BİTLİS EREN ÜNİVERSİTESİ
FEN BİLİMLERİ DERGİSİ

ISSN: 2147-3129 / e-ISSN: 2147-3188

Article Type	: Research Article	**Year**	: 2025
Received	: October 4, 2024	**Volume**	: 14
Revised	: February 8, 2025	**Issue**	: 1
Accepted	: February 12, 2025	**Pages**	: 163-178
DOI	: 10.17798/bitlisfen.1561298		

IMPROVING TEXT-TO-SQL CONVERSION FOR LOW-RESOURCE LANGUAGES USING LARGE LANGUAGE MODELS

Emir Öztürk [1]

[1] *Trakya University, Computer Engineering Department, Edirne, Türkiye, emirozturk@trakya.edu.tr*

ABSTRACT

Accurate text-to-SQL conversion remains a challenge, particularly for low-resource languages like Turkish. This study explores the effectiveness of large language models (LLMs) in translating Turkish natural language queries into SQL, introducing a two-stage fine-tuning approach to enhance performance. Three widely used LLMs Llama2, Llama3, and Phi3 are fine-tuned under two different training strategies, direct SQL fine-tuning and sequential fine-tuning, where models are first trained on Turkish instruction data before SQL fine-tuning. A total of six model configurations are evaluated using execution accuracy and logical form accuracy. The results indicate that Phi3 models outperform both Llama-based models and previously reported methods, achieving execution accuracy of up to 99.95% and logical form accuracy of 99.95%, exceeding the best scores in the literature by 5–10%. The study highlights the effectiveness of instruction-based fine-tuning in improving SQL query generation. It provides a detailed comparison of Llama-based and Phi-based models in text-to-SQL tasks, introduces a structured fine-tuning methodology designed for low-resource languages, and presents empirical evidence demonstrating the positive impact of strategic data augmentation on model performance. These findings contribute to the advancement of natural language interfaces for databases, particularly in languages with limited NLP resources. The scripts and models used during the training and testing phases of the study are publicly available at https://github.com/emirozturk/TT2SQL.

Keywords: Llama2, Llama3, Nl to Sql, Phi3, Turkish text to sql.

1 INTRODUCTION

The application of large language models (LLMs) has significantly expanded across various fields, including tasks such as classification, detection, and clustering [1], [2], [3]. One specific area of interest is translating natural language text into SQL queries, a task known as

text-to-SQL [4]. Generally, text-to-SQL algorithms involve converting natural language statements into SQL queries to retrieve the desired data from a database [5], [6], [7].

The main challenges in text-to-SQL tasks include handling the complexity and variability of natural language, managing different database schemas, and ensuring the accuracy and efficiency of the generated SQL queries. These challenges are further amplified when dealing with languages other than English, due to differences in syntax, semantics, and available resources. The complexity arises from the need to correctly interpreting the user's intent, which can be expressed in diverse ways, and accurately map it to a corresponding SQL query that adheres to the structure and constraints of the target database. Variability in natural language includes different ways to ask the same question, use of synonyms, and varying sentence structures. When this task extends to other languages, additional layers of complexity such as idiomatic expressions, grammatical structures, and specific linguistic characteristics further challenge the model's capabilities.

LLMs have revolutionized numerous natural language processing (NLP) tasks due to their advanced reasoning and contextual understanding capabilities. Their application in the text-to-SQL domain is particularly noteworthy because it involves understanding the semantics of natural language queries and converting them into structured SQL queries executable by a database. This task requires not only linguistic competence but also an understanding of database schemas and the ability to translate complex logical constructs into correct and efficient SQL code.

Before the use of language models, various methods were employed for text-to-SQL conversion. [8] proposed DialSQL, a dialogue-based framework leveraging human interaction to improve the accuracy of text-to-SQL models. [9] introduced IRNet, a neural approach decomposing the text-to-SQL generation process into three phases: schema linking, intermediate representation generation, and SQL inference. [10] focused on using graph neural networks (GNNs) to represent the database schema structure, enhancing the semantic parser's schema understanding. [11] presented execution guidance, a method utilizing SQL semantics by executing partially generated queries during decoding to filter out invalid candidates. [12] proposed INCSQL, a sequence-to-action parsing approach incrementally filling SQL query slots, exploring non-deterministic oracles to account for multiple correct SQL queries.

Following the success of generative models in reasoning and accurate response generation, these models were also applied to the text-to-SQL domain [13]. Generative models,

with their advanced language understanding and generation capabilities, have proven particularly effective in dealing with the complexities of text-to-SQL tasks. Their ability to generate diverse and contextually appropriate responses makes them suitable for handling the nuances of natural language queries and converting them into precise SQL statements.

While most research has been conducted in English, there have been studies in other languages as well. [14] conducted a pilot study for Chinese SQL semantic parsing by creating the CSpider dataset, translating English questions from the Spider dataset into Chinese, and experimenting with word-based and character-based encoders. [15] introduced the first large-scale text-to-SQL semantic parsing dataset for Vietnamese, extending and evaluating two strong semantic parsing baselines, EditSQL and IRNet, on their dataset and exploring various configurations to improve performance. [16] introduced TableQA, a large-scale Chinese text-to-SQL dataset focusing on table-aware SQL generation, proposing two table-aware approaches to address entity linking and answerability challenges. [17] developed a framework for cross-lingual text-to-SQL semantic parsing that translates non-English utterances into SQL queries based on an English schema, evaluating their method on Chinese, Vietnamese, Farsi, and Hindi using the XSPIDER and XKAGGLE-DBQA datasets. [18] proposed REX, a framework for cross-lingual text-to-SQL semantic parsing leveraging English translations to bridge the language gap for non-English utterances, demonstrating REX's performance on Chinese and Vietnamese datasets.

