Gr-Spider: Revealing the Power of Greek Text-to-SQL

Panagiotis Korovesis[†] panosk@di.uoa.gr
Ioannis Fourfouris[†] gfourfouris@di.uoa.gr
National and Kapodistrian University of Athens[†]

1 Introduction

Text-to-SQL, also known as Natural Language to SQL (NL2SQL), is a task in natural language processing that aims to convert human language queries into structured SQL queries. This involves analyzing the input text to understand its meaning and then mapping it to the appropriate SQL query components such as SELECT, FROM, WHERE and JOIN clauses.

While several benchmark datasets exist for English, there are few for other languages [1], [2] and particularly none for Greek. Considering that SQL serves as a universal semantic representation, we think that is valuable to explore its application in languages beyond English.

In this study, we introduce the Gr-Spider dataset, which is, to the best of our knowledge, the first Greek dataset for Text-to-SQL applications. Gr-Spider is essentially the Greek adaptation of the widely known Spider dataset [3]. The dataset was created by translating the English Spider dataset into Greek. For the translation process, we utilized Large Language Models (LLMs). Then, two annotators verified and post-edit the translations.

To evaluate the translated dataset, we used two advanced baseline Text-to-SQL models: RAT-SQL [4] and DAIL-SQL [5]. To support the Greek language, we enhanced DAIL-SQL with the paraphrase-multilingual-MiniLM-L12-v2 sentence transformer [6]. Additionally, we enhanced RAT-SQL by incorporating the mT5-base language encoder [7], allowing us to train the model on our local machine. These multilingual models help to reduce the impact of linguistic differences in the schema and address some of the challenges in SQL structure.

As part of the course, we translated only a subset of the initial dataset, in order to meet the project's deadline.

2 The Greek Spider Dataset

2.1 Spider Overview

Spider is a comprehensive, large-scale dataset for semantic parsing and text-to-SQL tasks, annotated by 11 Yale college students. The dataset encompasses 10181 questions and 5693 unique complex SQL queries across 200 databases, each with multiple tables, covering 138 distinct domains. It established a new benchmark for complex, cross-domain semantic parsing and text-to-SQL tasks by ensuring different SQL queries and database schemas in the training and testing sets. This setup demands that models generalize effectively to both novel SQL queries and unfamiliar database schemas.

SQL queries are categorized into four difficulty levels: Easy, Medium, Hard, and Extra Hard. We will refer to these categories as **(E), (M), (H) and (Ex)**. The difficulty is determined by the number and complexity of SQL components, selections and conditions. Queries that utilize a greater number of SQL keywords such as GROUP BY, ORDER BY, INTERSECT, nested subqueries, column selections and aggregators are classified as harder.

Due to the course's time constraints, for the remaining work we used only a subset of the data. More specifically, we split the development data into the required files (train_spider.json, train_others.json, dev.json) to match the proportions of the data in the original files, as shown in Table 1. We also utilize related schema information, which encompasses 20 distinct schemas, 81 tables and 441 columns, along with the related 1034 questions. From now on, we will refer to this dataset as 'Mini-Spider' and the equivalent dataset in Greek as 'Mini-Gr-Spider'.

	Original Spider	Mini-Spider
Category	**Data Count (Percentage)**	
dev	1034 (10.7%)	110 (10.7%)
train_spider	7000 (72.2%)	747 (72.2%)
train_others	1659 (17.1%)	177 (17.1%)
Total	9693	1034

Table 1: Comparison of Original Spider and Mini-Spider Data.

2.2 Translation Pipeline

The translation process is structured into two steps:

1. First, we translate the schema, including table and column names.
2. Next, the question translation is performed using the translated schema as a reference.

To increase efficiency and avoid manual translation of all information, we first used Large Language Models in order to translate the Spider dataset. Section 2.3 details the overall LLM selection.

For the translation task, we designed specific system prompts, which are outlined in Table 2. Additionally, we created a special input format to feed into the language models to obtain the translated information. Examples of these inputs can be seen in Figure 1.

Then, we, two data science master's students, correct and post-edit the translations, utilizing any gathered contextual information.

