Corpora for Automatically Learning to Map Natural Language Questions into SQL Queries

Alessandra Giordani, Alessandro Moschitti

Department of Computer Science and Information Engineering
University of Trento
Via Sommarive 14
38100 - POVO (TN) - Italy
agiordani, moschitti@disi.unitn.it

Abstract

Automatically translating natural language into machine-readable instructions is one of major interesting and challenging tasks in Natural Language (NL) Processing. This problem can be addressed by using machine learning algorithms to generate a function that find mappings between natural language and programming language semantics. For this purpose suitable annotated and structured data are required. In this paper, we describe our method to construct and semi-automatically annotate these kinds of data, consisting of pairs of NL questions and SQL queries. Additionally, we describe two different datasets obtained by applying our annotation method to two well-known corpora, GEOQUERIES and RESTQUERIES . Since we believe that syntactic levels are important, we also generate and make available relational pairs represented by means of their syntactic trees whose lexical content has been generalized. We validate the quality of our corpora by experimenting with them and our machine learning models to derive automatic NL/SQL translators. Our promising results suggest that our corpora can be effectively used to carry out research in the field of natural language interface to database.

1. Introduction

In this research, we present the design, construction and use of two datasets each consisting of pairs of NL questions and SQL queries. We represent these pairs by means of their syntactic trees and we use kernel functions and machine learning algorithms to derive the mapping between the two languages.

In particular, starting from a given set of correct pairs of questions and the related SQL queries, we produce incorrect pairs and additional correct pairs, i.e. negative and positive examples, respectively. Then we model a representation of the above question/query pairs in terms of syntactic structures, i.e. we build pairs of syntactic parse trees automatically derived by off-the-shelf natural language parsers and an ad-hoc SQL parser.

The resulting dataset is then used to learn the mapping between the two languages. First we train a classifier on the above pairs for selecting the correct queries for a question. Then, for a new question and a given set of available queries, we produce the set of pairs containing such question and then we use the classifier to rank pairs in terms of correctness. We select the top scored pair to find the query that answers the given question.

We ran experiments starting from the available datasets GEOQUERIES and RESTQUERIES (Tang and Mooney, 2001), creating two new corpora. Results show that the approach is viable since we obtain a fairly high accuracy and that indeed the corpora we generated and evaluated are valid language resources.

In the remainder, Section 2 shows the methodology to construct and annotate the corpora, Section 3 discusses the experimental setup and results and finally, Section 4 draws conclusions.

2. Dataset Construction and Annotation

The goal of this research is to map NL questions into SQL queries based on a machine learning approach; consequently, we need to have training data, i.e. a set of positive and negative examples. In practical cases, we can assume to have a set of positive examples consisting of correct question/query pairs, i.e. such that the execution of the query retrieves a correct answer for the question. Assuming the availability of negative examples is a more strong assumption since providing the correct query for an user information need is a more natural task than providing the incorrect solution.

Therefore, we need techniques to generate negative examples from an initial set of correct pairs. Unfortunately, this is not a trivial task since when mixing a question and a query belonging to different pairs we cannot assume to only generate incorrect pairs, e.g. when swapping two different queries x with y in the two pairs:
$\langle$*What are some good restaurants in San Francisco?*, $x\rangle$
$\langle$*What is a good place in San Francisco?*, $y\rangle$
we obtain other two correct pairs. To generate a gold standard dataset we would need to manually check this aspect thus we design an algorithm to limit the human supervision. It consists of the following steps:

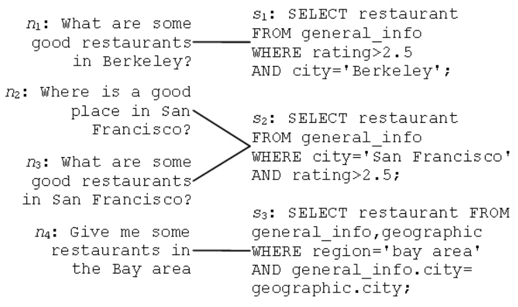
- Generalizing concept instances: substitute the involved concepts in questions and their related field values in the SQL queries, e.g. WHERE condition, by means of variables expressing the category of such values (e.g. San Francisco becomes *VarCity*).

