Neural Approaches for Natural Language Interfaces to Databases: A Survey

Radu Iacob[1], Florin Brad[2], Elena-Simona Apostol[1], Ciprian-Octavian Truică[1], Ionel Hosu[1], Traian Rebedea[1]
[1]University Politehnica of Bucharest, Romania
[2]Bitdefender, Bucharest, Romania
{radu.iacob,elena.apostol,ciprian.truica}@upb.ro
{ionel.hosu,traian.rebedea}@upb.ro
fbrad@bitdefender.com

Abstract

A natural language interface to databases (NLIDB) enables users without technical expertise to easily access information from relational databases. Interest in NLIDBs has resurged in the past years due to the availability of large datasets and improvements to neural sequence-to-sequence models. In this survey we focus on the key design decisions behind current state of the art neural approaches, which we group into *encoder* and *decoder* improvements. We highlight the three most important directions, namely linking question tokens to database schema elements (*schema linking*), better architectures for encoding the textual query taking into account the schema (*schema encoding*), and improved generation of structured queries using autoregressive neural models (*grammar-based decoders*). To foster future research, we also present an overview of the most important NLIDB datasets, together with a comparison of the top performing neural models and a short insight into recent non deep learning solutions.

1 Introduction

Semantic parsing is the task of mapping natural language (NL) utterances to a formal meaning representation (MR), mainly with the purpose of querying an information source, i.e. a database or a knowledge graph. These machine-readable MRs are sometimes referred to as logical forms (LF). MRs can be expressed using abstract, language-agnostic representations, such as λ-calculus expressions (Zettlemoyer and Collins, 2012) or abstract meaning representations (Banarescu et al., 2013). However, they can also be represented directly in a programming or query language, for example as Python code (Ling et al., 2016) or SQL (structured query language) commands (Zhong et al., 2017).

The text-to-SQL semantic parsing task, also known as Natural Language Interfaces to Databases (NLIDB), has gathered strong research interest even in the first wave of AI (Androutsopoulos et al., 1995). Interest in NLIDBs resurfaced in the last five years due to the success of large end-to-end neural architectures for conditional text generation (Cho et al., 2014; Bahdanau et al., 2015) and the availability of larger datasets (Zhong et al., 2017; Brad et al., 2017; Yu et al., 2018c; Yu et al., 2019a).

Early systems adopted pattern-matching, handcrafted grammars and rules, syntactic parsers, and intermediate language representations, which were usually deployed in a multi-stage system (Androutsopoulos et al., 1995). The main difficulties for the successful application of NLIDB systems include: fuzzy linguistic coverage (user does not know when the system truly understands the query), linguistic vs. conceptual failures (NLIDB failure caused either by linguistic coverage or the inability to resolve database entities), language ambiguity (e.g. incorrect quantifier scoping, incorrect attachment modifier, nominal compound problem, anaphora resolution, elliptical constructs), and out-of-domain queries.

However, recent solutions treat the NLIDB problem as a sequence-to-sequence transduction task. In this case, a large end-to-end neural network processes the input sequence (NL utterance X, plus database context C) and generates the output Y autoregressively. This approach overcomes the need to learn separate linguistic components and assemble them together. Most solutions employ extensions

Proceedings of the 28th International Conference on Computational Linguistics, pages 381–395
Barcelona, Spain (Online), December 8-13, 2020

of the Seq2Seq architecture (Cho et al., 2014; Sutskever et al., 2014), which has replaced multi-stage approaches for a wide range of NLP problems.

The objective of this paper is to explore and structure the recent advances in neural solutions for NLIDB. Although the use of neural networks and deep learning for this task is quite recent, we consider that the current neural NLIDB solutions are mature and it is important to acknowledge the most successful contributions in order to plan future research. As this topic is extensive, we do not aim to cover historically important systems that are not using deep learning, but only provide a brief talk about new solutions. For the interested reader, we strongly recommend recent surveys that cover a wide range of non-deep learning systems (Affolter et al., 2019; Kamath and Das, 2019). To some extent, our work aims to supplement them with the relevant contributions of neural approaches trained on large datasets.

This paper is structured as follows. In Section 2 we frame the task of creating a NLIDB in relation to the type of data that is available for training and evaluation. In Section 3 we provide a detailed description of the key decisions behind the design of current state of the art neural solutions. Section 4 grounds the exploration of neural NLIDB models with a brief presentation of the latest solutions that do not use deep learning. Finally, in Section 5 we highlight the most effective innovations that have been introduced recently in the field and how they have affected the performance of NLIDB systems.

2 Task Description

NLIDB systems aim to translate a NL user question into an executable SQL command. We begin this section by outlining how to formulate this goal according to the type of data available for training and validation. Next, we present a set of representative datasets along with the most popular metrics used to assess the performance of NLIDBs.

2.1 Types of Supervision

In the traditional **fully supervised** setting, the NLIDB is represented by a parametric model, trained on a dataset consisting of paired NL utterances and associated SQL queries. However, the process of devising such a dataset can be a costly and time consuming endeavour, as it requires hiring annotators with strong domain expertise to formulate the SQL commands.