It should be highlighted that the final translation at each stage results from a cross-check between ours (in other words, the annotators) post-editing efforts.

Type	**System Prompt**
Table	I will give you table names from a schema in English and you will translate it to Greek. Please return only the translated table name.
Column	I will give you English column names from tables and you will translate them to Greek. Please return only the translated column.
Question	I will give you English questions and you will translate them to Greek. Please return only the translated question.

Table 2: System Prompts for Translation.

2.3 Large Language Models

Large Language Models (LLMs) are trained on large amounts of text, which helps them learn various language patterns and cultural contexts. This enhances the accuracy and relevance of their translations, which is why we opt for LLMs in the translation process.

In that sense, we examined three models for their suitability for English-to-Greek translation namely Llama 3 [1], Meltemi [2] and Gpt-3.5 Turbo [3]. Llama 3 and Meltemi were provided by Ollama, whereas Gpt-3.5 Turbo was accessible through the Microsoft Azure OpenAI Service.

To identify the optimal model, we manually translated 3 schemas into Greek, comprising of 18 tables and 89 columns. Subsequently, we fed the 258 relevant questions into the models using the aforementioned system and input prompt and obtained the translated questions. We then validated these translations to assess their accuracy.

Table 3 shows that Gpt-3.5 Turbo translated the most questions correctly, leading us to select it for the subsequent translation tasks. The 'No one' entry shows the percentage of questions that were incorrectly translated by all the language models.

[1] https://ollama.com/library/llama3
[2] https://ollama.com/ilsp/meltemi-instruct
[3] https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models#gpt-35

```
Translate the column "breed_code" to Greek to fit the context of the table:
Table: ράτσες
Columns: breed_code, breed_name
```

(a) Column Input Prompt.

```
Translate the question "How many singers do we have?" to Greek. You can only use keywords provided below:
Table: στάδιο
Columns: αναγνωριστικό σταδίου, τοποθεσία, όνομα, χωρητικότητα, υψηλότερο, χαμηλότερο, μέσος όρος
Table: τραγουδιστής
Columns: αναγνωριστικό τραγουδιστή, όνομα, χώρα, όνομα τραγουδιού, έτος κυκλοφορίας τραγουδιού, ηλικία, είναι άνδρας
Table: συναυλία
Columns: αναγνωριστικό συναυλίας, όνομα συναυλίας, θέμα, αναγνωριστικό σταδίου, έτος
Table: τραγουδιστής σε συναυλία
Columns: αναγνωριστικό συναυλίας, αναγνωριστικό τραγουδιστή
```

(b) Question Input Prompt.

Figure 1: Example Input Prompts for Translation.

LLM	Correct Translation
Gpt-3.5 Turbo	93.80%
Meltemi	20.16%
Llama3	17.83%
No One	2.71%

Table 3: Accuracy of Translation Models.

It is worth noting that translated questions produced by LLMs, which included additional text such as explanations for why the question was translated a certain way or definitions of tables and columns, were marked as incorrect due to the models' 'babbling' behavior, especially in Llama3 and Meltemi. This is the main reason why these models have not achieved good accuracy.

For the remaining data, we continue the translation process outlined in Section 2.2.

2.4 Challenge of Dataset Translation

Based on our initial investigation, we have identified the most common errors that arise when translating Text-to-SQL datasets from English to Greek, covering both schema and questions.

Ambiguity in Context and Domain Knowledge: Some schema information lacks clarity regarding its intended reference. For instance, the term 'Student' may be relevant to a college attendee, a school-level student or even a doctoral candidate. Additionally, there are abbreviations such as 'shp_feature_1' whose specific meaning is unclear. In such cases, we refer to the questions that use this information to validate the logic.

Non-1-1 Association: Some English words do not have exact equivalents in Greek. For instance, the word 'State' can be translated as 'Politeia', but with a different meaning. In English, it typically denotes each of the federal states constituting the United States, whereas in Greek, it refers to the organization of a city-state or the form of government.

Entity polysemy: Some words have multiple meanings in Greek. For example, the word 'Transcript' can refer to a general written or printed material or a record of study.

