NLProv: Natural Language Provenance

Daniel Deutch
Tel Aviv University
danielde@post.tau.ac.il

Nave Frost
Tel Aviv University
navefrost@mail.tau.ac.il

Amir Gilad
Tel Aviv University
amirgilad@mail.tau.ac.il

ABSTRACT

We propose to present NLProv: an end-to-end Natural Language (NL) interface for database queries. Previous work has focused on interfaces for specifying NL questions, which are then compiled into queries in a formal language (e.g. SQL). We build upon this work, but focus on presenting a detailed form of the *answers* in Natural Language. The answers that we present are importantly based on the *provenance* of tuples in the query result, detailing not only which are the results but also their *explanations*. We develop a novel method for transforming provenance information to NL, by leveraging the original NL question structure. Furthermore, since provenance information is typically large, we present two solutions for its effective presentation as NL text: one that is based on provenance factorization with novel desiderata relevant to the NL case, and one that is based on summarization.

1. INTRODUCTION

Developing Natural Language (NL) interfaces to database systems has been the focus of multiple lines of research (see e.g. [10, 1, 9]), with the motivation typically being the difficulty of writing database queries in a formal language. In this work we complement these efforts by supporting NL *answers* to NL queries. The answers that we provide are *detailed and justified*, accounting not only for the requested information but also for crucial details regarding the *reasons* for it to qualify as an answer.

For instance, consider the Microsoft Academic Search database (http://academic.research.microsoft.com) and consider the NL query in Figure 1. A state-of-the-art NL query engine, NaLIR [10], is able to transform this question into the SQL query which is also shown (as a Conjunctive Query, for convenience of later development) in Figure 1. When evaluated, the query returns the expected organizations. But the user who asked this question is likely to further be interested in *why* was each organization name returned as an answer, i.e. the authors associated with the organization, their papers, and the venues and years of the publications. Such additional information, typically referred to as *provenance*, both enriches the answers and allows for their validation.

We propose here a novel approach of presenting provenance information for NL questions, again as sentences in Natural Language. A first key idea in our solution is to leverage the *structure of the question* in constructing the NL answer. In particular, we use and modify the code of NaLIR[1] so that we store exactly which parts of the NL question translates to which parts of the formal query. Then, we evaluate the formal query using a provenance-aware engine (we use SelP [4]), further modified so that it stores which parts of the query "contribute" to which parts of the provenance. By composing these two "mappings" (text-to-query-parts and query-parts-to-provenance) we infer which parts of the question text are related to which parts of the provenance. Finally, we use the latter information in an "inverse" manner, to translate the provenance to NL text.

A second key idea is related to the provenance size. In typical scenarios, a single answer may have multiple explanations (multiple authors, papers, venues and years in our example). A naive solution is to formulate and present a separate sentence corresponding to each explanation. The result will however be, in many cases, very long and repetitive. As observed already in previous work [2, 12], different assignments (explanations) may have significant parts in common, and this can be leveraged in a *factorization* that groups together multiple occurrences. In our example, we can e.g. factorize explanations based on author, on paper name (relevant for multi-authored papers), on conference name or on year. In the context of e.g. [2, 12], one prefers factorizations the lead to representations of smaller size. In our context, some factorizations are better than others *because they lead to answers whose NL semantics better meet the user expectations.* For instance, in the question of Fig. 1, the author name is semantically close to the organization name and so we prefer answers in which this semantic proximity is maintained. We formalize that, and devise a greedy algorithm to find small-sized factorizations that furthermore satisfy this desideratum. We further translate factorized representations to concise NL sentences.

Last, we propose *summarized* explanations, based on e.g. the number of papers published by each author or the overall number of papers published by authors of each organi-

Proceedings of the VLDB Endowment, Vol. 9, No. 13
Copyright 2016 VLDB Endowment 2150-8097/16/09.