The aim of these studies is to perform text-to-SQL conversions in low-resource languages with limited data. Researchers have created datasets and adapted existing models to these languages, addressing their unique challenges. These efforts contribute to extending semantic parsing research to languages other than English, increasing linguistic diversity in this field. Besides low-resource languages, there have also been multilingual studies. In [19], text-to-SQL operations were conducted in English, Portuguese, Spanish, and French. However, none of these studies included the Turkish language.

To conduct these studies and make comparisons, common datasets consisting of text and corresponding SQL queries are essential. Many datasets have been provided and their contents reviewed for this purpose [20]. For English, the Spider dataset has become the standard for benchmarking [21]. Multilingual datasets are crucial for developing and evaluating models that can work across different languages. Therefore, the Spider dataset has been translated or manually labeled in other languages [14], [22], [23], [24]. Additionally, a multilingual version of the Spider dataset has been made available for multilingual studies [25]. Besides Spider,

other text-to-SQL datasets in different languages have also been provided [15], [18]. However, for Turkish, the lack of a dedicated dataset has been a significant barrier.

Although text-to-SQL conversion has been explored in various languages, there has been a notable absence of such models and datasets for the Turkish language. To address this gap, [28] introduced TUR2SQL, the first publicly available cross-domain Turkish text-to-SQL dataset. This dataset comprises pairs of natural language statements and their corresponding SQL queries, facilitating objective comparison and evaluation of text-to-SQL models for Turkish, similar to the Spider dataset for English. The authors also conducted experiments using SQLNet [27] and ChatGPT on TUR2SQL, demonstrating the feasibility of text-to-SQL conversion in Turkish and providing baseline performance for future research.

In this study, the goal is to perform unguided text-to-SQL conversion using the TUR2SQL dataset and six fine-tuned language models.

The limited availability of Turkish models and datasets, coupled with the relatively few studies conducted in this area, has highlighted a significant gap in current research. Existing datasets often fall short in fully capturing the unique structural and semantic nuances of the Turkish language, which results in suboptimal outcomes. To further improve these results, this study employs training with Turkish-specific datasets before training with sql dataset to enhance the reasoning capacity of the models in Turkish. This approach aims to enable a more accurate understanding of Turkish linguistic intricacies, ultimately leading to the generation of more precise and reliable SQL queries.

For this process, in the initial phase, these language models are fine-tuned with TUR2SQL, and results are obtained. In the subsequent phase, to enable the models to better understand Turkish instructions, the models are first trained with a Turkish instruction dataset [28] and then fine-tuned with the TUR2SQL dataset.

The contributions of this study are as follows:

- Given that Turkish is a low-resource language and there are no specialized models trained on Turkish text-to-SQL tasks prior to this study, training multiple models capable of generating SQL from Turkish text and making these models publicly available introduces a significant novelty.
- To enhance the model's performance, the LLaMA and Phi models are first trained on a Turkish instruction dataset before being fine-tuned on SQL data. Consequently, Turkish LLaMA and Phi models for instructions are developed. Although the

primary goal of the study is not to develop a Turkish language model, these models have also been made publicly available.

- The study provides a comparative analysis by examining the EX and LF results obtained from other low-resource or multilingual studies. Despite differences in datasets and languages, this analysis evaluates the acceptability and robustness of the model's performance.

The second section of the study provides detailed information about the dataset, models, and training process. The third section presents the experimental results. The analysis includes a comparison with previous studies to highlight the advancements made in text-to-SQL conversion for Turkish. The final section presents the conclusions and discusses the implications of the results.

2 MATERIAL AND METHOD

The Tur2SQL dataset includes a metadata file containing tables and their IDs, as well as natural language instructions for querying these tables and the corresponding SQL queries. Additionally, for guided training, the dataset includes fields within the JSON files indicating the indices of keywords such as SELECT and COUNT. An example of a record in the Tur2SQL dataset is shown in Figure 1.


```
{
  "question": "Çiftliklerin koyun sayılarını getir.",
  "query_tok": [
    "SELECT",
    "koyun_sayisi"
  ],
  "table_id": "100046",
  "sql": {
    "agg": 0,
    "sel": 5,
    "conds": []
  },
  "phase": 1,
  "query": "SELECT koyun_sayisi",
  "question_tok": [
    "Çiftliklerin",
    "koyun",
    "sayılarını",
    "getir",
    "."
  ]
}
```


Figure 1. An example record in Tur2SQL dataset.

Upon examining the dataset example, it is noted that the keyword "FROM" is not present. Instead, the dataset uses a variable `table\_id` to indicate which table the query is

directed at. The `table\_id` values are also provided with names instead of ids in the database file used for result calculation. For the training stage, the IDs are first matched with the table names in the database. Subsequently, the relevant table name is appended with the "FROM" clause before the WHERE clause if present, or at the end of the query otherwise. The processed data is then used for training and testing. The steps for preparing the training data are illustrated in Figure 2.

