳其语会导致稳定但有限的性能下降。相比之下，智能体推理流水线在不同语言间表现出更强的稳健性，这表明显式的任务分解、模式接地和中间推理步骤能够有效缓解语言带来的挑战。尽管标准多语言基线仍难以进行监督微调，但较新的指令微调模型在土耳其语环境下展现出更清晰、可扩展性更强的学习表现。

除了当前范围，BIRDTurk 还为若干重要研究方向奠定了基础。未来的扩展包括：构建完全原生的土耳其语文本到 SQL 基准，以补充基于翻译的评估；开发结合翻译问题与土耳其语原生撰写问题的混合数据集；以及借鉴 Spider 2.0，将 BIRDTurk 扩展至更具智能体特征、面向企业且支持多轮评估的设置。这些方向将进一步提升该基准的真实度，并拓宽其在实际部署场景中的适用性。

总体而言，BIRDTurk 填补了土耳其语文本到 SQL 研究中的一个关键空白，使研究人员能够在真实的数据库条件下开展受控的跨语言评估，并系统分析各种建模范式，同时也为构建更先进、更符合土耳其语特点的评估框架奠定了基础。

局限性

尽管做出了这些贡献，BIRDTurk 仍存在若干局限，在解读结果时应予以考虑。

- BIRDTurk 通过翻译构建，而非由母语者撰写土耳其语问题。虽然这便于与 BIRD 进行受控的跨语言比较，但可能无法充分反映自然出现的土耳其语查询表述。

- 翻译依赖于单一的大语言模型（Gemini），尽管经过后处理与基于 CLT 的验证，仍可能引入系统性的风格偏差。

- 基于 CLT 的验证可在数据集层面提供质量保证，但无法替代在单个样本层面进行的穷尽式人工标注。

- 由于采用基于翻译的构建过程，某些土耳其语特有的语言现象（例如省略、语用推理）很可能代表性不足。

尽管翻译噪声在统计上受到约束，一小部分语言复杂或高度隐含的查询仍可能受到影响。

伦理考量

许可与版权。 BIRDTurk 是基于原始 BIRD 基准测试构建的衍生作品（Li 等人，2023）。我们严格遵守源材料最新的使用条款，该材料依据 知识共享署名-相同方式共享 *4.0* 国际版（*CC BY- SA 4.0*）许可发布。为遵守“相同方式共享”条款，BIRDTurk 依据相同许可发布，以确保所有衍生改进成果继续向研究界开放且可访问。我们明确承认原始数据集创作者的知识产权，并贡献这一适配版本，以促进跨语言语义解析领域可复现且协作性的进步。

生成式人工智能的使用。 生成式人工智能仅用于协助语言编辑。本研究中的所有科学贡献、数据构建与分析以及诠释均为作者原创，并完全由作者完成。

致谢

我们衷心感谢 Roketsan Inc. 和 Google Gemini 学术奖励计划提供的支持，使本研究得以开展实验并使用计算资源。

参考文献

Orevaoghene Ahia、Sachin Kumar、Hila Gonen、Kyunghyun Jung、Yulia Tsvetkov 和 Noah A. Smith。2023。所有语言的成本都相同吗？商业大语言模型时代的分词。载于《*2023* 年自然语言处理经验方法会议论文集（*EMNLP*）》，第 10760–10773 页。计算语言学协会。

Alfred V Aho、Monica S Lam、Ravi Sethi 和 Jeffrey D Ullman。2006。《编译原理、技术与工具》，第 2 版。培生教育。

Saleh Almohaimeed、Saad Almohaimeed、Mansour Al Ghanim 和 Liqiang Wang。2024。Ar-Spider：阿拉伯语文本到 SQL。载于第 *39* 届 *ACM/SIGAPP* 应用计算研讨会论文集（*SAC '24*），第 1024–1030 页。

