PAUQ🕷: Text-to-SQL in Russian

Daria Bakshandaeva[1,*] **Oleg Somov**[2,3,*], **Ekaterina Dmitrieva**[4,5,*],
Vera Davydova[5], **Elena Tutubalina**[4,5,6]

[1]University of Helsinki, [2]SberDevices, [3]MIPT, [4]HSE University, [5]Sber AI, [6]AIRI
Correspondence: pauq.sql@gmail.com

Abstract

Semantic parsing is an important task that allows to democratize human-computer interaction. One of the most popular text-to-SQL datasets with complex and diverse natural language (NL) questions and SQL queries is Spider. We construct and complement a Spider dataset for the Russian language, thus creating the first publicly available text-to-SQL dataset in Russian. While examining dataset components—NL questions, SQL queries, and database content—we identify limitations of the existing database structure, fill out missing values for tables and add new requests for underrepresented categories. We select thirty functional test sets with different features for evaluating the capabilities of neural models. To conduct the experiment, we adapt baseline models RAT-SQL and BRIDGE and provide in-depth query component analysis. Both models demonstrate strong single language results and improved accuracy with multilingual training on the target language. In this work, we also study tradeoffs between automatically translated and manually created NL queries. At present, Russian text-to-SQL is lacking in datasets as well as trained models, and we view this work as an important step toward filling this gap.

1 Introduction

Semantic parsing (SP) is the task of transforming a natural language (NL) utterance to a formal machine-understandable representation. Such representations include a variety of forms: from parsing linguistic features to generating a code in a specific programming language. In this paper, we focus on a SP subtask – mapping of NL questions to Structured Query Language (text-to-SQL).

Machine learning systems which are based on supervised learning and aimed at solving text-to-SQL, continue to appear and evolve actively (Kim and Lee, 2021; Cai et al., 2021; Gan et al., 2022). The key ingredient in the process of training is the data: a parallel corpus of NL questions and corresponding SQL queries along with the databases. The majority of text-to-SQL datasets consist of questions and database contents written in English. As a consequence, the development of such models has been limited to the English language. Spider dataset (Yu et al., 2018) remains one of the most popular benchmarks in the field due to the variety of domains and complexity of the questions.

This paper presents the Russian version of the Spider: PAUQ[1], the first text-to-SQL dataset in Russian[2]. In PAUQ, all three components have been modified and localized: the NL questions, the SQL queries, and the content of the databases. During this in-depth work, we discover several limitations of the original Spider and propose ways to overcome them. The new version of Spider database collection is presented. Apart from being bilingual, it is more complete – in order to use not only exact match as the evaluation metric but also execution accuracy, avoiding false-positive results. We also complement the dataset with the new samples of underrepresented types, including questions regarding columns with binary values, columns containing date and time values, and the ones that have a fuzzy and partial match with the database content.

We adapt and evaluate on PAUQ two strong ML models relating to different types of architecture: RAT-SQL (grammar-based) and BRIDGE (sequence-to-sequence). We compare the performance of these models in terms of component understanding of SQL and schema linking. Our evaluation demonstrates that both models present strong results with monolingual training and improved accuracy in a multilingual set-up.

[*] These authors contributed equally to this work. The order of author names was randomly determined.

[1] <u>P</u>ioneer d<u>A</u>taset for r<u>U</u>ssian text-to-S<u>Q</u>L

[2] https://github.com/ai-spiderweb/pauq

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 2355–2376
December 7-11, 2022 ©2022 Association for Computational Linguistics

Another important result of this work is the development of functional test sets, subsets of samples in English and Russian with particular features. They can be used by other researchers to evaluate the models' performance on particular types of questions – thus, in a more precise and informative way. Our experiments with machine-translated data suggest that in terms of resources expended and model performance, it could be beneficial to use machine translation for easy cases (such as questions not containing values) and manual translation for questions that imply referring to the database values. We regard this work as an important step in Russian text-to-SQL. Furthermore, we hope that our contributions will help other researchers in adaptation of Spider to other languages.

2 Related Work

Text-to-SQL is a topic that has been actively studied in recent years with a number of datasets and benchmarks (Hemphill et al., 1990; Zelle and Mooney, 1996; Ana-Maria Popescu and Kautz, 2003; Li and Jagadish, 2014; Navid Yaghmazadeh and Dillig, 2017; Victor Zhong and Socher, 2017). The goal of the Spider benchmark is to develop NL interfaces to cross-domain databases. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. In Spider 1.0, different complex SQL queries and databases appear in train and test sets. To do well on it, systems must generalize well to not only new SQL queries but also new database schemas.

There have been several attempts to adapt the Spider dataset to other languages. The first one is Chinese Spider (CSpider) (Min et al., 2019) with original questions from the Spider dataset translated from English into Chinese. In this case, the translation is not literal: a specific cultural localization of some values was conducted. The principles of this localization, however, were not explicitly stated. Authors also compare results obtained by the models trained on the machine-translated questions with the results of the models that were trained on NL queries translated manually and subjected to localization. The study shows that models trained on human translation significantly outperform machine-translation-based. This fact highlights the idea that adaptation of English datasets to other languages requires professional translation and localization. However, databases content in CSpider was not complemented by new localized values, so its impossible to make a complete evaluation of the obtained results.

Another two specific versions of the Spider dataset are Vietnamese (Nguyen et al., 2020) and Portuguese (José and Cozman, 2021). In the case of Vietnamese, not only NL questions were translated, but also the database schema, including table and column names along with values in SQL queries. Translated values, unlike those of Chinese Spider, were not localized; they also were not added to the databases content. In line with the experimental setup of research on CSpider, a comparison between manually-translated and machine-translated versions of the Vietnamese dataset was conducted. In Portuguese Spider only NL questions, excluding database values, were translated, thus no modifications affecting the database schema or content were made. As can be seen, in all currently existing versions of the Spider dataset, although they differ in principles by which the language adaptation is carried out, not all components of the original dataset are modified: database content is left unchanged and isn't aligned with the questions. It also means that execution accuracy can't be used to measure models' performance, thus only exact match is reported.

3 Dataset

When managing the translation process of the original Spider dataset into Russian, we were guided by a similar experience with the translation into Chinese (Min et al., 2019). After careful review of the Chinese Spider (CSpider), we identified two aspects that were not reflected in it – and which we would like to take into account: the formulation of the principles according to which the original English query has to be modified during translation in order to comply with Russian language and culture; the insertion of the modified values into database content so that models could be evaluated with the execution accuracy correctly. As database content is enriched by Russian values, we should pay regard to another significant issue – unambiguity of the questions. This principle also means that Russian values shouldn't be direct translations of English values as in this case it's not obvious what results we expect to get: should SQL queries be formulated so that English values are taken into consideration or should it be Russian values or, perhaps, a combination of both languages. For real-world databases in Russia, just like in Brazil, according to (José and Cozman, 2021), it is common to have English table and column names although the content can be in Russian. For this reason, all table and column names in SQL queries and database schemas remain in English.

The main principle of translation is the following: if there are only tables/columns names in an SQL query and no values, such query and corresponding text question are not subjected to localization or modification – they are just translated; in case an SQL query contains values, this query as well as corresponding text question are changed on the basis of several criteria foremost among which is the following: Russian values should not be literal translations of English ones, but rather should be close analogues that are at the same time absent in the original database content. Principles of values localization could be find in Appendix A.

The translation from English to Russian is undertaken by a professional human translator. The original Spider dataset consists of 10,181 samples, however the test set is closed which means that only 9,691 of them are available for processing. Preliminary work includes dividing questions into those that have values – thus, need modifications – and those that do not. For questions with values, lists with all corresponding column's values from database are obtained in order to ease the selection process of Russian values that do not have English direct analogues in the database. The translator is in advance informed of the SQL query structure and provided with the instruction in which all basic principles of the translation are stated. The translator is encouraged to translate question as closely to the original text as possible and rephrase it as necessary to obtain most natural Russian text. We also strive for a variety of questions, that is why no textual patterns are used.