Alternatively, **weakly supervised** models can be trained using only NL interrogations paired with correct answers. This approach diminishes the difficulty of annotating the NL utterances with SQL queries. However, training the model becomes harder as it requires to search an exponentially large program space with only sparse, binary rewards. Since only the answer is available, it is necessary to guide the search by executing partially generated queries on the target database during training. Notably, the search may also result in *spurious* code (e.g. semantically incorrect SQL queries that produce the expected answer). In this scenario, NLIDB systems employ **policy gradient algorithms** such as REINFORCE (Sutton et al., 2000) or MAPO (Memory Augmented Policy Optimization) (Liang et al., 2018).

The strengths of the two approaches can be combined in a unitary framework. For example, even in a fully supervised setting, we can employ policy learning to augment the generation of query clauses which admit multiple correct solutions (Zhong et al., 2017). On the other hand, a model trained with weak supervision can actively request extra supervision for a small subset of training examples which would maximally improve its performance (Ni et al., 2020).

A novel direction is to jointly train a model for the related tasks of generating NL summaries of SQL queries and producing the SQL queries that correspond to NL utterances. This **dual supervision** framework can be used to augment policy gradient algorithms (Hagopian et al., 2019) and to leverage corpora of unlabeled NL utterances and queries in a semi-supervised setting (Ye et al., 2019).

Another possible source of supervision is user feedback. Users can actively improve the model by indicating if the prediction of the NLIDB system is correct (Iyer et al., 2017a) or they can help solve ambiguities during the generation process (Gur et al., 2018; Li and Jagadish, 2014).

Dataset	#Q	#SQL	#DB	#Domain	#Table/DB	Avg. Q Size	#Nested SQL
ATIS	5,280	947	1	1	32	10.47	315
GeoQuery	877	247	1	1	6	7.48	167
Restaurants	378	378	1	1	3	10.13	4
Scholar	817	193	1	1	7	6.58	7
Academic	196	185	1	1	15	13.20	7
Yelp	128	110	1	1	7	9.86	0
IMDB	131	89	1	1	16	10.22	1
SENLIDB	25,670	22,625	1	1	29	8.15	6936
WikiSQL	80,654	77,840	26,251	-	1	11.65	0
Advising	4,570	211	1	1	18	10.90	22
Spider	10,181	5,693	200	138	5	12.67	844
OTTA	3,792	3,792	5	5	12	13.53	0

Table 1: Comparison between text-to-SQL datasets for the single-turn interaction scenario.

2.2 Datasets

A straightforward solution for acquiring data is to ask domain experts to annotate NL questions, collected from real users, with the corresponding SQL command. **ATIS** (Dahl et al., 1994), **GeoQuery** (Zelle and Mooney, 1996), **Restaurants** (Popescu et al., 2003), **Yelp**, **IMDB** (Yaghmazadeh et al., 2017), **Advising** (Finegan-Dollak et al., 2018), and **Spider** are examples of datasets obtained in this manner. A slightly different approach is to have developers annotate their SQL commands with an NL description, an approach used for building the **SENLIDB** dataset (Brad et al., 2017). Most datasets use a limited number of annotators, while others have annotations collected from a large number of users or even obtained using crowdsourcing (e.g. SENLIDB).

Scholar (Iyer et al., 2017a) is a dataset that was built interactively. The SQL query associated to a new question was generated automatically with a semantic parser and the users decided if the execution result matched their expectation. Alternatively, we can achieve a better coverage of the possible SQL queries in the corpus by enumerating them and asking users to supply NL annotations (Li and Jagadish, 2014). A problem with this approach is that annotators typically do not have good knowledge of SQL or the database schema. A workaround is to provide them automatically generated NL interpretations of the target logical form and only ask for paraphrases (Wang et al., 2015). This approach of inverting the data acquisition process has led to the creation of large corpora such as **WikiSQL** (Zhong et al., 2017). Nevertheless, it may introduce an inherent bias as the original NL interpretations were generated with a set of simple rules. Another possibility, showcased by **OTTA** (Deriu et al., 2020), is to supply annotators with logical forms that are easier to understand than the corresponding SQL.

In Table 1 we present several representative statistics describing the datasets used for a **single-turn interaction** - while most of these datasets are domain specific, recent proposals aim at generalisation (e.g. Spider, OTTA). Moreover, there are also corpora aimed at multi-turn NLIDB interactions. **SParC** (Yu et al., 2019b) is a dataset derived from **Spider** that tackles the challenge of handling **multi-turn interactions**. The rationale is that, in real-world scenarios, users tend to query a database in a sequential manner, and each query may be understood *contextually*, both in relation to what is being asked presently, as well as what was previously asked. At the same time, we may also wish to have the NLIDB converse with the user, by asking for clarifications or confirmations using NL. This dialogue formulation of the problem is the focus of the related **CoSQL** corpus (Yu et al., 2019a).

Dataset split. To assess the performance of a model, it is standard practice to ensure that a sample used for evaluation is not accessible during training. In the context of NLIDBs, there are three dimensions to consider when splitting the dataset. In a **question-based** split, the exact same question should not repeat in the training and test sets. However, we allow paraphrases for the same SQL command to appear in both sets. A concern with this approach is that, if all SQL commands from the test set are also encountered during training, we only evaluate how robust the model is to different ways of expressing a known set of queries. It does not however ensure that the model can produce novel SQL queries by composing the patterns it was trained on. A **query-based split** (Finegan-Dollak et al., 2018) alleviates this concern by ensuring that SQL commands do not repeat across splits. Finally, in a **database split** (Yu et al., 2018c),

all questions pertaining to the same database appear in only one split. This enables us to test how well the model generalizes to new databases. Further splitting the test set based on the complexity of the target SQL (Yu et al., 2018c) helps to obtain a better insight about the abilities of the model.