Daria Bakshandaeva、Oleg Somov、Ekaterina Dmitrieva、Vera Davydova 和 Elena Tutubalina。2022 年。PAUQ：俄语文本到 SQL。载于《计算语言学协会会议论文集：*EMNLP 2022*》，第 2355–2376 页，阿布扎比，阿拉伯联合酋长国。计算语言学协会。

Google DeepMind。2025 年。Gemini 2.5：借助先进推理、多模态、长上下文和新一代智能体能力推动前沿发展。*arXiv* 预印本。

Longxu Dou、Yan Gao、Mingyang Pan、Dingzirui Wang、Wanxiang Che、Dechen Zhan 和 Jian-Guang Lou。2023 年。Multispider：迈向多语言文本到 SQL 语义解析的基准测试。载于*AAAI* 人工智能会议论文集，第 37 卷，第 12745–12753 页。

GülşenEryiˇgit、JoakimNivre 和 Kemal Oflazer。2008。土耳其语依存句法分析。《计算语言学》，34(3):357–389。

William Feller。1968。《概率论及其应用导论》，第 *1* 卷，第 3 版。Wiley，纽约。

Catherine Finegan-Dollak、Jonathan K. Kummerfeld、LiZhang、Karthik Ramanathan、Sesh Sadasivam、Rui Zhang 和 Dragomir Radev。2018。改进文本到 SQL 评估方法。载于第 *56* 届计算语言学协会年会论文集（第 *1* 卷：长篇论文），第 351–360 页，澳大利亚墨尔本。计算语言学协会。

H. Ozan Hacıfazlıoˇglu、Vahid Partovi Nia 和 Ercan Kuruoˇglu。2024。针对土耳其语微调大语言模型。*arXiv* 预印本 *arXiv:2407.15185*。Dilek Z Hakkani-Tür、Kemal Oflazer 和 Gökhan Tür。2002。土耳其语文本的统计形态消歧。《国际模式识别与人工智能杂志》，16(04):381–402。Binyuan Hui、Jian Yang、Zeyu Cui、Jiaxi Yang、Dayiheng Liu、Lei Zhang、Tianyu Liu、Jialong Huang、Tong Yu、Ganqu Wang 等 1 人。2024。通义千问 Qwen2.5-Coder 技术报告。*arXiv* 预印本 *arXiv:2409.12186*。

Ali Buˇgra Kanburoˇglu 和 F. Boray Tek。2023。TUR2SQL：一个跨领域土耳其语文本到 SQL 数据集。载于第八届国际计算机科学与工程会议（*UBMK*）论文集，第 206–211 页。IEEE。

AliBuˇgraKanburoˇglu 和 Faik Boray Tek。2024。TUR-Spider：土耳其语文本到 SQL 数据集及基于大语言模型的研究。《*IEEE Access*》，12:169379–169387。

Fangyu Lei、Jixuan Chen、Yuxiao Ye、Ruisheng Cao、Dongchan Shin、Hongjin Su、Zhaoqing Suo、Hongcheng Gao、Wenjing Hu、Pengcheng Yin 等 1 人。2025。Spider 2.0：评估大语言模型在真实企业文本到 SQL 工作流中的表现。载于国际学习表征会议（*ICLR*）论文集。即将发表。

Jinyang Li、Binyuan Hui、Ge Qu、Jiaxi Yang、Binhua Li、Bowen Li、Bailin Wang、Bowen Qin、Ruiying Geng、Nan Huo 等 1 人。2023。大语言模型能否已胜任数据库接口？一项面向大规模数据库的文本到 SQL 研究基准。在《神经信息处理系统进展（*NeurIPS*）》第 36 卷，第 42330–42357 页。

Qingkai Min、Yuefeng Shi 和 Yue Zhang。2019。中文 SQL 语义解析初步研究。载于*2019* 年自然语言处理实证方法会议暨第九届自然语言处理国际联合会议（*EMNLP-IJCNLP*）论文集，第 3652–3658 页，中国香港。计算语言学协会。

Kemal Oflazer。1994。土耳其语形态学的双层描述。文学与语言计算，9(2):137–148。

Mohammadreza Pourreza 和 Davood Rafiei。2023。DIN-SQL：通过自我纠正实现文本到 SQL 的分解式上下文学习。载于 神经信息处理系统进展，第 36 卷，第 36339–36348 页。Curran Associates, Inc.

