Explaining Queries over Web Tables to Non-Experts

Jonathan Berant, Daniel Deutch, Amir Globerson, Tova Milo and Tomer Wolfson
Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel

***Abstract*—Designing a reliable natural language (NL) interface for querying tables has been a longtime goal of researchers in both the data management and natural language processing (NLP) communities. Such an interface receives as input an NL question, translates it into a formal query, executes the query and returns the results. Errors in the translation process are not uncommon, and users typically struggle to understand whether their query has been mapped correctly. We address this problem by explaining the obtained formal queries to non-expert users. We introduce novel query explanations that provide a graphic representation of the query cell-based provenance (in its execution on a given table). Our solution augments a state-of-the-art NL interface over web tables, enhancing it in both its training and deployment phase. Experiments, including a user study conducted on Amazon Mechanical Turk, show our solution to improve both the correctness and reliability of the NL interface.**

I. Introduction

Natural language interfaces have been gaining significant popularity, enabling ordinary users to write and execute complex queries. A prominent paradigm for developing NL interfaces is *semantic parsing*, which is the mapping of NL phrases into a formal language. As Machine Learning techniques are standardly used in semantic parsing, a training set of question-answer pairs is provided alongside a target database [1], [2]. The parser is a parameterized function that is trained by updating its parameters such that questions from the training set are translated into queries that yield the correct answers.

A crucial challenge for using semantic parsers is their reliability. Flawless translation from NL to formal language is an open problem, and even state-of-the-art parsers tend to err. Without any explanations of the executed query, users are left wondering whether its results are actually correct. Consider the example in Figure 1, displaying a table of Olympic games and the question *"Greece held its last Olympics in what year?"*. Given this question, a semantic parser would generate multiple candidate queries and return the evaluation result of its top candidate. The user will only be presented with the evaluation result, *2004*, having no clear indication whether his question was correctly parsed. Ensuring the system has executed a correct query (rather than simply returning a correct answer in a particular instance) is essential, as it enables reusing the query as the data evolves over time.

Our approach is to design *provenance-based* [3], [4] query explanations that are extensible, domain-independent and immediately understandable by non-expert users. We enhance an existing semantic parser for querying tables by introducing a novel component featuring our query explanations. Following the parsing of an input NL question, candidate queries are explained to users, allowing non-experts to choose the one that best fits their intention. The immediate application is to improve the quality of obtained queries at deployment time over simply choosing the parser's top query (without user feedback). Furthermore, we show how query explanations can be used to obtain user feedback which is used to retrain the Machine Learning system, thereby improving its performance.

Question: "Greece held its last Olympics in what year?"

MAX(Year)	Country	City
1896	Greece	Athens
...	...	...
2004	Greece	Athens
2008	China	Beijing
2012	UK	London
2016	Brazil	Rio de Janeiro

u : *maximum value in column Year where Country is Greece.*

Fig. 1. Lambda DCS query, `max(R[Year].Country.Greece)` is explained by utterance u and by the table's provenance-based highlights.

TABLE I
WikiTableQuestions Examples

Return the difference in engine size between Luigi Arcangeli and Louis Chiron.
Which country has more silver medals, the US or China?
In what position did team Penske finish the first year a Honda engine was used?
How many championships were in the $150,000 category?
Which was the only episode to gain a 7 or higher rating?

Setting: For our NL interface we use the state-of-the-art semantic parser of [5], mapping complex NL questions into queries over web tables. We test our solution on the WikiTableQuestions benchmark dataset [2] which includes over 20,000 complex NL questions formulated by actual users on thousands of extracted web tables.

As our formal query language over tables we use *lambda DCS*, a standard query language in the NLP community [1], [2], [6]. As lambda DCS is geared towards queries one would write in a search engine rather than database ones, its queries receive a single table as their input. Lambda DCS enables us to formulate highly complex queries supporting operations such as sorting, aggregation and intersection (Table I contains complex NL questions that can be expressed in lambda DCS).