Dataset augmentation. Synthesizing new samples automatically, without input from human annotators, has been used to improve the performance of NLIDB systems. One approach is to employ a **fixed set of rules** to derive new samples for the training set (Sun et al., 2020). For instance, values in a WHERE clause can be generated by matching question tokens to values in the table cells. Alternatively, we may create new samples from database agnostic templates, that match NL phrases to SQL clauses (Basik et al., 2018). We can further improve the linguistic variation of the resulting samples through automatic paraphrasing (Pavlick and Callison-Burch, 2016; Basik et al., 2018). Alternatively, there have been attempts to train neural networks to produce novel samples. Firstly, we can train a model to **generate NL interpretations** from the SQL query using a subset of the available corpora (Guo et al., 2018). The generator network may also be trained together with a discriminator as part of a GAN framework (Goodfellow et al., 2014; Xiong and Sun, 2019). Finally, in a **back-translation** setting (Sennrich et al., 2015) we train two networks jointly: one that synthesizes SQL from questions and one that generates the NL utterance that corresponds to the synthesized query (Sun et al., 2020).

2.3 Evaluation Metrics

Two key aspects are considered when evaluating the performance of a NLIDB: its effectiveness in answering user queries and how easy it is for the end user to interact with the system. Effectiveness is typically measured by analyzing the execution accuracy or the surface form of the predicted SQL command. Usability can be measured indirectly by analyzing the amount of time spent during an interaction (Li and Jagadish, 2014) or the number of distinct interactions (Gur et al., 2018) required to reach the desired answer. In some cases, users have been asked to provide a rating for the system (Li and Jagadish, 2014).

Execution accuracy is computed by checking if the result of the generated SQL matches the expected value. While conceptually straightforward, this metric may unfairly reward spurious code while penalizing solutions that are almost correct. Moreover, this procedure requires access to the full content of the target database and may be computationally expensive. There have been few systems, such as PRECISE (Popescu et al., 2004), for which correctness can be proved and thus they can achieve a perfect precision. However, PRECISE rejects queries that contain a token that cannot be directly linked to an entity in the database schema. Therefore, it requires extensive engineering effort to adapt its lexicon to a new domain, without which it suffers from low recall.

Alternatively, we can compare **the surface form** of the generated SQL code with the available ground truth query. This approach may incorrectly judge two semantically equivalent SQL commands as being different, but there are several techniques to alleviate this issue. First, we have to insure that both queries follow the same coding style conventions (e.g. standardize capitalization, names of aliases) (Finegan-Dollak et al., 2018). Second, we should perform our verification separately for each SQL clause, allowing us to acquire a better understanding about the specific weaknesses of the model.

3 Methodology

Neural models have become the standard approach for NLIDB solutions. The typical neural architecture consists of an encoder and a decoder component, in a Seq2Seq approach. In this section, we explore the most significant design considerations for the two components, as presented in recent literature.

3.1 Input Encoder

The input provided to an NLIDB is the database schema and the natural language query. In a content sensitive setting, the model may also incorporate the information stored in the database (Yu et al., 2018a; Iyer et al., 2017a). In addition, a model can leverage meta knowledge of the target database, such as a language model that provides the probability that a phrase references a column name (Wang et al., 2018b). Moreover, some tasks require additional contextual information such as the time an interaction takes place (Finegan-Dollak et al., 2018) or the content of previous interactions (Yu et al., 2019b).

Initial approaches modeled the question and the relevant schema context (e.g. table and column names) as a single concatenated sequence, encoded using recurrent neural networks (RNN) (Zhong et al., 2017; Wang et al., 2017). Alternatively, the input components may be encoded in distinct steps. This allows, for example, to enhance the representation of the question by applying an attention mechanism (Bahdanau et al., 2015) on the output of the column encoder (Xu et al., 2017; Dong and Lapata, 2018).

The representation of a column can be obtained by running a separate RNN over the pre-trained embeddings of the words in the column name (Xu et al., 2017). A simpler alternative is to sum the corresponding word embeddings (Yu et al., 2018a). This solution can be augmented by adding type information and the description of the table name (Yu et al., 2018b) to disambiguate between columns with the same name from different tables. Additionally, the representation may benefit from contextualization by passing all column representations through an RNN encoder (Dong and Lapata, 2018) or through a self-attention mechanism (Shi et al., 2018). This approach paved the way for using large, pre-trained, transformer-based architectures (Vaswani et al., 2017), such as BERT (Devlin et al., 2019). Recent solutions (He et al., 2019; Hwang et al., 2019; Guo and Gao, 2019; Choi et al., 2020) have enhanced the encodings of all input components by passing their respective elements concatenated through a BERT encoder.

