Research Report

DBPal: A Novel Lightweight NL2SQL Training Pipeline

NATURAL LANGUAGE (NL) IS A PROMISING ALTERNATIVE INTERFACE TO DATABASE MANAGEMENT SYSTEMS (DBMSs) BECAUSE IT ENABLES NON-TECHNICAL USERS TO FORMULATE COMPLEX QUESTIONS. RECENTLY, DEEP LEARNING HAS GAINED TRACTION FOR TRANSLATING NATURAL LANGUAGE TO SQL. HOWEVER, THE CORE PROBLEM WITH EXISTING DEEP LEARNING APPROACHES IS THAT THEY REQUIRE AN ENORMOUS AMOUNT OF MANUALLY CURATED TRAINING DATA IN ORDER TO PROVIDE ACCURATE TRANSLATIONS. WE PRESENT DBPAL THAT USES A NOVEL TRAINING PIPELINE TO LEARN NL2SQL INTERFACES WHICH SYNTHESIZES TRAINING DATA AND, THUS, DOES NOT RELY ON MANUALLY CURATED TRAINING DATA.

Benjamin Hättasch Nadja Geisler

Carsten Binnig

Introduction

In order to effectively leverage their data, DBMS users are required to not only have prior knowledge about the database schema (e.g., table and column names, entity relationships) but also a working understanding of the syntax and semantics of SQL. Unfortunately, despite its expressiveness, SQL can often hinder non-technical users from exploring and making use of data stored in a database. These requirements set "a high barrier to entry" for data exploration and have, therefore, triggered new efforts to develop alternative interfaces that allow non-technical users to explore and interact with their data conveniently. For example, imagine a doctor wants to look at the age distribution of patients with the longest stays in a hospital. To answer this question, the doctor would either need to write a complex nested SQL query or work with an analyst to craft the query. Even with a visual exploration tool (e.g., Tableau, Vizdom), posing such a query is nontrivial, since it requires the user to perform multiple interactions with an understanding of the nested query semantics. Alternatively, with an NL interface, the query is as simple as stating: "What is the age distribution of patients who stayed longest in the hospital?"

Based on this observation, a number of "natural language interfaces to databases" tools (NLIDBs) have been proposed that aim to translate natural language to SQL (NL2SQL). The first category of solutions are rule-based systems, e.g., NaLIR (Li and Jagadish, 2014), which use fixed rules for performing translations. Although effective in specific instances, these approaches are brittle and do not generalize well without substantial additional effort to support new use cases. More recently, deep learning techniques have gained traction for NL2SQL, since similar ideas have achieved success in the related domain of machine translation. For example, generic sequence-to-sequence (seq2seq) (Zhong et al., 2017) models have been successfully used in practice for NL2SQL translation, and more advanced approaches like Syntax SQLNet (Yu et al., 2018), which augments deep learning models with a structured model that considers the syntax and semantics of SQL, have also been proposed.

However, a crucial problem with deep learning approaches is that they require an enormous amount of training data in order to build accurate models. The aforementioned approaches have largely ignored this problem and assumed the availability of large, manually-curated training datasets (e.g., using crowd-sourcing). In almost all cases, however, gathering and cleaning such data is a substantial undertaking that requires a significant amount of time, effort, and money.

Moreover, existing approaches for NL2SQL translation attempt to build models that generalize to new and unseen databases, yielding performance that is generally decent but does not perform as well as running new queries on the databases used for training. That is, the training data used to translate queries for one specific database, such as queries containing words and phrases pertaining to patients in a hospital, does not always allow the model to generalize to queries in other domains, such as databases of geographical locations or flights.

In order to address these fundamental limitations, in a recent SIGMOD 2020 paper (Weir et al., 2020), we proposed DBPal, a fully pluggable NL2SQL training pipeline that can be used with any existing NL2SQL deep learning model to improve translation accuracy. DBPal implements a novel training pipeline for NLIDBs that synthesizes its training data using the principle of weak supervision.

The basic idea is to leverage various heuristics and existing datasets to automatically generate large (and potentially noisy) training data instead of manually handcrafting training examples. In its basic form, only the database schema is required as input to generate a large collection of pairs of NL queries and their corresponding SQL statements that can be used to train any NL2SQL deep learning model.

In order to maximize our coverage across natural-linguistic variations, DBPal also uses additional input sources to automatically augment the training data through a variety of techniques. One such augmentation step, as an example, is an automatic paraphrasing process. The goal of these augmentation steps is to make the model robust to different linguistic variations of the

same question (e.g., "What is the age distribution of patients who stayed longest in the hospital?" and "For patients with the longest hospital stay, what is the distribution of age?").

In the evaluation of our SIGMOD publication, we show that DBPal, which requires no manually crafted training data, can effectively improve the performance of a state-of-the-art deep learning model for NL2SQL translation. Our results demonstrate that an NLIDB can be effectively bootstrapped without requiring manual training data for each new database schema or target domain. Furthermore, if manually curated training data is available, such data can still be used to complement our proposed data generation pipeline.