- Clustering generalized pairs: each cluster represents the information need about a target semantic concept, e.g. “good restaurants in VarCity”, common to questions and queries. The clustering can be perfomed semi-automatically exploiting the semantic equivalency between the pairs' members and its transitivity closure (requires a limited human supervision for validation).

- Pairing questions and queries of distinct clusters, i.e. the Cartesian product between the set of questions and the set of queries belonging to the pairs of a target cluster. This allows to find new positive examples that were not present in the initial corpus.
- Final dataset annotation: consider all possible pairs, i.e. Cartesian product between all the questions and queries of the dataset, and annotate them as negatives if they have not been annotated as positives in the previous steps.

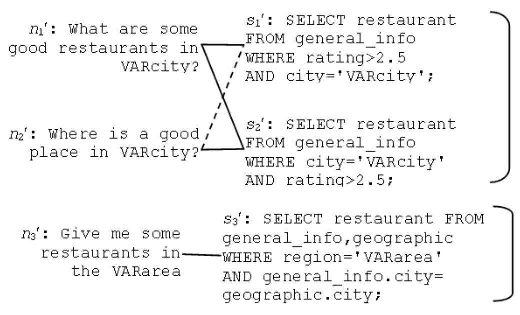
An example of the entire process is shown in Figure 1 and 2, which reports questions and queries of a restaurant domain. More in detail, in Figure 1 there is a set of four pairs containing four distinct questions and their three related queries (connected by lines) whereas in Figure 2 four generalized pairs are shown.

In the question and query pair $\langle n_1, s_1\rangle$, since in s_1 *Berkeley* is associated with the column *city*, its occurrences in n_1 and s_1 are *generalized* with the concept/variable *VARcity*. We note that, after substituting instances with variables, both n_1 and n_3 are generalized into n'_1, which can then be paired with two distinct SQL queries, i.e. s'_1 and s'_2. This is correct since there can be more SQL queries that correctly retrieve an answer to an NL question. We define them to be *semantically equivalent*, i.e. $s'_1 \equiv s'_2$. Conversely, there can be many NL questions that map to the same query, e.g. $n_2 \equiv n_3$.



```
n1: What are some          s1: SELECT restaurant
good restaurants           FROM general_info
in Berkeley?               WHERE rating>2.5
                           AND city='Berkeley';
n2: Where is a good
place in San               s2: SELECT restaurant
Francisco?                 FROM general_info
                           WHERE city='San Francisco'
n3: What are some          AND rating>2.5;
good restaurants
in San Francisco?
                           s3: SELECT restaurant FROM
n4: Give me some           general_info,geographic
restaurants in             WHERE region='bay area'
the Bay area               AND general_info.city=
                           geographic.city;
```

Figure 1: Example of the initial corpus



```
n1': What are some         s1': SELECT restaurant
good restaurants in        FROM general_info
VARcity?                   WHERE rating>2.5
                           AND city='VARcity';

n2': Where is a good       s2': SELECT restaurant
place in VARcity?          FROM general_info
                           WHERE city='VARcity'
                           AND rating>2.5;

n3': Give me some          s3': SELECT restaurant FROM
restaurants in             general_info,geographic
the VARarea                WHERE region='VARarea'
                           AND general_info.city=
                           geographic.city;
```

Figure 2: Generalized corpus, divided in two clusters (*identified by the two brackets*).

Once the pairs have been generalized, we *cluster* them according to their semantic equivalence so that we can automatically *derive new positive* examples by swapping their members. For example, in Figure 2 s'_1 and s'_2 retrieve the same results so we manually verify that they are semantically equivalent queries and we assign them to the same cluster (CL1), i.e. information need about good restaurants in a city (with a rating larger than 2.5 stars). Alternatively, we can also consider that n'_1 and n'_2 are both paired with s'_2 to derive that they are equivalent, avoiding the human intervention. Concerning s'_3, it retrieves a result set different form the previous one so we can automatically assign it to a different cluster (CL2), i.e. involving questions about restaurants in a region. Note that, once n'_2 is shown to be semantically equivalent to n'_1, we can pair them with s'_1 to create the new pair (indicated by the dashed line) $\langle n'_2, s'_1\rangle$. Indeed the negative example set is $\langle n'_3, s'_1\rangle, \langle n'_3, s'_2\rangle, \langle n'_1, s'_3\rangle, \langle n'_2, s'_3\rangle$.