2.5 Dataset Analysis

While Gpt-3.5 Turbo performed excellent in the translation process, we continued to refine the schema and the questions, to maintain a high-quality standard for the data. Details of the statistics of the post-editing data can be found in Table 4.

The *is_altered* information in the table reveals the percentage of questions that were modified between the initial Gpt translation and the final translated version. As mentioned earlier, each annotator verified and post-edited the Gpt translation. The final translation resulted from a verification process

	Tables	Columns	Questions
is_altered	24/81 (29.6%)	209/441 (47.3%)	531/1034 (51.3%)
cross_check	21/24 (87.5%)	112/209 (23.5%)	192/531 (36.1%)

Table 4: Statistics of post-editing for Tables Columns and Questions.

of the post-edits. The *cross_check* information indicates the percentage of disagreement between the two annotators (based only on the data that were altered).

Regarding the adjustments, we reorganized the content within sentences, eliminated unnecessary words and refined the sentence structures to sound more natural. Additionally, we corrected table and column names where needed. This was necessary because Gpt, occasionally, did not strictly follow the input keywords provided in the prompt. Instead, it translated the content from scratch.

For this Mini-Spider translation, we spent more than two days in total on translation and data review.

3 Experiments

Existing work on this topic uses the full version on this Spider Dataset. Thus, since we are using the Mini-Spider and Mini-Gr-Spider respectively, the results presented in this paper are not directly comparable with the existing literature. They can however be compared with each other to produce meaningfull insights. The conducted experiments can be separated into two distinct categories, based on the tool used for Text-to-SQL task.

First we used a modified version of the RAT-SQL for a traditional machine learning approach to the Text-to-SQL translation task. Then we used a modified version of DAIL-SQL which mainly relies on Software as a Service (SaaS) LLMs.

3.1 RAT-SQL Experiments

The RAT-SQL [8] framework which uses relation-aware self-attention to combine global reasoning over the schema entities and question words with structured reasoning over predefined schema relations. This approach is proven to perform better at generalizing to unseen database schemas. RAT-SQL manages to encode the database relations in an accessible way for the semantic parser and to map columns and their mentions in a given natural language query. This framework is limited however only to English questions.

The mRAT-SQL model, as introduced in Multispider [1], incorporates multi-language models like mT5 and mBart, facilitating the training and inference processes across all supported languages of the models. The authors of the paper, managed to get an average exact match score of 0.667 across 7 languages.

The framework was originally designed with 7 languages in mind. During the pre-processing phase, schema linking takes place. This is achieved by lemmatizing the questions as well as the tables and the columns of the database. Then the columns and tables are mapped to the questions, where possible. The lemmatizer of choice for this framework is Simplemma [9]. Through experimenting we found out that this lemmatizer performed worse at the schema linking process for the Greek dataset, when compared with the Stanza [10] lemmatizer. As such, we replaced Simplemma with Stanza for some of the experiments.

3.1.1 Setup

In existing works [1], [2], the experiments were conducted using the original English schema. Regardless of the language of the questions the same schema was used. This results in experiments column/table and question match is for the largest part absent. We have to clarify that the framework encodes all schema information, such as columns, tables, foreign keys, for each question, independent of the column/table question match.

We introduce a new approach by also translating the column and table names. Specifically, the tables.json file, which contains the schema details, includes four essential attributes among others.

- table_names_original: Contains the original names of tables.
- table_names: Contains curated versions of table names.
- column_names_original: Contains the original names of columns.
- column_names: Contains curated versions of column names.

We choose to translate the table_names and column_names, which are the properties that used for schema linking, while keeping all other fields unchanged to maintain their original values. This ensures that the predicted SQL queries, reference the correct tables and columns. We will refer to this schema as **gr-schema**. We will also refer to the original version of the file simply as **schema**.

All the scores presented in this paper are extracted by inferencing the **dev set**. This is done as the project deadline did not allow us to translate the **test set**. It's really important to keep in mind that both the dataset used for train and the dataset used for inference **use the same databases**. This fact greatly improves the inference scores, as opposed to using a database not present during the training.