Overview of DBPal

In the following, we first discuss the overall architecture of a NLIDB and, then, discuss DBPal, our proposed training pipeline based on weak supervision that synthesizes the training data from a given database schema.

Figure 1 shows an overview of the architecture of our fully functional prototype NLIDB, which consists of multiple components, including a user-interface that allows users to pose NL questions that are automatically translated into SQL. The results from the user's NL query are then returned to the user in an easy-to-read tabular visualization.

At the core of our prototype is a neural translator, which is trained by DBPal's pipeline, that translates incoming NL queries coming from a user into SQL queries. Importantly, our fully pluggable training pipeline is agnostic to the actual translation model; that is, DBPal is designed to improve the accuracy of existing NL2SQL deep learning models (e.g., SyntaxSQLNet) by generating training data for a given database schema.

Training Phase. During the training phase, DBPal's training pipeline provides existing NL2SQL deep learning models with large corpora of synthesized training data (Figure 2). This training pipeline consists of three steps to synthesize the training data: (1) generator, (2) augmentation, and (3) lemmatizer. Once training data is synthesized by DBPal's pipeline, it can then be used (potentially together with existing manually curated training data) to train existing neural translation models that can be plugged into the training pipeline.

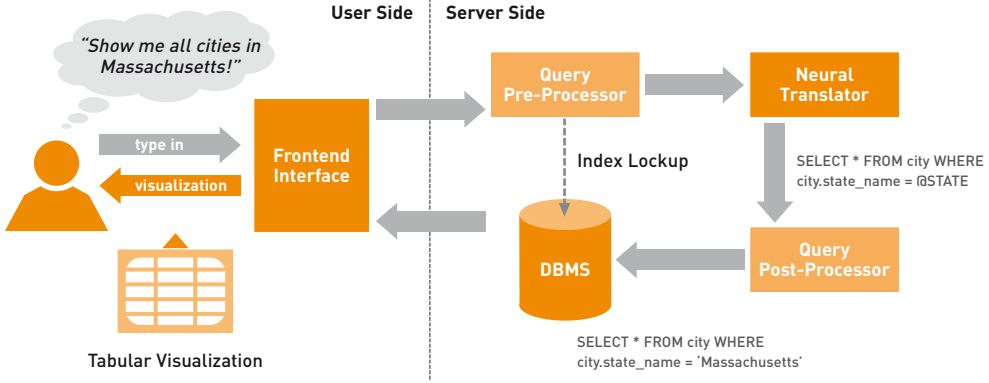

Figure 1: Overview of DBPal's Architecture

Runtime Phase. The runtime phase can leverage a model (neural translator) that was trained by DBPal, as shown on the right-hand side of Figure 2. The parameter handler is responsible for replacing the constants in the input NL query with placeholders to make the translation model independent from the actual database and help to avoid retraining the model if the underlying database is updated. For example, for the input query shown in Figure 2 (i.e., "What are cities whose state is Massachusetts?"), the parameter handler replaces "Massachusetts" with the appropriate schema element using the placeholder @STATE. The lemmatizer then combines different variants of the same word to a single root. For example, the words "is", "are", and "am" are all mapped to the root word "be". Then, the neural translator works on these anonymized NL input queries and creates output SQL queries, which also contain placeholders. In the example shown in Figure 2, the output of the neural translator is: SELECT name FROM cities WHERE state = @STATE. Finally, the post-processor replaces the placeholders with the actual constants such that the SQL query can be executed.

DBPal's Training Pipeline

The basic flow of the training pipeline is shown on the left-hand side of Figure 2. In the following,

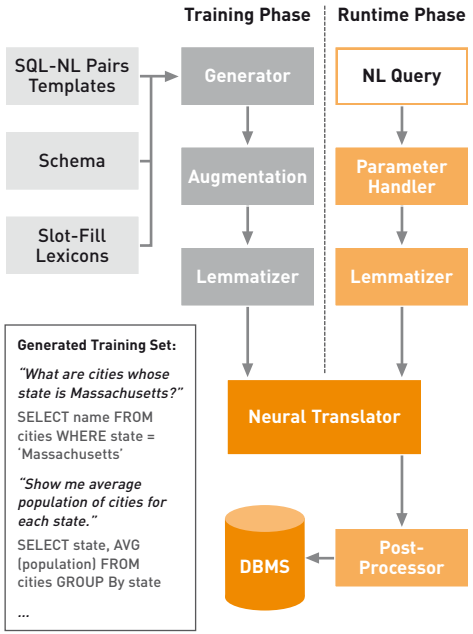

Figure 2: Training and Runtime Phase

we describe the training pipeline and focus in particular on the data generation framework.

Generator. In the first step, the generator uses the database schema along with a set of seed templates that describe typical NL-SQL pairs to generate an initial training set. In the second step, augmentation, the training data generation pipeline then automatically adds to the initial training set of NL-SQL pairs by leveraging existing general-purpose data sources and models to linguistically modify the NL part of each pair.