P. J. Price. 1990年. 口语语言系统评估：ATIS领域。载于《语音与自然语言：*1990*年*6*月*24*日至*27*日在宾夕法尼亚州希登谷举行的研讨会论文集》。

Bowen Qin、Binyuan Hui、Lihan Wang、Min Yang、Jinyang Li及其他1人。2022. text-to-SQL解析综述：概念、方法与未来方向。*arXiv*预印本 *arXiv:2208.13629*。

Phillip Rust 等 1 位作者。2021。《你的分词器有多好？多语言语言模型的单语性能研究》。载于计算语言学协会第 *59* 届年会论文集，第 3118–3135 页。

Cagri Toraman、Eyup Halit Yilmaz、Furkan ̧Sahinuç 和 OguzhanOzcelik。2023。分词对语言模型的影响：土耳其语分析。*ACM* 亚洲低资源语言与信息处理汇刊，22(4)。

Çağrı Toraman、Ahmet Kaan Sever、Ayse Aysu Cengiz、Elif Ecem Arslan、Görkem Sevinç、Mete Mert Birdal、Yusuf Faruk Güldemir、Ali Buğra Kanburoğlu、Sezen Felekoğlu、Osman Gürlek、Sarp Kantar、Birsen ̧Sahin Kütük、Büşra Tufan、Elif Genç、Serkan Coşkun、Gupse Ekin Demir、Muhammed Emin Arayıcı、Olˇgun Dursun、Onur Gungor 等 3 位作者。2026。土耳其语基准测试：用于评估土耳其语大语言模型的基准测试。*arXiv* 预印本 *arXiv:2601.07020*。

Anh Tuan Nguyen、Mai Hoang Dao 和 Dat Quoc Nguyen。2020 年。越南语文本到 SQL 语义解析试点研究。载于计算语言学协会 *EMNLP 2020* 论文集，第 4079–4085 页，线上。计算语言学协会。

Elif Ecem Umutlu、Ayse Aysu Cengiz、Ahmet Kaan Sever、Seyma Erdem、Burak Aytan、Busra Tufan、Abdullah Topraksoy、Esra Darıcı 和 Cagri Toraman。2025 年。低资源语言基准数据集质量评估：土耳其语案例研究。载于第四届生成、评估与指标研讨会论文集（*GEM2*），第 471–487 页，奥地利维也纳及线上会议。计算语言学协会。

Tao Yu、Rui Zhang、Kai Yang、Michihiro Yasunaga、Dongxu Wang、Zifan Li、James Ma、Irene Li、Qingning Yao、Shanelle Roman、Zilin Zhang 和 Dragomir Radev。2018 年。Spider：用于复杂跨领域语义解析和文本到 SQL 任务的大规模人工标注数据集。载于 *2018* 年自然语言处理经验方法会议论文集，第 3911–3921 页，比利时布鲁塞尔。计算语言学协会。

John M. Zelle 和 Raymond J. Mooney。1996年。利用归纳逻辑程序设计学习解析数据库查询。载于《第十三届全国人工智能会议（*AAAI*）论文集》，第 2 卷，第 1050–1055 页，波特兰，俄勒冈州。AAAI出版社。

Victor Zhong、Caiming Xiong 和 Richard Socher。2017年。Seq2sql：使用强化学习从自然语言生成结构化查询。*arXiv* 预印本 *arXiv:1709.00103*。