During the development phase, we found out that the suggested lemmatizer performed 10% worse at table linking, when compared with the Stanza lemmatizer. The exact percentages are shown in Table 5.

Lemmatizer	**Dataset**	**Schema**	**Column Match**	**Table Match**
Simplemma	Mini Spider	schema	0.987	0.873
Stanza	Mini Gr Spider	schema	0.0	0.0
Simplemma	Mini Gr Spider	gr_schema	0.89	0.662
Stanza	Mini Gr Spider	gr_schema	0.885	0.763

Table 5: Percentage of Linked Questions.

We chose to train models using both lemmatizers to observe how the difference in table/column linking coverage affects the total inference scores.

3.1.2 Results

RAT-SQL allows the use of different large multilingual models. Due to hardware restrictions, all of the experiments presented were performed using **mT5-base** [7].

In order for our measures to be comparable, we needed a **baseline**. To achieve this, we trained the RAT-SQL Model with the *Mini Spider* dataset for 10000 steps using the original schema. We will refer to this model as **baseline**. The inference results achieved by this model on the *Mini Spider dev* can be shown in the table bellow:

Lemmatizer	**Checkpoint**	**Dataset**	**Schema**	**E**	**M**	**H**	**Ex**	**All**
Simplemma	8460	Mini Spider	schema	0.808	0.805	0.722	0.560	0.736

Table 6: Inference Results of RAT-SQL using Mini Spider.

The first set of experiments includes training the model with the Mini Gr Spider dataset, using different combinations of lemmatizers and schemas. It should be noted that both the English and Greek models were trained using the same batch size and max steps:

- Batch Size: 4
- Max Steps: 1000

The aim of these experiments was to observe:

- The effect of Schema Linking, which is directly correlated with the lemmatizer of choice.
- mT5 model's performance when faced with language that's not English.

Lemmatizer	Checkpoint	Dataset	Schema	E	M	H	Ex	All
Simplemma	9260	Mini Gr Spider	schema	0.913	0.775	0.467	0.476	0.697
Stanza	9680	Mini Gr Spider	gr-schema	0.783	0.750	0.667	0.476	0.687
Stanza	9970	Mini Gr Spider	schema	0.739	0.800	0.600	0.429	0.677

Table 7: Inference Results of RAT-SQL using Mini Gr Spider.

The model with the highest score is the one using the Simplemma lemmatizer. We will refer to this model as **Optimal Greek**. We can see that all models have very similar performance on the *dev set*, independent of the schema used.

The total scores of the models trained on the Mini Gr Spider are lower that their English counterpart. This is expected due to the nature of the multilingual model and the accuraccy of the embeddings produced for each language.

It's apparent that column/table matching did not have a major role in the accuracy of these tests. The reason is that that both datasets use the same databases. Even though column/table matching did not work with the original schema, RAT always encodes the information of the whole database along with each question. Given that the connections in the Spider dataset are not consider difficult the model manages to correctly find the links from the available information. As a result, during the training process the model also became accustomed with the databases and managed to predict the questions of the *dev set*.

It is our understanding that the importance of proper schema linking will become apparent when inferencing with the translated *test set*, as it uses different databases, unknown to the model. In that context the model will have to relly more heavily on the map between questions and tables/columns.

The second set of experiments consists of different inferences on the baseline and the best performing Greek model, *Optimal Greek*. With recent advancements in the area of large multi-purpose LLMs, the main question that rises is if it is still necessarily to manually translate such datasets and train language specific models. The two most common alternatives of this approach are:

1. Train a multilingual model only in English and inference it directly in the target language.
2. Translate questions from any language to English using LLMs, inference a model trained on English data with these machine translated questions.

To address this alternatives we have performed a series of no-shot experiments. We used both the *Baseline* model and the *Optimal Greek* and inferenced them it with different datasets.