Embeddings. Models can leverage word embeddings pre-trained on large corpora (Mikolov et al., 2013; Pennington et al., 2014). A better alternative for handling rare words is to compute embeddings for individual characters (Hashimoto et al., 2017) or sub-word units (Sennrich et al., 2016). These embeddings may be either fine-tuned during training or kept fixed to avoid over-fitting (Wang et al., 2016). Finally, by using BERT or other transformers, the model can discover contextual relations between sub-word units.

Schema linking. The process of identifying relationships between question tokens and entities from the database, such as *table names*, *column names* or the actual *cell values* is called schema linking. For instance, given the question *How many games have Lakers played?*, the token *games* may be linked to a column named *Game* in the database. Schema linking can also occur between schema entities themselves, for instance the column *Game* may be a primary key in the table *Season*.

For a statistical model it may be difficult to acquire the knowledge to perform this task, since references to entities encountered at test time may be rare or absent from the training set. In particular, the conventional practice in neural encoders of replacing rare words with a unique out-of-vocabulary symbol can affect the ability to instantiate the correct schema elements during code generation (Dong and Lapata, 2016). Moreover, at test time, the solution needs to be robust to misspellings, alternative representations or abbreviations. Another source of ambiguity is the fact that a value may belong to multiple columns.

Linking as a preprocessing step. We may perform linking as a preprocessing step, by replacing the entity occurrences from the question with generic placeholders. These placeholders typically comprise the type of the entity, as well as a numeric ID to differentiate between two entities of the same type appearing in the question (e.g., replace *Jersey* with *city_name_1*) (Dong and Lapata, 2016; Iyer et al., 2017a). Alternatively, the numeric IDs can be omitted, to prevent the model from being biased by the arbitrary order of encountered tokens (Suhr et al., 2018). During training, this can be achieved by matching variable names present in the target code with word spans appearing in the question. However, applying the same technique on the test set artificially simplifies the task (Dong and Lapata, 2016).

If the full content of the target database is accessible at test time, one method is to devise a search engine for all the entities that occur within. The search engine can be queried using n-grams from the original question with a TF-IDF scheme, until a close match is encountered (Iyer et al., 2017a). Edit distance metrics have also been used to detect partial matches between spans of words from the question and schema elements (Shaw et al., 2019). Aside from methods that account for the surface-level form of the compared items, it is also possible to integrate semantic metrics, for example by computing distances in the word-embedding space (Wang et al., 2018b).

When the model is only allowed access to the database schema, external knowledge bases may be employed. For example, TypeSQL uses Freebase to classify entities in five broad categories, deemed

representative for the type of entities encountered in the target dataset: *person*, *place*, *country*, *organization*, and *sport* (Yu et al., 2018a). Other approaches proposed to use a knowledge graph, such as ConceptNet (Speer et al., 2017), to identify relations (e.g. 'type of', 'related terms') between question n-grams and the concept denoted by a column or table name (Guo et al., 2019).

Problems may appear when a question contains multiple references to columns and their associated values. For example, given the question *Which team, founded in 1947, has won the title in 2012?*, the values *1947* and *2012* both describe years, which are compatible with the possible columns *Year founded* and *Division titles*. Figuring out the match between a column name and a detected value can be tackled by analyzing the constituency parse tree of the question (Wang et al., 2018b). Finally, some entities can only be correctly identified using additional contextual information, such as the time of the interaction (e.g, *tomorrow* from the question *Which plane leaves tomorrow?*). This task can be delegated to off-the-shelf, specialized semantic parsers (Lee et al., 2014).

Linking and semantic parsing learned jointly. Another recent approach for schema linking is to formulate it as a sequential tagging problem, which improves the semantic parser by jointly training it with the linking model. The linking model can be trained using full supervision only (Chang et al., 2020) or in combination with weak supervision (Dong et al., 2019), by rewarding it for predicting entities from the SQL query.

Graph neural networks. Entities, question tokens, code tokens, and the relationships between these elements can also be encoded with a *graph neural network* (Li et al., 2016). The graph network can either be used in conjunction with existing bidirectional RNN encoders (Bogin et al., 2019; Song et al., 2019) or as a sub-layer in transformer-based encoder and decoder blocks (Shaw et al., 2019). To ease the constraint of encoding only known relations, Relation-Aware Transformer (Wang et al., 2020) encodes arbitrary relations between the question and schema elements.

3.2 SQL Decoder

Monolithic decoder. Drawing inspiration from successes in the field of machine translation (Luong et al., 2015), early neural decoders leveraged recurrent neural networks (RNN) to sequentially generate the target command (Dong and Lapata, 2016; Iyer et al., 2017b). Thus, an RNN decoder was used to sequentially compute the probability score for each SQL token, conditioned on the input context and the previously generated tokens. The input context is encoded as described in the previous sub-section, while soft-attention mechanisms (Bahdanau et al., 2015) have been widely employed to better account for the input components most relevant for generating each token.

A straight-forward representation for the previously generated tokens is the hidden state of the decoder cell from the prior step (Iyer et al., 2017b). This representation can be improved by accounting for the fact that SQL code has a clearly defined hierarchical structure, with higher-level structures encompassing lower-level constructs. The decoder can learn to model this characteristic by augmenting, during the generation of a new structure, the input to the neural cell with the hidden state of the corresponding parent structure (Dong and Lapata, 2016).

