**Provenance for Natural Language Queries**

**DEVELOPMENT EFFORT**

**BY**

Krunal Vipinchandra Sevak [A20405875]

Parth Girishkumar Patel [A20405581]

Tai Nguyen [A20356881]

Group 6

 Spring 2018

Presented to the

College of Science

Illinois Institute of Technology

ACKNOWLEDGMENTS

The support of Illinois Institute of Technology's Center for Computer Science is gratefully acknowledged. Thanks are also due to Prof. Boris Glavic for many helpful discussions, to Xing Niu for reviewing this on an earlier draft of this paper.

**PROBLEM STATEMENT:**

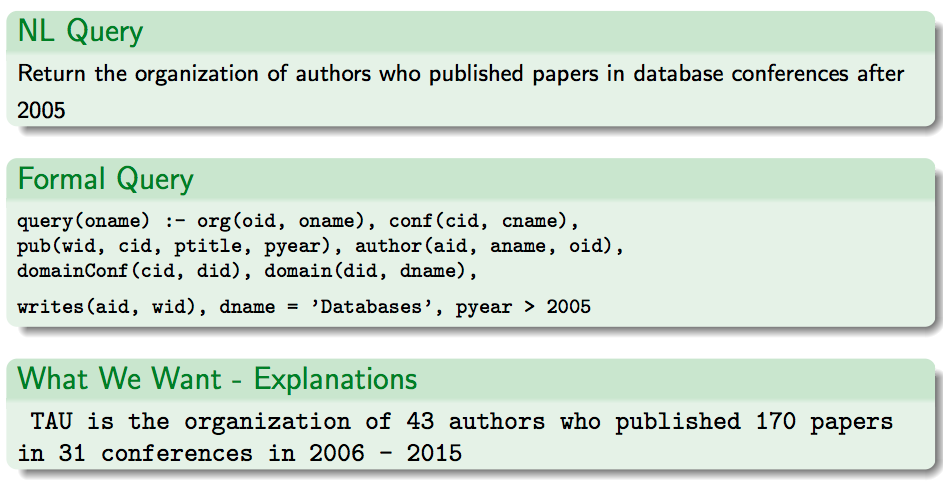
Multiple lines of research have developed Natural Language (NL) interfaces for formulating database queries. For this our focus is on presenting a highly detailed form of the answers in NL. The answers that we present are importantly based on the provenance of tuples in the query result, detailing not only the results but also their explanations.

**ABSTRACT:**

For dealing with this problem, We develop a novel method for transforming provenance information to NL, by leveraging the original NL query structure. This problem is very large and complex, but there are two solutions for this problem, ***Factorization*** and ***Summarization***. We have implemented our solution in an end-to-end system supporting questions, answers and provenance, all expressed in NL with the experiments, including a user study, quality of solution and its scalability.

**INTRODUCTION:**

Developing Natural Language (NL) interfaces to database frameworks has been the focal point of various lines of research. As an example, consider the Microsoft Academic Search database and consider the NL query in Figure 1a. A state-of-the-art NL query engine, NaLIR, is able to transform this NL query into the SQL query also shown (as a Conjunctive Query, which is the fragment that we focus on in this paper). When evaluated using a standard database engine, the query returns the expected list of organizations. However, the answers (organizations) in the query result lack justification, which in this case would include the authors affiliated with each organization and details of the papers.

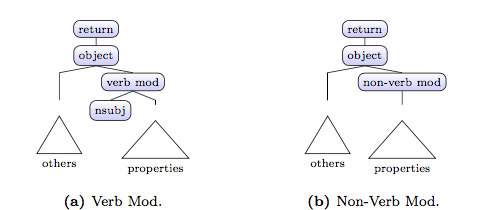


We propose a novel approach of presenting provenance in- formation for answers of NL queries, again as sentences in Natural Language.

**NL TO FORMAL QUERIES:**

We start by recalling some basic notions from NLP, as they pertain to the translation process of NL queries to a formal query language. A key notion that we will use is that of the ***syntactic dependency*** tree of NL queries.

