Journal of the Faculty of Engineering and Architecture of Gazi University 41:2 (2026) 1273-1288

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

A comparative evaluation of retrieval-augmented generation and retrieval-augmented fine-tuning methods for SQL query generation

Metin Bilgin*, Mehmet Bozdemir
Department of Computer Engineering, Faculty of Engineering, Bursa Uludağ University, 16110, Bursa, Türkiye

Highlights:

- Comparison of RAG and RAFT approaches for Turkish Text-to-SQL
- Accuracy gains on multi-table queries via Chain-of-Thought RAFT
- Effect of table count and relational complexity on SQL generation metrics

Keywords:

- RAG
- RAFT
- SQL
- Table selection
- LLMs

Article Info:

Research Article
Received: 04.10.2025
Accepted: 09.05.2026

DOI:

10.17341/gazimmfd.1796962

Acknowledgement:

This study was supported by the Bursa Uludağ University Scientific Research Projects Unit (BAP) under the FGA-2026-2534 grant

Correspondence:

Author: Metin Bilgin
e-mail: metinbilgin@uludag.edu.tr
phone: +90 312 224 275 5263

Graphical/Tabular Abstract

Natural language database querying facilitates data access for users without technical expertise. This study comparatively examines two fundamental approaches based on Large Language Models (LLMs) for this task: Retrieval-Augmented Generation (RAG) and Retrieval-Augmented Fine-Tuning (RAFT). In the RAG approach, relevant table structures are retrieved from a vector database and presented to the LLM as context. In the RAFT approach, the model is fine-tuned directly using a dataset that includes not only the relevant table definitions but also step-by-step reasoning (Chain-of-Thought, CoT) logic, thereby training it specifically for the task. The performance of these two methods was analyzed to determine the most effective strategy for generating SQL from Turkish natural language (Figure A).

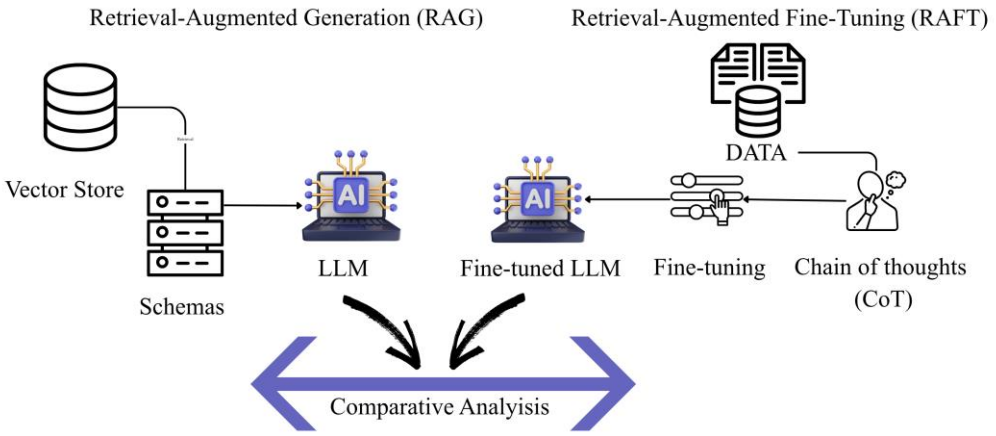


Figure A. Flow Chart of the Study

Purpose: The primary objective of this study is to automatically generate SQL queries from natural language expressions of users without SQL knowledge. Within this scope, the performance of the traditional RAG (Retrieval-Augmented Generation) approach is analysed comparatively with the RAFT (Retrieval-Augmented Fine-Tuning) method, aiming to determine the most effective method for generating SQL from Turkish natural language.

Theory and Methods: The study compared the RAG and RAFT methods. The GPT-4o, Gemini-1.5-pro, and Qwen2.5-14B-Instruct models were fine-tuned using the RAFT method on a test set comprising 1006 SQL queries from 15 different databases. In the RAG approach, table selection was performed using the OpenAI 'text-embedding-3-large' model and the Qdrant vector database. Model performance was evaluated using the Exact Match (EX), Table Similarity, and SQL Similarity metrics.

Results: RAFT significantly outperformed RAG across all models. Gemini-1.5-pro (84%) and GPT-4o (83%) achieved the highest Exact Match (EX) scores with RAFT, demonstrating comparable high performance. Qwen2.5-14B also showed a strong performance with 74%. The ablation study on GPT-4o showed that while CoT has minimal impact on simple queries, it provides significant performance gain in hard queries (4 tables), nearly matching the Oracle (upper bound) performance.

Conclusion: The findings indicate that the RAFT approach provides higher accuracy and structural consistency, particularly in complex queries, compared to the RAG method. This demonstrates that in the task of generating SQL from natural language, not only the model's capacity but also the fine-tuning strategy employed plays a decisive role in model performance.

Journal of the Faculty of Engineering and Architecture of Gazi University 41:2 (2026) 1273-1288

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Yapılandırılmış sorgu dilinde sorgu üretimi için erişim destekli üretim ve erişim destekli uyarlama yöntemlerinin karşılaştırmalı değerlendirmesi

Metin Bilgin*, Mehmet Bozdemir
Bursa Uludağ Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 16110, Bursa, Türkiye

Ö N E Ç I K A N L A R

- Türkçe Text-to-SQL için RAG ve RAFT yaklaşımlarının karşılaştırması
- Chain-of-Thought destekli RAFT ile çok tablolu sorgu doğruluğunda artış
- Tablo sayısı ve ilişkisel karmaşıklığın SQL üretim metriklerine etkisi

Makale Bilgileri
Araştırma Makalesi
Geliş: 04.10.2025
Kabul: 09.05.2026

DOI:
10.17341/gazimmfd.1796962

Anahtar Kelimeler:
RAG,
RAFT,
SQL,
tablo seçimi,
büyük dil modelleri

ÖZ
Doğal dilden veritabanı sorgusu üretimi, son kullanıcıların karmaşık SQL bilgisine ihtiyaç duymadan verilere erişimini kolaylaştıran kritik bir görevdir. Bu çalışmada, düşük kaynaklı bir dil olan Türkçe için Büyük Dil Modellerinin (LLM) Text-to-SQL performansları, Retrieval-Augmented Generation (RAG) ve Retrieval-Augmented Fine-Tuning (RAFT) yaklaşımları üzerinden karşılaştırmalı olarak analiz edilmiştir. Bird ve Spider veri kümelerinden derlenen 1.886 sorguluk (880 eğitim, 1006 test) hibrit bir veri seti kullanılmıştır. GPT-4o, Gemini-1.5-pro ve Qwen2.5-14B modelleri, şema eşleştirme yeteneğini artırmak amacıyla "doğru" ve "çeldirici" tabloların bir arada sunulduğu Chain-of-Thought (CoT) destekli RAFT yöntemiyle ince ayar işlemine tabi tutulmuştur. Sonuçlar, RAFT yönteminin tüm modellerde RAG'a üstünlük sağladığını göstermektedir. Gemini-1.5-pro (%84) ve GPT-4o (%83) en yüksek Exact Match (EX) skorunu paylaşırken, Qwen2.5-14B %74 ile rekabetçi bir performans sergilemiştir. GPT-4o üzerindeki ablasyon çalışması, CoT bileşeninin dört tablolu karmaşık sorgularda performansı önemli ölçüde artırdığını ve modelin gürültülü veriye rağmen Oracle sınırına yaklaştığını ortaya koymuştur. Bu çalışma, Türkçe sorgularda yalnızca anlamsal bağlamın yetersiz kaldığını ve yapısal akıl yürütmenin RAFT ile eğitilmesinin zorunlu olduğunu nicel verilerle kanıtlamaktadır.

A comparative evaluation of retrieval-augmented generation and retrieval-augmented fine-tuning methods for SQL query generation

H I G H L I G H T S

- Comparison of RAG and RAFT approaches for Turkish Text-to-SQL
- Accuracy gains on multi-table queries via Chain-of-Thought RAFT
- Effect of table count and relational complexity on SQL generation metrics

Article Info
Research Article
Received: 04.10.2025
Accepted: 09.05.2026

DOI:
10.17341/gazimmfd.1796962

Keywords:
RAG,
RAFT,
SQL,
table selection,
LLMs

ABSTRACT
Natural language to database query generation is a critical task that facilitates data access for end-users without requiring complex SQL knowledge. In this study, Text-to-SQL performance of Large Language Models (LLMs) for Turkish, a low-resource language, is comparatively analyzed using Retrieval-Augmented Generation (RAG) and Retrieval-Augmented Fine-Tuning (RAFT) approaches. A hybrid dataset of 1,886 queries (880 training, 1,006 testing) compiled from Bird and Spider benchmarks was used. GPT-4o, Gemini-1.5-pro, and Qwen2.5-14B models were fine-tuned using the Chain-of-Thought (CoT) supported RAFT method, where "oracle" and "distractor" tables are presented together to enhance schema linking capabilities. Results indicate that RAFT outperforms RAG across all models. Gemini-1.5-pro (84%) and GPT-4o (83%) shared the highest Exact Match (EX) scores, while Qwen2.5-14B achieved a competitive 74%. An ablation study on GPT-4o demonstrated that the CoT component significantly improved performance in complex queries involving four tables, approaching the Oracle upper bound despite noisy data. This study quantitatively demonstrates that semantic context alone is insufficient for Turkish natural language queries and that fine-tuning structural reasoning with RAFT is essential.

*Sorumlu Yazar / Yazarlar / *Corresponding Author / Authors: *metinbilgin@uludag.edu.tr, 512231003@ogr.uludag.edu.tr / Tel: +90 312 224 275 5263

1. Giriş (Introduction)

Dil modelleme, 1950'lerde Shannon'ın bilgi teorisini dile uygulamasıyla başlamış ve o zamandan beri doğal dil işleme, konuşma tanıma ve makine çevirisi gibi alanların temelini oluşturmuştur [1]. LLM (Large Language Models)'ler insan benzeri metin üretmek için tasarlanmış yapay zekâ sistemleridir. Bu sistemlerin ilk ortaya çıkışı 2017 yılında yayınlanan bir makaleye dayanmaktadır ve bu makale doğal dil işleme alanında büyük bir dönüm noktası olmuştur [2]. Makaledeki transformer mimarisi temel alınarak bugünkü büyük dil modellerinin temeli atılmıştır. Geniş çapta ilk olarak tanınan LLM, 2019 yılında OpenAI tarafından geliştirilen GPT-2 modeli olmuştur. Bu model daha önce geliştirilen bütün modellerden çok daha büyük olup 1,5 milyar parametreye sahiptir. Ayrıca bu model, fine-tune işlemi yapılmadan birçok NLP (Natural Language Processing) görevinde kullanılmıştır. [3]. Sonraki yıllarda OpenAI GPT-3, GPT-3.5 (ChatGPT), GPT-4o ve son olarak da GPT o1-preview modelini çıkartmıştır [4]. OpenAI'ın öncülüğünü yaptığı bu alana Google, Facebook ve Microsoft gibi firmalar da katılarak kendi dil modellerini oluşturmuşlardır. Büyük dil modellerinin yaygınlaşması, Türkçe gibi görece daha düşük kaynaklı dillerdeki doğal dil işleme (NLP) çalışmalarına da ivme kazandırmıştır; nitekim güncel literatürde, LLM'lerin ince ayar (fine-tuning) süreçlerinden geçirilerek Türkçe soru-cevaplama gibi karmaşık görevlerde yüksek doğruluk oranlarına ulaşabildiği gösterilmektedir [5].

RAG büyük dil modellerine dış kaynaklardan veri ekleyerek doğruluğu ve güvenirliği arttıran bir yaklaşımdır. LLM'lerin bilgiye erişiminin sınırlı olduğu durumlarda sıkça kullanılmaktadır [6]. RAG son yıllarda büyük dil modelleri arasında popülerlik kazanmıştır. Parametrik bellek gerektiren dil modelleri ile yapılan doğal dil işleme çalışmaları bazı zorlukları beraberinde getirmektedir. Model eğitimi, zaman ve maliyet açısından oldukça yoğun bir süreç olarak kullanıcıların karşısına çıkmaktadır. Parametrik modellerin eğitimi uzun ve zaman alıcı olduğu için RAG gibi bellek gerektirmeyen yaklaşımlar ön plana çıkmıştır [7]. RAG ile gerekli olan bilgiler dış kaynaklardan çekilerek model eğitimi sırasında ortaya çıkan zaman ve maliyet gibi sorunlar etkili bir şekilde çözümlenmiştir. RAG ile büyük dil modeline verilen bilgiler zenginleştirilerek daha isabetli ve tutarlı sonuçlar vermesi sağlanmıştır. Bu sayede büyük dil modellerinin en büyük sorunlarından biri olan halüsinasyon görmesinin de büyük ölçüde önüne geçilmiştir [8] .

Veritabanı, birbiriyle ilişkili tabloların bir arada tutulduğu koleksiyonlardır. Bu koleksiyonlar ile veriler kolay bir şekilde manipüle edilebilmektedir. Veritabanı modelleri aşağıdaki gibi 8 kategoriye ayrılmaktadır.

- Düz model veya tablo modeli
- Hiyerarşik Veri Modeli
- Ağ veri modeli
- İlişkisel Veri Modeli
- Nesne Yönelimli Veri Modeli
- Nesne İlişkisel Veri Modeli
- Çoklu Ortam Veri Modeli
- Dağıtık Veri Modeli

İlişkisel veri tabanları günümüzde en çok kullanılan veri tabanı sistemlerindendir. Bu veritabanları, verilerin satır ve sütunlar halinde yapılandırıldığı tablolardan oluşur. Bu tablolar arasında çeşitli ilişkiler bulunmaktadır. Bu ilişkiler sayesinde veriler arasında tutarlılık, bütünlük ve bağlantılar korunur [9].