Contributions: We introduce a novel multilevel cell-based provenance model for the lambda DCS query language, implemented through the use of *provenance-based highlights*. These highlights are paired with a more conventional form of query explanations via *NL utterances* [7]. One of our key contributions is showing empirically that combining NL utterances with our provenance-based highlights greatly accelerates user understanding of complex queries. We demonstrate the effectiveness of our query explanations by enhancing a semantic parser for querying tables. At deployment, users

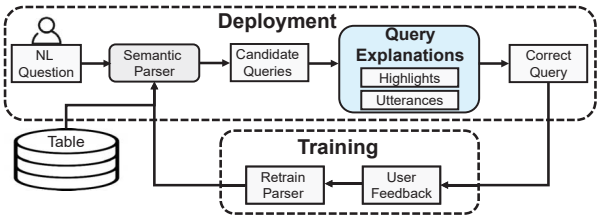

Fig. 2. System Architecture.

are able to make an informed choice of the query, based on our explanation mechanism. We further leverage the use of explaining queries in order to retrain the semantic parser on procured user feedback (feedback being pairs of NL questions and their correct query). User studies conducted via Amazon Mechanical Turk (AMT) show our query explanations to be applicable to non-experts. Furthermore, user feedback was beneficial for improving the correctness of generated queries, both at parser deployment and in its training.

II. System Overview

In Figure 2 we describe the general workflow of our system. Given an NL question on a corresponding table, we use the semantic parser in [5] to parse the question into a set of candidate lambda DCS queries. Our interface will generate the relevant query explanations for each candidate query, displaying an NL utterance and highlighting the provenance data (Section IV). Query explanations are presented to non-technical users for them to select the correct formal-query representing the NL question. User feedback in the form of question-query pairs is stored in order to retrain the parser.

III. Provenance for Lambda DCS

The tracking and presentation of provenance data has been extensively studied in the context of relational queries [3], [4]. We use provenance information for explaining the query execution on a given web table by designing a model for *multilevel cell-based provenance*. The model's categorization of provenance cells serves as a form of query explanation, implemented through our provenance-based table highlights.

A. Model Definitions

Given query Q and table T, the execution result, denoted by $Q(T)$, is either a collection of table cells, or a numeric result of an aggregate or arithmetic operation. We define three *cell-based provenance functions*, $P_O(Q,T), P_E(Q,T), P_C(Q,T)$. Given Q, T these functions output a set of table cells and aggregate functions.

Function $P_O(Q,T)$ returns all cells output by $Q(T)$ or, if $Q(T)$ is the result of an arithmetic or aggregate operation, returns all table cells involved in that operation in addition to the aggregate function itself. $P_E(Q,T)$ returns cells and aggregate functions used during the query execution. $P_C(Q,T)$ returns all table cells in columns that are either projected or aggregated on by Q. Using our model, we provide multilevel cell-based provenance for all lambda DCS operators. Example provenance for several operators is provided in Table II. We use c to denote a table cell with value $T[c]$, while denoting specific cell values by u, v. Each cell c belongs to a table record, $record(c)$ with a unique index, $record(c).Index$. We distinguish between two types of lambda DCS formulas: formulas returning values are denoted by $vals$ while those returning entire table records by $records$.

Example 3.1: We explain the provenance of the following lambda DCS query, `R[Year].City.Athens`. It returns the values of column Year in records where column City is Athens, thus $P_O(Q,T)$ will return all cells containing these values.

$$P_O(\mathbf{R}[Year].City.Athens, T) = \{c \mid c \in Year \wedge record(c).Index \in City.Athens.Indices\}$$

The cells involved in executing Q include the output cells $P_O(Q,T)$ but also the provenance of sub-formula `City.Athens`, defined as cells of column City with Athens.

$$P_E(\mathbf{R}[Year].City.Athens, T) = P_O(\mathbf{R}[Year].City.Athens, T) \cup P_E(City.Athens, T)$$
$$P_E(City.Athens, T) = \{c \mid c \in City \wedge T[c] = Athens\}$$

The provenance of the columns of Q is simply all cells appearing in columns Year and City.

$$P_C(\mathbf{R}[Year].City.Athens, T) = \{c \mid c \in Year \vee c \in City\}$$

IV. Explaining Queries

To allow users to understand formal queries we must provide them with effective explanations. We implement the multilevel provenance model introduced in Section III, producing query explanations we term *provenance-based highlights*. For each provenance function (P_O, P_E, P_C) we uniquely highlight its cells, creating a visual explanation of the query execution.