As regards the updating of the database content, we added all localized/modified values to the corresponding fields. Without this, fair evaluation of the models with the execution accuracy would be impossible as for the questions containing absent values, the ground truth and predicted requests – incorrect as well as correct – would return the same result when executed. In case of binary values, we delete English analogues in order to avoid ambiguity and keep the column's values binary.

A verification of the translated questions and their conformity with the queries, and an updating of the databases are undertaken by 4 computer science students. At first, all databases with corresponding questions in Russian and SQL queries are divided between annotators. Each of the students checks the correctness of a question and a query; if the values in the query has been changed (there is also a special mark indicating it), the annotator adds the new values to the corresponding fields of the database table(s) so that when the database is accessed by the query, the added values are retrieved. After that the very same question and query are cross-checked by another annotator who also makes sure that the database request works as expected.

We made a revision of the original Spider dataset: for those SQL queries that return None or empty list, corresponding values are added to the original databases; databases that were empty (*geo, scholar, yelp, academic, imdb, restaurants*) are filled with values. Further information on these changes could be find in subsection 4.1.

Following research that has been conducted previously on building the Spider dataset for other languages, we also constructed machine-translated (MT) dataset, using Yandex Translate API (Yandex, 2022). The experiments with Chinese and Vietnamese Spider versions show that the performance of the models trained on such data drops compared to that of the models trained on human-translated dataset. The quality of MT dataset certainly depends on the quality of the machine translation model itself; in addition, there is even more problematic issue – the translation of values. Without human revision and database updating, values in NL questions will not match database content in most cases. In order to examine the trade-off between the performance of the models trained on the data and the resources spent to obtain this data, we also experiment with the combined dataset, in which questions without values are machine-translated whereas for questions that include values, manual translation is used to avoid inconsistency (see subsection 5.2).

4 Comparative Analysis

Besides *manual translation of the questions* from the original Spider we (i) add **new samples**, (ii) insert in the tables **new values**, and (iii) compose a suit of **functional test sets**. In order to assess whether the basic properties of the Spider are not violated and whether its limitations are improved, we make a detailed analysis of the datasets and present it in the Appendix B. Here we outline the motivation for the updates, highlight the main differences and define an approach for estimation of the key dataset features like complexity, balance and diversity.

4.1 Component Analysis

The dataset for text-to-SQL task consists of three components: (i) the content of the databases, (ii) NL questions, and (iii) SQL queries. We divide our comparative analysis of Spider and PAUQ into three parts, respectively.

4.1.1 Databases Contents

The number of values in PAUQ **increased by 2%** compared to Spider. The main reasons for this are the following: the requirement that there should be **no ambiguity** between Russian and English elements (so all new Cyrillic values are unique and have no direct English analogues in the column) and the aspiration **to reduce the likelihood of false-positive errors**. The content of the databases and its relations with the NL questions can negatively affect the evaluation of the text-to-SQL models (Kim et al., 2020). One of the common drawbacks of **execution metric** is that it compares results of ground-truth and generated SQL requests and, therefore, tends to give false positive results for accidentally matching returns. The most common case is a null match when a ground-truth SQL request refers to non-existent values and coincides with many inappropriate SQL requests by return.

We analyse such errors made by the models on the original dataset and make several modifications that lead to the following changes in PAUQ compared to Spider: 1. All empty tables are filled with values. 2. The amount of empty columns decreases by more than 2 times (from 86 to 32). 3. All mentioned columns have non-zero sizes. 4. The requests are constructed as follows: a non-zero set of rows in the database corresponds to the set of conditions in a query.

To estimate the maintaining of the dataset **balance** and **diversity**, we calculate the set of quantitative indicators (Appendix B.1). The key results are: 1. The variety of column sizes and content increases. In PAUQ, there is less standard set of column values (in Spider, relatively low diversity is a result of automatic table generation – see Appendix B.1.2). 2. Russian values have the same range of contained token amounts. 3. We found out that in Spider, values longer than 8 tokens are not mentioned in the requests. In PAUQ, a set of requests to the entities of bigger token amounts is included (Appendix B.1.3).

Mapping question parts and database entities is often the major and the most complex part of text-to-SQL translation systems (Kim et al., 2020, Lei et al., 2020, Wang et al., 2021b). Thus, questions containing tokens that are present in several entities at once are particularly difficult. The challenging case is shown in Fig. 1. There are two requests mentioning the token “name”. It is often encountered in the database (as column name and as a value in the column), so it’s impossible simply map text tokens into entities. Hence, a model has to process the context of the request.

Thus, the amount of intersections of entity names within a database determines the potential **complexity of the database set**. We increase the indicator of overlapping entities in PAUQ (Appendix B.1.4), making the entity linking task harder for text-to-SQL models.

4.1.2 Questions

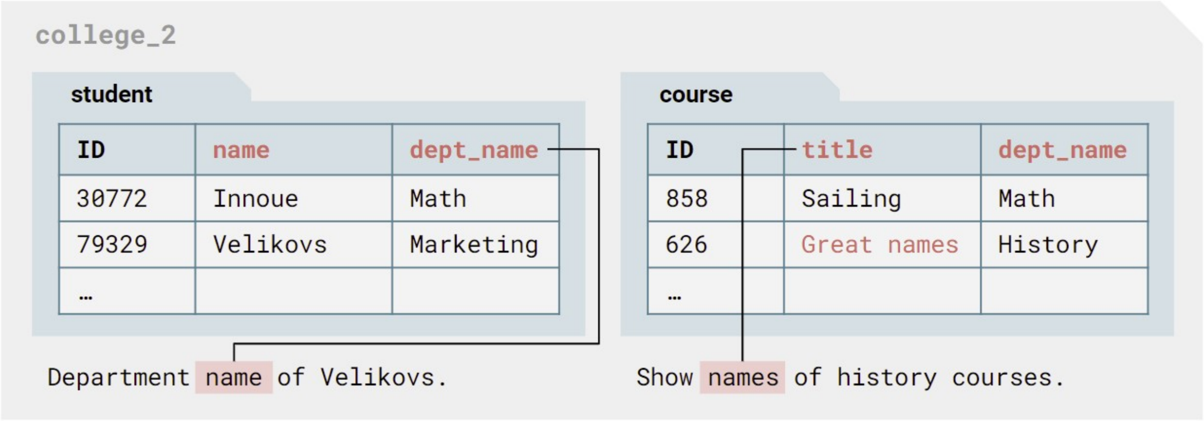
The questions translated into Russian are slightly shorter than the English ones due to peculiarities of the Russian language: see Fig. 2. Yet, the quality properties of the Spider questions are preserved (Appendix B.2.1).

To enlarge the coverage of database entities used in questions we enrich the set of PAUQ questions with new samples related to 15 tables that are not used in the original Spider question set (Appendix B.3). The number of question patterns is also increased by adding questions containing new **request template words** (these are the words that frame the request – imperative verbs, polite words and expressions, etc. – see Appendix B.4).

4.1.3 Queries

During the translation process we “repaired” more than 10 Spider samples (see Appendix B.5). The problems were mostly connected with the ambiguity of the questions and SQL structures.

One of the natural characteristics of the text-to-SQL dataset is the diversity of SQL patterns. We find out some quantitative imbalances in the set of queries (Appendix B.6). We note that these artefacts also exist in other text-to-SQL datasets like



college_2

student

ID	name	dept_name
30772	Innoue	Math
79329	Velikovs	Marketing
…		

course

ID	title	dept_name
858	Sailing	Math
626	Great names	History
…		

Department name of Velikovs.

Show names of history courses.

Figure 1: Example of a database containing different entities with the same names. To translate the questions in which they are mentioned, it is necessary to consider the location of the correct DB content and the context in the request.