Chamberlin ve Boyce tarafından 1974 yılında IBM laboratuvarında ortaya çıkan SEQUEL (Yapılandırılmış İngilizce Sorgu Dili) bugün kullanılan SQL dilinin kökenini oluşturmaktadır [10]. SEQUEL, IBM'in System R adlı ilişkisel bir veri tabanı yönetim sisteminde (Relational Database Management System [RDBMS]) depolanan verilere erişerek bunları yönetmenin basit bir yolu olarak tasarlanmıştır. SEQUEL ismi İngiliz hava taşıma şirketi Hawker Siddeley tarafından daha önceden tescil edildiği için bu isim daha sonraları SQL olarak değiştirilmiştir. Günümüzde MySQL, SQLite ve PostgreSQL gibi popüler açık kaynak kodlu SQL dilleri geliştirilmiştir. Günümüzde ORACLE ve Microsoft firmasının sunduğu ticari anlamdaki SQL sistemleri yaygın olarak kullanılmaktadır [11]. Bu dil veritabanı yönetim sistemlerinde verilerin yönetilmesi ve manipülasyonu için kullanılmaktadır. SQL dilinin kullanıcılara sunduğu kolaylıklara rağmen bu dili kullanabilmek için sorgu dilinin temel yapısını iyi bilmek gerekmektedir. Bilindiği gibi SQL veritabanları ilişkisel veri modelleri üzerine kurulmuştur ve ACID (Atomicity, Consistency, Isolation, Durability) gibi özellikleri desteklemektedir. Özellikle veriler üzerinde CRUD (Create, Read, Update, and Delete) gibi operasyonları kolaylıkla gerçekleştirilmektedir [12]. Ayrıca transaction ve procedure gibi yapılar SQL dilinin yeteneklerini artıran unsurlardır. Görüldüğü üzere SQL dili bize birçok kolaylık sağlamasına rağmen bu dili kullanmak yoğun bir bilgi ve birikim gerektirmektedir.

Bu çalışmanın motivasyon noktası, SQL dilinde uzman olmayan kişilerin veritabanı yönetim sistemlerinde bulunan verilere erişim ve analiz süreçlerini kolaylaştıracak altyapının oluşturulmasını sağlamaktır. Bu doğrultuda, son kullanıcıların doğal dilde ifade ettikleri bir sorguyu sisteme girdiklerinde, bu talebe karşılık gelen SQL ifadesinin otomatik olarak üretilebilmesi hedeflenmektedir. Üretilen SQL sorgusu, ilgili veritabanı üzerinde eş zamanlı olarak çalıştırılır ve sonuçlar kullanıcıya SQL kodu ya da görsel bir arayüz (View) biçiminde sunulabilmektedir. Doğal dilde ifade edilen sorguyla ilişkili veriler ise, veritabanı şemalarının yer aldığı bir vektör veritabanı üzerinden elde edilmektedir [13].

Ancak burada önemli bir husus, büyük dil modellerinin (LLM) sınırlılıklarının farkında olunması gerekliliğidir. LLM tabanlı yaklaşımlar zaman zaman hatalı veya bağlama uygun olmayan sorgular üretebilmektedir. Bu nedenle, sistemin son kullanıcıya tamamen güvenli ve hatasız bir otomasyon olarak sunulması yerine, bir öğrenme aracı ya da verimlilik artırıcı bir yardımcı sistem olarak konumlandırılması daha uygun olacaktır. Çalışma kapsamında LLM'in uzmanlığın yerini alması değil, uzmanlık gerektiren alanlara erişimi kolaylaştırması ve kullanıcının sistemle etkileşimini basitleştirmesi amaçlanmaktadır.

Bu çalışmada, doğal dilden SQL kodu üretimi üzerine hem RAG (Retrieval-Augmented Generation) hem de RAFT (Retrieval-Augmented Fine-Tuning) yöntemleri kullanılmıştır. Araştırmanın temel amacı, RAG yöntemiyle elde edilen SQL üretim başarımlarını, RAFT yöntemiyle elde edilen sonuçlarla karşılaştırmalı olarak değerlendirmektir. Çalışmada özellikle RAFT yöntemi üzerinde yoğunlaşılmıştır. Kullanılan veri setinde yer alan hatalı, eksik ya da boş çıktı üreten sorgular ayrıntılı biçimde tespit edilip elden geçirilmiş ve veri seti RAFT yöntemine uygun formata dönüştürülmüştür. Ardından, önerilen büyük dil modelleri ile fine-tune (ince ayar) işlemleri gerçekleştirilmiştir. Son aşamada ise, RAFT yöntemiyle elde edilen sonuçlar hem kendi içinde, hem de RAG yöntemiyle üretilen skorlarla model bazında karşılaştırılmıştır. Elde edilen bulgular, RAFT yönteminin doğal dilden SQL kodu üretiminde daha yüksek doğruluk ve tutarlılık sağladığını göstermektedir.

2. Literatür Çalışmaları (Literature Studies)

Bu bölümde ilgili alanda yapılan bilimsel çalışmalar ile ilgili bilgiler sunulmaktadır. Yapılan literatür taramasında, Text-to-SQL alanında çok sayıda çalışma gerçekleştirildiği görülmüştür. Bu çalışmaların

büyük bir bölümü, İngilizce dilinde hazırlanmış veri setleri üzerinde, büyük dil modellerinin eğitilmesi ya da ince ayar (fine-tuning) yapılması temeline dayanmaktadır. Ayrıca, son dönemde büyük dil modelleri ile RAG (Retrieval-Augmented Generation) ve benzeri yaklaşımlar kullanılarak, doğal dilden SQL sorgusu üretimi konusunda önemli ilerlemeler kaydedilmiştir. Mevcut literatür incelendiğinde, bu alandaki araştırmaların temel olarak; prompt mühendisliği, ince ayar (fine-tuning) mimarileri ve düşük kaynaklı diller (özellikle Türkçe) üzerine yoğunlaştığı görülmektedir.

Modelleri yeniden eğitmek yerine modele sunulan bağlamı ve istemleri optimize ederek başarıyı artırmayı hedefleyen çalışmalar, literatürün ilk ana eksenini oluşturmaktadır. Bu kapsamda Dong vd. [14], ChatGPT tabanlı zero-shot Text-to-SQL yöntemi olan C3'ü önermiştir. Yöntem, özel "Clear Prompting", "Calibration with Hints" ve "Consistent Output" bileşenlerinden oluşur. Spider veri setinde %82,3 execution accuracy ile zero-shot yöntemler arasında SOTA sonuç alınmıştır ve bu yöntem, pratikte az sayıda token ile yüksek başarı sağlamıştır. Benzer şekilde Li vd. [15], LLM tabanlı Text-to-SQL için iki aşamalı, prompt güçlendirmeli bir iyileştirme yöntemi olan PET-SQL'i sunmuşlardır. İlk aşamada örneklerden PreSQL üretilmekte, ikinci aşamada ise şema bağlantısı daha net hale getirilip nihai SQL üretimi yapılmaktadır. Ayrıca, farklı LLM'ler arasında tutarlılık analiziyle hataların önüne geçilmekte ve Spider benchmark'ında %87,6 execution accuracy elde edilmiştir. Karmaşık sorguların çözümünde ise Pourreza ve Rafiei [16], Text-to-SQL görevinde LLM'lerin başarısını artırmak için görevi alt parçalara bölen ve bu alt çözümleri LLM'e besleyerek daha yüksek doğruluk sağlayan bir yaklaşım sunmuşlardır. Çalışma, özellikle in-context learning ile sıfırdan veya az örnekle öğrenmenin ötesine geçip, kendi kendine düzeltme mekanizmalarını da içermekte ve Spider veri setinde yeni bir SOTA (state-of-the-art) başarı elde etmiştir. Mai vd. [17], Text-to-SQL görevinde in-context learning sırasında örnek seçiminin performansa etkisini analiz etmiş ve MARLO adlı, metadata bağımsız gömlemeler öğrenen bir yöntem önermiştir. Bu yöntem, daha yapısal ve görev odaklı örnek seçimiyle Spider veri kümesinde %2,9 oranında daha yüksek doğruluk sağlamıştır.

Modelin iç ağırlıklarının göreve özgü güncellendiği ince ayar (fine-tuning) yaklaşımları veya hibrit mimariler üzerine yapılan araştırmalar ise, özellikle karmaşık veritabanı yapılarında daha yüksek kararlılık sunmayı hedeflemektedir. Li vd. [18], Text-to-SQL dönüşümünde şema bağlantısı ve SQL iskeletinin birlikte ele alınmasının oluşturduğu zorlukları azaltmak amacıyla, bu iki adımı birbirinden ayıran yeni bir framework (RESDSQL) önermişlerdir. Bu yöntemde, öncelikle yalnızca en alakalı şema ögeleri seçilmekte ve ardından model, önce SQL iskeletini, sonra da tam sorguyu oluşturmaktadır. Bu yapı, özellikle karmaşık SQL operatörlerinin bulunduğu durumlarda başarımı artırmış ve Spider veri kümesinde yüksek doğruluk elde edilmiştir. Gao vd. [19], LLM tabanlı Text-to-SQL sistemleri için kapsamlı bir benchmark çalışması sunmuşlardır. Prompt mühendisliğinde kullanılan farklı stratejiler sistematik olarak karşılaştırılmış ve açık kaynak LLM'lerin hem in-context learning, hem de supervised fine-tuning ile Text-to-SQL görevinde potansiyeli detaylı şekilde incelenmiştir. Çalışmanın sonucunda DAIL-SQL isimli entegre bir çözümle Spider veri kümesinde %86,6 execution accuracy'ye ulaşılmıştır. Daha güncel hibrit yaklaşımlarda ise Gao vd. [20] adında çoklu jeneratörlü, toplu (ensemble) bir Text-to-SQL framework'ü önermiştir. Hem prompt mühendisliği hem de fine-tuning stratejilerini birleştirerek çeşitli benchmarklarda (özellikle Spider'da %89,65 doğrulukla) SOTA performans elde edilmiştir. Zhu vd. [21], Text-to-SQL alanında LLM (Büyük Dil Modeli) tabanlı yöntemlere dair kapsamlı bir literatür taraması sunmuşlardır. Çalışmada prompt mühendisliği, fine-tuning ve agent tabanlı modeller gibi stratejiler sınıflandırılmış; değerlendirme metrikleri ve önde gelen veri kümeleri detaylıca özetlenmiştir. Bu çalışma, alandaki trendleri ve mevcut zorlukları anlamak için temel bir kaynak sunmaktadır. Literatürdeki en güncel yaklaşımlardan biri olan Rossiello vd. [22], metinden SQL üretimi için 'Rasyonalizasyon Modelleri' (Rationalization Models) önermişlerdir. Bu çalışmada, büyük bir öğretmen modelden (Teacher LLM) elde edilen 'adım adım akıl yürütme' (Chain-of-Thought) verileriyle daha küçük modeller (Llama 3.1-8B) üzerinde ince ayar (fine-tuning) yapılmıştır. Araştırmacılar, modele sadece nihai SQL sorgusunu değil, sorguyu oluştururken izlediği ara adımları ve mantıksal açıklamaları (rationales) da öğreterek, özellikle karmaşık ve çok tablolu sorgularda (BIRD veri seti üzerinde) başarının önemli ölçüde arttığını göstermişlerdir.

Text-to-SQL literatürünün büyük çoğunluğu İngilizce odaklı olmakla birlikte, düşük kaynaklı bir dil olan Türkçe üzerine yapılan çalışmalar sınırlı sayıdadır. Kanburoğlu ve Tek [23] yaptıkları çalışmada doğal dilin SQL'e dönüştürülmesini incelemişlerdir. Kullandıkları veri seti Text-to-SQL isimli ve dili İngilizcedir. Türkçe veriseti noktasında eksiklik olduğu için TUR2SQL adında 10809 adet doğal dil ve SQL karşılıklarından oluşan bir veriseti hazırlamışlardır. SQLNet ve ChatGPT modellerini kullanarak TUR2SQL veriseti üzerinde çeşitli çalışmalar gerçekleştirmişlerdir. ChatGPT'nin kullanılan veri seti üzerinde daha yüksek performans sergilediğini göstermişlerdir. Kanburoğlu ve Tek [24] tarafından yapılan başka bir çalışmada doğal dil ifadelerinin SQL'e dönüştürülmesi incelenmiştir. Çalışmalarında WikiSQL ve Spider verisetleri kullanılmıştır. SQLNet, RAT-SQL, Seq2SQL gibi modelleri eğiterek sistemin başarısını ölçmüşlerdir. WikiSQL veriseti üzerinde SeaD adlı model %92,9'luk execution accuracy başarısı göstermiştir. Spider veriseti üzerinde ise T5-SR-3b modeli %79,9'luk bir exact match accuracy başarımı göstermiştir. Demirkiran vd. [25] tarafından geliştirilen çalışmada TUR2SQL veriseti kullanılarak T5,SQLCoder ve GPT-3.5 Turbo modelleri fine-tune edilmiştir. Veriseti Türkçe doğal dildeki sorgular ve şemların bir arada olduğu bir yapıdan oluşmaktadır. Veriler belirtilen modeller ile fine-tune edildiğinde SQLCoder (With Schema Context) ile %97 , T5 (With Schema Context) ile %98 , GPT-3.5 Turbo ile %94'lük bir EX başarısı elde edilmiştir. Başka bir çalışmada ise Kanburoğlu ve Tek [26], Türkçe için karmaşık ve iç içe (nested) sorgu eksikliğini gidermek amacıyla Spider veri setini insan çevirisiyle Türkçeye uyarlayarak TURSpider veri setini literatüre kazandırmışlardır. Çalışmalarında, Turkcell-LLM ve Trendyol-LLM gibi Türkçe destekli modelleri bu veri seti üzerinde ince ayar (fine-tuning) yaparak eğitmişler ve GPT-4 gibi kapalı kaynak modellerle kıyaslamışlardır. Ayrıca, modelin kendi çıktısını düzelttiği Chain-of-Feedback (CoF) yöntemini uygulayarak, ince ayar yapılmış Türkçe modellerin, karmaşık sorgularda dahi rekabetçi sonuçlar (execution accuracy) verebildiğini göstermişlerdir.

