# COVER: Covering the Semantically Tractable Questions

**Michael Minock**
KTH Royal Institute of Technology, Stockholm, Sweden
Umeå University, Umeå Sweden.
minock@kth.se, mjm@cs.umu.se

## Abstract

In semantic parsing, natural language questions map to meaning representation language (MRL) expressions over some fixed vocabulary of predicates. To do this reliably, one must guarantee that for a wide class of natural language questions (the so called *semantically tractable questions*), correct interpretations are always in the mapped set of possibilities. Here we demonstrate the system COVER which significantly clarifies, revises and extends the notion of semantic tractability. COVER is written in Python and uses NLTK.

## 1 Introduction

The twentieth century attempts to build and evaluate natural language interfaces to databases are covered, more or less, in (Androutsopoulos et al., 1995; Copestake and Jones, 1990). While recent work has focused on learning approaches, there are less costly alternatives based on only lightly naming database elements (e.g. relations, attributes, values) and reducing question interpretation to graph match (Zhang et al., 1999; Popescu et al., 2003). In (Popescu et al., 2003), the notion of *semantically tractable questions* was introduced and further developed in (Popescu et al., 2004). The semantically tractable questions are those for which, under strong assumptions, one can guarantee generating a correct interpretation (among others). A focus of the PRECISE work was reducing the problem of mapping tokenized user questions to database elements to a max-flow problem. These ideas were implemented and evaluated in the PRECISE system, which scored 77% coverage over the full GEOQUERY corpus.

Here we demonstrate the system COVER, which extends and refines the basic model as follows: a) we explicitly describe the handling of self-joining queries, aggregate operators, sub-queries, etc.; b) we employ theorem proving for consolidating equivalent queries and including only queries that are information bearing and non-redundant; c) we more cleanly factor the model to better isolate the role of off-the-shelf syntactic parsers. In practice, our extended model leads to improved performance (12% absolute coverage improvement) on the full GEOQUERY corpus.

## 2 COVER

Figure 1 shows the components of COVER. The only configuration is a simple domain specific lexicon which itself may be automatically derived from the database instance and lexical resources. There are three core processing phases, which generate a set of MRL expressions from the user's question and two auxiliary processes which use off-the-shelf syntactic analysis and theorem proving to prune MRL expressions from the constructed set. We cover the three core phases here and the two auxiliary processes in section 3. For more detail see (Minock, 2017).

### 2.1 Phase 1: Generating All Mappings

A *mapping* is defined as an *assignment* of all word positions in a user's question to database elements (relations, attributes, values). Thus a mapping for the six-word question "what are the cities in Ohio?" would consist of six assignments. Since eventually mappings might generate MRL expressions, each assignment is marked by a type to indicate its use as either a *focus*, a *condition* or a *stop* assignment (to be ignored).

In COVER's first phase candidate mappings for a question are generated. For example in the question "what are the cities in Ohio?", under the correct mapping: 'what' is assigned to `City` and is marked as a focus type; 'are' and 'the' are marked

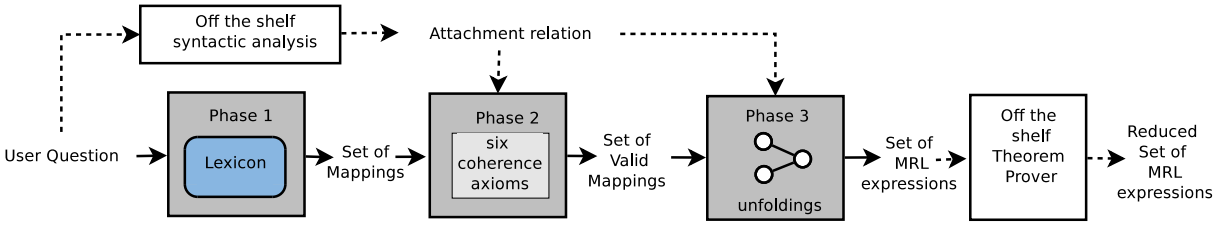


Figure 1: Overall processing paths of COVER.

as stop assignments and are assigned to the empty element; 'cities' is assigned to the relation `City` and is marked as condition type; 'in' is assigned the foreign key attribute `state` from the relation `City` to the relation `State` and is marked as a condition type; 'Ohio' is assigned to the value `'Ohio'` of attribute `name` of the relation `State` and is marked as a condition type. The knowledge to determine such mappings comes from a very simple domain lexicon which assigns phrases to database elements. (For example 'Ohio' in the lexicon is assigned to the database element representing the `State.name='Ohio'`). Note matching is based on word stems. Thus, for example, 'cities' matches 'city'.