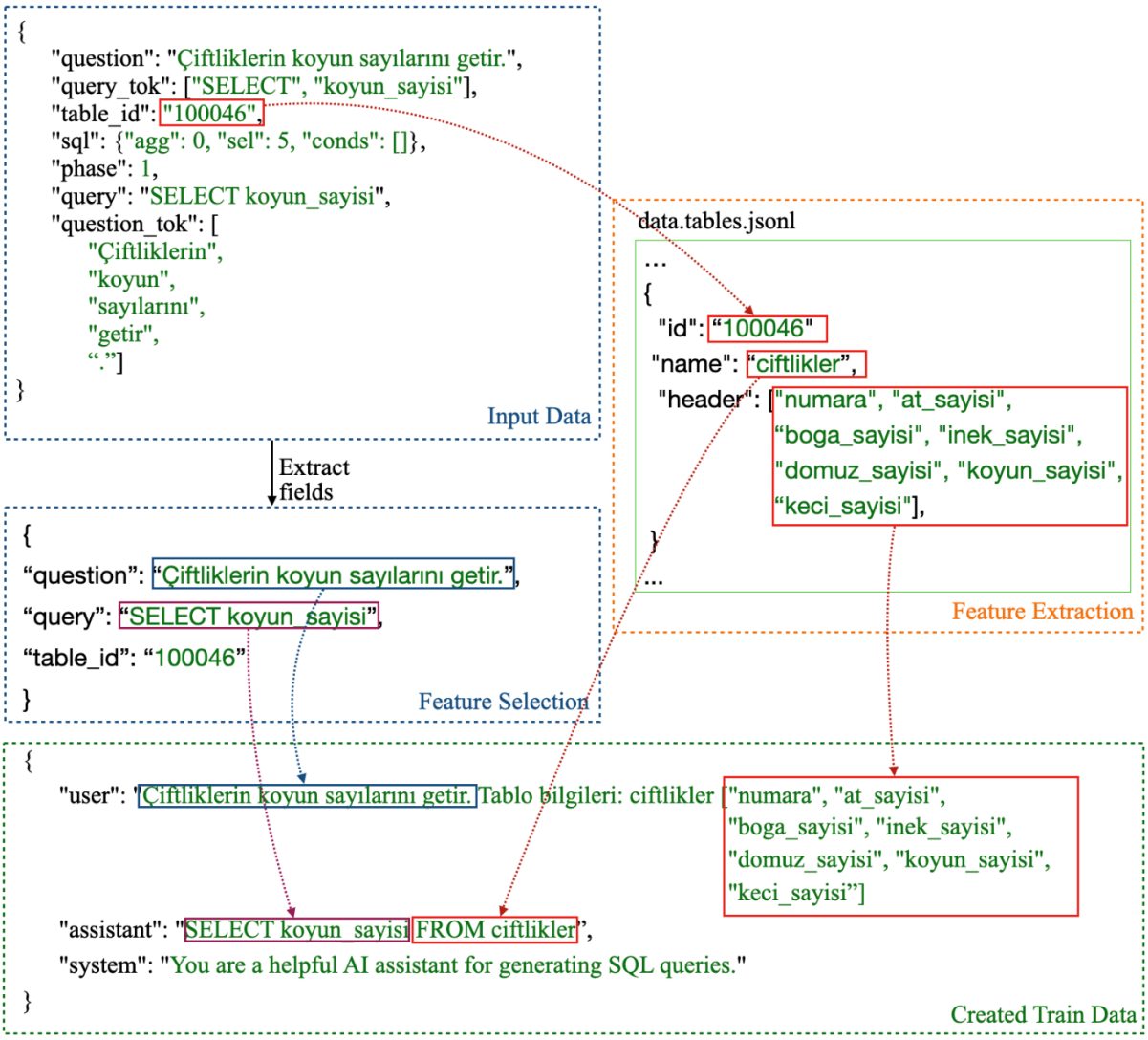

Figure 2. Preprocessing steps of training data.

As seen in Figure 2, a JSON file containing records with three key fields is provided to the models for training. Among these fields, the system field defines the nature of the response that the LLM model will generate. The system instruction acts as a guideline for the model's responses, nudging the AI to prioritize user satisfaction, offer assistance, and avoid generating harmful or offensive content. The user field contains the desired natural language text for the

query and information about the table from which the query is requested. In Figure 2, the example sentence means "Retrieve the number of sheep on the farms" and the "Tablo bilgileri" field shows the table information. The assistant field contains the response that needs to be generated based on the requested query.

In this study, since unguided training is being conducted, the indices of the keywords are not used. Only the natural language request text and the SQL query are utilized. The dataset is divided into three different files: train, dev, and test. The train and dev files are used for training, while the test file is used for evaluating the results. This approach focuses on leveraging the natural language requests to train models without explicit guidance, which can help in developing models that generalize better to unseen queries.

For training the dataset, the following models are used: “meta-llama/Llama-2-7b-chat-hf” [29], “meta-llama/Meta-Llama-3-8B-Instruct” [30], and “microsoft/Phi-3-mini-4k-instruct” [31]. As these models do not inherently support the Turkish language, two approaches are taken for model training. Initially, the models are trained with the Tur2SQL dataset, and the test results are obtained. The resulting models are named Llama2-7b-chat-SQL, Llama3-8b-instruct-SQL, and Phi3-mini-4k-instruct-SQL. Since the SQL queries' length does not exceed a certain number of tokens, the input and output token numbers are reduced in the models to improve training and test performance. This token reduction helps to optimize memory usage and training time, ensuring efficient handling of the dataset without compromising model accuracy.