Human intervention is required when there are two clusters whose queries retrieve the same result set but are not apparently related to equivalent questions. In such cases the semantic equivalence of questions should be checked by an human, that indeed doesn't need to be an SQL expert. Suppose that we have two clusters, one regarding infomation need about best restaurant in a city and one about Italian restaurant in a city. It could be the case that for a certain city the best restaurant is also Italian, so both queries retrieves the same result. In general these two concepts are different but if this information is not present in the database, the algorithm can't decide if the new pair is a positive or a negative example after its members are swapped.

It is worth noting that with the generalization process, we introduce redundancy that we eliminate by removing duplicated questions and queries. Thus, the output dataset is usually smaller than the initial one. However the number of training examples will be larger, not only because of the introduction of negatives but also due to the automatic discovering of new positives.

Next section illustrates the experiments on two datasets we generated using the above algorithm. In particular, since we want to carry out mapping at syntactic level, we represent pairs by means of their syntactic trees. For deriving question parse tree we use the Charniak's syntactic parser (Charniak, 2000) while for queries we implemented an adhoc SQL parser (Giordani and Moschitti, 2009a). The tree representation of pair $\langle n'_1, s'_1\rangle$ is shown in Figures 3 and 4.

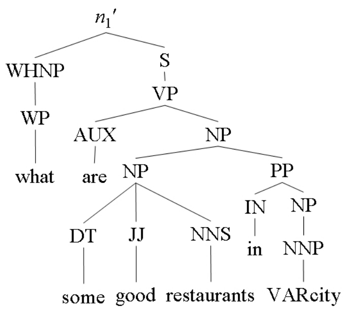


Figure 3: Syntactic parse tree of generalized question n'_1.

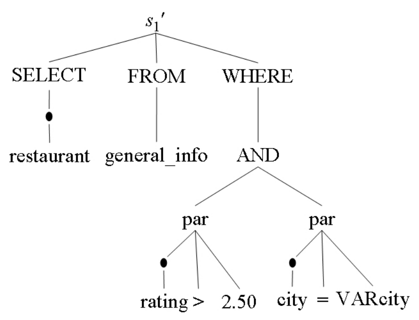


Figure 4: Syntactic parse tree of generalized query s_1'.

3. The Experiments

In order to derive an automatic translator of natural language (NL) questions into their associated SQL queries we start from a corpus of pairs annotated as correct when the SQL query answers to the question and incorrect otherwise. Then we train SVMs, encoding the structural representation of the pairs by means of different types of kernels (i.e. linear, polynomial and tree kernels and their combinations) to automatically exploit the associative patterns between NL and SQL syntax. Finally to map new questions in the dataset of the available queries, we rank the latter by means of the question classifier score and select the top one. One can argue that the set of all possible queries is infinite and that this approach is practically useless. Actually the queries usually asked towards a database are a finite subset of meaningful queries that can be generated starting from database metadata, e.g. MySQL INFORMATION-SCHEMA, and applying SQL grammar rules.

For a formal definition of our learning approach we remind the reader to the work described in (Giordani and Moschitti, 2009b; Giordani and Moschitti, 2009a).

To generate the datasets we applied our algorithm to GEOQUERIES250 and RESTQUERIES corpora. Questions in both corpora were originally collected from a web-based interface and manually translated into logical formulas in Prolog by Mooney's group (Tang and Mooney, 2001). Popescu et al. (Popescu et al., 2003) manually converted them into SQL. Thanks to our clustering algorithm we discovered and fixed many errors and inconsistencies in SQL queries. The first corpora is about geography questions. After the generalization process the initial 250 pairs were reduced to 155 pairs containing 154 NL questions and 79 SQL queries. We found 76 clusters, from which we generated 165 positive and 12.001 negative examples. The second dataset regards questions about restaurants. The initial 250 pairs were generalized by 197 pairs involving 126 NL questions and 77 SQL queries. We clustered these pairs in only 26 groups which led to 852 positive and 9.702 negative examples.

We experimemted with both extended corpora using 10-fold cross validation. For each corpus, every fold contains a different subset of generalized questions (10%) paired with all available generalized queries. So, in the first corpora, every fold consists of approximately 15 questions each paired with 79 SQL, among which almost only one retrieves the correct answer. In the second one, every fold contains an average of 12 questions each paired with 77 SQL where almost four of them are correct pairs.