Ancak, literatürde Türkçe dilinde ve RAG veya RAFT tabanlı bir yöntemin kullanıldığı herhangi bir çalışmaya rastlanmamıştır. RAG yönteminde, modelin yanıt üretirken dış kaynaklardan bilgi getirmesi sağlanmakta ve modelin bilgi tabanlı eksikleri tamamlanmaktadır. Buna karşılık, RAFT (Retrieval-Augmented Fine-Tuning) yaklaşımı ise, geri getirme ve bağlamsal veri sağlama adımlarının doğrudan modelin ince ayar (fine-tune) sürecine entegre edilmesiyle modelin çok daha hassas ve göreve özgü sonuçlar üretmesini hedeflemektedir. Bu durum, hem Türkçe dilinde hem de RAG/RAFT mimarisiyle doğal dilden SQL üretimi gerçekleştiren bu çalışmanın alanında özgün ve öncü bir katkı sunmasını sağlamaktadır. Bu bağlamda, literatürde öne çıkan çalışmaların yöntem, kullanılan temel model (LLM), hedef dil, veri seti ve CoT (Zincirleme Düşünce) kullanımı açısından detaylı karşılaştırması Tablo 1'de sunulmuştur.

Bu çalışmanın literatüre temel katkısı, mevcut yöntemlerden farklı olarak, yalnızca RAG tabanlı otomatik şema seçimiyle SQL sorgusu üretmenin yanı sıra, RAFT yaklaşımında özellikle Chain of Thought (CoT) akıl yürütme süreçlerini öne çıkararak modelin adım adım

Tablo 1. Literatürdeki ilgili çalışmaların karşılaştırmalı özeti (Comparative summary of related studies in the literature)

Çalışma	Yöntem / Yaklaşım	Temel Model (LLM)	Dil	Veri Seti	CoT Kullanımı
Li vd. [18] (RESDSQL)	Fine-Tuning (Şema ve İskelet Ayrıştırma)	T5-Base / Large	İngilizce	Spider	Hayır
Pourreza ve Rafiei [16] (DIN-SQL)	Prompt Müh. (Decomposition + Self-Correction)	GPT-4 / CodeX	İngilizce	Spider / BIRD	Evet
Dong vd. [14] (C3)	Zero-Shot Prompting (Hints + Self-Consistency)	ChatGPT (GPT-3.5)	İngilizce	Spider	Hayır
Gao vd. [19] (DAIL-SQL)	Prompt Müh. (Selection & Org.) + SFT	GPT-4 / LLaMA-2	İngilizce	Spider	Hayır
Rossiello vd. [22]	Fine-Tuning (CoT Distillation + Rationales)	Llama 3.1 (8B/70B)	İngilizce	BIRD	Evet (Rationales)
Kanburoğlu ve Tek [23]	Fine-Tuning (Temel)	ChatGPT / SQLNet	Türkçe	TUR2SQL	Hayır
Demirkiran vd. [25]	Fine-Tuning (Schema Context)	SQLCoder / T5	Türkçe	TUR2SQL	Hayır
Kanburoğlu ve Tek [26]	Fine-Tuning + Chain-of-Feedback (CoF)	Turkcell-LLM / GPT-4	Türkçe	TURSpider	Evet (CoF)
Önerilen Yöntem	RAFT (Retrieval-Augmented Fine-Tuning)	GPT-4o, Gemini-1.5, Qwen2.5	Türkçe	Bird & Spider (TR)	Evet (CoT)

mantıksal çıkarım yapabilmesini sağlamasıdır. RAFT kapsamında, modelin yalnızca doğru şema bilgisini değil, aynı zamanda doğal dil sorgusunu mantıksal olarak parçalayarak, alt adımlar halinde SQL sorgusu üretmesi hedeflenmiştir. Bu süreçte, klasik seq2seq eğitiminden farklı olarak, modelin çok aşamalı reasoning ve CoT tabanlı çözüm yollarını öğrenmesi sağlanmış ve bu sayede daha karmaşık ve çok adımlı sorgulara yüksek doğrulukla yanıt üretme kapasitesi artırılmıştır. Ayrıca, çalışmada RAG ve RAFT yöntemlerinin performansları çeşitli metrikler açısından karşılaştırılmış, CoT süreçlerinin özellikle RAFT performansına katkısı detaylı biçimde analiz edilmiştir. Bu yönüyle çalışma, Türkçe doğal dilde ve RAG/RAFT mimarisiyle SQL sorgusu üreten, aynı zamanda CoT süreçlerini inceleyen ve karşılaştırmalı analiz yapan ilk örneklerden biri olarak alana önemli bir katkı sunmaktadır.

3. Materyal ve Yöntem (Materials and Methods)

Bu bölümde, Türkçe doğal dil sorgularından SQL üretimi için önerilen RAFT (Retrieval-Augmented Fine-Tuning) ve RAG (Retrieval-Augmented Generation) yaklaşımlarının metodolojik çerçevesi sunulmuştur. Çalışma, veri seti hazırlığı, vektör tabanlı geri getirme (retrieval) mekanizması, Chain-of-Thought (CoT) veri üretimi ve model ince ayar (fine-tuning) süreçlerini kapsamaktadır. Deneysel tasarım, literatürde kabul gören standartlar ve SOTA (State-of-the-Art) modeller referans alınarak oluşturulmuştur.

3.1. Veri Seti (Data Set)

Çalışmanın deneysel altyapısı, metin-SQL (Text-to-SQL) alanında standart kabul edilen Bird [27] ve Spider [28] veri kümeleri üzerine inşa edilmiştir. Bu veri kümelerinin seçilme nedeni, farklı zorluk seviyelerinde (kolay, orta, zor) karmaşık veritabanı şemalarını ve gerçek dünya senaryolarını içermeleridir. Eğitim Seti: RAFT modelinin eğitimi için, 12 farklı veritabanını kapsayan toplam 880 adet sorgu seçilmiştir.

Test Seti: Modelin genelleme yeteneğini (generalization) ölçmek amacıyla, eğitim setinde hiç yer almayan (zero-shot setting) 15 farklı veritabanına ait 1006 adet sorgu test için ayrılmıştır. Literatürde Text-to-SQL görevleri için 500-1000 arası örneklem grubu, istatistiksel anlamlılık ve genelleme kapasitesini ölçmek için yeterli kabul edilmektedir [27], [28]. 1006 sorguluk bu hacim; toplamda 90 farklı tablo yapısını ve çeşitli ilişkisel operasyonları bünyesinde barındırarak, modelin belirli şemaları ezberlemesini önlemek ve daha önce karşılaşılmayan alanlardaki (cross-domain) başarısını kanıtlamak adına yeterli derinliğe ve çeşitliliğe sahiptir. Veri setlerinin veritabanı bazlı istatistiksel dağılımı Tablo 2'de sunulmuştur.

Modelin eğitimi sırasında kullanılan veri formatı, giriş (input) olarak doğal dil sorgusu ve veritabanı şemasını; çıkış (output) olarak ise adım adım akıl yürütme (Chain-of-Thought) ve SQL sorgusunu içerecek şekilde yapılandırılmıştır. Eğitim setinde kullanılan veri yapısına ait bir örnek aşağıda sunulmuştur:

Örnek Veri Yapısı
Doğal Dil Sorgusu (Question): "İngilizce dilini kullanan kaç TV kanalı var?"
Bağlam (Context - Tablo Yapısı): CREATE TABLE TV_Channel (id TEXT PRIMARY KEY, series_name TEXT, Country TEXT, Language TEXT, ...);
Model Çıktısı (CoT Answer):
Reasoning:
1. İngilizce dilini kullanan TV kanallarının sayısını bulmak için 'TV_Channel' tablosunda 'Language' sütununda 'English' değerine sahip kayıtları sorgulamalıyım.
2. Bu kayıtların sayısını hesaplayarak sonuca ulaşacağım.
Answer: SELECT count(*) FROM TV_Channel WHERE Language = "English";

3.2. CoT (chain-of-thought) Answer Oluşturma Süreci (Chain-of-Thought (CoT) Answer Generation Process)

RAFT sürecinde modelin adım adım akıl yürütme (Chain-of-Thought, CoT) yeteneği kazanabilmesi için, eğitim verilerinin mantıksal çözüm adımlarını içerecek şekilde yapılandırılması gerekmektedir. Veri setinin model eğitimi için gerekli formata dönüştürülmesinde kullanılan kod yapısı Şekil 1'de sunulmuştur. Ham veri seti başlangıçta bu yapıda olmadığından, her bir sorguya karşılık gelen bağlam, ilgili ve ilgisiz tüm şemalar bir arada olacak şekilde hazırlanmıştır. Bu şemalar etiketlenmeden modelin "Context" girdisine dahil edilmiş; modelden ise bu bağlam altında CoT biçiminde yanıt üretmesi beklenmiştir.

Bu amaçla, OpenAI'ın GPT-4o dil modeli bir API aracılığıyla kullanılmış ve Şekil 2'de ayrıntıları verilen prompt (istem) şablonu uygulanmıştır. İlgili şablonda, "question" (soru) ve SQLite şemaları ile beklenen düşünce zinciri tanımlanmıştır. Modele, "few-shot learning" yöntemi dikkate alınarak örnek bir CoT yanıtı da prompt'a eklenmiştir. Bu yapı sayesinde, GPT-4o modelinin her bir veri için CoT üretmesi sağlanmıştır. Sonuç olarak, elimizde bulunan 880 adet verinin tamamı için CoT süreci bu yöntemle gerçekleştirilmiştir.

Tablo 2. Eğitim ve test veri setlerinin istatistiksel dağılımı (Statistical distribution of training and test datasets)

Eğitim (Fine-Tuning) Veri Seti			Test (Değerlendirme) Veri Seti		
Veritabanı Adı	Sorgu Sayısı	Tablo Sayısı	Veritabanı Adı	Sorgu Sayısı	Tablo Sayısı
hr	124	7	architecture	17	3
world	120	3	cars	82	4
restaurant	117	3	chinook	84	11
store	106	13	coffee_shop	18	4
video_games	101	8	company_employee	16	3
music	100	4	college	170	11
student_transcripts	76	11	company_office	40	3
voter	64	2	customer_complaints	46	4
tvshow	62	3	department_store	88	14
allergy	6	2	election	68	3
e-learning	2	11	movie	98	3
dorm	2	2	sakila	82	14
-	-	-	soccer	106	7
-	-	-	social_media	76	3
-	-	-	voter	15	3
Toplam	880	69	Toplam	1006	90

```
with open(output_file, "w", encoding="utf-8") as f:
    for _, row in df.iterrows():
        data = {
            "messages": [
                {
                    "role": "system",
                    "content": "Sen, doğal dildeki soruları SQLite sorgularına çeviren akıllı bir
asistansın..."
                },
                {
                    "role": "user",
                    "content": f"### Soru:\n{row['question']}\n\n### Şema:\n{row['context']}\n\n###
Gerekçelendirme ve SQL:"
                },
                {
                    "role": "assistant",
                    "content": row['cot_answer']
                }
            ]
        }
        f.write(json.dumps(data, ensure_ascii=False) + "\n")
```

Şekil 1. GPT-4o modeli için JSONL eğitim verisinin python koduyla hazırlanması
(Preparation of JSONL training data for GPT-4o model using python code)

3.3. RAFT (Retrieval-Augmented Fine-Tuning)

RAFT, dil modellerini RAG senaryolarına hazırlamak için geliştirilmiş bir ince ayar yöntemidir. Standart RAFT uygulamalarında model, ilgili (oracle) ve ilgisiz (distractor) belgelerin karışık olduğu bir bağlamdan doğru bilgiyi çıkarmayı öğrenir [29]. Bu çalışmada, SQL üretimi gibi yüksek hassasiyet gerektiren bir görevde modelin şema eşleştirme (schema linking) performansını artırmak amacıyla Kılavuzlu RAFT (Oracle-Guided RAFT) stratejisi uygulanmıştır. Bu strateji iki aşamadan oluşur:

Eğitim (Training) Kılavuzlu Öğrenme: Eğitim aşamasında modelin, karmaşık veritabanı şemaları arasından doğru tabloyu seçme mantığını (reasoning) daha hızlı kavraması hedeflenmiştir. Bu nedenle, modele girdi (input) olarak soru ve tüm şemaların (context) yanı sıra, doğru cevabı içeren ilgili şemalar (oracle schemas) da açıkça sunulmuştur. Bu ek bilgi, modelin dikkatini (attention) doğru tablolara yönlendirmesini sağlamış ve 'Chain-of-Thought' (CoT) üretim kalitesini artırmıştır.