In the second phase, to improve the models' understanding of the table name context or better interpret the desired instructions, the models are first trained with Turkish instruction dataset which can generate Turkish responses to Turkish inputs. Subsequently, in the second phase, these models are further fine-tuned with the Tur2SQL dataset to obtain new performance results. The resulting models are named as Llama2-7b-chat-Turkish-SQL, Llama3-8b-instruct-Turkish-SQL, and Phi3-mini-4k-instruct-Turkish-SQL. This two-step approach aims to enhance the models' understanding of Turkish syntax and semantics before fine-tuning them specifically for text-to-SQL tasks, thereby improving their overall performance and accuracy.

A general schema of the model training stages is illustrated in Figure 3.

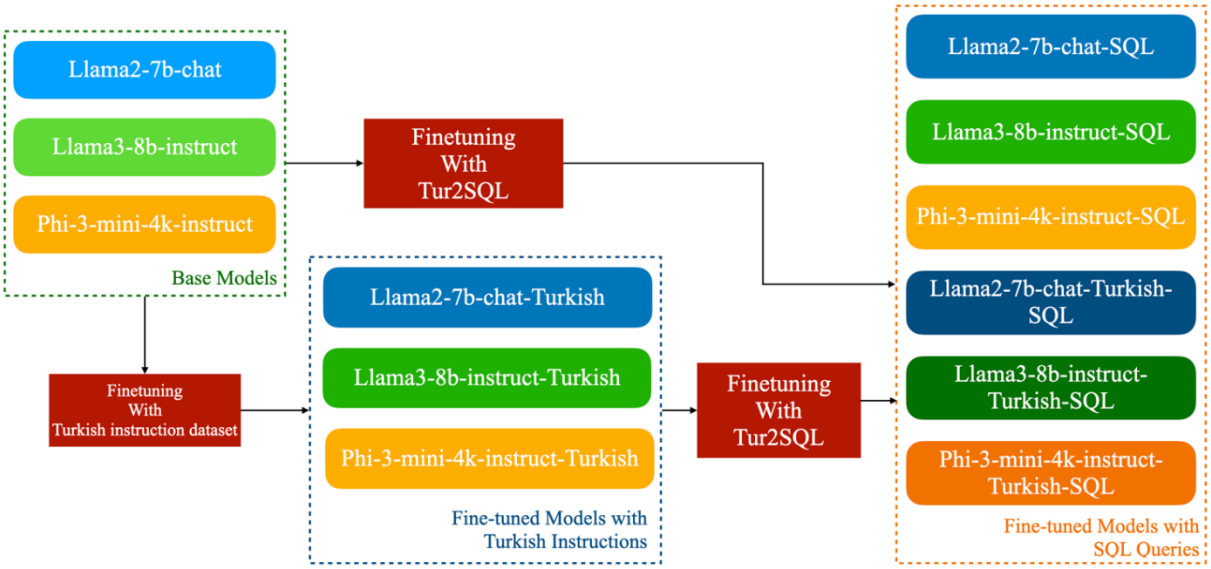

Figure 3. Overall model fine-tuning stages.

For training SuperAdapters library on Github is used. Since the datasets used were preprocessed and structurally presented, no additional preprocessing was performed in the study, except for converting them into a prompt format suitable for the given languages. During the fine-tuning phase, LoRa and PEFT were used, with the learning rate set to 0.00025. Adam was used as the optimizer, and different target modules were trained for each model. Training is conducted for 50 steps for each dataset. Training is stopped when the loss values of the training and test datasets converged and their changes minimized. As a result of the training phase, six models are obtained. These models and their train and validation loss values on step 50 are presented in Table 1 and training and validation loss graphs are given in Figure 4. The training process involves iterating over the data multiple times (steps), adjusting the model parameters to minimize the difference (loss) between the predicted SQL queries and the actual queries in the dataset. The loss values indicate how well the model is learning to perform the task, with lower values signifying better performance.

Table 1. Train and validation losses of fine-tuned models for 50 steps.

Model	Train Loss	Validation Loss
Llama2-7b-chat-SQL	0.0001	0.00008
Llama2-7b-chat-Turkish-SQL	0.0001	0.00010
Llama3-8b-instruct-SQL	0.0001	0.00004
Llama3-8b-instruct-Turkish-SQL	0.0001	0.00005
Phi3-mini-4k-instruct-SQL	0.0001	0.00009
Phi3-mini-4k-instruct-Turkish-SQL	0.0001	0.00051