[1] We are extremely grateful to Fei Li and H.V. Jagadish for generously sharing with us the source code of NaLir, and providing invaluable support.

```
return the organization of authors who published papers
in database conferences after 2005
```

```
query(oname) :- org(oid, oname), author(aid, aname, oid),
pub(wid, cid, ptitle, pyear), conf(cid, cname),
domainConf(cid, did), domain(did, dname), writes(aid, wid),
dname = 'Databases', pyear > 2005
```

Figure 1: NL Query and CQ Q

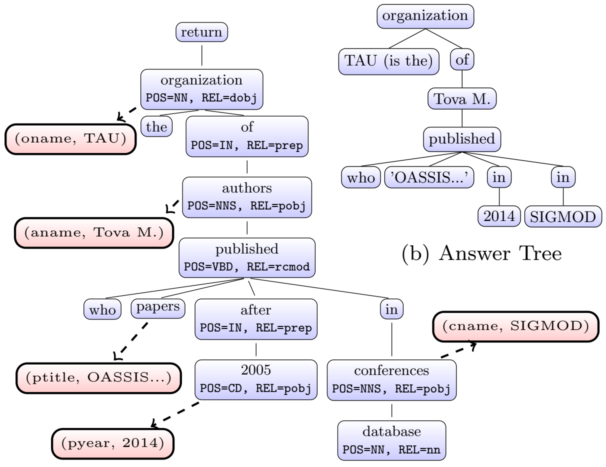
zation. Such summarizations incur by nature a loss of information but are typically much more concise and easier for users to follow. We show a tight correspondence between factorization and summarization: every factorization gives rise to multiple possible summarizations, each obtained by counting the number of sub-explanations that are "factorized together". Consequently, factorizations that compare favorably to others with respect to semantic optimality and conciseness also give rise to better summarizations. NLProv computes and presents such NL summarizations to the provenance, of varying levels of granularity.

2. TECHNICAL DETAILS

We briefly explain, via an example, the technical development underlying NLProv.

Natural Language Database Queries. We start by exemplifying an NL query and its *dependency tree* [11] that describes both the syntactic and semantic roles of terms.

EXAMPLE 2.1. *Re-consider the NL query in Figure 1; its dependency tree is depicted in Figure 2a (ignore for now the arrows). Every node is associated with a part-of-speech (POS) tag reflecting its syntactic role in the sentence (e.g. "organization" is a noun, denoted "NN", and "published" is a verb in past tense, denoted "VBD"). Each node is also associated with a relation (REL) tag, reflecting the semantic relation of its sub-tree with its parent. For instance, the REL of "published" is rcmod ("relative clause modifier") meaning that it describes a property of "authors".*



(b) Answer Tree

(a) Query Tree

Figure 2: Question and Answer Trees

The dependency tree is transformed by NaLIR, based also on schema knowledge, to SQL. We focus in this work on SPJU queries and present our example as a Conjunctive Query (CQ), see Q in Figure 1. NLProv augments NaLIR by keeping track of the mapping it produces from dependency tree nodes to query variables.

```
(oname,TAU)∧(aname,Tova M.)∧(ptitle,OASSIS...)∧
          (cname,SIGMOD)∧(pyear,14')∨
(oname,TAU)∧(aname,Tova M.)∧(ptitle,Querying...)∧
          (cname,VLDB)∧(pyear,06')∨
(oname,TAU)∧(aname,Tova M.)∧ (ptitle,Monitoring..)∧
          (cname,VLDB)∧(pyear,07')∨
(oname,TAU)∧(aname,Slava N.)∧(ptitle,OASSIS...)∧
          (cname, SIGMOD)∧(pyear,14')∨
(oname,TAU)∧(aname,Tova M.)∧(ptitle,A sample...)∧
          (cname,SIGMOD)∧(pyear,14')∨
(oname,UPENN)∧(aname,Susan D.)∧(ptitle,OASSIS...)∧
          (cname,SIGMOD)∧(pyear,14')
```

Figure 3: Value-level Provenance

EXAMPLE 2.2. *Reconsider the tree t in Figure 2a and the CQ Q in Figure 1. Some of the nodes in t are mapped to variables of Q. For example, the word "organization" corresponds to the head variable (oname) of Q. Similarly the word "authors" corresponds to aname in Q, etc.*

Provenance Model and Mapping. After compiling a formal query corresponding to the user's question, we evaluate it and keep track of *provenance*, to be used in explanations. Multiple notions of provenance appear in the literature, mostly focusing on recording the input *tuples* used in assignments. For explaining query results in NL, this is insufficient: we need *value-level* provenance.