Grammar aware decoder. With the previously mentioned approach, the neural model has to discover the syntax rules of the target SQL dialect from the available training data. One way to tackle this challenge is to incorporate explicit syntactical constraints during generation. To achieve this, we can train the decoder to produce the Abstract Syntax Tree (AST) of the target code, instead of the final code tokens. From the AST we can deterministically generate a code sequence that respects the SQL syntax (Yin and Neubig, 2017; Yin and Neubig, 2019). We can further alleviate the difficulty of learning the complete SQL syntax by restricting the grammar to a minimal set of production rules necessary for solving the queries in a particular dataset (Lin et al., 2019). Moreover, we can improve the decoder by adding common *code idioms* as valid grammar actions (Shin et al., 2019; Iyer et al., 2019). The code generation process thus interleaves low-level syntactic steps with high-level semantic constructs.

An alternative method to enhance the decoder is to target, instead of SQL, a simpler intermediate representation, such as SemQL (Guo et al., 2019). SemQL code is designed to be easier to describe

in NL than SQL, by allowing multiple SQL conditions to be specified by the same SemQL constructs. Therefore, translating first to SemQL, than deterministically converting the result to SQL, also decreases the variability of the generated queries.

Coarse-to-Fine. Since understanding a statement requires syntactic followed by semantic comprehension, it may be beneficial to target the two aspects separately. This can be achieved by decomposing the generation process in two stages, following a coarse-to-fine approach (Charniak et al., 2006). First, a high-level sketch is produced. The sketch is intended to model the syntax of the target language, while semantic details, such as the names of columns and tables, are abstracted using placeholders. During the next stage, the placeholders are replaced with concrete values. The sketch can be generated step by step (Hosu et al., 2018), using a grammar-aware decoder (Song et al., 2019; Guo et al., 2019), or selected from a fixed number of templates (Dong and Lapata, 2018; Finegan-Dollak et al., 2018; Lee et al., 2019).

Modular decoder. The SQL queries in datasets such as WikiSQL have a very simple structure, with a very small number of clauses. In this situation, we can tackle the problem by breaking the decoding process in two steps. First, the model generates a simple sketch comprising the fixed set of clauses that are necessary for solving the task (e.g. SELECT, FROM, WHERE). Then, dedicated modules for each type of clause are used to determine the correct parameters (Xu et al., 2017).

This slot-filling approach demonstrates good performance on a simpler dataset such as WikiSQL (Hwang et al., 2019; He et al., 2019). However, it adapts poorly to more complex scenarios. In particular, it is difficult to augment the sketch to allow for nested queries. One way to handle this challenge is to repeat the process for each nested clause, while allowing the slot-filling modules to predict a special slot value when a sub-query is necessary (Choi et al., 2020).

Copying mechanism. The names of referenced database entities can be copied directly from the user query or database schema. Adapted from pointer networks (Vinyals et al., 2015), a copying mechanism grants the decoder, at each step, the ability to choose between generating a new token from the vocabulary or copying an input token (Gu et al., 2016b). Using an additional memory mechanism helps diminish the chances of erroneously copying the same element repeatedly (Guo et al., 2019).

This technique is particularly useful when the user has multiple, consecutive interactions with the NLIDB system. In this setting, the model has the ability to copy entire segments from the previously generated SQL queries (Suhr et al., 2018) or to edit specific tokens within them (Zhang et al., 2019).

Inference. A popular strategy for improving the decoding results during inference is to guide the search based on the execution results of partially generated queries (Wang et al., 2018a; Shi et al., 2018; Boullanger and Dumonal, 2019; He et al., 2019; Lyu et al., 2020). Thus, if during execution no result is returned or a runtime error is triggered, the partial query can be safely discarded.

Another general technique that can be applied at this stage is to train a separate model to re-rank the answers generated during beam search (Kelkar et al., 2020). This solution can improve accuracy if the correct result does appear in the beam set, but with a lower probability than an incorrect alternative.

4 Non-Deep Learning NLIDB Approaches

In this section, we survey recent solutions that do not use deep learning (DL). When it comes to the non-DL NLIDB approaches, the main design classes are *i)* keyword-based; *ii)* pattern-based; *iii)* parsing-based; *iv)* grammar-based (Affolter et al., 2019).

Keyword-based solutions. Keyword-based approaches use inverted indexes to map the tokens extracted from natural language to existing concepts in the database and return a query. These approaches cannot express complex query intents accurately or define proper aggregation queries. For example, QaldGen (Singh et al., 2019) is a framework that uses the keyword-based approach to construct SPARQL queries over a knowledge graph for question answering systems.

Pattern-based solutions. Pattern-based methods augment the keyword-based solutions by adding information extracted through basic NLP methods, i.e. stopword removal, part-of-speech tagging, stemming and lemmatization, etc. These approaches are used to extract concepts and aggregation operations from user queries. RecipeFinder (Zhan et al., 2019) is a pattern-based system that uses human-computer interaction and basic NLP methods to extract entities and resolve question ambiguity.

Parsing-based solutions. Parsing-based approaches extend pattern-based systems by using advanced NLP techniques to parse and extract the grammatical structure of the questions. NaLIR (Li and Jagadish, 2014) and Sqlizer (Yaghmazadeh et al., 2017) build intermediate representations of the user question, leveraging a dependency parser and a semantic parser, respectively. The representations are iteratively refined using rules and heuristics and then translated to the target SQL.