### 2.2 Phase 2: Determining Valid Mappings

Valid mappings must observe certain constraints. For example the question "what is the capital of New York?" is valid when 'New York' maps to the state, but is not valid when 'New York' maps to the city. "What is the population of the Ohio River?" has no valid mappings because under the given database there is no way to fit the `population` attribute with the `River` table. Specifically the following are six properties that valid mappings must observe.

1. **There exists a unique focus element**. There always exists a unique focus element that is the attribute or relation upon which the question is seeking information. For efficiency this is actually enforced in phase 1.

2. **All relation focus markers are on mentioned relations**. If a relation is a focus, then it must be *mentioned*. A relation is mentioned if either it is explicitly named in the question (e.g. 'cities') or if a primary value[1] of the relation (e.g. 'New York' for the relation `City`) is within the question.

3. **All attribute focus markers are on mentioned, rooted attributes**. An attribute focus marker (e.g. 'what') must not only explicitly match an attribute (e.g. 'population'), but such an attribute must also be *rooted*. An attribute is rooted if the relation of the attribute is mentioned.

4. **Non-focus attributes satisfy correspondences**. Unless they are the focus, attributes must pair with a value (e.g. in "cities with *name Springfield*"), or, in the case that the attribute is a foreign key (e.g. "cities *in* the *state*..", the attribute must pair with the relation or primary value of the foreign key (e.g. "cities *in Ohio*").

5. **Value elements satisfy correspondences**. Values are either primary (e.g. "New York") or must be paired with either an attribute (e.g. "... city with the *name* of *New York*" ..."), or via ellipsis paired with a relation (e.g. "... the *city New York*").

6. **All mentioned elements are connected**. The elements assigned by the mapping must form a connected graph over the underlying database schema.

This leads to the core definition of this work:

**Definition 1** *(Semantically Tractable Question) For a given question $q$, lexicon $\mathcal{L}$, $q$ is semantically tractable if there exists at least one valid mapping over $q$.*

### 2.3 Phase 3: Generating Logical Formulas

Given a set of valid mappings, COVER's third phase is to generate one or more MRL expressions for each valid mapping. To achieve this we unfold the connected graph of valid mappings (see property 6 in section 2.2) into meaningful full

[1]Primary values(Li and Jagadish, 2014), stand, or for the most part stand, for a tuple of a relation. For example 'Springfield' is a primary value of City even though it is not a key value.

graphs. This is complicated in the self-joining case where graphs will have multiple vertices for a given relation. For example the valid mapping for "what states border states that border New York?" maps to two database relations: `State` and `Borders`. But the corresponding unfolded graph will be over three instantiations of `State` and two of `Borders`. Our algorithm gives a systematic and exhaustive process to compute all reasonable unfolded graphs. And then for each unfolded graph we generate the set of possible attachments of conditions and selections. In this end this gives a set of query objects which maybe directly expressed in SQL for application over the database or to first order logic expressions for satisfiability testing via an automatic theorem prover.

## 3 Auxiliary Processes

Figure 1 shows two auxiliary processes that further constrain what are valid mappings as well as which MRL expressions are returned.

### 3.1 Integrating Syntax

COVER uses an off-the-shelf parser to generate a syntactic attachment relation between word positions. This attachment relation is then used to sharpen the conditions in the properties 2, 3, 4 and 5 of valid mappings. In short correspondences and focus-value matches must be over word positions that are attached in the underlying syntactic parse. This has the effect of reducing the number of valid mappings. For example let us consider "what is the largest city in the smallest state?". If our syntactic analysis component correctly determines that 'largest' does not attach to 'state' and 'smallest' does not attach to 'city', then an erroneous second valid mapping will be excluded. Attachment information is also used to constrain which MRL expressions can be generated from valid mappings.

### 3.2 Integrating Semantics and Pragmatics

Many of the MRL expressions generated in phase three are semantically equivalent, but syntactically distinct. The second auxiliary process uses a theorem prover to reduce the number of MRL expressions (the shortest one from each equivalence class) that need to be paraphrased for interactive disambiguation. This is achieved by testing pairwise query containment over all the MRL expressions in the MRL set produced in the third phase.

| Case | Coverage | Avg/Med # interpretations |
|---|---|---|
| full | 780/880 (89%) | 7.59/2 |
| no-equiv reduction | 780/880 (89%) | 19.65/2 |

Table 1: Evaluation over GEOQUERY

Theorem proving is also used to enforce pragmatic constraints. For example we remove queries that do not bear information, or have redundancies within them that violate Gricean principles. This is principally achieved by determining whether a set-fetching query necessarily generates a single or no tuple where the answer is already in the question. For example a query retrieving "the names of states where the name of the state is New York and the state borders a state" does not bear information. Finally it should be noted, one can add arbitrary domain rules (e.g. *states have one and only one capital*) to constrain deduction. This would allow more query equivalencies and cases of pragmatics violations to be recognized.