Given a query Q and table T, our procedure divides cells into four types, based on their multilevel provenance functions. To help illustrate the query, each cell type is highlighted differently: *Colored cells* are equivalent to $P_O(Q,T)$ - the cells returned by Q as output or used to compute the final output. *Framed cells* are equivalent to $P_E(Q,T)$ - the cells and aggregate functions used during query execution. *Lit cells* are equivalent to $P_C(Q,T)$ - the cells of columns projected by the query. Cells that are unrelated to the query have no highlights applied to them.

Example 4.1: Consider the lambda DCS query,

```
sub(R[Total].Nation.Fiji, R[Total].Nation.Tonga).
```

The query returns the *"difference in column Total between rows where Nation is Fiji and Tonga"*. Figure 4 displays the highlights generated for this query, lighting all of the query's columns, framing its provenance cells and coloring the cells that comprise its output. In this example, all cells in columns *Nation* and *Total* are lit. The cells *Fiji* and *Tonga* are part of $P_E(Q,T)$ and are therefore framed. The cells in $P_O(Q,T)$, containing *130* and *20*, are colored as they contain the values used to compute the final result.

Examples of provenance-based highlights are provided for several lambda DCS operators in Figures 3-5.

TABLE II
EXAMPLE PROVENANCE FOR SEVERAL QUERY OPERATORS

Operator	Query (lambda DCS)	Example	Translation	Provenance
Column Values	`R[C].records`	`R[Year].City.Athens`	"value of column Year where City is Athens"	$P_O(Q) = \{c \mid c \in C \ \wedge \ record(c).Index \in records.Indices\}$ $P_E(Q) = P_O(Q) \cup P_E(records)$ $P_C(Q) = \{c \mid c \in C\} \cup P_C(records)$
Aggregation on Values	`aggr(vals)` aggr $\in$ {count, max, min, sum, avg}	`sum(R[Year].City.Athens)`	"the sum of values in column Year where City is Athens"	$P_O(Q) = P_O(vals) \cup \{AGGR\}$ $P_E(Q) = P_O(Q)$ $P_C(Q) = \{c \mid c \in C\}$
Intersection of Records	$records_1 \sqcap records_2$	`City.London` $\sqcap$ `Country.UK`	"rows where value of City is London and also where value of Country is UK"	$P_O(Q) = P_O(records_1) \cap P_O(records_2)$ $P_E(Q) = P_E(records_1) \cup P_E(records_2)$ $P_C(Q) = P_C(records_1) \cup P_C(records_2)$

Year	Country	City	...
1896	Greece	Athens	...
...	...	...	...
2004	Greece	Athens	...
2008	China	Beijing	...
2012	UK	London	...
2016	Brazil	Rio de Janeiro	...

Fig. 3. Superlative (values). *Between London or Beijing who has the highest value of Year.*

Nation	Gold	...	Total
New Caledonia	120	...	288
Tahiti	60	...	144
Fiji	33	...	130
Samoa	22	...	73
Tonga	4	...	20
...	...	...	...

Fig. 4. Difference (values). *Difference in column Total between Fiji and Tonga.*

"What was the highest growth rate of Madagascar in the 1980s?"

Row	Country	Year	...	MAX(Growth)
14266	Madagascar	1986	...	2.945
14270	Madagascar	1983	...	2.877
14454	Burkina Faso	2011	...	3.085

Fig. 5. Scaling highlights to a large table by selecting three table rows.

We note that different queries may possess identical provenance-based highlights. In such cases the user should refer to the NL utterances of the queries in order to distinguish between them. Our explanation methods are complementary, with provenance-based highlights providing quick visual feedback while utterances serve as detailed descriptions.