Model	Checkpoint	Dataset	E	M	H	Ex	All
Baseline	8460	MT Gr to En	0.557	0.585	0.556	0.440	0.545
Baseline	8460	Gr	0.231	0.195	0.167	0.120	0.182
Optimal Greek	9260	MT En to Gr	0.769	0.707	0.500	0.400	0.618
Optimal Greek	9260	En	0.769	0.537	0.333	0.320	0.509

Table 8: No Shot Results on Best Performing Models.

The notation of the table 8 is the following:

- MT Gr to En: Machine Translation of the Greek questions to English.

- Gr: The Greek questions found in the dev.json file of Mini Gr Spider.
- En: The original English questions found in the dev.json file of Mini Spider.

The worse performance was given by inferencing a model in a different language than the one it was trained on. By taking a look at the train statistics for mT5 [7], we can see that 3,928,733,379 English examples were used as opposed to only 68,577,376 Greek examples. It is thus safe to assume that this model performes better when faced with English text rather than Greek. This deduction, also explains the difference between the baseline model and the best Greek one.

Models trained with the machine translated data also performed worse than their counterparts. Despite the added context provided to the LLM for a most accurate translation it failed to provide comparable results with the human curated dataset.

The decline in performance can primarily be attributed to the polysemy of the Greek language and the failure of Language Learning Models to fully grasp the context of the question. With the provided database information, the ideal translation should align with the specified database tables and columns. However, this has not been accomplished to a satisfactory level. The human translation process proves to be still necessary.

3.2 DAIL-SQL Experiments

The DAIL [5] framework relies on large open or closed source LLMs for the Text-to-SQL task. It provides several options for the representation of the information leading to intricate prompts aimed at maximizing the performance of LLMs. The main areas of focus are:

- Schema Linking: Same logic as the RAT-SQL schema linking.
- Question Representation: Whats the best way to represent the Question along with the database information in a prompt that can be easily understood from the LLM.
- Example Selection: How can we find the most relevant examples from the train set given an unknown question.
- Number of Examples: How the number of included examples affects the overall performance of the model.

We will refer to Question Representation as **QA** and Example Selection as **ES** for the purposes of this paper.

For the *Schema Linking* the **Stanford Core-NLP** [11] lemmatizer is used.

For the *Example Selection*, text embeddings are generated for the questions of the *train set* and for the given question, using the **paraphrase-multilingual-MiniLM-L12-v2** [6] sentence transformer model. Then, one of the available distance metrics is applied and the most similar examples are selected.

After setting these options, DAIL first generates a documents with prompts for all the questions of the dev set. Then it queries the model with the prompts, cleans and saves the generated SQL.

Evaluation is not a part of the DAIL framework. As such, in order to evaluate the generated SQL, we used the test suite evaluation metric [12]. This testing suite was published by the creators of the Spider Dataset and calculates two main metrics:

- Exact Match: One-to-One match between produced SQL and the correct one
- Execution Match: Two queries are regarded as correct if they return the same results when executed.

This supervised fine-tunning method proved to be the best, taking the first place at the Spider leader-board with an impressive score of 0.866.

3.2.1 Setup

DAIL, same as RAT-SQL, was designed with the English Language in mind. As such, we had to replace *Stanford Core-NLP* [11] with *Stanford Stanza* [10] Lemmatizer, since it also supports Greek. We also had to replace the *bert-base-nli-mean-tokens* sentence transformer with a multilingual one. We chose the **paraphrase-multilingual-MiniLM-L12-v2** sentence transformer, as it's the most commonly used for the greek language.

3.2.2 Results

We mentioned that the DAIL framework can work with a number of models such as GPT-4, DAVINCI and even open source LLMs like Llama2 and CodeLlama. When deciding which model to use, we chose to go with **Gpt-3.5-Turbo** since it's the most powerful model we had access to. As such, all the tests bellow are executed on this model.