We focus on a sub-class of queries handled by NaLIR, namely that of Conjunctive Queries, possibly with comparison operators (=,>,<) and logical combinations thereof (NaLIR further supports nested queries and aggregation). The corresponding NL queries in NaLIR follow one of the two (very general) abstract forms described in below Figure an object (noun) is sought for, that satisfies some properties, possibly described through a complex sub-sentence rooted by a modifier (which may or may not be a verb, a distinction whose importance will be made clear later).



The dependency tree is transformed by NaLIR, based also on schema knowledge, to SQL. We focus in this work on NL queries that are compiled into Conjunctive Queries (CQs).

The translation performed by NaLIR from an NL query to a formal one can be captured by a mapping from (some) parts of the sentence to parts of the formal query.

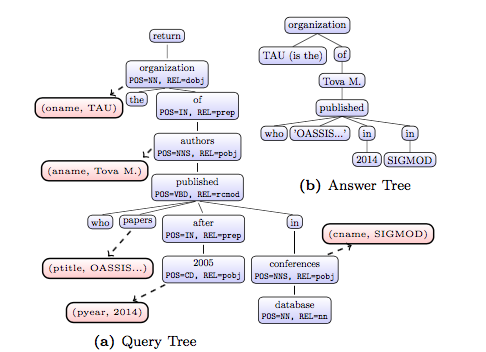
**MAPPINGS AND ANSWER TREE – Single Assignment:**

We now start describing our transformation of provenance to NL. We show it first for the case where the query has a single assignment with respect to the input database. The solution will serve as a building block for the general case of multiple assignments.

**For Example:**

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

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



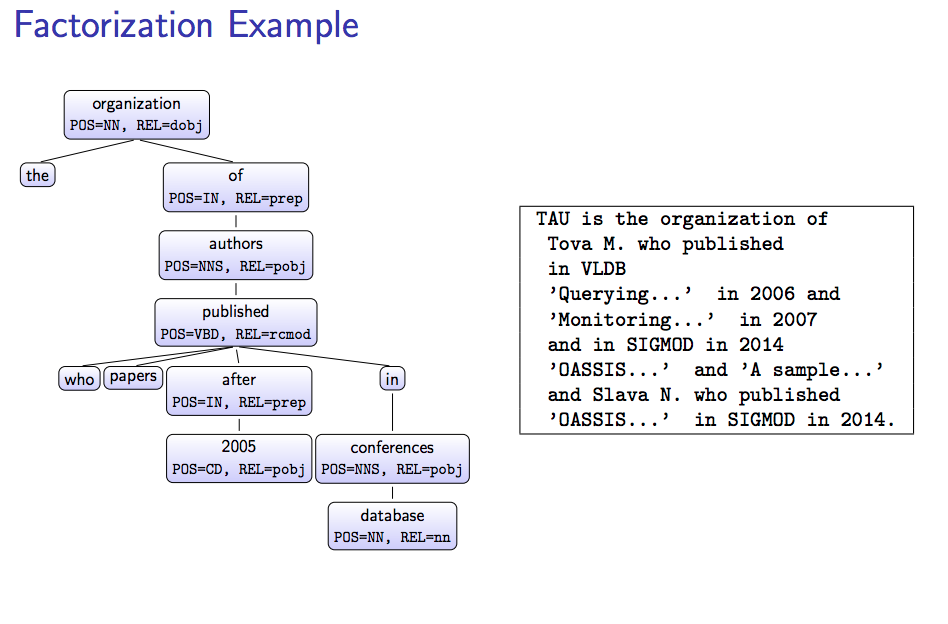
The translation performed by NaLIR from an NL query to a formal one can be captured by a mapping from (some) parts of the sentence to parts of the formal query.

**FACTORIZATION:**

Definition: Let P be a provenance expression. We say that an expression f is a factorization of P if f may be obtained from P through (repeated) use of some of the following axioms: distributivity of summation over multiplication, associativity and commutativity of both summation and multiplication.

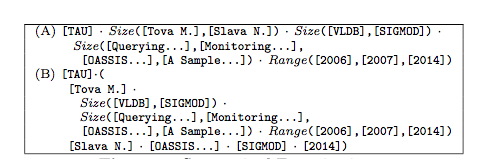
How do we measure whether a possible factorization is suitable/preferable to others? Standard desiderata are that it should be short or that the maximal number of appearances of an atom is minimal. On the other hand, we factorize here as a step towards generating an NL answer; to this end, it will be highly useful if the (partial) order of nesting of value annotations in the factorization is consistent the (partial) order of corresponding words in the NL query. We will next formalize this intuition as a constraint over factorizations.