Test (Inference) Gerçek Senaryo: Modelin ezberlemesi (overfitting) engellenerek genelleme yeteneğinin ölçülmesi amacıyla, test aşamasında oracle bilgisi tamamen kaldırılmıştır. Model, tıpkı gerçek bir RAG senaryosunda olduğu gibi, kendisine sunulan karışık şema havuzu (context) içerisinden ilgili tabloyu kendisi tespit etmek ve SQL sorgusunu üretmek zorunda bırakılmıştır.

Şekil 3 incelendiğinde, RAFT'ın temel amacının büyük dil modellerini (LLM), "açık kitap (open book)" sınav ortamına benzer bir şekilde, kendisine verilen dış dokümanlar üzerinden hem doğru cevabı bulmaya hem de ilgisiz (distractor) dokümanları görmezden gelmeye teşvik etmek olduğu görülmektedir

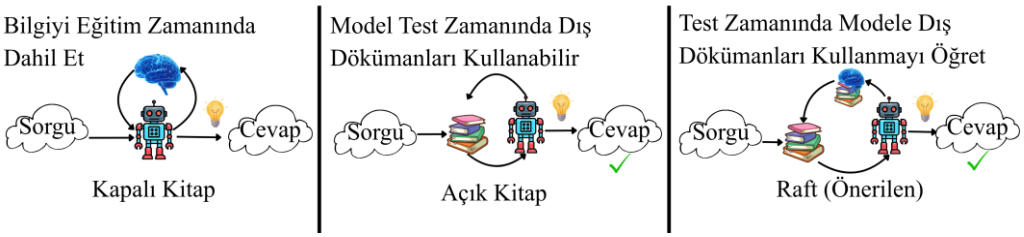
```
prompt_template = f"""
Aşağıda doğal dildeki sorgu (question) ve SQLite sorgusuna göre bir Düşünce Zinciri
(cot_answer) örneği verilmiştir.
question:
'Metallica' sanatçısının kaç albümü var?
SQLite:
SELECT COUNT(*) AS AlbumCount
FROM Album
JOIN Artist ON Album.ArtistId = Artist.ArtistId
WHERE Artist.Name = 'Metallica';

Reasoning:
1. 'Metallica' sanatçısının albüm sayısını bulmak için, `Artist` tablosunda 'Metallica' adına
karşılık gelen `ArtistId` değerini bulmalıyım.
2. `Artist` tablosunda `Name` sütununda 'Metallica' adına karşılık gelen `ArtistId` değerini
sorgulayacağım.
3. Elde ettiğim `ArtistId` değerini `Album` tablosunda kullanarak, `ArtistId` değerine sahip tüm
albümleri sorgulamalıyım.
4. `Album` tablosunda, bu `ArtistId` değerine sahip tüm kayıtların sayısını hesaplayarak sonucu
elde edeceğim.

Yukarıdaki örneği dikkate alarak, **aşağıda verilen question ve SQLite sorgusuna göre bir
düşünce zinciri (cot_answer) oluşturun. Yalnızca bu bilgilere odaklanın ve farklı bir bağlamdan
bilgi eklemeyin.**

(Adım sayısı en az 2, en fazla da 6 olabilir. Türkçe olarak cevapla.
question:
{question}
SQLite:
{SQLite}
Reasoning:?
"""
```

Şekil 2. Chain-of-Thought (CoT) oluşturulmasında kullanılan prompt yapısı (Prompt structure used for Chain-of-Thought (CoT) generation)



Şekil 3. RAFT'ın (Retrieval-Augmented Fine-Tuning) klasik yöntemlerle karşılaştırılması [29]
(Comparison of RAFT (Retrieval-Augmented Fine-Tuning) with classical methods)

Retriever tarafından getirilen birden fazla belge:

- En az biri oracle/golden (doğru, ilgili) belge
- Diğerleri ise distractor (yanlış, ilgisiz) dokümanlar

Modelden Beklenen Çıktı (Output):

- Soruya yalnızca ilgili (golden/oracle) doküman(lar) üzerinden, adım adım akıl yürütme (Chain-of-Thought, CoT) ile açıklamalı ve alıntı yaparak cevap üretmesi.

3.4. RAG (Retrieval-Augmented Generation)

RAG birçok NLP görevi için giderek daha da popüler hale gelmektedir. LLM'lerin uzmanlık alan bilgisi gerektirdiği yerlerde sıklıkla başvurulan bir alandır. LLM'e RAG sayesinde daha doğru sorgu ve context'ler verilmektedir. Örneğin müşteri chatbot'ları müşterilerine konu ile daha alakalı içerikler üretmektedir [8].

Eş. 1'deki X vektör veritabanından aranmak istenen sorgudur. X bağlamı göz önüne alındığında ilgili K bilgisi Eş. 3'e göre ilk olarak derlemden veya veritabanından alınır. Alınan bilgi daha sonra ilgili prompt'a girdi olarak verilir.

$$X = [x_1, x_2, \ldots, x_n] \quad (1)$$

$$Y = [y_1, y_2, \ldots, y_n] \quad (2)$$

$$K = \text{Retrieval}(X) \quad (3)$$

$$P(Y \mid X) = \prod_t P(y_t \mid X, K, y_1, \ldots, y_{t-1}) \quad (4)$$

Eş. 2'de ifade edilen Y üretilen metindir. Prompt K, kodlayıcıyı harici bilgileri dahil etmeye yönlendirir. Eş. 4'e göre X ve K verileri ele

alınarak her seferinde *Y* çıktı değeri hesaplanır. Bütün bu değerlerin olasılıklarını arka arkaya çarparak genel formüle ulaşılır [30]. Bu teorik çerçeve doğrultusunda, çalışmada kurulan RAG mimarisi herhangi bir ince ayar (fine-tuning) gerektirmeden, sorgu anında en alakalı tablo yapılarını modele sunmak üzere tasarlanmıştır. Sistemin teknik konfigürasyonu ve kullanılan hiperparametreler aşağıda detaylandırılmıştır:

Embedding (Gömme) Modeli: Doğal dil sorgularının ve tablo şemalarının vektör temsilleri, anlamsal yakalamadaki başarısı nedeniyle OpenAI'ın text-embedding-3-large modeli ile oluşturulmuştur.

Vektör Boyutu (Dimension): Her bir veri 3072 boyutlu vektör uzayında temsil edilmiştir.

Vektör Veritabanı: Vektörlerin saklanması ve hızlı aranması için Qdrant veritabanı kullanılmıştır.

Benzerlik Metriği: Sorgu ve şema vektörleri arasındaki anlamsal yakınlığı ölçmek için Kosinüs Benzerliği (Cosine Similarity) metriği tercih edilmiştir.

Geri Getirme (Retrieval) Stratejisi: Her sorgu için benzerlik skoru en yüksek olan ilk K=5 (Top-K) tablo yapısı bağlama eklenmek üzere seçilmiştir. Bu değer seçilirken; veri setindeki sorguların genellikle en fazla 3-4 tablo birleşimi (JOIN) gerektirdiği dikkate alınmış, böylece modelin bağlam penceresi gereksiz bilgilerle (noise) doldurulmadan maksimum kapsayıcılık hedeflenmiştir. Doğal dildeki kullanıcı sorgusu embedding modeli ile vektörleştirildikten sonra Qdrant üzerinde arama yapılır. Belirlenen eşik değerine ve Top-K kuralına göre getirilen en alakalı tablo şemaları, kullanıcı sorgusu ile birleştirilerek büyük dil modeline (LLM) "prompt" olarak sunulur. Böylece model, sadece ilgili tabloları görerek doğru SQL sorgusunu üretmeye yönlendirilir.

Deneylerde kullanılacak modellerin seçiminde, güncel Büyük Dil Modeli (LLM) ekosistemini temsil eden farklı mimari ve erişim modellerinin performans spektrumunu belirlemek hedeflenmiştir. Bu doğrultuda; kapalı kaynaklı modellerin endüstri standardı ve en yüksek kapasiteli temsilcisi olan GPT-4o, Google'ın karmaşık akıl yürütme ve geniş bağlam penceresi için optimize ettiği üst segment modeli Gemini-1.5-pro ve açık kaynak dünyasında kendi parametre sınıfında (14B) en verimli sonuçları veren Qwen2.5-14B tercih edilmiştir. Bu seçki hem ticari API tabanlı çözümlerin hem de yerel olarak koşturulabilen (on-premise) açık kaynaklı modellerin Türkçe Text-to-SQL görevindeki başarısını objektif bir şekilde karşılaştırma imkânı sunmaktadır.

3.5 GPT-4o (Generative Pretrained Transformer-4o)

GPT-4, OpenAI tarafından geliştirilen, çok büyük parametreli ve çok modlu (metin + görsel) bir dil modelidir. Metin ve görsel girdileri işleyip metin çıktısı üretebilen GPT-4, klasik dil modelleme görevlerinde insan seviyesine yakın başarı sergilemekte ve MMLU, baro sınavı gibi çok çeşitli akademik ve mesleki sınavlarda önceki nesil modelleri belirgin şekilde geride bırakmaktadır. Model mimarisi Transformer tabanlıdır ve ön-eğitim sürecinden sonra RLHF (insan geribildirimiyle güçlendirme) ile hizalanmıştır. GPT-4, çok dilli destek sunmakta ve düşük kaynağa sahip dillerde dahi iyi sonuçlar vermektedir. Bununla birlikte, modelin güvenilirliği tamamen garanti değildir ve hatalı bilgi üretme potansiyeli taşır [4].

GPT-4o, OpenAI'ın 2024'te tanıttığı, GPT-4'e göre çok daha hızlı çalışan ve çoklu modalite (metin, görsel, ses) desteğine sahip en yeni büyük dil modelidir. Düşük gecikme süresi ve gelişmiş dil anlama yetenekleriyle GPT-4'ten ayrışmaktadır. Sejun Oh (2024) tarafından yapılan bir çalışmada, GPT-4o'nun Kore üniversite giriş sınavı (CSAT) matematik sorularında %49,46 doğruluk oranına ulaştığı raporlanmıştır. Model, çoktan seçmeli sorularda güçlü performans gösterirken, açık uçlu ve karmaşık matematiksel sorularda insan seviyesine henüz erişememiştir. Yine de GPT-4o, hız ve pratiklik açısından eğitim uygulamaları için önemli avantajlar sunmaktadır [31].

3.6 Gemini-1.5-pro Modeli (Gemini-1.5-pro Model)

Gemini-1.5-pro, Google'ın geliştirdiği yeni nesil multimodal dil modellerinden biridir. Yüksek verimlilik ve düşük gecikme amacıyla tasarlanan bu model, 2 milyon token'a kadar uzun bağlam işleyebilmekte ve metin, görsel, ses, video gibi çoklu veri türlerinde üstün performans göstermektedir. Gemini-1.5-pro özellikle "needle-in-a-haystack" (samanlıkta iğne) gibi uzun bağlamda bilgi bulma görevlerinde %99'dan fazla doğruluk oranına ulaşmakta; çoklu modaliteler arası geri çağırma ve akıl yürütme görevlerinde benzerlerinden ayrışmaktadır. Ayrıca, hızlı cevap üretimi ve düşük kaynak tüketimi sayesinde büyük ölçekli uygulamalarda etkin biçimde kullanılabilmektedir [32].

3.7 Qwen2.5-14B-Instruct-bnb-4bit Modeli (Qwen2.5-14B-Instruct-bnb-4bit Model)

Qwen2.5-14B-Instruct, Alibaba tarafından geliştirilen, 14 milyar parametreli, açık ağırlıklı ve yönerge tabanlı büyük bir dil modelidir. Model, 18 trilyon token'dan oluşan yüksek kaliteli veriyle ve gelişmiş denetimli ince ayar ile eğitilmiştir; özellikle dil anlama, muhakeme, matematik ve kodlama görevlerinde çarpıcı sonuçlar göstermektedir. Qwen2.5 serisi, uzun bağlam yönetimi (128K token), çok dilli destek ve yapısal veri anlayışı gibi ileri düzey yeteneklerle öne çıkar. 4-bit quantization (bnb-4bit) sayesinde model, düşük kaynak ortamlarında da verimli biçimde çalıştırılabilmekte ve pratik kullanımda yüksek hız ve düşük maliyet sunmaktadır. Gerçekleştirilen değerlendirmelerde, Qwen2.5-14B-Instruct modeli hem genel hem de matematiksel yeteneklerde kendi boyutundaki modeller arasında rekabetçi performans sergilemektedir [33].