Scaling to Large Tables: Our query explanations can be easily extended to tables with numerous records. When employing our provenance-based highlights to large tables it might seem intractable to display them to the user. However, as the highlights are meant to explain the candidate query, we can indicate its semantics by highlighting a subsample of the table. An intuitive solution can be used to achieve a succinct sample. First, we compute the three cell-based provenance sets (Section III) then build corresponding record sets, $R_O(Q,T), R_E(Q,T), R_C(Q,T)$ (with cells mapped to their table records). To best illustrate the query we sample one record from each of the following sets: $R_O(Q,T)$, $R_E(Q,T) \setminus R_O(Q,T)$ and $R_C(Q,T) \setminus R_E(Q,T)$. In the special case of a query containing arithmetic difference (Figure 4), we select two records from $R_O(Q,T)$, one for each subtracted value. Sampled records are ordered according to their order in the original table. The example in Figure 5 contains three table rows selected from a large web table.

V. CONCRETE APPLICATIONS

We harness our methods for explaining formal queries to non-experts in order to enhance an existing NL interface.

Deployment: When deployed, our interface is given a table and a corresponding user question. Using the semantic parser [5] it generates a set of candidate queries ranked by their likelihood of being correct. We display the top-k candidates to users through our explanations (utterances and highlights), enabling them to identify which queries are correct translations and which should be discarded. In contrast to the baseline parser, our system displays users its top-k candidates, allowing them to modify the top query when necessary.

Training on Feedback: User feedback is also used to enhance the system correctness through training. The mapping from NL to formal queries is learned by the semantic parser of [5], trained on the WIKITABLEQUESTIONS dataset [2]. We improve the semantic parser by retraining it on pairs of questions and formal queries that were marked as their correct translations by users. While it is known that training a semantic parser on question-query pairs improves its performance [8], up until now this was only achieved by employing users familiar with the query language. We are the first to illicit such *annotations* without any reliance on experts [8]–[10].

VI. EXPERIMENTS

Following the presentation of concrete applications for our methods we have designed an experimental study to measure the effect of our query explanation mechanism. As our main evaluation metric we define the parser *correctness* as the percentage of questions where the *top-ranked* candidate query was a *correct translation* of the NL question to lambda DCS. In addition to correctness, we also measured the *mean reciprocal rank* (MRR) to evaluate the average correctness of all candidate queries generated.

A. Interactive Parsing at Deployment

We conducted a user study to measure both the quality of our query explanations and their ability to improve the real-time performance of the semantic parser. The participants were anonymous workers recruited through the the Amazon Mechanical Turk (AMT) crowdsourcing platform. Focusing on non-experts, our only requirements were that participants be over 18 years old and reside in a native English speaking country. Our study included 35 participants, a significant number compared to previous works on evaluating NL interfaces [4], [11], [12]. Rather than relying on a fixed set of NL test questions [4], [11] we presented each worker with 20 distinct questions that were randomly selected from the WIKITABLEQUESTIONS benchmark dataset for question answering over tables. Displaying the top-k results allowed

TABLE III
USER STUDY - SUCCESS RATES

distinct questions	explanations	avg. success
405	2,835	78.4%

The number of distinct questions and candidate query explanations presented to users.

TABLE IV
USER WORK-TIME (MINUTES) ON 20 QUESTIONS

method	avg	median	min	max
Utterances + Highlights	16.2m	16.6m	6.45m	22.5m
Utterances	24.7m	20.7m	17.5m	35.4m

workers to improve the baseline parser in cases where the correct query was generated, but not ranked at the top. We chose to display top-k queries with $k = 7$. We made sure to validate that our choice of k was sufficiently large, so that it included the correct query when generated.

Explanation Quality: We first wished to validate the quality of our query explanations by assessing the ability of non-experts to correctly identify candidate queries given their explanations. Results in Table III show the effectiveness of our approach with non-expert success at 78.4%. We then sought to measure the effect to which our provenance-based highlights accelerate user understanding. Dividing users into two groups of 10 workers each, the first group was presented with both highlights and utterances as their query explanations, while in the other group users relied solely on NL utterances. Table IV shows our provenance-based explanations cut the average and median work-time by 34% and 20% respectively, this while achieving the same correctness results.