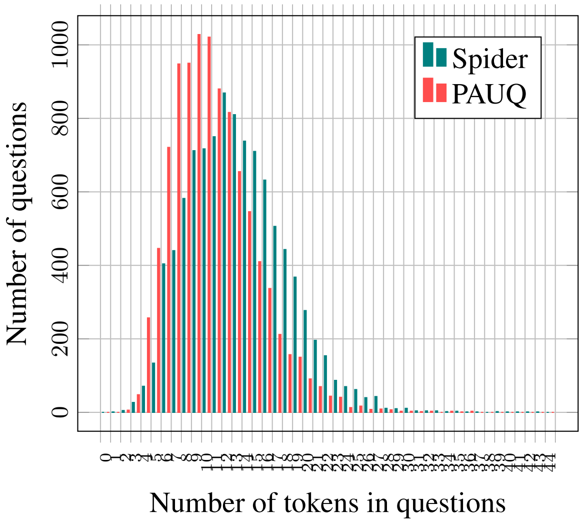


Figure 2: Distribution of question sizes (in tokens).

WikiSQL and ATIS (Finegan-Dollak et al., 2018).

4.2 New requests

Our observations show that some categories of requests are underrepresented in Spider. Therefore, we added 213 **new samples** to PAUQ, divided into five groups to diversify and supplement existing suit and to make the dataset more balanced. These groups are the following: 1. Requests containing **“long” values** (containing more than 4 tokens). 2. Samples containing references to one of the two possible values (which are opposed to each other) from the column (we refer to such columns as **binary columns**; querying this type of values is also different from referring to other types from the semantic point of view. 3. Queries containing **date** or **time** filters. 4. Requests with **“fuzzy”** mentioning of entities (e.g. using synonyms, words reordering, etc.). 5. Queries with **empty** return (which conditions, for instance, don’t correspond to any row of the database). This suite is separated from others to prevent false positive errors on it.

The **details** on new requests can be found in Appendix C.

BRIDGE and RAT-SQL (text-to-SQL models experiments with which are described in detail in the Section 5) achieve 19.2% and 21.1% of exact match accuracy, respectively, being trained on the existing English and Russian data and evaluated on the set of new samples. That is 3 times less than accuracy obtained on all Spider questions (Table 1). These results demonstrate the complexity of added examples.

4.3 Functional Test Sets

Measuring the performance of the models with a single number (e.g. exact match score) is a simple and convenient way. Yet, this makes it difficult to identify model’s weak points and estimate its ability to cope with the hard cases.

The complexity of requests is defined based on the number of SQL components and is divided into four categories: “Easy”, “Medium”, “Hard” and “Extra hard” (Yu et al., 2018). But this division is not universe. In particular, (Lei et al., 2020) shows the example that “is actually difficult as it requires a model to perform logic reasoning”, but is classified as “medium”. SQL-request, corresponding to the question *“What is the total surface area of the continents Asia* *and* *Europe?”*, has a really simple structure: `SUM(surface area) FROM country WHERE continent =`

`«Asia» OR continent = «Europe»`. However, text-to-SQL models struggle to choose the correct logical connective because it differs from the connective used in the question.

One of the most effective approaches to in-depth detailed analysis is the generation of evaluation suits - multiple (probably overlapping) **test sets** (also called **challenge sets**), that assess specific capabilities of a model. Now it applies to the widest range of NLP tasks, e.g., (Röttger et al., 2020) introduced a suite of functional tests for hate speech detection models, (Cho et al., 2021) proposed augmented test sets for the dialogue state tracking (DST) task.

To encourage text-to-SQL researchers to investigate the soft spots of the models, we extracted from our dataset the subsets of samples with different features divided into three classes:

- **Features of NL questions**: (1) use of synonyms for existing database entities (e.g. in the question *"What is the **final station** for the train 56701?"*, a term "final station" refer to the column name "destionation"), (2) atypical size:"short" and "long" questions (the ones for which the ratio of the number of tokens in the query to the number of tokens in the question, has extreme value), (3) questions that contain several different logical structures at once (*"How many students are over 18 **and** do not have allergy to food type **or** animal type?"*), etc. Other features with corresponding examples are presented in the Tables 10, 11, 12 from Appendix C.

- **Features of SQL queries**: (1) nested queries (`SELECT * FROM participants WHERE name in (SELECT id FROM winners)`), (2) queries with several JOIN operations, (3) queries with several aggregations for one column (*"**Maximal, minimal average** temperature last year"*), (4) and, vice versa, several columns with one aggregation (*"Maximal **temperature** and **pressure** last year"*), etc. Details can be found in the Table 13 from Appendix C.

- **Database features**: (1) questions with entities that cause the ambiguity problem for the models (e.g. a request *"Name of user with id 26"* mentions column "Name" from the table "Employers", while there is the column with the same title in the table "Workers"), (2) queries with the binary filters like 0/1, "yes" / "no", ... (A request *"Number of cancelled flights"* filters all flights that have value "No" in the the column "Completed"); (3) mention of the multi-sentence values in the request, etc. Other examples can be found in the Table 9 from Appendix C.

The choice of these particular features is based notably on the analysis of the results described in the evaluation sections of different papers (the references can be found in the Appendix C). The validity of this choice is confirmed by low results obtained on the extracted suit in comparison with the scores obtained on the whole set of questions. Thus, on all functional test sets (excluding "simple" and "extra simple" requests created for comparison reasons) the average exact match accuracy is 0.23 (BRIDGE) and 0.27 (RAT-SQL) while on the full development dataset the results are 0.55 and 0.57 correspondingly. All metric values and the list of test set descriptions are presented in Tables 9, 10, 11, 12, 13 from Appendix.

5 Experiments

In this paper, we provide an empirical evaluation of baseline neural network models for text-to-SQL on our dataset. To date, there are many different models present and there are three major categories of these models (Katsogiannis-Meimarakis and Koutrika, 2021): (i) sequence-to-sequence, (ii) grammar-based, and (iii) sketch-based slot filling. We have analyzed the Spider leaderboard (spi, 2022) and concluded that the sketch-based slot filling method does not stand up to the competition with the other two methods, which leaves us with two other approaches. We kept in mind the following requirements for models during our selection:

1. Both models have different nature of schema encoding, architecture, and decoding processes. This way we can test the method's performance on new datasets and languages;

2. Since we are also complementing Spider with the new database content we want to explore how it affects the result scores. Therefore, models must work with database content;

3. The models have identical encoder models for us to exclude language modeling bias.

Train Data	Infer Data	BRIDGE	RAT-SQL
Spider	Spider	0.60 / 0.60	0.66 / 0.63
MT RU	PAUQ	0.49 / 0.40	0.46 / 0.37
MT + HT RU	PAUQ	0.52 / 0.49	0.52 / 0.5
PAUQ	PAUQ	0.52 / 0.48	0.51 / 0.49
RU + ENG	Spider	0.68 / 0.65	0.66 / 0.65
RU + ENG	PAUQ	0.55 / 0.50	0.57 / 0.53

Table 1: Exact match (left) and execution accuracy (right) results.

	MT + HT	PAUQ	RU + ENG
easy	0.60 / 0.61	0.65 / 0.57	0.65 / 0.65
medium	0.70 / 0.59	0.71 / 0.66	0.70 / 0.66
hard	0.77 / 0.73	0.77 / 0.73	0.81 / 0.80
extra	0.83 / 0.78	0.79 / 0.78	0.82 / 0.82

Table 2: Component errors intersection for BRIDGE (left) and RAT-SQL (right).

According to our requirements, we utilized two popular Spider models – RAT-SQL (grammar-based) (Wang et al. 2021b) and BRIDGE (sequence-to-sequence) (Lin et al. 2020). RAT-SQL is an encoder-decoder framework that uses relation-aware transformer within the encoder to model alignments between database schema and content and question tokens. The decoder of the model is tree-structured and generates abstract syntax tree in the context-free SQL grammar. BRIDGE, in turn, utilizes database schema and content as input to the model. It has an encoder-decoder architecture with the pointer-generator network using beam-search. The model generates queries in execution-guided order.