For Example:

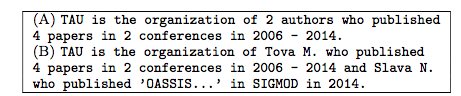


**SUMMERIZATION:**

we employ summarization, as follows. First, we note that a key to summarization is understanding which parts of the provenance may be grouped together. For that, we use again the mapping from nodes to query variables: we say that two nodes are of the same type if both were mapped to the same query variable. Now, let n be a node in the cir- cuit form of a given factorization f. A summarization of the sub-circuit of n is obtained in two steps. First, we group the descendants of n according to their type. Then, we summa- rize each group separately. The latter is done in our imple- mentation simply by either counting the number of distinct values in the group or by computing their range if the values are numeric. In general, one can easily adapt the solution to apply additional user-defined “summarization functions” such as “greater / smaller than X” (for numerical values) or “in continent Y” for countries.



Summarized Factorization



Summarized Sentences

**SAMPLE SCALIBILITY REPORT:**

Computation time as a function of the number of assignments.

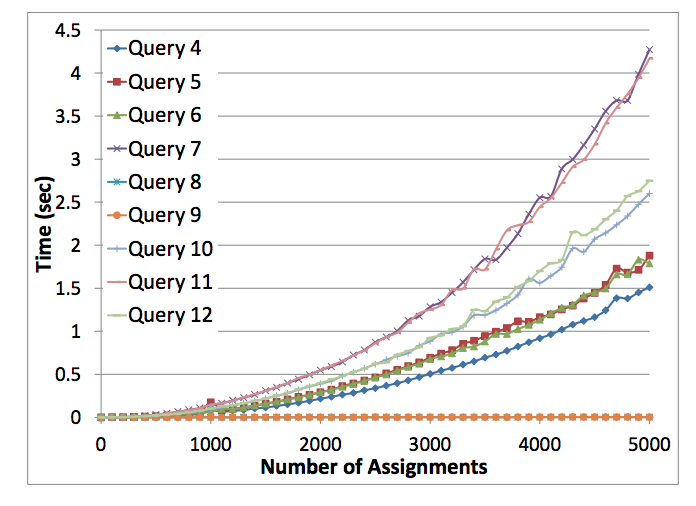
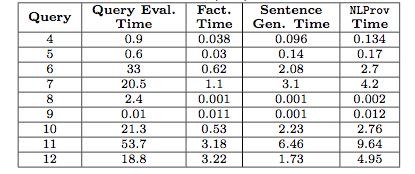


Table includes, for each query, the runtime required by our algorithms to transform provenance to NL in factorized or summarized form, for all query results (as explained in Section, we can compute the factorizations independently for each query result). We show a breakdown of the execution time of our solution: factorization time, sentence generation time, and total time incurred by NLProv (we note that the time to compute summarizations given a factorization was negligible). For indication on the complexity level of the queries, we also report the time incurred by standard (provenance-oblivious) query evaluation, using the mySQL engine. We note that our algorithms perform quite well for all queries (overall NLProv execution has 16% overhead), even for fairly complex ones.

**CONCLUSION:**

We have studied in this paper, for the first time to our knowledge, provenance for NL queries. We have devised means for presenting the provenance information again in Natural Language, in factorized or summarized form. There are two main limitations to our work. First, a part of our solution was designed to fit NaLIR, and will need to be replaced if a different NL query engine is used. Specifically, the “sentence generation” module will need to be adapted to the way the query engine transforms NL queries into formal ones; our notions of factorization and summarization are expected to be easier to adapt to a different engine. Second, our solution is limited to Conjunctive Queries. One of the important challenges in supporting NL provenance for further constructs such as union and aggregates is the need to construct a concise presentation of the provenance in NL (e.g. avoiding repetitiveness in the provenance of union queries, summarizing the contribution of individual tuples in aggregate queries, etc.).

**REFERENCE:**

* <https://github.com/navefr/NL_Provenance/>
* <http://www.cs.tau.ac.il/~amirgilad/papers/VLDB17.pdf>