A 面向 BIRDTurk 的提示工程

为确保完全可复现，我们提供了整个流程中使用的完整提示模板集合。这些提示根据其在系统中的功能角色进行组织。

用于数据翻译和模式本地化的提示会强制采用严格定义的仅 JSON 输出格式，并明确禁止对 SQL 词元或模式标识符进行任何非预期或隐式修改。

用于推理和 *SQL* 生成的提示会明确指定任务级约束、业务逻辑护栏和 SQLite 特定的执行规则，从而确保生成的查询在语法上有效且在语义上正确。

最后，用于*DIN-SQL* 提示翻译 的提示旨在作为训练和评估的受控即插即用替代方案。这些提示保留了 DIN-SQL 框架原有的结构和约束，仅翻译自然语言指令，不改变底层语义。

A.1 模式映射提示

模式映射

系统：
你是一名严谨的数据标注助手。请严格遵守规则。

用户：
你将收到一个从 SQLite 文件中提取的数据库模式包。
你的任务是为数据库、表和列标识符提供土耳其语映射。

仅返回包含以下键的有效 JSON：

```
{
  "数据库_标识符_土耳其语":  "<仅含 ASCII 字符的小写蛇形_命名>",
  "翻译": {
    "<EN_标识符>":  "<TR_标识符>",
    ...
  }
}
```

输入模式包（JSON）：
<<模式 –包 –JSON>>

规则（严格）：
1) 范围：仅翻译标识符（数据库/表/列名）。不要翻译数据值。
2) 输出格式：
- db_id_tr：仅使用 ASCII 字符的小写蛇形命名。
- 翻译：每个 TR_IDENTIFIER 均使用仅含 ASCII 字符的小写蛇形命名。
- 土耳其语字符必须转换：ç->c, ğ->g, ı->i, ö->o, ş->s, ü->u.
3）保持标识符简洁：
- NOT 添加额外的解释性词语。
- 仅翻译英文标识符中明确出现的内容。
4）保持一致性：
- 在整个映射中，对反复出现的子术语采用一致的译法（例如，名称，日期，
↪ COUNT, 比率）。
- 在适当情况下保留常见缩写不变：id, api, ip, url, 通用唯一识别码, JSON, xml,
↪ http, 安全超文本传输协议, sql.
5) 未知缩写：
- 如果标识符是未知缩写（例如，"frpm"），保持其不变（仍为小写
↪ 蛇形_命名）。
6) 唯一性偏好（尽力而为）：
- 尽量避免在同一个数据库中发生冲突。
- 如果可能会发生冲突，优先添加最少量的上下文词元，而不是较长的短语。
7) 请勿输出任何额外的键、注释或 Markdown。仅输出 JSON。

A.2 联合问题—证据翻译Pr ompt

问题—证据翻译

系统：
你是一名谨慎的文本到 SQL 数据集翻译助手。请严格遵守规则。

用户：
你将收到：
- question_en：一个英语问题 - evidence_std：一段证据文本，其中的模式标识符已经过标准化，因此每个模式标识符都以土耳其语 ASCII 格式的蛇形命名法写在反引号内。-（可选）sql_en 和/或 sql_tr，供参考（不要改写 SQL）

你的任务：
将问题_en 和证据_std 翻译成正式、流畅的土耳其语，切勿修改反引号内的任何内容。仅返回有效 JSON，且必须恰好包含以下键：{ "question_tr": "...", "evidence_tr": "..." } 输入（JSON）：{ "question_en": "<<QUESTION_EN>>", "evidence_std": "<<EVIDENCE_STD>>", "sql_en": "<<SQL_EN_OPTIONAL>>" } 规则（严格）：1）反引号内内容只读：- 将 `...` 中的所有内容保持原样（不得编辑、调整空格或更改大小写）。↪- 不得添加或删除反引号。2）保留对含义至关重要的词元：- 不得更改数字、数值范围、单位或日期格式。- 如果文本中出现带引号的字符串字面量（例如 'Directly funded'），不得更改。3）保留逻辑意图：- 必须正确保留比较级/最高级意图（最高/最低、最多/最少、前 k 名、至少/至多）。↪- 如果问题暗示前 k 名，必须在土耳其语中明确表达这一意图。4）必须保留证据结构：- 如果证据包含方程/比例/运算符（=、/、 >、 <、 >=、 <=），必须保留相同的结构

.