oid	oname
1	UPENN
2	TAU

Rel. *org*

aid	aname	oid
3	Susan D.	1
4	Tova M.	2
5	Slava N.	2

Rel. *author*

wid	cid	ptitle	pyear
6	10	"OASSIS..."	2014
7	10	"A sample..."	2014
8	11	"Monitoring..."	2007
9	11	"Querying..."	2006

Rel. *pub*

aid	wid
4	6
3	6
5	6
4	7
4	8
4	9

Rel. *writes*

cid	cname
10	SIGMOD
11	VLDB

Rel. *conf*

cid	did
10	18
11	18

Rel. *domainConf*

did	name
18	Databases

Rel. *domain*

Table 1: DB Instance

EXAMPLE 2.3. *Re-consider our running example query and consider the database in Table 1. The assignments to the query are represented in Figure 3 as a DNF expression. Each of the 6 clauses stands for a different assignment, and the atoms are pairs of the form (var, val) so that var is assigned val in the particular assignment. We only record variables to which a query word was mapped (these are the relevant variables for formulating the answer).*

By composing the mappings from the question's dependency tree to query variables, and the assignments of query variables to values from the database, we associate different parts of the question with values.

EXAMPLE 2.4. *Continuing our running example, consider the assignment represented by the first clause of Figure 3.*

Further reconsider Fig. 2a, and now note that each node is associated with a pair (var, val) of the variable to which the node was mapped, and the value that this variable was assigned in this particular assignment.

Generating a sentence for a single assignment. The structure of the question's dependency tree and the mapping to database values are then used to construct a detailed NL answer. As a first step, we demonstrate how a single assignment is transformed into an NL sentence.

Example 2.5. *Reconsider Figure 2a. To generate an answer, we follow the question structure, "plugging-in" mapped database values. For each node n that is mapped to a value, we either replace n with its value or "connect" n to a new node including the value. The choice of transformation is based on whether n has a* modifier *as a child and if so, which kind of modifier it is. Intuitively, modifiers describe the object in n, and so depending on their existence/kind, replacing/adding n will have different effects on the sentence. For instance, "authors" has a verb modifier ("published") describing it, and so it is safe to replace "authors" with the value it is mapped to ("Tova M."). In contrast, the modifier of "organization" is prepositional ("of") and so replacing it with the value ("TAU") will result in an improper sentence; instead we add a node "TAU" along with connecting words (in this case "is the"). The "conference" node has a noun modifier ("database"), in which case both the node and its modifier are replaced by "SIGMOD". Leaves (e.g. "papers") are simply replaced by values. After further transformations (e.g. replacing "after" by "'in"), we finally obtain a tree representation of the answer (Fig. 2b). Converting it to a sentence (details omitted), we obtain:*

```
TAU is the organization of Tova M. who published
'OASSIS...' in SIGMOD in 2014
```

We next generalize the construction to account for multiple query results as well as multiple explanations for each result. A naive solution in this respect is to generate a sentence for each explanation of each result. Already for the small-scale example presented here, this would result in a long and unreadable answer. Instead, we have implemented two solutions: the first based on the idea of provenance factorization [12], and the second providing a summarized form.

Factorization. Different explanations in the provenance expression typically share significant parts, which may be exploited for representing it in a more succinct way.

Example 2.6. *Re-consider the DNF in Figure 3. Rewriting it in a factorized (non-DNF) form would be much more succinct, as e.g. the same organization and the same author name appear in multiple conjuncts. Two possible factorizations are shown in Figure 4 (keeping only the values and omitting the variable names for brevity).*

$$\begin{array}{l}
\texttt{[TAU]} \wedge \\
A\left\{\begin{array}{l}
(\texttt{[Tova M.]} \wedge \\
B\left\{\begin{array}{l}
(\texttt{[VLDB]} \wedge \\
\quad (\texttt{[2006]} \wedge \texttt{[Querying...]} \\
\quad \vee \texttt{[2007]} \wedge \texttt{[Monitoring...]})) \\
\vee \texttt{[SIGMOD]} \wedge \texttt{[2014]} \wedge \\
\qquad (\texttt{[OASSIS...]} \vee \texttt{[A sample...]}))
\end{array}\right\} B \\
\vee \texttt{[Slava N.]} \wedge \texttt{[OASSIS...]} \wedge \texttt{[SIGMOD]} \wedge \texttt{[2014]})
\end{array}\right\} A \\
\vee \texttt{[UPENN]} \wedge \texttt{[Susan D.]} \wedge \texttt{[OASSIS...]} \wedge \texttt{[SIGMOD]} \wedge \texttt{[2014]}
\end{array}$$