Grammar-based solutions. Grammar-based systems rely on rules to transparently limit the flexibility of the user question. This approach enables auto-suggestion mechanisms (Song et al., 2015; Ferré, 2017) that guide the user to write only queries that can be correctly interpreted by the system.

5 Discussion

Non-DL vs DL approaches. Non-DL systems focus on delivering a good user experience through mechanisms such as auto-suggestion and user interaction. However, they typically depend on rules which are tailored for particular domains or databases. This limits their evaluation to relatively small, single-domain datasets (e.g. MAS, IMDB, Yelp). On the other hand, DL solutions have also been successfully applied in recent years, boosted by the availability of large datasets and the flexibility of neural sequence learners, such as the Seq2Seq architecture. For training and testing, they leverage corpora featuring multiple databases and domains, such as WikiSQL and Spider. While the two strands of work have evolved from different research communities, a common evaluation testbed is required to make the strengths and weaknesses of these lines of work more apparent.

Comparison of DL approaches. In Table 2 we report the *exact matching* accuracy for different models on the Spider (Yu et al., 2018c) development set. While many models have been developed and evaluated against the WikiSQL dataset, we opt to compare existing state-of-the-art approaches using the more challenging Spider dataset, which better reveals the relevant design choices of these models. To highlight the gains offered by different components described in Section 3, we list the models in increasing accuracy order. We also group models and their ablated variants for better readability. We consider a Seq2Seq baseline augmented with attention (Bahdanau et al., 2015) and copying (Gu et al., 2016a) mechanisms, as reported by IRNet (Guo et al., 2019). We list the most important architecture decisions in the columns, which we group into *encoder* decisions (schema linking and encoding variations), *decoder* decisions, and *other* training and inference decisions.

Schema linking. Schema linking is a crucial step and provides a significant boost, as highlighted by the ablated vs full RAT-SQL model (14.4% relative accuracy). The models in the table with no schema linking compensate by a more flexible schema encoding (self-attention in EditSQL) or a more complex decoder (grammar decoder in SyntaxSQLNet).

Leveraging structure in the database schema. Choosing the appropriate model for schema encoding is another important aspect, with more complex approaches, either based on Graph Neural Networks (GNNs) or Transformers, generally outperforming bidirectional LSTMs. EditSQL for instance has a higher EM accuracy than the simplest SyntaxSQLNet model (36.4% vs 18.9%), even without a grammar-based decoder. This difference mainly comes from the transformer-based schema encoding, which better captures the relationships between question tokens and schema elements.

Leveraging code structure. The third set of improvements comes from leveraging the grammatical structure of the SQL commands via a grammar-aware decoder (either stand-alone or modular). For instance, the simplest SyntaxSQLNet model outperforms TypeSQL by a 10.9% margin, likely due to the fact that it is guided by the SQL grammar. The BERT-free variant of IRNet also outperforms GNN

Table 2: Exact match (EM) accuracy on the Spider dev set for the latest NLIDB solutions. The columns denote the most important architecture decisions (SL - Schema Linking, DA - Data Augmentation).

		Encoder				Decoder					Other		
	SL	**BiLSTM**	**Transformer**	**GNN**	**BERT**	**LSTM**	**Modular**	**Grammar**	**C2F**	**DSL**	**DA**	**Re-ranking**	**EM**
Seq2Seq baseline (Guo et al., 2019)		✓				✓							4.1
TypeSQL (Yu et al., 2018a)	✓	✓					✓						8.0
SyntaxSQLNet (Yu et al., 2018b)		✓					✓	✓					18.9
SyntaxSQLNet (Yu et al., 2018b)		✓					✓	✓			✓		24.8
SyntaxSQLNet (Yu et al., 2018b)					✓		✓	✓			✓		25.0
GNN (Bogin et al., 2019)	✓	✓				✓		✓					34.9
GNN (Bogin et al., 2019)	✓			✓		✓		✓					40.7
GNN (Kelkar et al., 2020)	✓			✓	✓	✓		✓					51.3
GNN + Bertrand-DR (Kelkar et al., 2020)	✓			✓	✓	✓		✓				✓	57.9
EditSQL (Zhang et al., 2019)		✓	✓			✓							36.4
EditSQL (Zhang et al., 2019)		✓	✓		✓	✓							57.6
EditSQL + Bertrand-DR (Kelkar et al., 2020)		✓	✓		✓	✓						✓	58.5
IRNet (Guo et al., 2019)	✓	✓				✓		✓	✓				53.2
IRNet (Guo et al., 2019)	✓				✓	✓		✓	✓	✓			61.9
RYANSQL (Choi et al., 2020)			✓				✓						43.4
RYANSQL (Choi et al., 2020)					✓		✓						66.6
RAT-SQL (Wang et al., 2020)			✓			✓		✓					46.2
RAT-SQL (Wang et al., 2020)	✓		✓			✓		✓					60.6
RAT-SQL (Wang et al., 2020)	✓		✓		✓	✓		✓					69.7

(53.2% vs 40.7% accuracy), even though the latter uses a more complex schema encoding. The difference is again likely due to IRNet using a grammar-aware decoder.