- 仅翻译周围的自然语言。5）正式土耳其语风格指南：- 优先使用正式指令：“Listeleyiniz”（请列出）、“Belirtiniz”（请说明）、“Gosteriniz”（请展示）、“Hesaplayiniz”（请计算）。- “how many” -> “Kac ... vardir?”（有多少……？）- “average” ->“Ortalama ... nedir?”（……的平均值是多少？）6）如果 SQL 词元出现在反引号之外，请勿翻译：如果证据文本包含 SQL 关键字、子句、运算符或函数名，请原样保留（大小写和空格均不得更改）。包括：- 核心子句/关键字：SELECT, DISTINCT,FROM, WHERE, JOIN, INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN, ON, USING,GROUP BY, HAVING, ORDER BY, LIMIT,OFFSET, UNION, UNION ALL, INTERSECT, EXCEPT, WITH, AS, IN, NOT IN, EXISTS, NOT EXISTS, BETWEEN, LIKE,GLOB, IS NULL, IS NOT NULL, NULL, AND, OR, NOT, CASE, WHEN, THEN, ELSE, END, ASC, DESC - 聚合/函数：COUNT, SUM, AVG, MIN, MAX, CAST, COALESCE, NULLIF, SUBSTR, INSTR, LENGTH, LOWER, UPPER, TRIM, ROUND, ABS,

STRFTIME, DATE, DATETIME - 运算符/标点（不得更改）：=, !=, <>, >, <, >=, <=, +, -, *, /, %, ||, (,), ,, . 7）请勿添加新的约束或解释。8）输出必须是严格的 JSON，且仅包含上述两个键。不得添加其他文本。

A.3 大语言模型评审评分标准提示

大语言模型评审

系统：
你是土耳其语文本到 SQL 数据集的严格评估员。你只能输出有效的 JSON。

用户：
你将收到一条土耳其语文本到 SQL 数据集条目：- 问题_tr - 证据_tr（可能为空）- sql_tr（本地化的 SQL）。此外，还可能会看到原始 sql_en 以供参考。

你的任务：
评估土耳其语文本（问题_tr + 证据_tr）在语义上是否与 sql_tr 一致。请使用以下评分标准，并且只输出 JSON。

只返回 va lid JSON w ith exact ly this sch ema: { "意图_匹配": {"pass": true/false, "reason": "..."}, "约束_保留": {"pass": true/false, "reason": "..."}, "聚合_匹配": {"pass": true/false, "reason": "..."}, "排序_LIMIT_匹配": {"pass": true/false, "reason": "..."}, "证据_一致性": {"pass": true/false, "reason": "..."}, "字面值_处理": {"pass": true/false, "reason": "..."}, "总体_通过": true/false, "严重程度": "低" | "中" | "高", "建议的_修正": "..." } 输入（JSON）：{ "问题_tr": "<<QUESTION_TR>>", "证据_tr": "<<EVIDENCE_TR>>", "sql_tr": "<<SQL_TR>>", "sql_en": "<<SQL_EN_可选>>" }

评分标准定义：
1) 意图_匹配：- 问题_tr 是否恰好询问了 sql_tr 返回的内容？ 2) 约束_保留：- 隐含的筛选条件/条件是否反映在 sql_tr 中（反之亦然）？ 3) 聚合_匹配：- 如果文本隐含 COUNT/SUM/AVG/MIN/MAX 或分组，sql_tr 是否与之匹配？ 4) 排序_限制_匹配：- 如果文本隐含最高/最低/前 k 名，ORDER BY/LIMIT/OFFSET 是否匹配？ 5) 证据_一致性：- 如果存在证据_tr，其中的反引号标识符和计算是否与 sql_tr 一致？↪6) 字面量_处理：- 引号括起的字面量和常量是否得到一致处理（不翻译/更改值）？