Within the experiments, we seek to answer to following research questions (RQs): **RQ1** What is the performance difference between models which were trained on English, or on various Russian datasets, or in multilingual setup (English and Russian)? **RQ2** Do we need to use qualified human translation while adapting the original dataset to different language (in our case, Russian)? **RQ3** How does query component prediction differ depending on the language?

5.1 Overall Results

We evaluated performance of our models trained on 5 different types of NL data: 1. human-translated Russian (**PAUQ**); 2. machine-translated Russian (**MT**); 3. machine-translated Russian for queries without values and human-translated Russian for queries with values (**MT + HT**); 4. original English (**Spider**); 5. union of Russian (PAUQ) and English (Spider) datasets (**RU + ENG**).

The details on hyperparameters are presented in Appendix E. mBERT base language model is used as an encoder for all experiments on Russian data and in multilingual setup. For English models, we have replaced the base encoder BERT-large with BERT-base to get comparable results. For evaluation of Russian models (PAUQ, MT, MT + HT) we used PAUQ dev set and for English models – original Spider dev set. For all sets during training and testing, revised and complemented databases from PAUQ were used. For metrics calculation, we utilized original Spider evaluation script provided at https://github.com/taoyds/spider. Since we are working with multi-language data, we explored how training simultaneously on PAUQ and Spider would affect evaluation metrics. The experiment was conducted in the following way: we merged PAUQ with original English Spider NL – SQL pairs and used revised databases from PAUQ as target databases. After training the models on this data, we measured the performance on Spider development set and PAUQ independently. In Table 1 we present the obtained results. The answer to **RQ1** is that the English model on BERT-base has better performance than all Russian models on mBERT-base. However, training systems on combined English and Russian dataset increases performance on both languages.

5.2 Human vs Machine Translation Comparison

To answer **RQ2**, we have compared the performance of the models trained on manually translated PAUQ dataset with those trained on machine-translated (MT) or combined (MT + HT) data. Experiments showed that MT + HT models perform on par with the models trained on PAUQ. Hence, in order to get a qualitative dataset in Russian, it is enough to resort to manual translation only for queries containing values. If we compare these results with those of the models trained on MT data, we see a decrease in execution accuracy which corresponds to weak value matching ability of such models. This happens because MT data doesn't correspond well with the database values. This observation is supported by train value match accuracy: for PAUQ and MT + HT models it has 0.69 error rate and for MT 0.94.

5.3 Structure & Logic Understanding Error Analysis

We focus on structure and logic understanding component match errors to answer **RQ3**. Such components correspond to the core logic of the question projected on the schema and SQL syntax in the expected query. Selected components are *WHERE* (operations only), *SELECT* (aggregations only), *GROUP (no having), ORDER, AND/OR, IUEN, JOIN.* We use the following evaluation setup: for every predicted query in the development dataset extract structure components that are incorrect according to the component set match metric. Then we count all errors per component and scale them by the total amount of components in the development dataset to get the distribution of errors per component. We analyze and compare predictions of MT + HT, PAUQ, RU + ENG, and Spider models. MT + HT, PAUQ, RU + ENG models are evaluated on PAUQ development set, Spider models – on the original development set. These metrics are presented in Table 3. As we can see, for both languages models, on average, most often make incorrect predictions on *JOIN*, *ORDER* and *GROUP* components. English models, however, seem to perform better on these components while on other components (*SELECT, WHERE, AND/OR, IUEN*) the difference between models is not drastic.

We have also explored how structure and logic model's errors on Russian and English data differentiate based on the complexity of the Spider split queries in Table 2. The analysis of how predicted structure & logic components errors on PAUQ development set intersect with those on Spider development set reveals that the more complex the queries are, the more structure & logic component errors of models trained on different languages begin to intersect with each other.

5.4 Schema Linking Error Analysis

Along with semantic errors analysis, we intend to evaluate how the models trained on Russian queries cope with substituting the necessary database schema elements into the query.

To evaluate this we extracted gold and predicted database schema elements from queries generated by BRIDGE and RAT-SQL and calculated error rate per each element. These elements refer to query components such as *SELECT* (without aggregations), *WHERE* (value components)*, FROM, GROUP BY, ORDER BY*.

Table 4 illustrates the error rates related to particular components for BRIDGE and RAT-SQL models predictions from development subsets of each of four datasets (Spider, MT + HT, PAUQ, EN + RU). This percentage is calculated relative to the total number of queries from the development subset containing such components. Our study shows, that both models have very similar schema linking errors distribution. The average number of errors related to entity linking is much higher than the average number of errors related to structure and logic understanding. English models perform better results on schema linking – apparently, BERT encoder often fails with linking entities. All models perform well on database tables names prediction: FROM components, 8% of errors. The models show lower results on column names prediction: SELECT, WHERE (columns), ORDER BY, GROUP BY components, 23% of errors for Russian data sets. Predictions made on the English set are better in terms of column names linking, they contain only 18% of such errors.

Value Match The most notable difference is related to WHERE (values) component – it is the most difficult part for all English and Russian models. However, both models trained on PAUQ, MT+HT, RU+ENG make mistakes twice as often as those trained on Spider. Moreover, Russian MT model predictions fail to link value entities at all: since translation is made automatically, there are often no such entities in the database. That is one of the main reasons for using dataset created by annotators instead of less labour-intensive machine translation. BRIDGE model trained on English Spider performs better results on database values match than RAT-SQL because it augments model input with automatically extracted database cell values mentioned in the question to align the schema components with the NL utterance using the fuzzy match algorithm. However, since Russian is a highly inflectional language, fuzzy match works correctly in a much smaller amount of cases than in English. Thus, for Russian, both models perform reasonably poor on the value matching .

6 Conclusion

In this paper, we have presented the first public text-to-SQL dataset for the Russian language.

	SELECT	WHERE	JOIN	AND/OR	IUEN	ORDER	GROUP
Spider	0.04 / 0.04	0.04 / 0.04	0.35 / 0.27	0.10 / 0.10	0.10 / 0.11	0.18 / 0.18	0.26 / 0.21
MT + HT	0.03 / 0.05	0.04 / 0.04	0.39 / 0.34	0.10 / 0.09	0.11 / 0.12	0.24 / 0.23	0.32 / 0.33
PAUQ	0.04 / 0.04	0.05 / 0.06	0.32 / 0.38	0.08 / 0.10	0.11 / 0.11	0.26 / 0.23	0.33 / 0.33
RU + ENG	0.03 / 0.04	0.03 / 0.04	0.34 / 0.34	0.08 / 0.10	0.09 / 0.10	0.23 / 0.20	0.29 / 0.27

Table 3: Component error rate for BRIDGE (left), RAT-SQL (right) per structure and logic components.

	SELECT (w/o agg.)	WHERE (columns)	WHERE (values)	FROM	GROUP BY	ORDER BY
Spider	0.20 / 0.18	0.17 / 0.16	0.23 / 0.31	0.08 / 0.07	0.20 / 0.21	0.16 / 0.13
MT + HT	0.32 / 0.29	0.22 / 0.31	0.57 / 0.57	0.10 / 0.08	0.28 / 0.32	0.35 / 0.29
PAUQ	0.29 / 0.28	0.25 / 0.22	0.55 / 0.57	0.09 / 0.09	0.29 / 0.30	0.22 / 0.23
RU + ENG	0.29 / 0.24	0.28 / 0.28	0.56 / 0.57	0.08 / 0.07	0.29 / 0.24	0.22 / 0.19

Table 4: Schema linking error rate for BRIDGE (left), RAT-SQL (right) per database schema components.