Grammar-based decoders benefit even more when simplifying the grammar itself as much as possible. Decoding to SemQL (Guo et al., 2019) instead of SQL brings an average relative boost of 8.4%. More recently, integrating code idioms into the grammar further boosts performance (Shin et al., 2019; Iyer et al., 2019). All these examples highlight the importance of simplifying the search space of the decoder by taking advantage of the structural regularities of the SQL output. We anticipate this direction to be further exploited by future approaches.

Leveraging re-ranker methods. Using beam search in generative semantic parsers may narrowly miss the correct SQL among the generated SQL candidate list. To address this, re-ranker methods such as Bertrand-DR (Kelkar et al., 2020) can be added on top of existing models to further improve their performance. We expect this approach to become a standard performance boosting technique in the future.

Leveraging pre-trained contextualized representations. Another boosting method comes from incorporating pre-trained contextualized representations, such as BERT (Devlin et al., 2018), into the architecture. Both SyntaxSQLNet and IRNet greatly benefit from using these contextualized representations. Currently, the top 5 approaches in the Spider leaderboard[1] list BERT as part of the solution. We expect this research direction to be a fertile ground for further exploration. Specifically, we foresee that more sophisticated pre-training methods with self-supervision over structural, not just raw, data to perform even better than the current approaches.

6 Conclusion

In this paper, we have surveyed the most important contributions of deep learning approaches proposed for building NLIDB systems. Although the focus was on deep learning NLIDBs, we have also presented seminal ideas from important classical methods and from recent non-DL solutions. The main conclusion of the paper is that NLIDB research has seen a revival in the last years following the release of several large datasets and specific improvements in neural network architectures.

[1] `https://yale-lily.github.io/spider` (Last accessed on 30 June 2020)

The first set of enhancements are task-specific, accounting for structural information either in the database schema (schema encoding) or the SQL code (grammar-based decoding). Better schema encoding has been achieved with GNNs or transformer-based architectures that encode the schema entities as well as relationships between them. Syntax-based decoding is still mostly performed with recurrent architectures and further improvements have come from simplifying the SQL grammar itself.

The second improvement is in line with the recent success of transfer learning in natural language processing, namely leveraging transformer-based contextualized representations via a largely pre-trained BERT backbone. This backbone provides better word representations and can be used to encode the question and the database schema more efficiently by attending each other.

We expect more sophisticated models that better leverage the structural information of both the database schema and the SQL code to dominate future approaches. Moreover, as the performance gap on the single-turn task reduces, we also anticipate that the more complex multi-turn task will soon become the testbed of new methods, due to its difficulty and proximity to real world use cases.

Acknowledgements

The current work has been supported in part by UEFISCDI under grant PN-III-P2-2.1-PTE-2016-0109 Text2NeuralQL.

References

Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A Comparative Survey of Recent Natural Language Interfaces for Databases. *VLDB Journal*, 28(5).

I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(1):29–81.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

Fuat Basik, Benjamin Hättasch, Amir Ilkhechi, Arif Usta, Shekar Ramaswamy, Prasetya Utama, Nathaniel Weir, Carsten Binnig, and Ugur Cetintemel. 2018. Dbpal: A learned nl-interface for databases. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1765–1768, New York, NY, USA. Association for Computing Machinery.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565, Florence, Italy, July. Association for Computational Linguistics.

Geoffrey B Boullanger and Maxime Dumonal. 2019. Search Like a Human: Neural Machine Translation for Database Search. Technical report.

Florin Brad, Radu Iacob, Ionel Hosu, and Traian Rebedea. 2017. Dataset for a neural natural language interface for databases (NNLIDB). *CoRR*, abs/1707.03172.

Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Zero-shot text-to-sql learning with auxiliary task. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7488–7495. AAAI Press.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 168–175, New York City, USA, June. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *arXiv preprint arXiv:2004.03125*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriber. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. *Proceedings of the workshop on Human Language Technology*, pages 43–48.

Jan Deriu, Katsiaryna Mlynchyk, Philippe Schläpfer, Álvaro Rodrigo, Dirk Von Grünigen, Nicolas Kaiser, Kurt Stockinger, Eneko Agirre, and Mark Cieliebak. 2020. A methodology for creating question answering corpora using inverse data annotation. *CoRR*, abs/2004.07633.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to Logical Form with Neural Attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:731–742, May.

Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-Anonymous Encoding for Text-to-SQL Generation. 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)*, pages 5405–5414. Association for Computational Linguistics.

Sébastien Ferré. 2017. Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language. *Semantic Web*, 8(3):405–418.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, July.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016a. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016b. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, August. Association for Computational Linguistics.

Tong Guo and Huilin Gao. 2019. Content enhanced bert-based text-to-sql generation. *CoRR*, abs/1910.07179.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from SQL queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535. Association for Computational Linguistics.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. DialSQL: Dialogue Based Structured Query Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1339–1349. Association for Computational Linguistics.

Kaylin Hagopian, Qing Wang, Tengfei Ma, Yupeng Gao, and Lingfei Wu. 2019. Learning Logical Representations from Natural Languages with Weak Supervision and Back-Translation. In *Knowledge Representation & Reasoning Meets Machine Learning Workshop (KR2ML)*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, Copenhagen, Denmark, September. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. *CoRR*, abs/1908.08113.