规则 (strict)：1）不要重写sql_tr。只对其进行评估。2）理由必须简短（每条1-2句话）。3）overall_pass：- 仅当所有关键维度均通过时为True。- evidence_consistency仅在evidence_tr为空时才可以失败。4）严重程度：

- 高：可能存在语义不匹配或导致执行失败的问题 - 中：部分不匹配/歧义 - 低：大体正确；存在小问题 5 ） suggested_fix：- 若 overall_pass 为 true："（空字符串）。- 若overall_pass为false：给出最小修复，或标记为需人工复核。6）输出必须仅为严格的JSON。不得包含额外键，不得使用Markdown。

A.4 BIRD 上下文学习

上下文学习（BIRD）

系统_提示： |
你是面向 BIRD 基准（业务领域数据库）的文本到 SQL 专家模型。给定一个自然语言问题、数据库模式和可选提示，请生成一条正确的 SQLite SQL 查询。↪

BIRD 的关键规则：
1. 切勿使用预先计算的百分比/比率列（例如“符合条件的免费餐百分比（%）”）。务必根据基础列计算比率/百分比（例如“免费餐数 / 注册人数”）。例如，对于“符合条件的免费率”，请计算：CAST(`免费餐数` AS REAL) / `注册人数`

2. 使用模式中的精确列名——仔细查看列描述。如果问题提到“学校类型”，但模式中有“Educational Option Type”，则使用“Educational Option Type”。↪不要使用相似的列名——请从模式中核实准确的名称。

3. 使用值描述中的精确筛选值。如果值描述写的是“Continuation School”（而不只是“Continuation”），则使用该精确值。↪请查看模式中的值描述，以获取正确的筛选值。

4. 提供提示时，务必阅读并应用这些提示——其中包含关键业务逻辑。

关键指南：
1. 理解模式：
- 仅使用所提供模式中的表和列——密切关注列描述和值描述——检查外键关系，以确保使用正确的 JOIN——列名可能包含空格，请使用双引号：“Column Name”——核实准确的列名——不要使用相似的列名

2. 业务逻辑：
- 理解业务语义（关键绩效指标、比率、百分比、比例）- 适当处理 NULL 值（排除或使用 COALESCE 处理）- 使用 CAST 进行正确的类型转换（尤其是百分比）- 应用提示中提到的业务规则 - 根据基础列计算比率/百分比，绝不使用预先计算的列

3. 复杂计算：
- 计算百分比：CAST(分子 AS REAL) / 分母 * 100 - 计算比率/比例：CAST(count_A AS REAL) / count_B - 除法前筛除分母为零或 NULL 的情况 - 使用 CASE WHEN 实现条件逻辑 - 示例：“符合条件的免费餐比率” = CAST(`免费餐数量` AS REAL) / `入学人数`

4. 查询优化：
- 使用适当的连接（默认使用 INNER JOIN，需要时使用 LEFT JOIN）- 在查询中尽早应用筛选条件（WHERE 应先于 HAVING）- 使用子查询实现复杂聚合 - 如有需要，可考虑使用公用表表达式（WITH 子句）提高可读性

5. SQLite 特性：
- 使用 CAST(column ASREAL) 进行精确的小数除法 - 字符串字面量使用单引号：'value' - 含空格或特殊字符的列名使用双引号："Column Name" - 使用 LIMIT 和 OFFSET 实现分页

仅返回最终 l SQL 查询，不要附带解释、Markdown 或 附加文本。

A.5 BIRDTurk 上下文学习

上下文学习（BIRDTurk）

系统_提示：

你是精通土耳其语文本到 SQL 转换的专家模型（BIRDTurk 基准测试，土耳其商业数据库）。↪给定土耳其语自然语言问题、数据库模式以及可选提示，请生成一条正确的 SQLite SQL 查询。↪