Based on the Spider – large-scale, cross-domain benchmark composed of table interconnected databases and corresponding NL queries. We improved the original Spider by inserting the missing values, correcting errors, and adding new samples of poorly represented types. To answer the stated research questions, we conducted experiments with RAT-SQL and BRIDGE architectures trained on different combinations of Russian and English data. Based on the results obtained, we got the following key observations. Lexical and semantic diversity of questions, queries and table values was improved. Extracted categories of requests with different artifacts in mentioned entities, question formulation and logical structure show model weaknesses and open perspectives for further development. Although Russian and English differ considerably from each other, similar models show similar performance on these languages. As we can see, errors in the structure and logic components are quite similar, but errors related to schema linking are different: it is much harder for Russian models to generalize and match on these components than for English ones. Machine translation works in comparably for queries without values for selected text-to-SQL models, but human translation is needed for queries with values in order to get qualitative data and provide better results. Our experiments show that training on Russian and English languages simultaneously using a multilingual encoder brings a notable increase on both Russian and English development datasets. We mark the development of multilingual models as a prospect for future work.

7 Limitations

As our dataset is an adaptation of the Spider dataset to Russian language, it indeed inherits most of Spider's limitations. First of all, the data is still 'artificial' which means that it was created by a limited number of people specifically for training and evaluating text-to-SQL models, thus it lacks the diversity and complexity of natural data formed by questions that people formulate in order to get the desired information from the database. For instance, the real-world data contain NL queries that require common sense knowledge which can't be extracted directly from the database; ambiguous questions allowing various ways of interpretation that are quite frequent and queries with window functions that make the process easier and more convenient, – all of these aren't included in the Spider dataset, as well as in our. Some of these and other limitations have already been resolved in more recent datasets (Yu et al., 2019; Hazoom et al., 2021), some others we partially fulfill by our functional testsets. Another limitation concerns evaluation metrics – exact match and execution accuracy, which are the most commonly used to evaluate text-to-SQL models performance. However, the first one is too strict and prone to false negative results (Zhong et al., 2020) while the latter is problematic with respect to spurious questions and ambiguous questions (Hazoom et al., 2021). More sophisticated metrics such as proposed in Kim et al., 2020; Hazoom et al., 2021 may be used in future work to adequately evaluate model performance.

8 Ethical Considerations

The presented dataset have been collected in a manner which is consistent with the terms of use of the original Spider, which is distributed under the CC BY-SA 4.0 license. We also used the original evaluation code scripts from Spider repository.

The translation of queries from English to Russian is made by a professional translator; the database changes are made by annotators. All of them received fair compensation (more than the minimum wage in Moscow, Russia). We would like to thank the authors of the Spider for providing access to the original data. We also thank the translator and annotators for their time and effort.

References

2022. Repository for Spider dataset.

Oren Etzioni Ana-Maria Popescu and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 149–157.

Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. Sadga: Structure-aware dual graph aggregation network for text-to-sql.

Hyundong Cho, Chinnadhurai Sankar, Christopher Lin, Kaushik Ram Sadagopan, Shahin Shayandeh, Asli Celikyilmaz, Jonathan May, and Ahmad Beirami. 2021. Checkdst: Measuring real-world generalization of dialogue state tracking performance. *arXiv preprint arXiv:2112.08321*.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*.

Yujian Gan, Xinyun Chen, Qiuping Huang, and Matthew Purver. 2022. Measuring and improving compositional generalization in text-to-sql via component alignment.

Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R Woodward, Jinxia Xie, and Pengsheng Huang. 2021. Towards robustness of text-to-sql models against synonym substitution. *arXiv preprint arXiv:2106.01065*.

Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-sql in the wild: A naturally-occurring dataset based on stack exchange data. *CoRR*, abs/2106.05006.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Marcelo Archanjo José and Fábio Gagliardi Cozman. 2021. mrat-sql+gap: A portuguese text-to-sql transformer. *CoRR*, abs/2110.03546.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD/PODS '21, page 28462851, New York, NY, USA. Association for Computing Machinery.

Donggyu Kim and Seanie Lee. 2021. Self-supervised text-to-sql learning with header alignment training. *CoRR*, abs/2103.06402.

Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to sql: where are we today? *Proceedings of the VLDB Endowment*, 13(10):1737–1750.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-sql. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing.

Qingkai Min, Yuefeng Shi, and Yue Zhang. 2019. A pilot study for chinese SQL semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3650–3656. Association for Computational Linguistics.

Isil Dillig Navid Yaghmazadeh, Yuepeng Wang and Thomas Dillig. 2017. Sqlizer: Query synthesis from natural language. In *International Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM*, pages 63:1–63:26.

Anh Tuan Nguyen, Mai Hoang Dao, and Dat Quoc Nguyen. 2020. A pilot study of text-to-sql semantic parsing for vietnamese. *CoRR*, abs/2010.01891.

Paul Röttger, Bertram Vidgen, Dong Nguyen, Zeerak Waseem, Helen Margetts, and Janet B Pierrehumbert. 2020. Hatecheck: Functional tests for hate speech detection models. *arXiv preprint arXiv:2012.15606*.

Caiming Xiong Victor Zhong and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021a. Meta-learning for domain generalization in semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–379, Online. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2021b. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers.

Yandex. 2022. Yandex translation api site.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019. Sparc: Cross-domain semantic parsing in context. *CoRR*, abs/1906.02285.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411, Online. Association for Computational Linguistics.

A Principles of values localization

- abbreviations: if there is a widely used Russian-language analogue of the abbreviation, then it is translated and changed; if an unknown abbreviation is found, then it is kept in the original form, without transliteration;

- names of films/songs/metrics/events/technics etc.: such names are localized, that is, replaced with an analogue known in Russian culture (which is familiar to a native speaker from the media/culture); however, a name should be kept in English, if this name is used in the media in the original form;

- proper names: personal proper names are substituted by Russian-language analogues; company/brand names aren't translated;

- addresses: the addresses without specifying the country are localized;

- binary values (e.g. "yes/no", "true/false"): such values should be translated into Russian.

B Comparative Analysis

B.1 Database Content

B.1.1 Number of Entities

To quantify the scope of newly added values, we counted the number of different elements of databases.

Since the translation affected only the values in our tables, the number of databases, tables and columns hasn't changed. The total number of values **increased by 2%**.

- **Databases:** 166 entities (88.0% – train set, 12.0% – dev set).

- **Tables:** 876 entities (90.8% – train set, 9.2% – dev set).

- **Columns:** 4503 entities (90.2% – train set, 9.8% – dev set).

- **Values:** increased by 498 490 entities (2 587 unique values) from 23 435 505 to 23 933 995; 531 164 and 533 751 unique values respectively (88.4% – train set, 11.6% – dev set).

B.1.2 Table Sizes

In terms of diversity, the variety of column sizes increased from 122 variants of unique values to 151.

Fig. 3 shows the distribution of column sizes for "small" tables – tables shorter than 25 rows (the biggest table contains 510 437 rows).

An outlier at 15 is the result of automatic table values generation – filling table's columns with the values `{1, 2, ..., 15}`. In PAUQ, this artifact becomes **less** pronounced.

B.1.3 Value Sizes

All English values are preserved in PAUQ. Thus, the longest one also contains **31 tokens** and 213 symbols (*"Et totam est quibusdam aspernatur ut. Vitae perferendis eligendi voluptatem molestiae rem ut enim. Ipsum expedita quae earum*