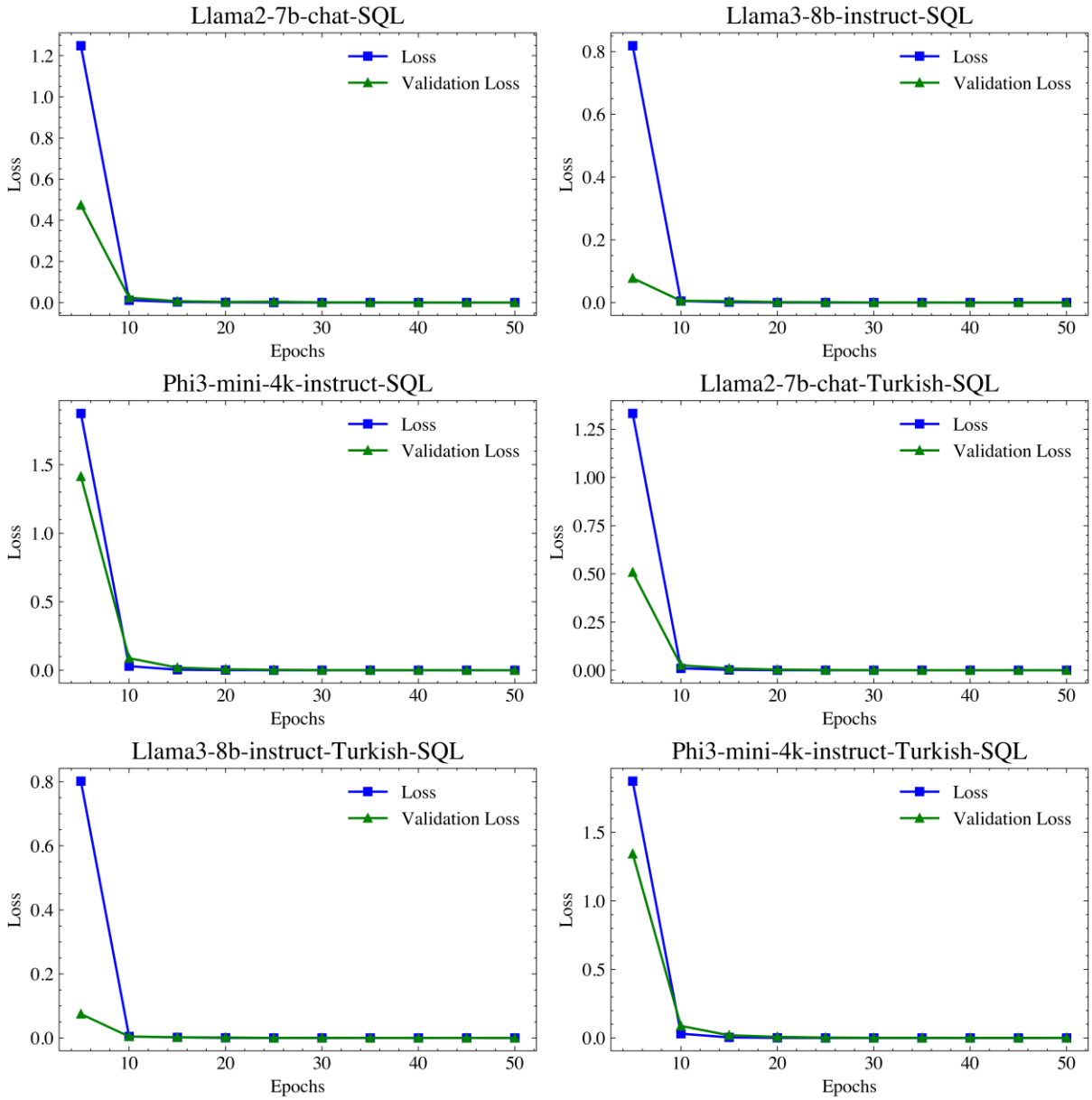

Figure 4. Training and validation loss graphs.

As seen in Table 1 and Fig 4, the train and validation loss values progress steadily without any signs of overfitting. The validation data, like the training data, also showed a reduction in the loss value.

3 RESULTS AND DISCUSSION

For performance evaluation of the test data, the ground truth SQL queries, referred to as gold SQL, are compared with the SQL queries predicted by the model. These comparisons are measured by execution accuracy (EX) and logical form accuracy (LF), as reported in the literature [32], [33]. Execution accuracy compares the generated query and the ground truth

query in terms of their results, typically by executing both queries on the same database and comparing their outputs. Logical form accuracy focuses on exact string matching between synthesized and ground truth queries but penalizes correct results without exact string matches [34]. In this study, to facilitate comparison with the results obtained in [26], the execution accuracy and logical form accuracy results of the models are obtained. The test file of the Tur2SQL dataset is used to obtain the results.

Each model uses specific prompt formats during the training and testing phases. These formats contain certain special tags and at the end of training, and the outputs require preprocessing. The LLaMA2 models generate outputs between </s> tags and produce tokens up to the maximum token limit specified. In contrast, the LLaMA3 models generate the <|eot_id|> tag up to the maximum token limit at the end of production. The Phi models, however, produce outputs directly without including any special tags. Consequently, it is necessary to process these tags for each model before calculating EX and LF results. The outputs generated by the models and their cleaned versions are presented in Table 2.

Table 2. Models and examples of predictions and cleaned predictions.