Table 1: Kernel combination accuracies (± Std. Dev)

Corpus	**BOW^2**	**TK^2**	**Advanced Best Kernel**
GEO	70.7±12.0	75.6±13.1	75.9±9.6
REST	37.1±16.2	74.5±14.0	84.7±11.5

We tested several models for ranking questions based on different kernel combinations (the interested reader can find their definition in (Giordani and Moschitti, 2009b; Giordani and Moschitti, 2009a)). Here, in Table 1, we just report mapping accuracy of using the set of word pairs as features from Questions and Queries: BOW^2, and the accuracy when tree kernels, i.e. pairs of tree fragments, are used as feature spaces, i.e.TK^2. In addition we list the best accuracy obtained with our advanced kernel combinations. The accuracy shown in the table is the rate of the number of times that the top ranked pair is the correct one[1]. This is fairly high considering that due to the heavily skewed class distribution of test sets the probability of randomly choosing the correct one among all the pairing is very low for the first corpus (1.4%) and a bit higher for the second one (5.6%).

The achieved accuracy is *near* the one of the best systems, 77.5% (GEOQUERIES) and 95.5%(RESTQUERIES), achieved by Krisp (Kate and Mooney, 2006). Note that such state-of-the-art system use, instead of an off-the-shelf parser (like our system), semantic trees manually designed (encoding the minimum representation semantics). Moreover, while also in evaluating Krisp, Kate and Mooney use 10-fold cross validation, our test sets are less trivial. Indeed original corpora were redundant in that both questions and queries were instantiated many times with similar values. Consider for example questions n_1 and n_3 in Figure 1. When evaluationg Krisp it may have happened that one of those similar questions was used for training the parser and the other to test it. As opposite using our extended corpora, since questions are generalized, it is never the case that the same question appear both in the training set and in the test set. There exist other state-of-the-art systems (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Tang and Mooney, 2001; Ge and Mooney, 2005) that were tested on an extension of GEOQUERIES but with different experimental-setup, so results are not directly comparable. However we perform similarly to Krisp, that compares favourably with them.

This demonstrates that our corpora are valid and accurate language resources for learning question-queries relationships.

4. Conclusions

In this paper, we describe the methodology to generate annotated data that we use to derive a mapping between natural language and programming language by automatically learning a model based the syntactic representation of the

[1]Although the Std. Dev. associated with the model accuracy is high, the one associated with the distribution of difference between the model accuracy is much lower, i.e. 5%

training examples. In our experiments we consider pairs of NL questions and SQL queries as training examples. These are annotated by means of our algorithm starting from a given initial annotation. In particular we experimented with the annotation available in GeoQueries and RestQueries corpora. We generated new datasets adding new positive pairs[2], creating negatives example set and also fixing some errors. Experimental results show that the evaluated corpora are valid and accurate language resources.

5. Acknowledgements

We would like to thank Rohit Kate for his help in obtaining the GeoQueries and RestQueries datasets. Special thanks go to Periklis Andrisos for providing the initial problem and discussing possible solutions. This research has been partially supported by the EC project "Trustworthy Eternal Systems via Evolving Software, Data and Knowledge" (EternalS, project number FP7 247758).

6. References

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*.

Ruifang Ge and Raymond Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 9–16, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Alessandra Giordani and Alessandro Moschitti. 2009a. Semantic mapping between natural language questions and sql queries via syntactic pairing. In *Proceedings of the 14th International Conference on Applications of Natural Language to Information Systems*. Springer-Verlag.

Alessandra Giordani and Alessandro Moschitti. 2009b. Syntactic structural kernels for natural language interfaces to databases. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, pages 391 – 406. Springer-Verlag.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st ICCL and 44th Annual Meeting of the ACL*, pages 913–920, Sydney, Australia, July. Association for Computational Linguistics.

Ana-Maria Popescu, Oren A Etzioni, and Henry A Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces*, pages 149–157, Miami. Association for Computational Linguistics.

L. R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477, Freiburg, Germany.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA, June. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666.

[2]The datasets will be publicly available at `http://disi.unitn.it/~iKernels/corpora.htm`