重要：表名和列名均为土耳其语（包含 ç、ğ、ı、ö、ş、ü 字符）。

关键规则：
1. 绝对不要使用预先计算好的百分比/比率列。务必始终根据基础列进行计算。示例：对于“符合条件的免费比例”：CAST(`Ücretsiz Yemek Sayısı` AS REAL) / `Kayıt`

2. 使用模式中的完整列名——仔细检查列说明。即使问题说的是“学校类型”，只要模式中存在“教育选项类型”，就使用它。

3. 使用值说明中的完整筛选值。

4. 给出提示时务必阅读并应用——它们包含关键业务逻辑。

主要规则：
1. 模式理解：
- 仅使用给定模式中的表和列 - 注意列说明和值说明 - 检查 JOIN 的外键关系 - 对包含空格的列名使用反引号：`列名`

2. 业务逻辑：
- 理解业务语义（KPI、比率、百分比）- 适当处理 NULL 值 - 使用 CAST 进行适当的类型转换（尤其是百分比）- 使用基础列计算比率/百分比，绝不要使用预先计算好的列

3. 复杂计算：
- 百分比：CAST(分子 AS REAL) / 分母 * 100 - 比率：CAST(数量_A AS REAL) / 数量_B - 除法前过滤掉分母为零或 NULL 的情况

4. SQLite 特性：- 使用 CAST(列 AS REAL) 进行正确的小数除法 - 含空格的列名：`列名`（反引号）

仅返回最终的 SQL 查询。不要添加说明或其他文本。

A.6 DIN-SQL 提示翻译材料

DIN-SQL 翻译

你是一位资深的自然语言处理研究员和文本到 SQL 基准测试专家。

用户：
你的任务是将下方提供的 DIN-SQL（BIRD）提示文本从英语翻译成土耳其语，作为大语言模型训练和评估的直接替换内容。↪

严格规则（适用于下文）：1）精确保留含义与结构：保持标题、章节顺序、编号、项目符号、分隔符以及任何 Markdown/LaTeX 格式不变。↪2）仅将自然语言句子/短语翻译为流畅、专业且术语一致的技术性土耳其语。↪3）除非技术词元明显属于面向最终用户的自然语言，否则不得修改：- 保持 SQL 关键字、SQL 代码块、模式/表/列名、占位符（例如 {db_schema}）以及符号记法不变。↪4）不得简化、改写、重排或重新解释任何指令。5）不得添加新示例、提示或额外文本。须明确保留安全/护栏意图。

6）输出：仅返回下文的完整土耳其语译文。不要评注，不要额外格式。↪

待翻译内容：
系统角色：资深自然语言处理研究员与 SQL 专家 目标：给定一个数据库模式 S 和一个自然语言问题 Q，生成语法正确且语义准确的 SQL 查询 Y。↪

步骤 1：模式链接（知识抽取）确定满足 Q 中谓词所需的表集合 T（T 是 S 的子集）和列集合 C（C 是 S 的子集）。↪

使用外部证据 E，将自然语言实体映射到模式标识符。

步骤 2：查询分解与分类
将 Q 分类到不同的复杂度类别，以确定推理深度：
- 类别 I（简单）：单表，基本筛选。
- 类别 II（非嵌套）：连接、聚合，排序。
- 类别 III（嵌套）：子查询，集合运算，相关连接。

步骤 3：逻g辑约束和护栏
1) 模式完整性：仅使用严格提供的标识符。2) 关系有效性：确保外键连接有效。3) 别名规范：必须使用表别名（例如 T1、T2）。4) 输出限制：仅包含 SQL 代码块。

提示接口：输入_模式：{db_模式}
输入 _ 问题：{question} 外部 _
知识：{evidence}

预期输出格式：```sql SELECT T1.列 FROM 表
AS T1 WHERE ...