4. Modellerin İnce Ayar (Fine-Tuning) Süreçleri
(Fine-Tuning Processes of Models)

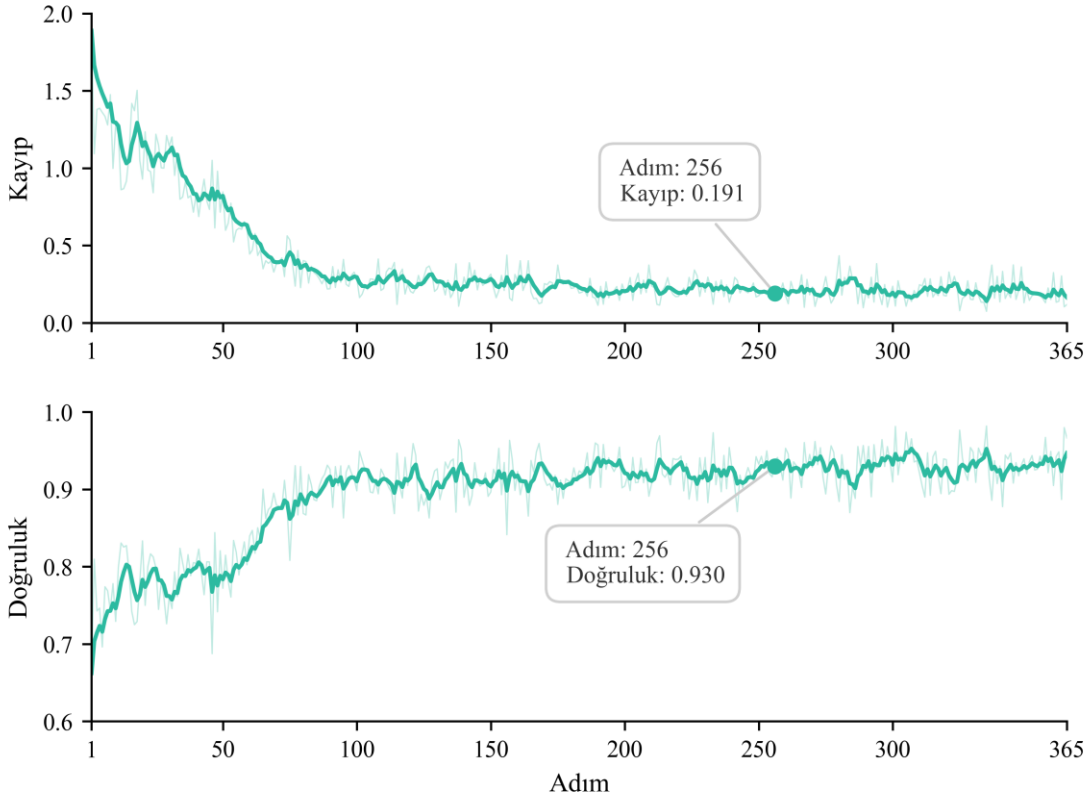
Bu bölümde, GPT-4o, Gemini-1.5-pro ve Qwen2.5-14B-Instruct-bnb-4bit modelleri üzerinde uygulanan fine-tuning prosedürleri ve veri hazırlık adımları detaylı olarak açıklanmıştır.

4.1. GPT-4o Modeli Üzerinde İnce Ayar Süreci (Fine-Tuning Process on GPT-4o Model)

Bu çalışmada, OpenAI tarafından sağlanan bulut tabanlı ince ayar (fine-tuning) arayüzü kullanılarak "gpt-4o-2024-08-06" temel modeli üzerinde özelleştirme yapılmıştır. Çalışmanın deneysel tasarımı kapsamında (Bkz. Bölüm 5), önerilen RAFT yönteminin etkinliğini ve bileşenlerinin (CoT, Distractor Schemas) katkısını ayrı ayrı analiz edebilmek amacıyla üç farklı model varyasyonu eğitilmiştir:

RAFT (CoT) Modeli: Çalışmanın ana önerisi olan modeldir. Eğitim setinde hem doğru şema (gold) hem de çeldirici (distractor) tablolar bulunmakta olup, model adım adım akıl yürütme (CoT) süreciyle eğitilmiştir (Model ID: ft:gpt-4o-2024-08-06:bursa-uludag:raft-to-sql:BIj9eHZW).

Oracle (No-Distractor) Modeli: Çeldirici tabloların etkisini ölçmek amacıyla, eğitim setinde sadece ilgili tabloların verildiği ve çeldiricilerin temizlendiği ideal senaryo modelidir (Model ID: ft:gpt-4o-2024-08-06:bursa-uludag:oracle:Cw5usGGk).

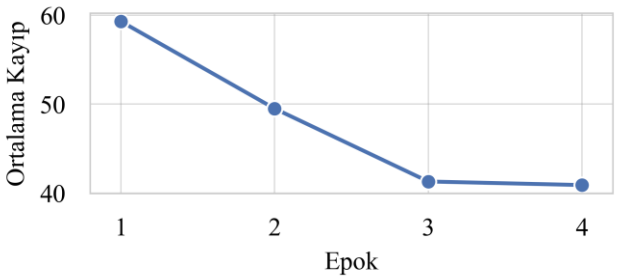


Şekil 4. GPT-4o modeli için fine-tune sürecinde izlenen kayıp (loss) ve doğruluk (accuracy) grafikleri
(Loss and accuracy graphs monitored during the fine-tune process for the GPT-4o model)

Baseline (No-CoT) Modeli: Akıl yürütme (reasoning) etkisini izole etmek amacıyla, eğitim verisinden CoT adımlarının çıkarıldığı ve sadece doğrudan "Soru → SQL" eşleşmesini içeren temel modeldir (Model ID: ft:gpt-4o-2024-08-06:bursa-uludag:no-cot:Cw59T8qz).
Eğitimde kullanılan toplam 880 örnekten oluşan veri seti, OpenAI standartlarına uygun olarak JSONL formatında yapılandırılmıştır. Bölüm 3.2'de detayları verilen veri hazırlama süreci sonucunda elde edilen bu yapı (Bkz. Şekil 1), modelin girdi formatına uygun hale getirilmiştir. Sistem mesajında modelin SQL uzmanı rolü tanımlanmış; kullanıcı mesajında doğal dil sorgusu ile birlikte veritabanı şeması (context) sunulmuş; asistan mesajında ise RAFT ve Oracle modelleri için beklenen akıl yürütme (CoT) süreci ve hedef SQL çıktısı yer alırken, Baseline model için CoT adımları çıkarılarak sadece nihai SQL sorgusu bırakılmıştır. İnce ayar sürecinde, tüm modeller için sonuçların karşılaştırılabilirliğini sağlamak adına eğitim parametreleri sabit tutulmuştur: 1 epoch, 1 batch size ve 1,0 öğrenme oranı çarpanı (learning rate multiplier) kullanılmıştır. Optimizasyon algoritması ve dropout gibi yapısal hiperparametreler için platformun sunduğu standart konfigürasyonlar (default AdamW) korunmuştur. Ana model olan RAFT (CoT) modeli için eğitim süresince kaydedilen kayıp (loss) değerlerinin değişimi Şekil 4'te sunulmuştur. Sürecin başında yaklaşık 1,1 olan kayıp değerinin, 100. adım itibarıyla 0,2 seviyelerine gerilemesi, modelin Türkçeden SQL üretme ve gerekçelendirme görevine başarıyla adapte olduğunu kanıtlamaktadır. Diğer modeller de benzer yakınsama (convergence) davranışları sergilemiştir.

4.2. Gemini-1.5-pro Modeli Üzerinde İnce Ayar Süreci
(Fine-Tuning Process on Gemini-1.5-Pro Model)

Gemini-1.5-pro-tuning modeli üzerindeki ince ayar süreci, Google'ın üretken yapay zeka API'si kullanılarak gerçekleştirilmiştir. Eğitim süreci boyunca modelin performansını izlemek amacıyla her epoch sonunda elde edilen ortalama kayıp (mean loss) değerleri analiz edilmiştir. Eğitim konfigürasyonu; 4 epoch, 4 batch size ve 0,001 öğrenme oranı (learning rate) olarak set edilmiştir. Modelin çıkarım (inference) parametreleri ise top-p: 0,95, top-k: 64 ve temperature: 1,0 değerlerinde sabitlenmiştir. Optimizasyon algoritması ve dropout gibi düşük seviyeli parametreler için platformun sunduğu standart mimari konfigürasyonlar (default) kullanılmıştır. Toplam 880 adımda tamamlanan bu sürecin sonunda, kayıp değerlerinin istikrarlı bir şekilde düştüğü Şekil 5'teki loss eğrisiyle doğrulanmıştır.

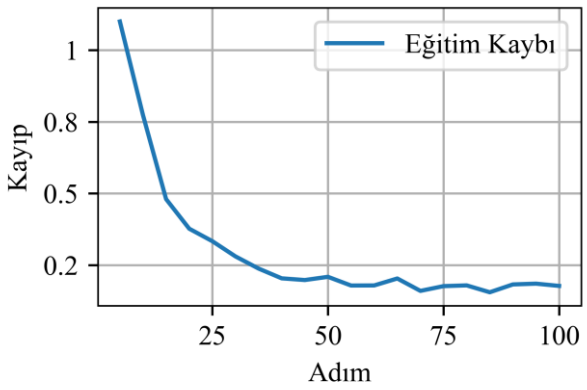


Şekil 5. Gemini-1.5-pro modeli için epoch bazında ortalama kayıp (mean loss) grafiği
(Average loss (mean loss) graph per epoch for the Gemini-1.5-pro model)

4.3. Qwen2.5-14B-Instruct-bnb-4bit Modeli Üzerinde İnce Ayar Süreci
(Fine-Tuning Process on Qwen2.5-14B-Instruct-bnb-4bit Model)

Qwen2.5-14B-Instruct-bnb-4bit modeli üzerinde gerçekleştirilen ince ayar sürecinde, Unsloth kütüphanesinin bellek optimizasyonu

sağlayan altyapısından yararlanılmıştır. Model, 880 örnekten oluşan Türkçe RAFT veri setiyle, LoRA (Low-Rank Adaptation) teknikleri kullanılarak eğitilmiştir. Eğitim konfigürasyonu, donanım verimliliğini artırmak amacıyla 4-bit quantization ve gradient checkpointing tekniklerini içermektedir. LoRA hiperparametreleri; rank (r) 16, alpha 16 ve dropout oranı 0 olarak ayarlanmış; hedef modüller (target modules) tüm lineer katmanları (q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj) kapsayacak şekilde genişletilmiştir. Optimizasyon sürecinde, bellek kullanımını düşüren adamw_8bit algoritması tercih edilmiştir. Diğer eğitim hiperparametreleri ise şu şekildedir: Maksimum adım sayısı (max_steps) 100 (yaklaşık 1 epoch), batch size 2, gradient accumulation 4, learning rate 2e-4, weight decay 0.01 ve warmup steps 5. Öğrenme oranı planlayıcısı (scheduler) olarak lineer yapı kullanılmış ve eğitim kaybının (training loss) istikrarlı düşüşü Şekil 6'da gösterilmiştir.



Şekil 6. Qwen2.5-14B modelinde loss grafiği
(Loss graph for the Qwen2.5-14B model)

5. Sonuçlar ve Tartışmalar (Results and Discussion)

Çalışmada kullanılan 1006 SQL test sorgusu üzerinden, RAFT yöntemiyle ince ayar yapılan üç farklı LLM'in genel başarım düzeyi karşılaştırmalı olarak analiz edilmiştir. Modellerin performans değerlendirmesinde literatürde yaygın olarak kullanılan üç temel metrik esas alınmıştır:

EX (Execution Accuracy): Üretilen SQL sorgusunun veritabanında çalıştırılmasıyla elde edilen sonucun, referans (doğru) SQL sorgusunun sonucuyla birebir eşleşip eşleşmediğini ölçen en kritik doğruluk metriğidir.
Table Similarity (Tablo Benzerliği): Modelin ürettiği sorguda kullandığı tabloların, gerçek sorguda olması gereken tablolarla ne oranda örtüştüğünü ifade eder.
SQL Similarity (SQL Benzerliği): Üretilen sorgunun sözdizimsel (syntactic) yapısının ve komut bileşenlerinin, referans sorguya olan yapısal benzerliğini gösterir.

Tablo 3'te GPT-4o, Gemini-1.5-Pro ve Qwen2.5-14B modellerinin RAFT tabanlı EX Skoru, Tablo Benzerliği ve SQL Benzerliği metriklerine göre performansları sunulmaktadır. Gemini-1.5-pro (0,84) ve GPT-4o (0,83) modelleri birbirine çok yakın ve yüksek düzeyde sorgu doğruluğu sağlamıştır. Önceki analizlerin aksine, Qwen2.5-14B modeli de 0,74'lük EX skoru ile rekabetçi bir performans sergilemiştir. Yapısal benzerlik ölçütleri olan Tablo Benzerliği ve SQL Benzerliği değerleri de GPT-4o ve Gemini-1.5-pro modellerinde oldukça yakın ve yüksek seviyededir (her ikisi için 0,90 üzeri). Yapısal benzerlik ölçütleri incelendiğinde, Qwen2.5-14B modelinin SQL Benzerliği konusunda 0,92 skoruna ulaşarak Gemini-1.5-pro ile aynı, GPT-4o'dan (0,89) ise daha yüksek bir sözdizimsel başarı gösterdiği dikkat çekmektedir. Bu bulgu, açık kaynaklı ve daha düşük parametreli (14B) modellerin de RAFT eğitimi ile ticari modeller kadar güçlü SQL yapıları kurabileceğini göstermektedir.

Tablo 3. RAFT ile ince ayarlanmış LLM'lerin genel başarısı
(Overall Performance of RAFT-Fine-Tuned LLMs)

LLM	EX	Tablo Benzerliği	SQL Benzerliği
GPT-4o	0,83	0,92	0,89
Gemini-1.5-pro	0,84	0,90	0,92
Qwen2.5-14B	0,74	0,85	0,92

Modellerin genel başarım düzeyleri incelendikten sonra, bu performansın veritabanı yapılarındaki karmaşıklık karşısındaki dayanıklılığını ölçmek amacıyla ayrıntılı bir analiz yapılmıştır. Bu doğrultuda, çalışmada 'Schema Complexity' (Şema Karmaşıklığı) kavramı; sorguda kullanılan tablo sayısı, JOIN operasyonlarının yoğunluğu ve SQL bileşenlerinin (GROUP BY, iç içe sorgular vb.) zorluk derecesi esas alınarak yapılandırılmıştır. Bu sınıflandırma yapılırken, Text-to-SQL literatüründe standart kabul edilen Spider benchmark metodolojisi [27], [28] referans alınmıştır. Söz konusu kategorizasyonun temel bilimsel gerekçesi, modellerin sadece basit seçim sorgularında değil, ilişkisel veritabanı derinliği arttıkça 'şema eşleştirme' (schema linking) ve mantıksal çıkarım becerilerini ne ölçüde koruyabildiğini ampirik olarak kanıtlamaktır. Böylece, "Modeller karmaşık ilişkisel yapılarda ne kadar etkilidir?" araştırma sorusuna yanıt aranmıştır.