## 4 Demonstration

Our demonstration, like PRECISE, is over GEOQUERY, a database on US geography along with 880 natural language questions paired with corresponding logical formulas. The evaluation method is exactly as in PRECISE(in conversation with Ana-Maria Popescu). First we prepare a lexicon over the GEOQUERY domain, then, given the set of 880 natural language/meaning representation pairs, the queries are run through the system and if the question is semantically tractable and generates one or more formal query expressions, then an expression equivalent to the target MRL expression must be within the generated set. Our experiment shows this to be the case.

Table 1 presents some results. By 'coverage' we mean, like in the PRECISE evaluation, that the answer is in the computed result set. Table 1 shows results for two cases: *full* has all features turned on; *no-equiv reduction* shows results when the auxiliary process described in section 3.2 is disengaged. Clearly we are benefiting from the use of a theorem prover which is used to reduce the size of returned query sets.

At EACL we will run our evaluation on a laptop, showing the complete configuration over GEOQUERY and welcoming audience members to pose interactive questions.

## 5 Discussion

To apply `Cover`, consider our strong, though achievable assumptions: a) *the user's mental model matches the database*; b) *no word or phrase of a question maps only to incorrect element, type pairs*; c) *all words in the question are syntactically attached*. Under such assumptions, every question that we answer is semantically tractable, and thus a correct interpretation is always included within any non-empty answer. Let us discuss the feasibility of these assumptions.

With respect to **assumption a**, it is difficult to determine if a user's mental model matches a database, but, in general, natural language interfaces do better when the schema is based on a conceptual (e.g. Entity-Relationship) or domain model, as is the case for GEOQUERY. Still COVER is not yet fully generalized for EER-based databases (e.g. multi-attribute keys, isa hierarchies, union types, etc.). We shall study more corpora (e.g. QALD (Walter et al., 2012)) to see what type of conceptual models are required.

With respect to **assumption b**, an over-expanded lexicon can lead to spurious interpretations, but that is not so serious as long as the right interpretation still gets generated. Any number of additional noisy entries could be added and our strong assumption b) would remain true. While we are investigating how to automatically generate 'adequate' lexicons (e.g. by adapting domain-independent ontologies, expanding domain-specific lists, or using techniques from automatic paraphrase generation), the question of how the lexicon is acquired and accessed (e.g. Querying over an API would require probing for named entities rather than simple hash look-up, etc.) is orthogonal to the contribution of this work.

We make **assumption c** because we want to guarantee that COVER lives up to the promise of always having within its non-empty result sets the correct interpretation. Still the control points for syntactic analysis are very clearly laid out in COVER. Given that interfaces should be usable by real users, keeping the interpretation set manageable while trying to keep the correct one in the set is useful, especially if a database is particularly ambiguous. Exploring methods to integrate off-the-shelf syntactic parsers to narrow the number of interpretations while not excluding correct interpretations will be future work. Specifically we will evaluate which off-the-shelf parsers and which notions of 'attachment' perform best.

A final question is, *is the semantically tractable class fundamental?* COVER has generalized the class from the earlier PRECISE work and nothing seems to block its further extension to questions requiring negation, circular self-joins, etc. Still, we expended considerable effort trying to extend the class to include queries with maximum cardinality conditions (e.g. "What are the states with the most cities?"). This effort foundered on defining a decidable semantics. But we also witnessed many cases where spurious valid mappings were let in while not finding a correct valid mapping ('most' seems to be a particularly sensitive word). Further study is required to determine the natural limit of the semantic tractability question class. How far can we go? Is there a limit? Our intuition says 'yes'. A related question is *is there a hierarchy of semantically tractable classes where the number of interpretations blows up as we extend the number of constructs we are able to handle?* Again, our intuition says 'yes'. COVER will be the basis of the future study of these questions.

## References


Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases. *NLE*, 1(1):29–81.

Ann A. Copestake and Karen Sparck Jones. 1990. Natural language interfaces to databases. *Knowledge Eng. Review*, 5(4):225–249.

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

Michael Minock. 2017. Technical description of COVER 1.0. *CoRR*.

Ana-Maria Popescu, Oren Etzioni, and Henry A. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *IUI-03, January 12-15, 2003, Miami, FL, USA*, pages 149 – 157.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases. In *COLING*, pages 141 – 147.

Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. 2012. Evaluation of a layered approach to question answering over linked data. In *ISWC, Boston, MA, USA*, pages 362–374.

Guogen Zhang, Wesley W. Chu, Frank Meng, and Gladys Kong. 1999. Query formulation from high-level concepts for relational databases. In *UIDIS*, pages 64–75.