The main idea is that each seed template covers a typical class of SQL queries (e.g., a SELECT-

FROM-WHERE query with a simple predicate). Composing the seed templates is only a minimal, one-time overhead, and all templates are independent of the target database, i.e., they can be reused for other schemas. Furthermore, in DBPal, we assume that the database schema provides human-understandable table and attribute names, but the user can optionally annotate the schema to provide more readable names if required. Deriving readable schema names automatically is an orthogonal issue.

The schema information is, then, used to instantiate these templates using table and attribute names. Additionally, manually predefined dictionaries (e.g., to cover synonyms) can be used to instantiate simple variations of NL words and phrases (e.g., "Show me" and "What is" for the SELECT clause). Currently, DBPal contains approximately 100 seed templates. A typical training set that can be generated from these templates contains around one million NL-SQL pairs for a simple, single-table database schema and around two to three million for more complicated schemas.

Augmentation. A core aspect of our pipeline is the augmentation step that automatically expands the training data produced by our generator in order to offer more accurate and linguistically robust translations. During augmentation, the training data generation pipeline automatically adds new NL-SQL pairs by leveraging existing general-purpose data sources and models to linguistically vary the NL part of each pair. The goal of the augmentation phase is to cover a wide spectrum of linguistic variations for the same SQL query, which represent different versions of how users might phrase the query in NL. This augmentation is the key to make the translation model robust and allows DBPal to provide better query understanding capabilities than existing standalone approaches.

Lemmatization. Finally, in the last step of the data generation procedure, the resulting NL-SQL pairs are lemmatized to normalize the representation of individual words. During this process, different forms of the same word are mapped to the word's root in order to simplify the analysis (e.g., "cars" and "car's" are replaced with "car"). The same lemmatization is applied at runtime during the aforementioned pre-processing step.

Conclusions and Future Work

We presented DBPal, a fully pluggable natural language to SQL (NL2SQL) training pipeline that generates synthetic training data to improve both the accuracy and robustness to linguistic variation of existing deep learning models. In combination with our presented data augmentation techniques, which help improve the translational robustness of the underlying models, DBPal is able to improve the accuracy of state-of-the-art deep learning models by up to almost 40%.

Longer term, we believe that an exciting opportunity exists to expand DBPal's techniques to tackle broader data science use cases, ultimately allowing domain experts to interactively explore large datasets using only natural language (Rogers et al., 2017). In contrast to the typical notion of one-shot SQL queries currently taken by DBPal, data science is an iterative, session-driven process, where a user repeatedly modifies a query or machine learning model after examining intermediate results until finally arriving at some desired insight, which will, therefore, necessitate a more conversational interface. These extensions would require the development of new techniques for providing progressive results (Turkay et al., 2018) by extending past work on traditional SQL-style queries and machine learning models.

Finally, we believe there are also interesting opportunities related to different data models, e.g., time series (Eichmann et al., 2017) and new user interfaces, e.g., query-by-voice (Lyons et al., 2016).

References

Eichmann, P.; Crotty, A.; Galakatos, A.; Zgraggen, E.:
Discrete Time Specifications in Temporal Queries.
In: CHI Extended Abstracts; Denver (CO), US, 2017.

Li, F.; Jagadish, H.:
NaLIR: An Interactive Natural Language Interface for Querying Relational Databases.
In: Proceedings of the ACM SIGMOD Conference; Snowbird (UT), US, 2014.

Lyons, G.; Tran, V.; Binnig, C.; Çetintemel, U.; Kraska, T.:
Making the Case for Query-by-Voice with EchoQuery.
In: Proceedings of the ACM SIGMOD Conference; San Francisco (CA), US, 2016.

Rogers, J. L. J.; Potti, N.; Patel, J.:
Ava: From Data to Insights Through Conversations.
In: Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR); Chaminade (CA), US, 2017.

Turkay, C.; Pezzotti, N.; Binnig, C.; Strobelt, H.; Hammer, B.; Keim, D.; Fekete, D.; Palpanas, T.; Wang, Y.; Rusu, F.:
Progressive Data Science: Potential and Challenges.
In: Working Paper, 2018.

Weir, N.; Utama, P.; Galakatos, A.; Crotty, A.; Ilkhechi, A.; Ramaswamy, S.; Bhushan, R.; Geisler, N.; Hättasch, B.; Eger, S.; Cetintemel, U.; Binnig, C.:
DBPal: A Fully Pluggable NL2SQL Training Pipeline.
In: Proceedings of the ACM SIGMOD Conference; Portland (OR), US, 2020.

Yu, T.; Yasunaga, M.; Yang, K.; Zhang, R.; Wang, D.; Li, Z.; Radev, D.:
SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL Task.
In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP); Brussels, Belgium, 2018.

Zhong, V.; Xiong, C.; Socher, R.:
Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.
In: Working Paper, 2017.