DAIL provides 18 different *question representation* types. Due to time limitations we chose to proceed with the *SQL* question representation for this paper, at it's the suggested representation in DAIL's repository[4]. The form of a question using the SQL representation method can be seen in Figure 2

```
/* Given the following database schema: */
CREATE TABLE "employee" (
"Employee_ID" int,
"Name" text,
"Age" int,
"City" text,
PRIMARY KEY ("Employee_ID")
)
CREATE TABLE "shop" (
"Shop_ID" int,
"Name" text,
"Location" text,
"District" text,
"Number_products" int,
"Manager_name" text,
PRIMARY KEY ("Shop_ID")
)
CREATE TABLE "hiring" (
"Shop_ID" int,
"Employee_ID" int,
"Start_from" text,
"Is_full_time" bool,
PRIMARY KEY ("Employee_ID"),
FOREIGN KEY (`Shop_ID`) REFERENCES `shop`(`Shop_ID`),
FOREIGN KEY (`Employee_ID`) REFERENCES `employee`(`Employee_ID`)
)
CREATE TABLE "evaluation" (
"Employee_ID" text,
"Year_awarded" text,
"Bonus" real,
PRIMARY KEY ("Employee_ID","Year_awarded"),
FOREIGN KEY (`Employee_ID`) REFERENCES `employee`(`Employee_ID`)
)
/* Answer the following: Find the districts in which there are both shops selling less than 3000 products and shops selling more than 10000 products. */
SELECT
```

Figure 2: SQL Question Representation.

Given this representation we run all example *selection methods* to compare the *Example Quality* metric, produced during the prompt generation step. This metric is calculated by getting the jaccard distance between the *SQL Skeleton* of the SQL statement corresponding to the question from the *dev set* and the *SQL Skeleton* of the selected example. If the examples are more than one, a mean is calculated. The exact scores are displayed in Table 9, where EQ indicates the example quality metric.

The *example selection methods* can be seperated into three distinct categories:

1. *Random*: Select the examples at random.

2. *Question Similarity Selection*: Embed the questions using a pre-trained language model. Then calculate the distance between the embeddings using a distance metric such as cosine or euclidean distance. Finally select the K N most similar ones. COSSIMILAR, EUCDISTANCE, EUCDISTANCETHRESHOLD belong in this category.

3. *Masked Question Similarity Selection*: Eliminate the negative influence of domain-specific information by replacing table names, column names, and values in all questions with a mask token. Then calculate the similarity as described in the previous category. EUCDISQUESTIONMASK and EUCDISSKLSIMTHR belong in this category.

[4]https://github.com/BeachWang/DAIL-SQL

4. *Query Similarity Selection*: Employ a preliminary model to generate an SQL for a given question. Then encode the generated SQL and SQL from examples into binary discrete syntax vectors according to their keywords. Select the N questions with the most similar queries

Since the final category depends on another preliminary model, we did not take it into account for this project.

Dataset	**QR**	**ES**	**EQ**
Mini Spider Dev	SQL	COSSIMILAR	0.576
Mini Gr-Spider Dev	SQL	COSSIMILAR	0.575
Mini Spider Dev	SQL	RANDOM	0.437
Mini Gr-Spider Dev	SQL	RANDOM	0.437
Mini Spider Dev	SQL	EUCDISTANCE	0.572
Mini Gr-Spider Dev	SQL	EUCDISTANCE	0.579
Mini Spider Dev	SQL	EUCDISTANCETHRESHOLD	0.0
Mini Gr-Spider Dev	SQL	EUCDISTANCETHRESHOLD	0.0
Mini Spider Dev	SQL	EUCDISSKLSIMTHR	0.885
Mini Gr-Spider Dev	SQL	EUCDISSKLSIMTHR	0.874
Mini Spider Dev	SQL	EUCDISQUESTIONMASK	0.666
Mini Gr-Spider Dev	SQL	EUCDISQUESTIONMASK	0.538

Table 9: No Shot Results on the Baseline Model

We decided to use the Euclidean Distance of the SQL Skeletons, or **EUCDISSKLSIMTHR** as it provided the best *Example Quality* scores for both languages. An illustration of the questions masked by this method is displayed in Figure 3. These questions are then embedded, and their similarity is computed.

```
find the <mask> in which there are both <mask> selling less than <unk> products and <mask> selling more than <unk> products
```

Figure 3: SQL Question Representation.