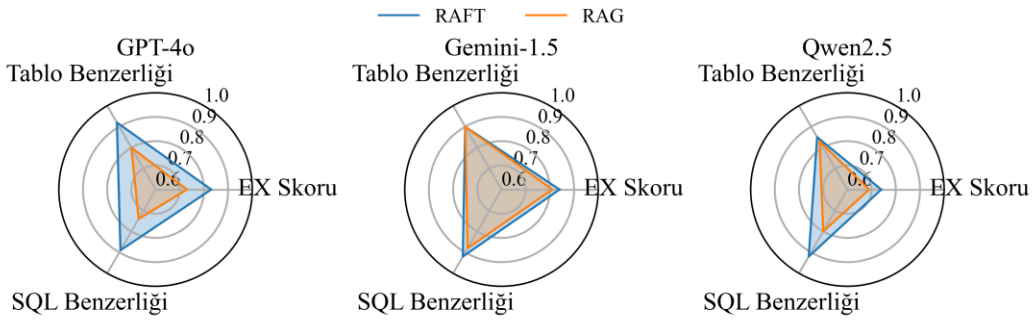
Tablo 4'te ise, RAG yaklaşımı kullanılarak yapılan deneysel analizlerin sonuçları sunulmuştur. Bu aşamada, doğal dildeki sorguların vektör temsili için OpenAI'ın "text-embedding-3-large" embedding modeli kullanılmış ve en uygun şemaların seçimi Qdrant vektör veritabanı aracılığıyla gerçekleştirilmiştir. Ardından, seçilen şemalar büyük dil modellerine sağlanarak SQL sorgusu üretimi yapılmıştır. Sonuçlar incelendiğinde, Gemini-1.5-pro modeli yine en yüksek EX Skoru (0,81), Tablo Benzerliği (0,90) ve SQL Benzerliği (0,88) değerlerine ulaşmıştır. GPT-4o ve Qwen2.5-14B modellerinin başarıları ise sırasıyla EX Skor için 0,73 ve 0,69, Tablo Benzerliği için 0,80 ve 0,83, SQL Benzerliği için ise 0,74 ve 0,80 olarak ölçülmüştür.

Elde edilen bulgular, embedding ve vektör tabanlı retrieval mimarisi kullanıldığında da Gemini-1.5-pro modelinin, diğer modellere kıyasla daha yüksek doğruluk ve yapısal tutarlılık sağladığını ortaya koymaktadır. Karşılaştırmalı analizde, RAFT yaklaşımının özellikle GPT-4o (0,73'ten 0,83'e) ve Qwen2.5-14B (0,69'dan 0,74'e) modellerinde anlamlı bir performans artışı sağladığı, Gemini-1.5-pro modelinde ise bu artışın daha sınırlı (0,81'den 0,84'e) olduğu gözlemlenmiştir. Bu durum, Gemini modelinin bağlam penceresini (context window) yönetme konusundaki yerleşik başarısının yüksek olduğunu; ancak GPT-4o ve Qwen modellerinin, görev odaklı ince ayar (RAFT) yapıldığında potansiyellerini çok daha verimli bir şekilde ortaya koyabildiğini göstermektedir.

Tablo 4. RAG ile LLM'lerin genel başarısı
(Overall performance of RAG-enhanced LLMs)

LLM	EX	Tablo Benzerliği	SQL Benzerliği
GPT-4o	0,73	0,80	0,74
Gemini-1.5-pro	0,81	0,90	0,88
Qwen2.5-14B	0,69	0,83	0,80

Şekil 7'de ise GPT-4o, Gemini-1.5 ve Qwen2.5-14B modellerinin RAFT ve RAG yöntemlerine göre üç temel ölçüte dayalı (EX Skor, Tablo Benzerliği, SQL Benzerliği) genel karşılaştırması radar grafik aracılığıyla sunulmuştur. Grafikler, her bir modelin yapısal benzerlik



Şekil 7. RAFT ve RAG yaklaşımlarının radar bazlı değerlendirmesi (Radar-based evaluation of RAFT and RAG approaches)

Tablo 5. Tablo sayılarına göre veritabanlarındaki sorguların başarımı (GPT-4o)
(Query performance in databases based on table count (GPT-4o))

		1 Tablo		2 Tablo		3 Tablo		4 Tablo	
Veritabanları	Toplam Sorgu	Sayı	EX	Sayı	EX	Sayı	EX	Sayı	EX
coffee_shop	18	16	1	2	1				
election	68	38	0,95	30	0,87				
chinook	84	32	0,97	50	0,72	2	1		
architecture	17	8	1	7	0,86	2	1		
company employee	16	11	0,91	1	1	4	0,75		
company office	40	32	0,94	2	1	6	0,83		
voter	15	9	0,89	5	1	1	1		
customer complaints	46	26	0,85	16	0,75	4	0,5		
movie	98	38	0,87	48	0,52	12	0,83		
social media	76	22	0,45	48	0,6	6	0,33		
sakila	82	40	0,9	36	0,92	6	0,67		
soccer	106	68	0,88	34	0,76	4	1		
cars	82			54	0,61	21	0,67	7	0,57
department store	88	46	0,8	28	0,81	12	1	2	1
college	170	104	0,92	44	0,82	20	0,85	2	1
Genel Toplam/ Ağırlıklı Ortalama	1006	490	0,88	405	0,73	100	0,78	11	0,73

ve doğruluk açısından hangi yöntemde daha yüksek başarı sergilediğini açık biçimde ortaya koymaktadır. GPT-4o modeli, her üç ölçütte de RAFT yönteminde RAG'a kıyasla bariz bir üstünlük göstermektedir. Gemini-1.5 modeli için ise RAFT ve RAG performansları görece daha dengelidir. Qwen2.5-14B modeli için durum incelendiğinde; özellikle SQL Benzerliği metriğinde modelin RAFT ile 0,92 seviyelerine ulaşırken, RAG yaklaşımında 0,80 seviyesinde kaldığı görülmektedir. Bu veriler, RAFT yönteminin sadece doğru sonucu bulmayı değil, aynı zamanda SQL dilinin sözdizimsel kurallarına (syntax) uyumu da RAG yöntemine göre daha güçlü bir şekilde garanti altına aldığını kanıtlamaktadır

Modellerin tablo sayılarına göre başarım karşılaştırmaları Tablo 5 (GPT-4o), Tablo 6 (Gemini-1.5-pro) ve Tablo 7 (Qwen2.5-14B) üzerinde sunulmuştur. GPT-4o, tüm tablo senaryolarında en dengeli ve kararlı performansı sergilemiştir; tek tablo içeren sorgularda 0,88, iki tablolu sorgularda 0,73, üç tablolu sorgularda 0,78 ve dört tablolu sorgularda 0,73 skorlarına ulaşmıştır. Bu durum, GPT-4o'nun çok tablolu ve karmaşık yapıdaki sorgularda dahi performans kaybı yaşamadan güçlü genel başarımını koruduğunu kanıtlamaktadır.

Gemini-1.5-pro modeli ise özellikle tek tablo içeren sorgularda en yüksek EX skoru olan 0,90'ı elde ederek öne çıkmıştır. Ancak tablo sayısı arttıkça modelin başarımında bir miktar düşüş gözlenmiş ve dört tablolu sorgularda EX skoru 0,64'e gerilemiştir. Bu bulgu, Gemini modelinin basit sorgularda çok etkili olmasına rağmen, ilişki sayısı ve karmaşıklık arttıkça GPT-4o'ya kıyasla daha fazla zorlandığını göstermektedir.

Qwen2.5-14B modeli ise, başlangıçta 0,82 olan tek tablo başarımıyla güçlü bir giriş yapmış, orta karmaşıklıktaki (2 ve 3 tablolu) sorgularda sırasıyla 0,56 ve 0,58 skorlarına düşmesine rağmen, en karmaşık senaryo olan dört tablolu sorgularda tekrar yükselişe geçerek 0,64 skoruna ulaşmıştır. Dört tablolu sorgularda Gemini-1.5-pro ile aynı başarıyı (0,64) yakalaması, modelin parametre boyutunun küçük olmasına rağmen RAFT yöntemiyle karmaşık mantıksal yapıları öğrenme kapasitesinin yüksek olduğunu ortaya koymaktadır.

Şekil 8'de, üç modelin tablo sayılarına göre EX skorlarının eğilimleri karşılaştırmalı olarak sunulmuştur. Grafik, GPT-4o'nun genel tutarlılığını ve karmaşık yapılara karşı dayanıklılığını açıkça ortaya koyarken; Gemini-1.5-pro'in daha basit tablo yapılarına karşı güçlü ama çok tablolu senaryolarda daha kırılgan olduğunu; Qwen2.5-14B'nin ise karmaşıklık arttıkça hızlı performans düşüşü yaşadığını gözler önüne sermektedir.

Genel olarak, bu sonuçlar; kullanılan verinin kalitesi, ince ayar yöntemi (RAFT), mimari yeterlilik ve eğitimin kapsamı gibi

Tablo 6. Tablo sayılarına göre veritabanlarındaki sorguların başarımı (Gemini-1.5-pro)
(Query performance in databases based on table count (Gemini-1.5-pro))

		1 Tablo		2 Tablo		3 Tablo		4 Tablo	
Veritabanları	Toplam Sorgu	Sayı	EX	Sayı	EX	Sayı	EX	Sayı	EX
coffee_shop	18	16	1	2	1				
election	68	38	1	30	0,77				
chinook	84	32	1	50	0,74	2	1		
architecture	17	8	1	7	0,86	2	1		
company employee	16	11	1	1	1	4	0,5		
company office	40	32	0,97	2	1	6	1		
voter	15	9	0,89	5	0,8	1	0		
customer complaints	46	26	0,96	16	0,75	4	0,5		
movie	98	38	0,76	48	0,52	12	0,83		
social media	76	22	0,73	48	0,73	6	0,17		
sakila	82	40	0,95	36	0,92	6	0,83		
soccer	106	68	0,87	34	0,68	4	1		
cars	82			54	0,72	21	0,76	7	0,57
department store	88	46	0,76	28	0,84	12	0,88	2	1
college	170	104	0,9	44	0,68	20	0,65	2	0,5
Genel Toplam/ Ağırlıklı Ortalama	1006	490	0,9	405	0,73	100	0,73	11	0,64

Tablo 7. Tablo sayılarına göre veritabanlarındaki sorguların başarımı (Qwen2.5-14B)
(Query performance in databases based on table count (Qwen2.5-14B))

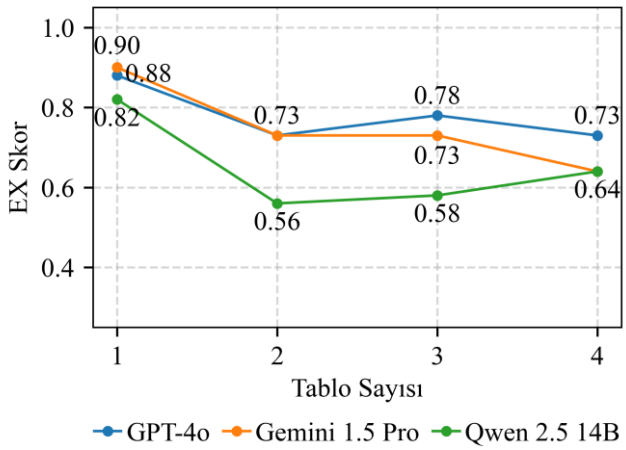
		1 Tablo		2 Tablo		3 Tablo		4 Tablo	
Veritabanları	Toplam Sorgu	Sayı	EX	Sayı	EX	Sayı	EX	Sayı	EX
coffee_shop	18	16	0,94	2	1				
election	68	38	0,76	30	0,17				
chinook	84	32	0,97	50	0,5	2	1		
architecture	17	8	1	7	0,71	2	1		
company employee	16	11	1	1	1	4	0,75		
company office	40	32	0,78	2	1	6	0,83		
voter	15	9	0,89	5	0,80	1	1		
customer complaints	46	26	0,77	16	0,69	4	0.25		
movie	98	38	0,76	48	0,5	12	0,33		
social media	76	22	0,77	48	0,69	6	0,50		
sakila	82	40	0,93	36	0,78	6	0.50		
soccer	106	68	0,74	34	0,50	4	0,50		
cars	82			54	0,69	21	0,71	7	0,71
department store	88	46	0,76	28	0,46	12	0,67	2	1
college	170	104	0,83	44	0,48	20	0,45	2	0
Genel Toplam/ Ağırlıklı Ortalama	1006	490	0,82	405	0,56	100	0,58	11	0,64

faktörlerin yalnızca doğru SQL üretimini değil, aynı zamanda karmaşık veri senaryolarına karşı model dayanıklılığını da doğrudan etkilediğini göstermektedir. Sonuç olarak, sadece yüksek model kapasitesi değil, ince ayar stratejisinin niteliği de başarımda belirleyici rol oynamaktadır

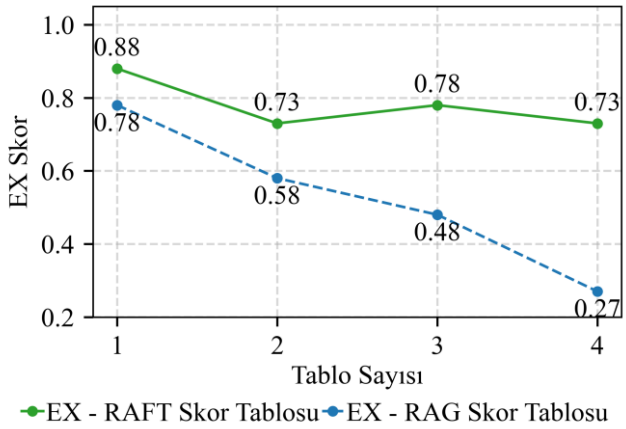
GPT-4o modeli referans alınarak, RAG ve RAFT yaklaşımlarında tablo sayısının artmasının model başarımı üzerindeki etkisi analiz edilmiştir. Özellikle her iki yöntemin EX Skor (Exact Match), Tablo Benzerliği ve SQL Benzerliği gibi performans metrikleri üzerindeki etkileri ayrıntılı olarak değerlendirilmiştir.