Model	Prediction	Cleaned Prediction
Llama2-7b-chat-SQL	SELECT *max(numara)* FROM *sehirler*</s></s>2. SELECT *count(numara)* FROM *kategori* </s> SELECT *count(num*	SELECT *max(numara)* FROM *sehirler*
Llama2-7b-chat-Turkish-SQL	SELECT *max(numara)* FROM *sehirler* </s></s>2. SELECT *count(numara)* FROM kategori </s> SELECT *count(num*	SELECT *max(numara)* FROM *sehirler*
Llama3-8b-instruct-SQL	SELECT *max(numara)* FROM *sehirler*<|eot_id|> <|eot_id|><|eot_id|><|eot_id|><|eot_id|><|eot_id|> <|start_header_id|><|eot_id|><|eot_id|><|eot_id|> <|eot_id|><|eot_id|><|end_header_id|><|eot_id|>	SELECT *max(numara)* FROM *sehirler*
Llama3-8b-instruct-Turkish-SQL	SELECT *max(numara)* FROM *sehirler* <|eot_id|> <|eot_id|><|eot_id|><|eot_id|><|eot_id|> <|eot_id|> <|start_header_id|> <|start_header_id|><|eot_id|> <|eot_id|><|eot_id|><|eot_id|><|start_header_id|> <|eot_id|>	SELECT *max(numara)* FROM *sehirler*
Phi3-mini-4k-instruct-SQL	SELECT *max(numara)* FROM *sehirler*	-
Phi3-mini-4k-instruct-Turkish-SQL	SELECT *max(numara)* FROM *sehirler*	-

As shown in Table 2, for LLaMA2 models, it is sufficient to take the portion of the output before the first </s> tag to obtain the cleaned result. Similarly, for LLaMA3 models, the portion of the output before the first <|eot_id|> tag is taken as the produced result. For Phi models, there was no need for a splitter.

The execution accuracy and logical form accuracy values of the trained models are presented in Table 3.

Table 3. Execution and Logical Form Accuracy results.

Model	Dev		Test	
	EX	LF	EX	LF
Llama2-7b-chat-SQL	85.95	85.31	85.89	85.29
Llama2-7b-chat-Turkish-SQL	89.15	85.04	86.03	85.01
Llama3-8b-instruct-SQL	93.25	93.25	92.96	92.73
Llama3-8b-instruct-Turkish-SQL	92.40	92.65	93.84	92.83
Phi3-mini-4k-instruct-SQL	99.72	99.42	99.81	99.81
Phi3-mini-4k-instruct-Turkish-SQL	99.96	99.95	**99.95**	**99.95**
SQLNet	-	-	40.19	39.03
ChatGPT	-	-	98.38	86.72

Table 3 presents the results of six different models trained in this study, as well as the results from SQLNet and ChatGPT previously tested in the TUR2SQL study. The latter two results did not involve any fine-tuning procedures. When examining the execution accuracy results, it is evident that the Phi models achieved the best performance, while the LLaMA models produced results comparable to ChatGPT. Additionally, as demonstrated by the results, training with the Turkish dataset improved execution accuracy for every model, since the instruction dataset helped the models learn the context of the natural language questions more effectively. Considering that the Phi models are smaller versions, it is expected that they would perform significantly better on the existing SQL dataset. The LLaMA models, being more generalized models, did not converge as effectively on the dataset as the Phi models. In terms of logical form accuracy, all models except for the LLaMA2 models achieved better results than ChatGPT. The LLaMA2 models, also, produced results very close to ChatGPT with only a 1\% difference. It is clear that in resource-constrained scenarios, the Phi models would be more suitable for use. Each model trained with the [28] dataset showed an average performance improvement of 1\% for the LLaMA models.

Despite the results provided by [26] for untrained SQLNet and ChatGPT models, there are no other trained models available for comparison. Execution accuracy and exact match accuracy values of other methods published in the literature are given in Table 4.

Table 4. Execution and Logical Form Accuracy results of studies conducted on Spider dataset.

Model	Dev		Test	
	LF	EX	LF	EX
Coarse2Fine	72.9	79.2	71.7	78.4
IncSQL	76.1	82.5	75.5	81.6
SQLova	81.6	87.2	80.7	86.2
X-SQL	83.8	89.5	83.3	88.7
HydraNet	83.6	89.1	83.8	89.2
IE-SQL	84.6	88.7	84.6	88.8
BRIDGE	86.2	91.7	85.7	91.1
SDSQL	86.0	91.8	85.6	91.4
SeaD	84.0	90.2	84.7	90.1

Table 4 presents the performance of various models on English datasets from existing literature. It is seen that the top-performing models on English datasets to achieve around 80-90\% execution accuracy and logical form accuracy. For instance, the BRIDGE model, which is one of the top performers in the literature, achieves 91.7\% execution accuracy and 86.2\% logical form accuracy on the test set. Therefore, the results from our Turkish dataset, where models trained achieve 85-99\% execution and logical form accuracy, are particularly acceptable.

Table 5 provides examples of the errors made by the models and the situations that classified as errors, which led to decreases in EX or LF values.

Table 5. Models and examples of false predictions.

Model	Query	Cleaned Prediction
Llama2-7b-chat-SQL	SELECT *tip* FROM *suruculer* WHERE *kod = 20202020*	SELECT *tip* FROM suruculer WHERE *kod = 20203020*
Llama2-7b-chat-Turkish-SQL	SELECT *alan* FROM *sehirler* WHERE *nufus = 15000000*	SELECT *alan* FROM sehirler WHERE *nufus = 1500*
Llama3-8b-instruct-SQL	SELECT *kalkis_tarihi* FROM *ucuslar* WHERE *kalkis_tarihi* = \	SELECT *kalkis_tarihi* FROM ucuslar WHERE *kalkis_tarihi* = \
Llama3-8b-instruct-Turkish-SQL	SELECT *kalkis_tarihi* FROM *ucuslar* WHERE *kalkis_tarihi* = \	SELECT *kalkis_tarihi* FROM ucuslar WHERE *kalkis_tarihi* = \
Phi3-mini-4k-instruct-SQL	SELECT *max(numara)* FROM *kopruler*	SELECT *max(uzunluk)* FROM *kopruler*
Phi3-mini-4k-instruct-Turkish-SQL	SELECT *cikis_tarihi* FROM *sarkilar*	SELECT *cikis_tarihi* FROM *sarkilar* WHERE *cikis_tarihi* = şarkı tablosu içerisinde bulunan çıkış tarihlerini ekrana\