Same as the RAT-SQL case, since we are using the Mini Spider dataset, we need a new base line. In order to get the base line we inferenced Mini Spider with the DAIL framework and got the following results:

Dataset	**Match Type**	**E**	**M**	**H**	**Ex**	**All**
Mini Spider	Exact Match	0.885	0.634	0.333	0.320	0.573
Mini Spider	Execution Match	1.000	0.756	0.667	0.440	0.727

Table 10: DAIL Inference with Mini Spider.

It's evident that the strength of this model lies in *Execution Match* rather than Exact Match. This means that the produced query may ber different, but produces the correct results.

The following results are generated by inferencing the DAIL framework with the Mini Gr-Spider Dataset.

Dataset	**Match Type**	**E**	**M**	**H**	**Ex**	**All**
Mini Gr-Spider	Exact Match	0.731	0.683	0.500	0.400	0.600
Mini Gr-Spider	Execution Match	0.885	0.756	0.778	0.560	0.745

Table 11: DAIL Inference with Mini Gr-Spider.

We can see that in this case, inferencing with *Mini Gr-Spider* provides *marginally better results, when compared with the baseline model.* This behaviour is rather unusual as it deviates from the expected drop we have observed in all cases during experimentation with RAT-SQL.

Considering that a black-box LLM is the main component of this implementation, an attempt to interpret the results is warranted. Firstly, it's plausible to suggest that the LLM's performance gap when processing either English or Greek data is relatively minor, when compared with *mT5*. Secondly, given that Gpt-3.5-Turbo stands as one of the best models in the area, it can deliver robust performance even when the examples provided are less than ideal. In essence, it has the capability to compensate for any less-than-optimal embeddings generated by the *paraphrase-multilingual-MiniLM-L12-v2* sentence transformer, which lead to subpar examples.

Furthermore, Greek is known for its linguistic richness, which allows it to express and convey meanings more effectively than English. This implies that more relevant examples could be chosen to be included in the input prompt for the question inference. This is of course an assumption that needs further investigation.

4 Conclusion and Feature Work

This paper introduced the Gr-Spider dataset, the first Greek dataset for Text-to-SQL applications. The dataset was created by translating the English Spider dataset into Greek, using Large Language Models. We verified and post-edited the translations, then evaluated the dataset using advanced baseline Text-to-SQL models like RAT-SQL and DAIL-SQL, enhanced with Greek language support. We conducted experiments, analyzed the results and highlighted the importance of multilingual models in overcoming linguistic differences in schema and SQL structure. Our efforts aimed to explore the application of SQL in languages beyond English, showcasing the potential for Greek language processing in structured query tasks.

Feature work will include *(1)* completing the entire translation for both train and test data, *(2)* training RAT-SQL with the large version of the selected multilingual pre-trained models while utilizing specific stemmers for Greek [5] and *(3)* evaluating different configurations in DAIL-SQL to determine the best for the Greek dataset.

References

[1] L. Dou, Y. Gao, M. Pan, D. Wang, W. Che, D. Zhan, and J.-G. Lou, "Multispider: towards benchmarking multilingual text-to-sql semantic parsing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 12 745–12 753.

[2] S. Almohaimeed, S. Almohaimeed, M. Al Ghanim, and L. Wang, "Ar-spider: Text-to-sql in arabic," in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024, pp. 1024–1030.

[3] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *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.

[4] P. Shi, P. Ng, Z. Wang, H. Zhu, A. H. Li, J. Wang, C. N. dos Santos, and B. Xiang, "Learning contextual representations for semantic parsing with generation-augmented pre-training," 2020.

[5] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-sql empowered by large language models: A benchmark evaluation," *CoRR*, vol. abs/2308.15363, 2023.

[6] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[7] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[5] https://github.com/skroutz/greek_stemmer

[8] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7567–7578.

[9] A. Barbaresi, "Simplemma," Jan. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7555188

[10] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[11] Stanford. (2011) Corenlp. [Online]. Available: http://nlp.stanford.edu:8080/corenlp/

[12] R. Zhong, T. Yu, and D. Klein, "Semantic evaluation for text-to-sql with distilled test suite," in *The 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020.