Şekil 9'da yalnızca GPT-4o modeli için, farklı tablo sayıları altında elde edilen EX skorlarının RAFT ve RAG yöntemleriyle nasıl değiştiği çizgisel olarak sunulmuştur. Grafik incelendiğinde, GPT-4o modelinin RAFT yönteminde tüm tablo senaryolarında RAG'a kıyasla daha yüksek EX skorları elde ettiği gözlemlenmektedir. Örneğin, tek tablo içeren sorgularda RAFT ile elde edilen skor 0,88 iken RAG yöntemiyle bu değer 0,78'de kalmıştır. Özellikle tablo sayısı arttıkça RAG yönteminin başarımı ciddi biçimde düşerken (örneğin dört tablolu sorgularda 0,27), RAFT yöntemi bu düşüşü daha dengeli biçimde yönetebilmiş ve dört tablolu sorgularda dahi 0,73'lük bir skor elde edilmiştir.

Bu sonuç, RAG yönteminin karmaşıklık arttıkça "gürültü" (noise) yönetiminde başarısız olduğunu; buna karşın RAFT yönteminin modelin dikkatini (attention) doğru tablolara odaklayarak karmaşık sorgularda dahi sürdürülebilir bir başarı sağladığını kanıtlamaktadır.



Şekil 8. RAFT yöntemiyle fine-tune edilen modellerin,tablo sayılarına göre EX skorları bakımından karşılaştırılması
(Comparison of models fine-tuned with the RAFT method in terms of EX scores according to the number of tables)



Şekil 9. GPT-4o modelinin, farklı tablo sayılarında RAFT ve RAG yöntemlerine göre EX skorlarının karşılaştırılması
(Comparison of the EX-scores of the GPT-4o model according to RAFT and RAG methods at different numbers of tables)

5.1. İstatistiksel Doğrulama ve Ablasyon Analizi
(Statistical Verification and Ablation Study)

Modellerin performans farklarının istatistiksel olarak anlamlı olup olmadığını belirlemek ve sonuçların güvenilirliğini doğrulamak amacıyla Bootstrap yöntemi kullanılmıştır. Test seti üzerinde %95 güven aralığı (Confidence Interval - CI) hesaplanarak, modellerin RAFT ve RAG yöntemleri altındaki kararlılıkları analiz edilmiştir. Tablo 8'de, her bir model ve yöntem için elde edilen başarı oranları ve hesaplanan güven aralıkları sunulmuştur.

Elde edilen sonuçlar incelendiğinde; GPT-4o modelinin RAFT yöntemiyle elde ettiği başarı (%83,35) ile RAG yöntemi (%72,62) arasında belirgin bir performans farkı olduğu görülmektedir. Güven aralıkları incelendiğinde, RAFT yönteminin alt sınırının (%77,04) RAG yönteminin üst sınırına (%77,67) oldukça yakın olduğu, ancak ortalama başarı puanındaki yaklaşık %11'lik farkın RAFT yönteminin üstünlüğünü net bir şekilde ortaya koyduğu görülmektedir.

Gemini-1.5-pro modelinde ise RAFT (%83,97) ve RAG (%80,81) skorları birbirine daha yakındır. Güven aralıklarının ([%79,00, %88,88] ve [%76,75, %84,87]) belirgin şekilde örtüşmesi, Gemini modelinin her iki yöntemde de kararlı ve birbirine yakın bir performans sergilediğini, ancak RAFT yönteminin ortalamada daha yüksek bir başarı potansiyeli sunduğunu göstermektedir.

Tablo 8. Modellerin RAFT ve RAG yöntemlerine göre istatistiksel doğrulaması (%95 CI)
(Statistical verification of models by RAFT and RAG methods (95% CI))

Model	Yöntem	EX	%95 Güven Aralığı (CI)
Gemini- 1.5-pro	RAFT	83,97	[0.7900, 0.8888]
	RAG	80,81	[0.7675, 0.8487]
GPT-4o	RAFT	83,35	[0.7704, 0.8880]
	RAG	72,62	[0.6743, 0.7767]
Qwen2.5-14B	RAFT	74,09	[0.6766, 0.8103]
	RAG	68,99	[0.6285, 0.7514]

Qwen2.5-14B modelinde ise RAFT (%74,09) ile RAG (%68,99) yöntemleri arasında yaklaşık %5'lik bir başarı farkı bulunmasına rağmen, geniş güven aralıkları ([%67,66, %81,03] ve [%62,85, %75,14]) nedeniyle istatistiksel olarak kesin bir ayrım yapmak güçtür. Ancak ortalama değerlerdeki artış, RAFT yönteminin Qwen modeli üzerinde de pozitif bir etkisi olduğunu doğrulamaktadır.

Modellerin başarısız olduğu durumları daha derinlemesine incelemek ve RAFT yönteminin hangi problem türlerini çözmede etkili olduğunu belirlemek amacıyla nicel bir hata analizi gerçekleştirilmiştir. Hatalar; Yanlış/Eksik JOIN, Eksik WHERE Koşulu, Halüsinasyon, Yanlış Kolon Seçimi ve Yanlış Tablo Seçimi olmak üzere kategorize edilmiştir. Tablo 9, modellerin RAFT ve RAG yöntemlerinde ürettikleri hatalı sorgu sayılarının dağılımını göstermektedir.

Analiz sonuçları şu kritik bulguları ortaya koymaktadır:

Yanlış Tablo Seçimi: Tüm modellerde RAG yönteminde en sık karşılaşılan hatalardan biri yanlış tablo seçimidir. RAFT yöntemi, bu hatayı tüm modellerde belirgin şekilde azaltmıştır. Özellikle Qwen2.5-14B modelinde yanlış tablo seçimi hatası 199'dan 106'ya, GPT-4o modelinde 82'den 61'e ve Gemini modelinde 82'den 70'e düşmüştür. Bu bulgu, RAFT sürecindeki "distractor" (çeldirici) tablolarla yapılan eğitimin, modelin şema eşleştirme (schema linking) yeteneğini doğrudan geliştirdiğini doğrulamaktadır.

Yanlış/Eksik JOIN: RAFT eğitimi, modellerin tablolar arası ilişki kurma (JOIN) mantığını da önemli ölçüde iyileştirmiştir. Yanlış veya eksik JOIN hataları; Gemini modelinde 182'den 75'e, Qwen modelinde 159'dan 109'a ve GPT-4o modelinde 94'ten 65'e gerilemiştir. Bu durum, CoT (Chain-of-Thought) desteğinin modelin sorgu mantığını kurarken tablolar arası ilişkileri daha doğru kurgulamasını sağladığını göstermektedir.

Filtreleme Hassasiyeti (WHERE Koşulları): Veriler, RAFT yönteminin modelleri belirli koşullarda daha "temkinli" (conservative) davranmaya ittiğini göstermektedir. Özellikle GPT-4o (5'ten 23'e) ve Gemini (14'ten 27'ye) modellerinde, RAFT sonrası "Eksik WHERE Koşulu" hatalarında RAG yöntemine kıyasla bir artış gözlemlenmiştir. Modeller, şema ve JOIN seçiminde kazandıkları yüksek hassasiyeti, filtreleme koşullarını oluştururken aşırı seçici davranarak dengelemiş görünmektedir. Ancak Qwen modeli bu eğilimin aksine, Eksik WHERE hatalarını da RAG'a kıyasla (72'den 22'ye) azaltmayı başarmıştır.

Halüsinasyon ve Kolon Seçimi: Halüsinasyon oranları genel olarak düşüktür, ancak GPT-4o modelinin RAG yönteminde 135 olan halüsinasyon sayısı RAFT ile 81'e düşerek modelin veriye sadakati artmıştır. Yanlış kolon seçimi hatalarında ise RAFT yöntemi Qwen

Tablo 9. Modellerin yöntemlere göre hata türleri dağılımı (Distribution of error types by models and methods)

Hata Türü	Gemini (RAFT)	Gemini (RAG)	Gpt (RAFT)	Gpt (RAG)	Qwen (RAFT)	Qwen (RAG)
Yanlış/Eksik JOIN	75	182	65	94	109	159
Eksik WHERE	27	14	23	5	22	72
Halüsinasyon	65	21	81	135	127	107
Yanlış Kolon Seçimi	64	42	48	94	103	68
Yanlış Tablo Seçimi	70	82	61	82	106	199
Toplam Hata	301	341	278	410	467	605

Tablo 10. GPT-4o modeli üzerinde CoT etkisinin karmaşıklığa göre analizi (Ablation Study)
(Comprehensive analysis of CoT impact on GPT-4o model based on complexity)

Karmaşıklık (Complexity)	Sorgu Sayısı	No-CoT (Baseline)	RAFT (CoT)	Oracle (Upper Bound)	CoT Etkisi (Gain)	Oracle Farkı (Gap)
Kolay (1 Tablo)	490	%87,35	%88,7	%90,61	%+1,02	%-2,24
Orta (2 Tablo)	405	%67,48	%72,86	%78,24	%+5,38	%-5,38
Zor (3 Tablo)	100	%63,54	%77,08	%79,17	%+13,54	%-2,08
Çok Zor (4 Tablo)	11	%36,36	%72,73	%72,73	%+36,37	%0,00
Genel (Overall)	1006	%76,43	%80,83	%84,30	%+4,41	%-3,47

(68'den 103'e) ve Gemini (42'den 64'e) modellerinde bir miktar artışa neden olurken, GPT-4o modelinde (94'ten 48'e) iyileşme sağlamıştır. Genel bir değerlendirme yapıldığında, Tablo 9 verileri RAFT yaklaşımının toplam hata yükünü tüm modellerde RAG yöntemine kıyasla belirgin şekilde azalttığını kanıtlamaktadır. Toplam hata sayılarındaki düşüş oranları (GPT-4o için ~%32, Qwen2.5 için ~%23 ve Gemini için ~%12), RAFT'ın modelin kararlılığını genel ölçekte artırdığını göstermektedir. Her ne kadar "Eksik WHERE" gibi spesifik alt kırılımlarda yöntemsel karakteristiğe bağlı kısmi artışlar gözlemlense de; doğru SQL üretiminin temelini oluşturan şema seçimi ve tablo ilişkilendirme (JOIN) gibi en kritik aşamalarda RAFT yöntemi, RAG'a göre tartışmasız bir üstünlük kurmuş ve daha güvenilir bir SQL üretim süreci sağlamıştır.

RAFT yönteminin başarısında, adım adım akıl yürütme (Chain-of-Thought, CoT) bileşeninin katkısını izole etmek amacıyla bir "ablation study" gerçekleştirilmiştir. Bu analizde, GPT-4o modelinin CoT olmadan (Baseline) ve CoT ile (RAFT) ürettiği sonuçlar, sorgu karmaşıklık seviyelerine (tablo sayısına) göre karşılaştırılmıştır. Ayrıca, modelin ulaşabileceği teorik üst sınır olan "Oracle" (doğru şemanın elle verildiği durum) performansı da referans olarak sunulmuştur. Elde edilen sonuçlar Tablo 10'da verilmiştir

Tablo 10 incelendiğinde, CoT mekanizmasının etkisinin sorgu karmaşıklığı ile doğru orantılı olarak arttığı görülmektedir. Tek tablolu basit sorgularda CoT kullanımı başarıyı sadece %1,02 artırırken, 3 tablolu sorgularda bu artış %13,54'e, 4 tablolu "Çok Zor" kategorisinde ise %36,37 gibi çarpıcı bir seviyeye ulaşmıştır. No-CoT modeli, 4 tablolu sorgularda %36,36 başarıda kalarak yetersiz kalırken; RAFT (CoT) modeli, adım adım akıl yürütme sayesinde bu oranı %72,73'e çıkararak Oracle (teorik üst sınır) skorunu yakalamıştır. Bu sonuç, karmaşık sorgularda başarının anahtarının sadece "doğru şemayı bulmak" değil, aynı zamanda "nasıl birleştirileceğini adım adım planlamak" olduğunu kanıtlamaktadır.
Diğer yandan, çeldirici (distractor) tabloların etkisinin analiz edildiği Oracle (Upper Bound) sütunu incelendiğinde, önerilen RAFT yönteminin ideal senaryoya oldukça yakın bir performans sergilediği görülmektedir. Genel ortalamada Oracle ile RAFT arasındaki fark sadece %3,47 seviyesinde kalmıştır. Daha da çarpıcı olan bulgu ise; modelin en çok zorlanması beklenen 4 tablolu "Çok Zor" kategorisinde, RAFT modelinin çeldirici tablolara rağmen Oracle skoruyla birebir aynı başarıyı (%72,73) yakalamış olmasıdır (Fark: %0,00). Bu durum, RAFT eğitim stratejisinin, modelin gürültülü (noisy) veri içerisinden doğru şemayı ayırt etme yeteneğini üst seviyeye taşıdığını ve en karmaşık senaryolarda dahi "ideal" çalışma ortamına eşdeğer sonuçlar ürettiğini kanıtlamaktadır. Sonuç olarak; basit sorgularda doğrudan üretim yeterli olabilirken, ilişkisel karmaşıklık arttığında CoT stratejisinin bir tercih değil, bir zorunluluk olduğu doğrulanmıştır.