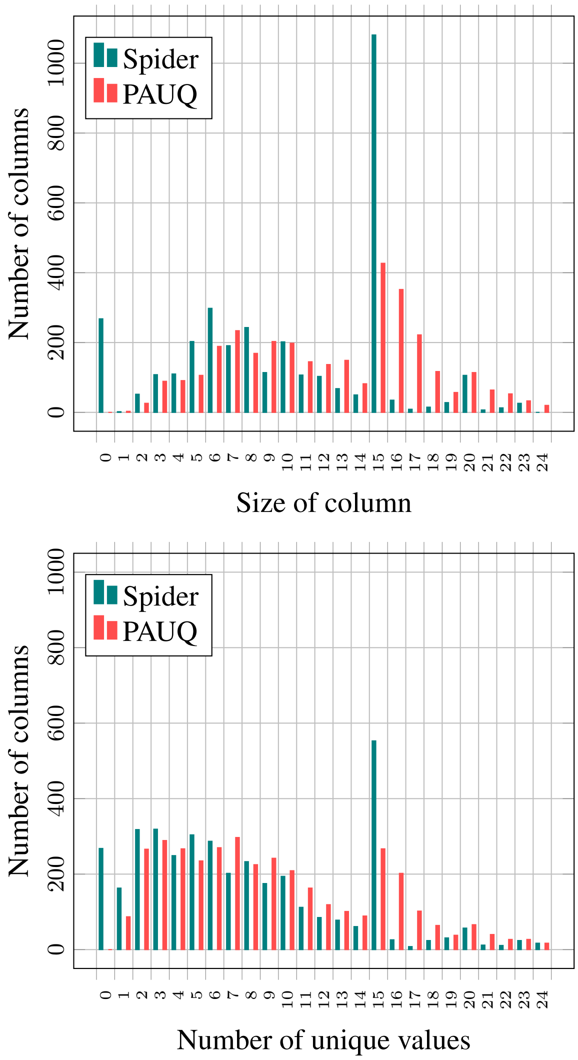


Figure 3: Distribution of column sizes.

unde est. Repellendus ut ipsam nihil accusantium sit. Magni accusantium numquam quod et.", DB `cre_Doc_Tracking_DB`). In Spider, however, the longest entity mentioned in the request consists of just **8 tokens**. In PAUQ, there is a set of requests referring to the entities expressed in bigger amount of tokens.

The length of Cyrillic values varies from 1 to 20 tokens and such values are also mentioned in the requests. The longest one is the following: *"Человеку сложно обидеть бога, – ответил Акинфий Иванович. – Худшее, что может случиться – вы его собой не заинтересуете"* (DB `twitter_1`).

B.1.4 Overlapping Entities

Almost every fourth Spider entity (table name, column name, column value) contains a token from some other element, but only an eighth of them is used in some query. While adding Russian values and new requests we tried **to increase both indicators** – quantity of overlapping and common tokens in questions that can cause ambiguity problem for text-to-SQL models.

For precise analysis we calculate the amount of overlapping tokens found in several entities in Spider and PAUQ. The results are presented in Table 5.

B.2 Questions

B.2.1 Length of Questions

- Average length of natural language query:
  - **English:** 13.2 tokens, 66.4 symbols;
  - **Russian:** 63.8 tokens, 10.6 symbols.
- The longest question:

  "Display the employee number, name (first name and last name), and salary for all employees who earn more than the average salary and who work in a department with any employee with a 'J' in their first name." (45 tokens, DB "movie_1", is the same for both languages).
- The shortest question:
  - **English:**
    "chi" (DB "scholar"). It's interesting that corresponding SQL request is rather long: `SELECT DISTINCT t1.paperid FROM venue AS t2 JOIN paper AS t1 ON t2.venueid = t1.venueid WHERE t2.venuename = "chi"`.
  - **Russian:**
    "Авторы NIPS" (DB "scholar").

B.3 Coverage by Mentions

An important advantage of Spider is a large number of multi-domain databases of different sizes. But not all DB elements are addressed in the requests. Thus, 92.0% of all tables are used, 48.3% of columns and less than 0.1% of all values (see Fig. 4).

We **add new mentions** of 15 tables that are not used in Spider, to PAUQ.

B.4 Request Template Words

A lot of queries contain standard request template words like different imperative verbs with or without affirmative “please”.

	Spider databases	**PAUQ databases**	**Spider queries**	**PAUQ queries**
Number of tokens found in:				
...entities of different types from one DB	7261	8637↑	1538 (21.2%)	1952 (22.6%)
...different **table** names from one DB	227	227	42 (18.5%)	101 (44.5%) ↑
...different **column** names from one DB	1425	1425	156 (10.9%)	215 (15.1%) ↑
...different **values** from one DB	5659	7070↑, of which 464 are Cyrillic	154 (2.7%)	234 (3.3%) ↑

Table 5: Overlapping DB entities

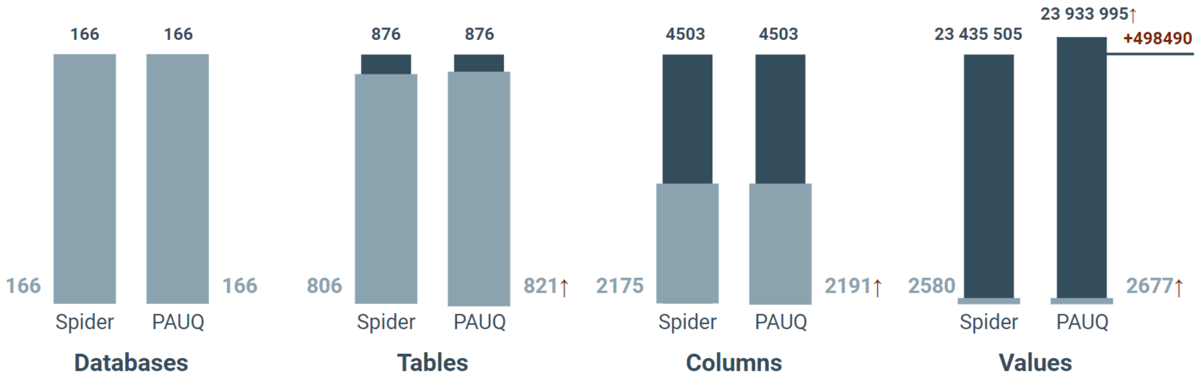


Figure 4: Coverage of database entities by mentions in questions. Left bar in every category corresponds to Spider, right bar – to PAUQ. Dark color is for the entities from the databases, light – for the entities used in the requests.

Some examples:

- ***Please, show** the most common type of ships.*
  - ***Покажи, пожалуйста,** наиболее распространенные типы кораблей.*
- ***Split** the number of killed ships by type.*
  - ***Разбить** количество потопленных кораблей по типам.*

This set is not too diverse, but is slightly expanded in the PAUQ. It is given in Table 6.

B.5 Queries

B.5.1 Corrections

During the translation process we have “repaired” more than 10 Spider samples. The problems were mostly connected with the ambiguity of the questions and logical patterns. E.g. in the query corresponds to the question *“What are the names of all reviewers that have given 3 **or** 4 stars for reviews¿ ‘* should be used union-operator instead of “INTERSECTION“ in the Spiderground truth. List of corrections is on the repository.

B.6 SQL query sets imbalances

In the Spider queries we found several quantatative imbalances:

1. One of the constructions is found one and a half times more often than the next variant (more details in Appendix B.5).
2. Queries are dominated by the aggregation function "COUNT" that is more than half of

Spider:	PAUQ:
What	Какой (-ая, -ое, -ие) / Какова
Find	Покажи (-те, -зать)
How many	Найди (-те, -ти) / Поищи
Who	Где
Which	Кто
Show	Перечисли (-те, -ть)
List	Отобрази (-те, -ть)
Return	Назови (-те)
Order	Сколько
Type	Отортируй (-те, -ть)
Whose	Топ
Give	Упорядочи
Count	Напечатай
Where	Который из
Top	Узнай
Rank	Выведи
Sort	Посмотри
Display	Как
How	Как много
	Детализируй
	Разбей
	Сравни (-те)
	Соответствие
	Нужно (-ы)
	Попробуй (-ы)

Table 6: Lists of request template words in natural language questions sorted by frequency of using in descending order

total amounts.

3. More than half of the requests query only one table.

B.6.1 SQL Query Structures

We replace all entity names, aggregations and filter constructions in the queries and extract 402 different constructions. The most common schemes are:

`"SELECT <column> FROM <table> WHERE <condition>"` (13.7%)

and

`"SELECT <column> FROM <table-1> JOIN <table-2> WHERE <condition>"` (9.0%).

Surprisingly, they are more common than the simplest one: `"SELECT <column> FROM <table>"` (8.3%). This ratio is approximately the same for the training and validation sets.