As shown in Table 5, all models produced a response to the given queries. All of the generated responses also produced correct SQL queries. Only the Phi3 model generated natural language texts at the end of the SQL query. Upon examination, it was generally found that the outputs contained errors due to typographical errors in the numbers or within the text. For example, in the first query example, the value 20202020 was generated as 20203020, and because it did not produce the correct result, it was classified as incorrect. However, it can be observed that the query would work if this typographical error were corrected. It was observed that, with a few exceptions, all outputs from the models had errors of this nature.

Since query construction is a text generation process, it does not involve reasoning, and therefore, common errors are mostly related to matching numbers and dates. Additionally, errors such as generating redundant data are encountered. To correct these errors in the query generation system, solutions should focus on detecting the endpoint of the query and identifying inconsistencies in values like dates and numbers within the query, ensuring they are corrected based on internal consistency checks.

4 CONCLUSION AND SUGGESTIONS

In this study, text-to-SQL task for the Turkish language, considered a low-resource language, is performed using open-source models. To accomplish this, three different models are trained following two different approaches, and a reliable dataset presented in the literature is used for training.

The results of this study illustrate the significant advancements in text-to-SQL performance achieved through the use of large language models (LLMs) in the context of the Turkish language. All the models exhibit same or better performance compared to models given in the literature, with the Phi3 models achieving the highest execution accuracy and logical form accuracy values.

4.1 Comparative Performance Analysis

Furthermore, the study reveals that all trained models outperform SQLNet, with models except the Phi-3 models closely matching ChatGPT's success. This indicates the substantial benefits of LLMs in handling text-to-SQL tasks, even in languages with limited resources. The Phi-3 models, although not as complex as the LLaMA models, surpass all methods, suggesting their potential utility in resource-constrained environments. The reason of this is the restricted

availability of datasets for Turkish, which is classified as a low-resource language. The success of the Phi-3 models demonstrates that effective text-to-SQL processing can be achieved with low resource usage. Thus, by selecting Phi-3 models, acceptable production success can be ensured with minimal resource consumption.

4.2 Impact of Turkish Instruction Data

The integration of Turkish instruction data in model training has significantly improved SQL query generation accuracy. This improvement is attributed to the enhanced understanding of natural language question contexts facilitated by the instruction dataset. On average, the pre-training with the Turkish instruction dataset resulted in a performance gain across all models. This step helped the models better grasp the linguistic nuances and typical query structures in Turkish, thereby improving their ability to generate accurate SQL queries from natural language inputs. The instruction-tuning phase effectively fine-tuned the models to handle Turkish-specific syntactic and semantic patterns, which are crucial for accurate text-to-SQL conversion.

Results highlights the transformative impact of LLMs on text-to-SQL tasks, particularly in low-resource languages like Turkish. The findings underscore the value of leveraging large, instruction-tuned models to achieve high accuracy in SQL query generation, paving the way for more effective and accessible natural language interfaces to databases. Additionally, the study demonstrates that even in the absence of extensive resources, strategic use of instruction data and careful model training can bridge the performance gap, enabling high-accuracy applications in diverse linguistic contexts.

4.3 Future Work

In future, further optimization of model architecture and training procedures could potentially enhance performance even further. Additionally, the creation or acquisition of a larger dataset will provide sufficient data to meet the requirements of models with more parameters. By fulfilling the data needs of these models, subsequent training endeavors can lead to the development of a significantly more generalized text-to-SQL model for the Turkish language. Also, investigating the combination of LLMs with other techniques, such as data augmentation and transfer learning, may also yield valuable insights and improvements.

Conflict of Interest Statement

There is no conflict of interest between the authors.

Statement of Research and Publication Ethics

The study is complied with research and publication ethics.

Artificial Intelligence (AI) Contribution Statement

This manuscript was entirely written, edited, analyzed, and prepared without the assistance of any artificial intelligence (AI) tools. All content, including text, data analysis, and figures, was solely generated by the author.

REFERENCES

[1] K. Mohamad and K. M. Karaoğlan, "Enhancing Deep Learning-Based Sentiment Analysis Using Static and Contextual Language Models," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 12, no. 3, pp. 712–724, 2023.

[2] K. M. Karaoğlan, "Novel approaches for fake news detection based on attention-based deep multiple-instance learning using contextualized neural language models," *Neurocomputing*, vol. 602, p. 128263, 2024.

[3] K. M. Karaoglan and O. Findik, "Enhancing Aspect Category Detection Through Hybridised Contextualised Neural Language Models: A Case Study In Multi-Label Text Classification," *Comput J*, p. bxae004, 2024.

[4] D. Gao *et al.*, "Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation", *arXiv preprint arXiv:2308.15363*, 2023.

[5] A. Kumar, P. Nagarkar, P. Nalhe, ve S. Vijayakumar, "Deep Learning Driven Natural Language Text-to-SQL Query Conversion: A Survey", *arXiv preprint arXiv:2208.04415*, 2022.