6. Sonuçlar (Conclusions)

Bu çalışmada, Türkçe doğal dilde ifade edilen sorgulardan SQL sorgusu üretilmesi görevi, Retrieval-Augmented Generation (RAG) ve Retrieval-Augmented Fine-Tuning (RAFT) yaklaşımları ile kapsamlı biçimde incelenmiştir. Deneysel süreçte, RAG aşamasında sorguların vektör temsili için yalnızca OpenAI'ın "text-embedding-3-large" embedding modeli ve Qdrant vektör veritabanı birlikte kullanılmış; RAFT aşamasında ise GPT-4o, Gemini-1.5-pro ve Qwen2.5-14B-Instruct-bnb-4bit modelleri üzerinde ince ayar (fine-tuning) uygulanmıştır.

Her iki yöntemde de, 1006 test sorgusu üzerinden EX Skoru, Tablo Benzerliği ve SQL Benzerliği gibi metriklerle performans değerlendirmesi yapılmıştır. Sonuçlar, Gemini-1.5-pro ve GPT-4o modellerinin RAFT yöntemiyle %83-84 bandında yüksek ve dengeli bir performans sergilediğini göstermiştir. Öte yandan, açık kaynaklı Qwen2.5-14B modeli de RAFT yöntemiyle eğitildiğinde %74 genel başarıya ulaşmış ve özellikle 4 tablolu karmaşık sorgularda %64 başarı oranıyla Gemini modeliyle eşdeğer bir performans sergileyerek rekabetçi bir alternatif olduğunu kanıtlamıştır.

Ayrıca, yapılan karşılaştırmalı analizde, RAFT yaklaşımının RAG yöntemine kıyasla tüm modellerde daha yüksek doğruluk sağladığı tespit edilmiştir. Özellikle ablasyon (ablation) çalışmaları, karmaşık sorgularda "adım adım akıl yürütme" (CoT) stratejisinin başarıyı %36 oranında artırdığını ve RAFT modelinin gürültülü veriye rağmen "Oracle" (ideal) performansını yakaladığını kanıtlamıştır. Bu

bulgular, LLM tabanlı doğal dilden SQL sorgusu üretiminde yalnızca modelin kapasitesinin değil, kullanılan ince ayar stratejisinin ve akıl yürütme yöntemlerinin de başarım üzerinde belirleyici bir rol oynadığını göstermektedir.

Bununla birlikte, çalışmanın sonuçları yorumlanırken dikkate alınması gereken bazı sınırlılıklar mevcuttur. İlk olarak, hata analizleri, RAFT yönteminin Gemini ve GPT-4o gibi modelleri "WHERE" koşullarını oluştururken daha temkinli (conservative) davranmaya ittiğini ve bu durumun, şema seçimindeki yüksek doğruluğa rağmen bazı filtreleme eksikliklerine yol açabildiğini göstermiştir.

İkinci olarak, çalışma kapsamında kullanılan Spider ve BIRD tabanlı veri setleri, akademik standartlara uygun olarak düzenlenmiş ve optimize edilmiş şema yapılarına sahiptir. Gerçek dünya senaryolarında sıklıkla karşılaşılan belirsiz sütun isimlendirmeleri, eksik veri tipleri veya normalizasyon hataları içeren "kirli" veritabanları üzerindeki performans, bu çalışmanın kapsamı dışında bırakılmıştır.

Üçüncü önemli kısıt, dil uyumsuzluğudur. Doğal dil sorgularının Türkçe, veritabanı şemalarının (tablo ve sütun adları) ise İngilizce olması, modellerin sadece SQL sözdizimini değil, aynı zamanda diller arası anlamsal eşleştirmeyi (cross-lingual mapping) de gerçekleştirmesini zorunlu kılmıştır. Bu durum, tamamen Türkçe veya tamamen İngilizce olan tek dilli senaryolara kıyasla model performansını baskılayan bir faktördür.

Ayrıca, RAFT yönteminde uygulanan "Chain-of-Thought" (CoT) stratejisi doğruluk oranını artırmakla birlikte, modelin ürettiği çıktı uzunluğunu (token sayısı) artırmaktadır. Bu çalışmada öncelik doğruluk (accuracy) metriklerine verildiğinden, artan token sayısının yanıt süresine (latency) ve operasyonel maliyete etkisi deneysel olarak ölçülmemiştir. Ticari modellerin yüksek eğitim maliyetleri ve açık kaynak model (Qwen2.5) deneylerinde donanım kısıtları nedeniyle uygulanan 4-bit niceleme (quantization) işlemi de modelin tam potansiyelinin gözlemlenmesini sınırlayan diğer teknik bariyerlerdir. Gelecek çalışmalarda, RAG yaklaşımının performansını artırmak amacıyla Türkçe dil yapısına daha uygun, çok dilli (multilingual) embedding modellerinin ve farklı vektör veritabanı indeksleme stratejilerinin test edilmesi önerilmektedir.

RAFT yönteminin etkinliğini artırmak adına ise iki önemli yol haritası öngörülmektedir. Birincisi, modelin "temkinli davranışını" (over-conservativeness) dengelemek ve filtreleme hatalarını azaltmak için DPO (Direct Preference Optimization) veya RLHF (Reinforcement Learning from Human Feedback) gibi ileri hizalama tekniklerinin uygulanmasıdır. İkincisi, Türkçe sorgular ile İngilizce veritabanı şemaları arasındaki anlamsal boşluğu (semantic gap) daraltmak için şema çevirisi yapan ara katmanların veya terim sözlükleriyle zenginleştirilmiş hibrit istem (prompt) yapılarının geliştirilmesidir.

Ayrıca, bu çalışmada kullanılan CoT yaklaşımının getirdiği çıkarım maliyetini (inference cost) düşürmek amacıyla, büyük modellerden (Teacher) elde edilen akıl yürütme yeteneklerinin daha küçük ve hızlı modellere (Student) aktarıldığı Bilgi Damıtma (Knowledge Distillation) yöntemleri üzerine odaklanılması, sistemin gerçek dünya senaryolarında kullanılabilirliğini artıracaktır.

Teşekkür (Acknowledgement)

Bu çalışma, Bursa Uludağ Üniversitesi Bilimsel Araştırma Projeleri Birimi (BAP) tarafından FGA-2026-2534 hibesi kapsamında desteklenmiştir.

Kaynaklar (References)

1. Minaee S., Mikolov T., Nikzad N., Chenaghlu M., Socher R., Amatriain X., Gao J., Large language models: A survey, arXiv preprint arXiv:2402.06196, 2024.
2. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I., Attention is all you need, Advances in Neural Information Processing Systems, 30, 5999-6009, 2017.
3. Naveed H., Khan A.U., Qiu S., Saqib M., Anwar S., Usman M., Akhtar N., Barnes N., Mian A., A comprehensive overview of large language models, arXiv preprint arXiv:2307.06435, 2023.
4. OpenAI, Achiam J., Adler S., Agarwal S., Ahmad L. vd., GPT-4 technical report, arXiv preprint arXiv:2303.08774, 2023.
5. Güven Z.A., Large-scale impact analysis on large language models for Turkish question-answering, Journal of the Faculty of Engineering and Architecture of Gazi University, 40 (3), 1787-1796, 2025.
6. Gao Y., Xiong Y., Gao X., Jia K., Pan J., Bi Y., Dai Y., Sun J., Wang M., Wang H., Retrieval-augmented generation for large language models: A survey, arXiv preprint arXiv:2312.10997, 2023.
7. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W., Rocktäschel T., Riedel S., Kiela D., Retrieval-augmented generation for knowledge-intensive NLP tasks, Advances in Neural Information Processing Systems, 33, 9459-9474, 2020.
8. Wang M., Shafran I., Soltau H., Han W., Cao Y., Yu D., El Shafey L., Retrieval augmented end-to-end spoken dialog models, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Güney Kore, 12056-12060, 14-19 Nisan, 2024.
9. Öztürk S., Atmaca H.E., İlişkisel ve ilişkisel olmayan (NoSQL) veri tabanı sistemleri mimari performansının yönetim bilişim sistemleri kapsamında incelenmesi, Bilişim Teknol. Derg., 10 (2), 199-212, 2017.
10. Chamberlin D.D., Boyce R.F., SEQUEL: A structured English query language, Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control, Ann Arbor, Michigan, ABD, 249-264, 1974.
11. Darwen H., SQL: A Comparative Survey, Bookboon.com, 2014.
12. Li Y., Manoharan S., A performance comparison of SQL and NoSQL databases, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Kanada, 15-19, 2013.
13. Han Y., Liu C., Wang P., A comprehensive survey on vector database: Storage and retrieval technique, challenge, arXiv preprint arXiv:2310.11703, 2023.
14. Dong X., Zhang C., Ge Y., Mao Y., Gao Y., Chen L., Lin J., Lou D., C3: Zero-shot text-to-SQL with ChatGPT, arXiv preprint arXiv:2307.07306, 2023.
15. Li Z., Wang X., Zhao J., Yang S., Du G., Hu X., Zhang B., Ye Y., Li Z., Zhao R., Mao H., PET-SQL: A prompt-enhanced two-round refinement of text-to-SQL with cross-consistency, arXiv preprint arXiv:2403.09732, 2024.
16. Pourreza M., Rafiei D., DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction, Advances in Neural Information Processing Systems, 36, 2023.
17. Mai C., Tal R., Mohamed T., Learning metadata-agnostic representations for text-to-SQL in-context example selection, arXiv preprint arXiv:2410.14049, 2024.
18. Li H., Zhang J., Li C., Chen H., RESDSQL: Decoupling schema linking and skeleton parsing for text-to-SQL, Proceedings of the AAAI Conference on Artificial Intelligence, 37 (11), 13067-13075, 2023.
19. Gao D., Wang H., Li Y., Sun X., Qian Y., Ding B., Zhou J., Text-to-SQL empowered by large language models: A benchmark evaluation, arXiv preprint arXiv:2308.15363, 2023.
20. Gao Y., Liu Y., Li X., Shi X., Zhu Y., Wang Y., Li S., Li W., Hong Y., Luo Z., Gao J., Mou L., Li Y., A preview of XiYan-SQL: A multi-generator ensemble framework for text-to-SQL, arXiv preprint arXiv:2411.08599, 2025.
21. Zhu X., Li Q., Cui L., Liu Y., Large language model enhanced text-to-SQL generation: A survey, arXiv preprint arXiv:2410.06011, 2024.
22. Rossiello G., Pham N., Glass M., Lee J., Subramanian D., Rationalization models for text-to-SQL, arXiv preprint arXiv:2502.06759, 2025.
23. Kanburoğlu A.B., Tek F.B., TUR2SQL: A cross-domain Turkish dataset for text-to-SQL, 8th International Conference on Computer Science and Engineering (UBMK), Antalya, Türkiye, 206-211, 2023.

24. Kanburoğlu A.B., Tek F.B., Text-to-SQL: A methodical review of challenges and models, Turk. J. Electr. Eng. Comput. Sci., 32 (3), 403-419, 2024.
25. Demirkiran F., Coşkun A.K., Kömeçoğlu Y., Kömeçoğlu B.B., Güven R., Enhancing text-to-SQL conversion in Turkish: An analysis of LLMs with schema context, 9th International Conference on Computer Science and Engineering (UBMK), Antalya, Türkiye, 1-5, 2024.
26. Kanburoğlu A.B., Tek F.B., TURSpider: A Turkish text-to-SQL dataset and LLM-based study, IEEE Access, 12, 169379-169387, 2024.
27. Li J., Hui B., Qu G., Yang J., Li B., Li B., Wang B., Qin B., Cao R., Geng R., Huo N., Zhou X., Ma C., Li G., Chang K.C.C., Huang F., Cheng R., Li Y., Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs, arXiv preprint arXiv:2305.03111, 2023.
28. Yu T., Zhang R., Yang K., Yasunaga M., Wang D., Li Z., Ma J., Li I., Yao Q., Roman S., Zhang Z., Radev D.R., Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belçika, 3911-3921, 2018.
29. Zhang T., Patil S.G., Jain N., Shen S., Zaharia M., Stoica I., Gonzalez J.E., RAFT: Adapting language model to domain specific RAG, arXiv preprint arXiv:2403.10131, 2024.
30. Yang H., Zhang M., Wei D., SRAG: Speech retrieval augmented generation for spoken language understanding, 6th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 279-283, 2023.
31. Oh S., Evaluating mathematical problem-solving abilities of generative AI models: Performance analysis of ChatGPT 4o1 and ChatGPT 4o using the Korean College Scholastic Ability Test, IEEE Access, 13, 1123-1135, 2024.
32. Gemini Team, Georgiev P., Lei V.I., Burnell R., Bai L., Gulati A. vd., Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, arXiv preprint arXiv:2403.05530, 2024.
33. Qwen, Yang A., Yang B., Zhang B., Hui B., Zheng B., Yu B. vd., Qwen2.5 technical report, arXiv preprint arXiv:2412.15115, 2024.