We extract rare and short structures (with less than 5 components, they occur less than in 1% of samples) and use them in our new samples:

- `SELECT <column> FROM <table> WHERE <ent> ORDER BY <ent> ASC` (0.02%)
- `SELECT <column> FROM <table> WHERE <condition> GROUP BY <column>` (0.02%)
- `SELECT <column-1>, <column-2>, <column-3>, <column-4> FROM <table> ORDER BY <column>` (0.02%)

B.6.2 Joined Tables

During the analysis of joined tables, we found out that in the validation set, the average amount of tables that need to be accessed to answer the NL question, is less than that in the train set (1.7 vs 1.4), as shown in Table 7.

We add to the development set requests which refer to more than 4 tables (in Spider, there is a lack of such questions).

Tables	Share of the total	Train/Dev ratio
1	57.01%	88:12
2	26.01%	87:13
3	11.36%	93:7
4	3.76%	97:3
5	1.63%	96:4
6	0.16%	100:0
7	0.06%	100:0
8	0.02%	100:0

Table 7: Number of tables used in one Spider request

B.6.3 Aggregations

Just for analytical purposes we calculate the amount of aggregation functions used in Spider. The most popular one is COUNT (see table 8). A curious fact is that MAX is requested almost twice as often as MIN.

C New Requests

C.1 Long Values

The presence of “long” values in databases is important, since accessing them in a query is usually

Aggregation	Share of the total
COUNT	65.45%
AVG	11.97%
MAX	10.02%
SUM	6.88%
MIN	5.68%

Table 8: Aggregations used in Spider requests

difficult for the text-to-SQL models. A user probably uses only a part of tokens and changes the order while mentioning them.

The values with the length greater than 8 tokens are not mentioned in the Spider queries. But it is an important case for the models that can predict values and for the systems used in real life applications.

We add 20 requests to the train set and 10 – to the test set – the requests that refer to the values longer then 8 tokens. As in real-world examples key word in question are disordered and not full-matched with entity names.

Example:

*"Name of film about **moose**, **composer** and **dentist**"*

→ "`A Touching Saga of a` **`Composer`** `And a` **`Moose`** `who must Discover a` **`Dentist`** `in A MySQL Convention` from column "`description`" of table "`film`" (DB `sakila_1`).

C.2 Binary Columns

Querying entities from the *columns with the two opposite values*[3] (**binary columns**) is different to querying other types of values from the semantic point of view.

Typically user mentions a value with or without the name of the column:

*"Number of participants from **New York**" ("Количество участников из **Новосибирска**")*, - refer to the value `New York` ("Новосибирск") from the column "`City`".

But when a value from the binary column is accessed, it is often just the name of the column with or without negative particle that is used. The value itself is not mentioned:

*"Number of **registered** participants" (``Количество **зарегестрированных** участников")*, - refer to value "`True`" from column `Is_registered`.

We extract all binary columns from Spider and divide them into five categories:

- "True" / "False" – 8 columns, 8 requests;
- "Yes" / "No" – 12 columns, 11 requests;
- "1" / "0" – 19 columns, 17 requests;
- Gender ("F" / "M", "1" / "0", ``муж" / ``жен", etc.) – 33 columns, 32 requests;
- Antonyms ("Satisfied" / "Unsatisfied" "good" / "bad", etc.) – 17 columns, 13 requests.

The use of values from different categories is slightly different from each other. We found out that the models under consideration show low results on these requests (exact match is 0.09 for BRIDGE and 0.18 for RAT-SQL). Besides, the amount of such samples in the dev set is extremely small. Thus, we decided to enrich the train set with 44 such queries and the dev set – with 8 queries.

C.3 Dates and Times

In Spider there is a wide variety of date and time data. Some examples are shown in Figure 5.

Unfortunately, in only 71 questions within the train set, some date or time information should be used as filters and there are just 4 such samples in the dev set. Since it looks natural to use this type of data in the requests, we added 48 queries (train: 39, dev: 9) that have different formats of date and time information and are not used in Spider.

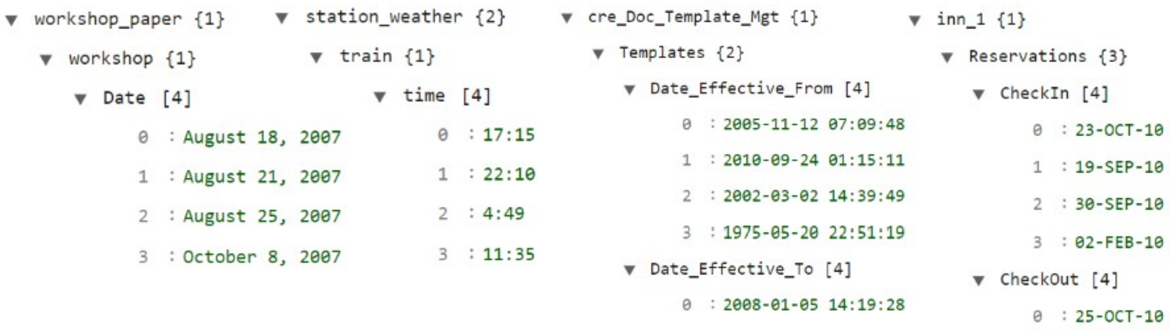
C.4 Fuzzy and partial matching

In Gan et al., 2021 it is stated that "existing text-to-SQL models typically rely on the lexical matching between words in natural language (NL) questions and tokens in table schemas". For this reason it is important to evaluate models on the requests containing nontrivial mention of a value from a database. It is hard to extract all such examples from Spider. Thus, for the evaluation purposes we added 62 new samples which can cause difficulties in the entity linking process. They can be divided into several types:

- **Fuzzy matching**: the use of synonyms or paraphrases for the value names in the question.

  *What is **the final station** for the train 56701?*

[3] Like: "Yes" / "No", "True" / "False", "1" / "0"



```
▼ workshop_paper {1}
  ▼ workshop {1}
    ▼ Date [4]
        0 : August 18, 2007
        1 : August 21, 2007
        2 : August 25, 2007
        3 : October 8, 2007
```

```
▼ station_weather {2}
  ▼ train {1}
    ▼ time [4]
        0 : 17:15
        1 : 22:10
        2 : 4:49
        3 : 11:35
```

```
▼ cre_Doc_Template_Mgt {1}
  ▼ Templates {2}
    ▼ Date_Effective_From [4]
        0 : 2005-11-12 07:09:48
        1 : 2010-09-24 01:15:11
        2 : 2002-03-02 14:39:49
        3 : 1975-05-20 22:51:19
    ▼ Date_Effective_To [4]
        0 : 2008-01-05 14:19:28
```

```
▼ inn_1 {1}
  ▼ Reservations {3}
    ▼ CheckIn [4]
        0 : 23-OCT-10
        1 : 19-SEP-10
        2 : 30-SEP-10
        3 : 02-FEB-10
    ▼ CheckOut [4]
        0 : 25-OCT-10
```

Figure 5: Example of date and time values in Spider

Какая ***конечная станция*** *у поезда №56701?*

→ A column "destination" (DB “station_weather”)

- **Part of speech change** (a common subtype of the previous type): the use of value names different from the original form used in the DB.

  Verbal form:

  What are the mountains ***called*** *in Ethiopia*

  Перечисли, как ***называются*** *все горы в России.*

  → A column “name” from the table “mountain” (DB “mountain_photos”)

  Adjective form:

  The number of matches in which the loser ***was higher*** *than the winner.*

  Количество матчей, в которых проигравший ***был выше*** *победителя.*

  → SELECT COUNT(*) FROM matches WHERE **loser_ht** > **winner_ht** (DB “wta1_1”)

- **Partial match**: only part of the tokens from the original entity name are used in the question.

  Location of ***Webber University***

  → Value “**Webber** International **University**” (DB “protein_institute”). In the mentioned column “Institution” there are other values with the keywords “Webber” and “University”.

- **Overlapping** (an important subtype of the previous type): the use of multiple intersecting mentions of several entities.