Parser Correctness: We have examined the effect to which our query explanations can help users improve the correctness of a baseline NL interface. Table V details the correctness results of the parser with and without user interaction. *Parser* denotes the baseline parser correctness, while *Users* denote the correctness where the end results is the candidate query marked by the user. The *Hybrid* scenario is the same as *Users* except when the user marked all queries as incorrect it will return the parser's top candidate. We define the *correctness bound* as the percentage of examples where the top-k candidate queries actually contain a correct result. Using the correctness-bound as an upper bound on our results shows the hybrid approach achieves 87% of its full potential, an encouraging results given that our non-experts possess no prior experience of their task.

B. Training on User Feedback

Through the use of query explanations non-expert users are able to *annotate* questions, i.e. assign to them their correct formal queries. Using non-expert workers from AMT we were able to collect 2,068 annotated questions from the WIKITABLEQUESTIONS training set. Following standard methodology, we split the annotated data into train and development sets. Out of our 2,068 annotated examples, 418 were selected as the development set, and 1,650 as the training set. The annotated development examples were used to evaluate the effect of our annotations on the parser correctness.

TABLE V
USER STUDY - CORRECTNESS RESULTS

	Parser	Users	Hybrid	Bound
correct examples	260/700	312/700†	**341**/700†	392/700
correctness	37.1%	44.6%	**48.7%**	56%

The † sign indicates statistical significance compared to the Parser baseline at the 0.01 level using the χ^2 test (with 1 degree of freedom).

TABLE VI
EFFECT OF USER FEEDBACK ON CORRECTNESS

train ex.	annotations	correctness	MRR
1650	1650	**49.8%**	**0.586**
1650	0	41.8%	0.499
11000	1650	**51.6%**	**0.60**
11000	0	49.5%	0.570

We experimented on two scenarios: (1) training the parser solely on 1,650 annotated examples; (2) incorporating our annotated examples into the entire WIKITABLEQUESTIONS training set. In each scenario two parsers were trained, one trained using annotations and the other without any use of annotations. To gain more robust results we ran our experiments on three different train/dev splits of our data, averaging the results. Table VI displays the results of our experiments. The spike in correctness further asserts the significance of annotated training data [8], [9] and shows that our system can learn from quality feedback collected entirely by non-experts.

VII. RELATED WORK

Building natural language interfaces for querying databases has been studied extensively [10]–[12]. The usage of procured user feedback to train semantic parsers was pursued in recent works [8], [10]. While such works relied on additional labeling by experts (familiar in SQL, SPARQL) we used solely the feedback of non-expert users. When regarding provenance models for relational queries [3], the work in [4] presents an interface complete with a provenance model for NL queries, albeit limited to handling Conjunctive Queries. The cell-based provenance we present supports further NL constructs such as union, aggregation, etc.

REFERENCES

[1] J. Berant *et al.*, "Semantic parsing on Freebase from question-answer pairs," in *EMNLP*, 2013.
[2] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," in *ACL*, 2015.
[3] B. Glavic *et al.*, "Using sql for efficient generation and querying of provenance information," in *In search of elegance in the theory and practice of computation*. Springer, 2013.
[4] D. Deutch *et al.*, "Provenance for natural language queries," in *VLDB*. VLDB Endowment, 2017.
[5] Y. Zhang *et al.*, "Macro grammars and holistic triggering for efficient semantic parsing," in *EMNLP*, 2017.
[6] P. Liang, "Lambda dependency-based compositional semantics," 2013.
[7] Y. Wang *et al.*, "Building a semantic parser overnight." in *ACL*, 2015.
[8] W. T. Yih *et al.*, "The value of semantic parse labeling for knowledge base question answering," in *ACL*, 2016.
[9] Y. Artzi and L. Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions," in *TACL*, 2013.
[10] S. Iyer *et al.*, "Learning a neural semantic parser from user feedback," in *ACL*, 2017.
[11] F. Li and H. Jagadish, "Constructing an interactive natural language interface for relational databases," in *VLDB*. VLDB Endowment, 2014.
[12] G. Koutrika *et al.*, "Explaining structured queries in natural language," in *ICDE*, 2010.