(a) f_1

```
[TAU] ∧
    ([SIGMOD] ∧ [2014] ∧
        ([OASSIS...] ∧
            ([Tova M.] ∨ [Slava N.]))
        ∨ [Tova M.] ∧ [A sample...])
    ∨ [VLDB] ∧ [Tova M.] ∧
        ([2006] ∧ [Querying...]
        ∨ [2007] ∧ [Monitoring...])
∨ [UPENN] ∧ [Susan D.] ∧  [OASSIS...] ∧ [SIGMOD] ∧ [2014]
```

(b) f_2

Figure 4: Provenance Factorizations

How do we measure the quality of a factorization? Natural desiderata are that it should be short or that the maximal number of appearances of an atom is minimal [12, 5]. On the other hand, in our case we factorize as a step towards generating an NL answer; so if the factorized expression structure is similar to that of the question, the answer is likely to better fit the user intention. We found that commonly used measures such as edit distance between questions and answers (see e.g. the survey in [6]) are unsuitable here, and we thus devise a novel "structural" condition that again leverages the information on mappings, as follows.

Let f be a factorization, let p be the query dependency tree, and let x, y be nodes of p such that y is a descendant of x. For every two atoms x', y' in f s.t. x, y were mapped to x', y' (for some derivation), the nesting depth (i.e. level in the circuit corresponding to f) of x' should be smaller or equal to that of y'. Intuitively, if x is "higher" than y in the hierarchy in the question, then its corresponding part of the answer should not be pushed below that of y.

Example 2.7. *f_2 is slightly shorter in terms of number of atoms than f_1. However, f_1 satisfies the above condition and its structure is indeed "more similar" to that of the query dependency tree. In contrast, in f_2 "[SIGMOD]" appears in a shallower nesting depth than "[Tova M.]", but "author" is an ancestor of "conferences" in the dependency tree of the query. If we were to generate a sentence based on f_2, we would have to face the problem of losing the intuitive direct connection reflected in the question between the organization and authors (as we have "pushed" the conference and paper names between them). Based on f_1, we generate the following sentence for the answer "TAU" (a similar sentence is generated for "UPenn"):*

```
TAU is the organization of
    Tova M. who published
        in VLDB
           'Querying...' in 2006 and
           'Monitoring...' in 2007
        and in SIGMOD in 2014
            'OASSIS...' and  'A sample...'
    and Slava N. who published
        'OASSIS...' in SIGMOD in 2014.
```

Summarization. In some cases, even an "optimal" factorized representation is too long and convoluted for users to follow. In these cases we need to summarize the provenance in some way that will preserve the "essence" of all assignments without actually specifying them. We do so by leveraging the factorization, and "summarizing" the brackets at any level. Summarization manifests by replacing a disjunction by the number of disjuncts, with further constructs to group e.g. numerical values, forming a range expression.

```
(A) TAU is the organization of 2 authors who published
4 papers in 2 conferences in 2006 - 2014.
(B) TAU is the organization of Tova M. who published
4 papers in 2 conferences in 2006 - 2014 and Slava N.
who published 'OASSIS...' in SIGMOD in 2014.
```

Figure 5: Summarized Sentences

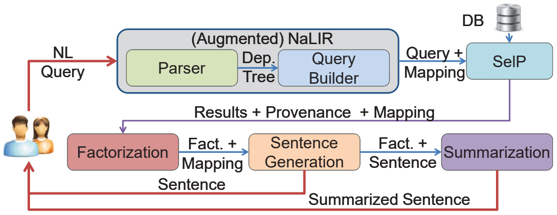


Figure 6: System Arch.

Example 2.8. *Re-consider the factorization f_1 from Figure 4, and now note the A and B brackets standing for levels of summarizations. The resulting sentences (for the "TAU" answer) are shown in Fig. 5. Summarizing at a higher level results in a shorter but less detailed summarization.*

Related Work. As mentioned in the Introduction, multiple lines of work (e.g. [10, 1, 9]) have proposed NL interfaces to database queries, and multiple lines of work have studied provenance tracking (e.g. [7, 2, 3]), but to our knowledge our work is the first to support NL provenance. We note that [8] has focused on the complementary problem of translating SQL *queries* (rather than their provenance) to NL.