Ionel Hosu, Radu Iacob, Florin Brad, Stefan Ruseti, and Traian Rebedea. 2018. Natural Language Interface for Databases Using a Dual-Encoder Model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 514–524.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017a. Learning a Neural Semantic Parser from User Feedback. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:963–973, apr.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017b. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Srinivasan Iyer, Alvin Cheung, and Luke Zettlemoyer. 2019. Learning programmatic idioms for scalable semantic parsing. *CoRR*, abs/1904.09086.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*.

Amol Kelkar, Rohan Relan, Vaishali Bhardwaj, Saurabh Vaichal, and Peter Relan. 2020. Bertrand-dr: Improving text-to-sql using a discriminative re-ranker. *arXiv preprint arXiv:2002.00557*.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Dongjun Lee, Jaesik Yoon, Jongyun Song, Sang-gil Lee, and Sungroh Yoon. 2019. One-shot learning for text-to-sql generation. *CoRR*, abs/1905.11499.

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

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Chen Liang, Google Brain, Mohammad Norouzi, Jonathan Berant, Quoc Le Google Brain, and Ni Lao. 2018. Memory Augmented Policy Optimization for Program Synthesis and Semantic Parsing. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 10015–10027. Curran Associates Inc.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *CoRR*, abs/1905.13326.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew W. Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *CoRR*, abs/1603.06744.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql. Technical Report MSR-TR-2020-7, Microsoft Dynamics 365 AI, March.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Ansong Ni, Pengcheng Yin, and Graham Neubig. 2020. Merging weak and active supervision for semantic parsing. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8536–8543. AAAI Press.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple ppdb: A paraphrase database for simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–148.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, , 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.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 141–147.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. 2019. Generating Logical Forms from Graph Representations of Text and Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 95–106. Association for Computational Linguistics.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *CoRR*, abs/1809.05054.

Richard Shin, Miltiadis Allamanis, Marc Brockschmidt, and Oleksandr Polozov. 2019. Program Synthesis and Semantic Parsing with Learned Code Idioms. In *Advances in Neural Information Processing Systems*, pages 10824–10834.

Kuldeep Singh, Muhammad Saleem, Abhishek Nadgeri, Felix Conrads, Jeff Z. Pan, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. 2019. QaldGen: Towards microbenchmarking of question answering systems over knowledge graphs. In *Lecture Notes in Computer Science*, pages 277–292. Springer International Publishing.

Dezhao Song, Frank Schilder, Charese Smiley, Chris Brew, Tom Zielund, Hiroko Bretz, Robert Martin, Chris Dale, John Duprey, Tim Miller, and Johanna Harrison. 2015. Tr discover: A natural language interface for querying and analyzing interlinked datasets. In Marcelo Arenas, Oscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015*, pages 21–37, Cham. Springer International Publishing.

Meina Song, Zecheng Zhan, and Haihong E. 2019. Hierarchical Schema Representation for Text-to-SQL Parsing With Decomposing Decoding. *IEEE Access*, 7:103706–103715, jul.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Alane Suhr, Srinivasan Iyer, Yoav Artzi, and Paul G Allen. 2018. Learning to Map Context-Dependent Sentences to Executable Formal Queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2238–2249. Association for Computational Linguistics.

Yibo Sun, Duyu Tang, Nan Duan, Yeyun Gong, Xiaocheng Feng, Bing Qin, and Daxin Jiang. 2020. Neural semantic parsing in low-resource settings with back-translation and meta-learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8960–8967. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342.

Wei Wang, Meihui Zhang, Gang Chen, H V Jagadish, Beng Chin Ooi, and Kian-Lee Tan. 2016. Database Meets Deep Learning: Challenges and Opportunities. *ACM SIGMOD Record*, 45(2):17–22.

Chenglong Wang, Marc Brockschmidt, and Rishabh Singh. 2017. Pointing out sql queries from text. Technical Report MSR-TR-2017-45, November.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018a. Execution-guided neural program decoding. *CoRR*, abs/1807.03100.

Wenlu Wang, Yingtao Tian, Hongyu Xiong, Haixun Wang, and Wei-Shinn Ku. 2018b. A transfer-learnable natural language interface for databases. *CoRR*, abs/1809.02649.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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*, pages 7567–7578, Online, July. Association for Computational Linguistics.

Hongvu Xiong and Ruixiao Sun. 2019. Transferable Natural Language Interface to Structured Queries Aided by Adversarial Generation. In *Proceedings - 13th IEEE International Conference on Semantic Computing, ICSC 2019*, pages 255–262. Institute of Electrical and Electronics Engineers Inc., March.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *CoRR*, abs/1711.04436.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–26, October.

Hai Ye, Wenjie Li, and Lu Wang. 2019. Jointly Learning Semantic Parser and Natural Language Generator via Dual Information Maximization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2090–2101. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.

Pengcheng Yin and Graham Neubig. 2019. Reranking for Neural Semantic Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4553–4559. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium, October-November. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. 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)*, pages 1962–1979, Hong Kong, China, November. Association for Computational Linguistics.

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 Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy, July. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *CoRR*, abs/1207.1420.

Huayi Zhan, Baivab Sinha, and Wei Jiang. 2019. Natural language question/answering with user interaction over a knowledge base. In *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science - AICS 2019*, pages 325–329. ACM Press.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. 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)*. Association for Computational Linguistics.

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