  ID *of* ***race*** *and drive for the first position*

  Идентификатор заезда *и водителя у первого места*

  → Columns “raceId” and “driverId” (DB “formula_1”). In this table there are other columns with the keywords “race” and “driver”, thus for a model it is important to understand that the word “ID” is connected not only with the word “race”, but also with the word “driver”.

C.5 Empty Return

As mentioned above, all queries were redesigned so that the result of their execution was non-empty set of cells. To prevent the occurrence of imbalance in the dataset we add another separate pool of samples, the conditions of which do not correspond to any row of the database. This set of queries can be easily excluded from the main dataset.

There are:

- Requests with empty returns;
- Zero-result requests with COUNT aggregation;
- Requests with AVG (average) aggregation that refer to the empty set of rows and thus produce NaN as a part of the return.

D Functional Test Sets

The descriptions and sizes of the functional test sets can be found below (Tables 9, 10, 11, 12, 13). In all tables, exact match accuracy of BRIDGE (values on the left) and RAT-SQL (values on the right) base models trained on PAUQ without new samples described at C, are presented.

E Experimental Setup

Both RAT-SQL and BRIDGE systems were trained on one Tesla V100 32 GB. We used Tensor2Struct package (Wang et al., 2021a) to train RAT-SQL. The hyperparameters are taken from the original implementation of RAT-SQL provided at https://github.com/berlino/tensor2struct-public. In monolingual setup, RAT-SQL models are trained for a maximum of 25k iterations, then the best checkpoint on the corresponding dev set in terms of exact match was picked (in all cases it is a checkpoint obtained after training in the range of 20k to 25k iterations). In multilingual setup, when training data is double-sized, the maximum number of iterations is increased to 40k.

As for BRIDGE, in all cases it was trained for a maximum of 20k iterations. Then the best checkpoint according to exact-match top-1 metric on the corresponding dev set was selected. During training, we used default hyperparameters from the original implementation of BRIDGE provided at https://github.com/salesforce/TabularSemanticParsing.

Challenge set	**Example and description**	**Accuracy**	**Training amount**	**Validation amount**
Mention of values from a binary column	«What are the names of the registered nurses?»	0.09 / 0.18	83	22
Mention of gender	«What is the average age of female students?»	0.67 / 0.67	82	3
Mention of dates or times	«How many songs have 4 minute duration?»	0.02 / 0.00	102	10
Mention of "long" values (more than 3 tokens)	«ID of documents in which we can read something about Google working process *→value «How Google people work.»*	0.62 / 0.54	51	13
Empty return	«Amount of papers in 2004», *query to value «2004» which is absent in the column «YEAR».*	0.40 / 0.50	20	10
Aggregation keyword in the mentioned value	«Average maximum pressure.» *It is hard to identify the aggregation here, because column «maximum pressure» starts with key word.*	0.66 / 0.33	193	6
Mention of some tokens from the DB entity that aren't used in correct SQL query	«What is the first and second line for all addresses?» *Refer to columns «line_1» and «line_w», while in the same table there are columns «first_name» and «last_name»*	0.43 / 0.34	1699	177

Table 9: List of test sets for database features. The words defining features are highlighted by color.

Challenge set	Example and description	Accuracy	Training amount	Validation amount
Logical: union and intersection of sub-requests	«How many concerns are there in year 2014 or 2015?» → `SELECT count(*) FROM concert WHERE YEAR = 2014 OR YEAR = 2015`; «What are the ids of students who both have friends and are liked?» → `SELECT student_id FROM Friend INTERSECT SELECT liked_id FROM Likes`	0.58 / 0.53	864	111
Logical: using of Any-or-All predicates.	«Find the average age of students who do not have any pet.» → `SELECT avg(age) FROM student where stuid NOT IN (SELECT stuid FROM has_pet)`	0.73 / 0.54	245	41
Logical: negations	«What are the name of the countries where there is not a single car maker?»	0.51 / 0.47	601	106
Logical: using multiple different logical connectives in one question	«How many students are over 18 and do not have allergy to food type or animal type?»	0.25 / 0.31	108	16
Logical: different connectives in question and SQL request	«What is the sum of budgets of the Marketing and Finance departments?» → `SELECT sum(budget) FROM department WHERE dept_name = «Marketing» OR dept_name = «Finance»` *Most often, the logical connective linking the names of entities in the question coincides with the logical link in the request. So, it is really hard to define where it should be replaced with opposite one. This demand deep understanding of semantics.*	0.00 / 0.00	3	1

Table 10: List of test sets for question **logical** features.

Challenge set	Example and description	Accuracy	Training amount	Validation amount
Fuzzy: partial mentions of entities	«Location of Webber University» → value «Webber International University» *In the mentioned column «Institution» there are other values with key words «Webbe» and «University».*	0.45 / 0.40	1988	201
Fuzzy: overlapping of several entities in one question	«Winners and losers names» → columns «winner_name» and «loser_name»	0.16 / 0.23	253	37
Fuzzy: using synonyms in question for some entity in DB	«What is the final station for the train 56701?» → column «destination» from table «train»	0.62 / 0.50	1354	189
Fuzzy: Determining aggregation by the comparative or superlative adjective from column name	«What is the age of the oldest dog?» → `SELECT MAX(age) FROM Dogs`	0.00 / 0.00	22	3

Table 11: List of test sets for questions with **fuzzy and partial mentions of entities**.

Challenge set	Example and description	Accuracy	Training amount	Validation amount
Long questions: Multiple sentences in question	«Find the maximum weight for each type of pet. List the maximum weight and pet type.» → `SELECT MAX(weight), petType FROM pets GROUP BY petType`	0.51 / 0.38	358	53
Long questions: the ratio of lengths of the question to length of the query is greater than 2/3 of all queries in the database	«What are the dog names, ages and weights of all the dogs that were abandoned (note that 1 stands for value «yes», and 0 stands for value «no» in the databases tables)» → `SELECT name, age, weight FROM Dogs WHERE abandoned_yn = 1`	0.00 / 0.00	33	4
Short questions: the ratio of lengths of the query to length of the question is greater than 2/3 of all queries in the database	«Parsing top papers» → `SELECT DISTINCT t4.citedpaperid, COUNT (t4.citedpaperid) FROM paperkeyphrase AS t2 JOIN keyphrase AS t1 ON t2.keyphraseid = t1.keyphraseid JOIN paper AS t3 ON t3.paperid = t2.paperid JOIN cite AS t4 ON t3.paperid = t4.citedpaperid WHERE t1.keyphrasename ="parsing" GROUP BY t4.citedpaperid ORDER BY COUNT (t4.citedpaperid) DESC`	0.00 / 0.00	305	1

Table 12: List of test sets for questions of **atypical sizes**.

Challenge set	**Example and description**	**Accuracy**	**Training amount**	**Validation amount**
One aggregation for multiple column entities	«What is the average latitude and longitude in San Jose?» → `SELECT avg(lat), avg(long) FROM station WHERE city = "San Jose"`	0.00 / 0.00	57	4
Multiple aggregations for one column entity	«What is the average, minimum, and maximum age for all French singers?» → `SELECT avg(age), min(age), max(age) FROM singer WHERE country = "France"`	0.67 / 0.84	807	90
Extra simple: SQL request with scheme `SELECT <column> FROM <table>`	«Album titles» → `SELECT title FROM albums`	0.67 / 0.84	721	92
Simple: SQL request with scheme `SELECT <column> FROM <table> WHERE <condition>`	«How many cars has over 6 cylinders?» → `SELECT COUNT(*) FROM CARS_DATA WHERE Cylinders > 6`	0.82 / 0.79	1028	131
One «JOIN» in SQL request		0.39 / 0.34	2233	320
More than one «JOIN» in SQL request		0.11 / 0.26	1579	88
Nested sub-queries in SQL query		0.25 / 0.33	320	12
Multiple columns in the «SELECT» part of request		0.44 / 0.39	4173	92
Multiple conditions for multiple column		0.00 / 0.00	1595	131

Table 13: List of test sets for SQL query features.