3. SYSTEM OVERVIEW

NLProv is implemented in JAVA with JAVAFX GUI using SceneBuilder, and runs on Windows 8. It uses MySQL server as its underlying database management system. Figure 6 depicts the system architecture. First, the user enters a query in Natural Language. This NL sentence is fed to the augmented NaLIR system which interprets it and generates a formal query. This includes the following steps: a parser [11] generates the dependency tree for the NL query. Then, the nodes of the tree are mapped to attributes in the tables of the database and to functions, to form a formal query. As explained above (see Example 2.2), to be able to translate the results and provenance to NL, NLProv stores the mapping from the nodes of the dependency tree to the query variables. Once a query has been produced, NLProv uses the SelP system [4] to evaluate it while storing the provenance (see Example 2.3), keeping track of the mapping of dependency tree nodes to parts of the provenance (see Example 2.4). The provenance information is then factorized (see Examples 2.6 and 2.7) and the factorization is compiled to an NL answer containing explanations (see Example 2.7).

Finally, the factorized answer is shown to the user. If the answer contains excessive details and is too difficult to understand, the user may choose to view summarizations of different nesting levels (see Example 2.8).

4. DEMONSTRATION SCENARIO

We will demonstrate that NLProv provides intuitive, human-readable explanations to answers of Natural Language database queries. The operation of NLProv will be demonstrated with respect to the Microsoft database of publications, through examples such as those presented in this short paper. The demonstration will interactively engage the audience, demonstrating the different facets of the system. Since the explanations are given in Natural Language, understanding them requires no prior knowledge on provenance (in fact they require no database knowledge) and so the demonstration will thus be suitable for the VLDB audience at large. For the first part of the demonstration we will use a set of pre-defined questions of varying complexity levels. We will first show the audience a single explanation for one of the answers, then consider all explanations and show the different variants of Natural Language explanations supported by the system: full (non-factorized) explanations, factorized explanations, and summarized explanations of varying granularity. We will show different options, in addition to those chosen by the system (and in particular the result of choosing other possible factorizations), demonstrating the superior quality of the results w.r.t. these alternatives. We will then allow participants to pose, in natural language, queries of their liking with respect to the publications dataset. We will again show the answers and explanations computed by NLProv. Last, we will allow participants to look "under-the-hood", showing the underlying queries and the generated provenance as boolean expressions, explaining the connection between different parts of the question, query and explanations, and highlight the manner in which NLProv computes answers and explanations.

Acknowledgements. This research was partially supported by the Israeli Science Foundation (ISF, grant No. 1636/13), by the Broadcom Foundation and by ICRC - The Blavatnik Interdisciplinary Cyber Research Center.

5. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16, 2002.

[2] A. P. Chapman, H. V. Jagadish, and P. Ramanan. Efficient provenance storage. In *SIGMOD*, pages 993–1006, 2008.

[3] S. B. Davidson and J. Freire. Provenance and scientific workflows: Challenges and opportunities. In *SIGMOD*, pages 1345–1350, 2008.

[4] D. Deutch, A. Gilad, and Y. Moskovitch. Selective provenance for datalog programs using top-k queries. *VLDB*, pages 1394–1405, 2015.

[5] K. Elbassioni, K. Makino, and I. Rauf. On the readability of monotone boolean formulae. *JoCO*, pages 293–304, 2011.

[6] M. Emms. Variants of tree similarity in a question answering task. In *LD*, pages 100–108, 2006.

[7] T. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[8] G. Koutrika, A. Simitsis, and Y. E. Ioannidis. Explaining structured queries in natural language. In *ICDE*, pages 333–344, 2010.

[9] D. Küpper, M. Storbel, and D. Rösner. Nauda: A cooperative natural language interface to relational databases. In *SIGMOD*, pages 529–533, 1993.

[10] F. Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, pages 73–84, 2014.

[11] M. Marneffe, B. Maccartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *LREC*, pages 449–454, 2006.

[12] D. Olteanu and J. Závodný. Factorised representations of query results: Size bounds and readability. In *ICDT*, pages 285–298, 2012.