[6] C. Wang, A. Cheung, ve R. Bodik, "Synthesizing Highly Expressive SQL Queries from Input-Output Examples", içinde *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2017, ss. 452-466.

[7] A. Giordani ve A. Moschitti, "Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked", içinde *Proceedings of COLING 2012: Posters*, 2012, ss. 401-410.

[8] I. Gür, S. Yavuz, Y. Su, ve X. Yan, "DialSQL: Dialogue Based Structured Query Generation", içinde *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, ss. 1339-1349.

[9] J. Guo *et al.*, "Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation", *arXiv preprint arXiv:1905.08205*, 2019.

[10] B. Bogin, M. Gardner, ve J. Berant, "Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing", *arXiv preprint arXiv:1905.06241*, 2019.

[11] C. Wang *et al.*, "Robust Text-to-SQL Generation with Execution-Guided Decoding", *arXiv preprint arXiv:1807.03100*, 2018.

[12] T. Shi, K. Tatwawadi, K. Chakrabarti, Y. Mao, O. Polozov, ve W. Chen, "InCSQL: Training Incremental Text-to-SQL Parsers with Non-Deterministic Oracles", *arXiv preprint arXiv:1809.05054*, 2018.

[13] A. Liu, X. Hu, L. Wen, ve P. Yu, "A Comprehensive Evaluation of ChatGPT's Zero-Shot Text-to-SQL Capability", *arXiv preprint arXiv:2303.13547*, 2023.

[14] Q. Min, Y. Shi, ve Y. Zhang, "A Pilot Study for Chinese SQL Semantic Parsing", *arXiv preprint arXiv:1909.13293*, 2019.

[15] A. T. Nguyen, M. H. Dao, ve D. Q. Nguyen, "A Pilot Study of Text-to-SQL Semantic Parsing for Vietnamese", *arXiv preprint arXiv:2010.01891*, 2020.

[16] S. Ningyuan, Y. Xuefeng, ve L. Yunfeng, "TableQA: A Large-Scale Chinese Text-to-SQL Dataset for Table-Aware SQL Generation". 2020. Erişim adresi: https://arxiv.org/abs/2006.01234

[17] P. Shi, R. Zhang, H. Bai, ve J. Lin, "XRICL: Cross-Lingual Retrieval-Augmented In-Context Learning for Cross-Lingual Text-to-SQL Semantic Parsing", *arXiv preprint arXiv:2210.13693*, 2022.

[18] P. Shi *et al.*, "Cross-Lingual Text-to-SQL Semantic Parsing with Representation Mixup", içinde *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, ss. 5296-5306.

[19] M. A. Jose ve F. G. Cozman, "A Multilingual Translator to SQL with Database Schema Pruning to Improve Self-Attention", *International Journal of Information Technology*, c. 15, sy 6, ss. 3015-3023, 2023.

[20] N. Deng, Y. Chen, ve Y. Zhang, "Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect", *arXiv preprint arXiv:2208.10099*, 2022.

[21] T. Yu *et al.*, "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task", *arXiv preprint arXiv:1809.08887*, 2018.

[22] A. Heakl, Y. Mohamed, ve A. B. Zaky, "AraSpider: Democratizing Arabic-to-SQL", *arXiv preprint arXiv:2402.07448*, 2024.

[23] D. Bakshandaeva, O. Somov, E. Dmitrieva, V. Davydova, ve E. Tutubalina, "PAUQ: Text-to-SQL in Russian", içinde *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, ss. 2355-2376.

[24] O. Somov ve E. Tutubalina, "Shifted PAUQ: Distribution Shift in Text-to-SQL", içinde *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, 2023, ss. 214-220.

[25] L. Dou *et al.*, "MultiSpider: Towards Benchmarking Multilingual Text-to-SQL Semantic Parsing", içinde *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, ss. 12745-12753.

[26] A. B. Kanburoğlu ve F. B. Tek, "TUR2SQL: A Cross-Domain Turkish Dataset For Text-to-SQL", içinde *2023 8th International Conference on Computer Science and Engineering (UBMK)*, IEEE, 2023, ss. 206-211.

[27] X. Xu, C. Liu, ve D. Song, "SQLNet: Generating Structured Queries from Natural Language without Reinforcement Learning", *arXiv preprint arXiv:1711.04436*, 2017.

[28] T. Kurtuluş, "turkish_73k_instruct_extended", *HuggingFace Dataset Repository*. HuggingFace.co, 2024. Erişim adresi: https://huggingface.co/datasets/tolgadev/turkish_73k_instruct_extended

[29] H. Touvron *et al.*, "LLaMA 2: Open Foundation and Fine-Tuned Chat Models", *arXiv preprint arXiv:2307.09288*, 2023.

[30] AI@Meta, "LLaMA 3 Model Card", 2024. Erişim adresi: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[31] M. Abdin *et al.*, "Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone", *arXiv preprint arXiv:2404.14219*, 2024.

[32] B. Qin *et al.*, "A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions", *arXiv preprint arXiv:2208.13629*, 2022.

[33] V. Zhong, C. Xiong, ve R. Socher, "Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning", *arXiv preprint arXiv:1709.00103*, 2017.

[34] A. B. Kanburoğlu ve F. B. Tek, "Text-to-SQL: A Methodical Review of Challenges and Models", *Turkish Journal of Electrical Engineering and Computer Sciences*, c. 32, sy 3, ss. 403-419, 2024.