Natural Language Interface for Relational Database

Midterm Report

Yilin Gao yilin.gao@duke.edu
Keping Wang kw238@duke.edu
Chengkang Xu cx33@duke.edu

October 25, 2016

1 Introduction

- Motivation of the problem with practical examples – why the problem is interesting and important.

  This project is going to build a natural language interface for relational databases, following the ideas of Fei Li (2014) Constructing an Interactive Natural Language Interface for Relational Databases. Here we only focus on how to select data (no insert, add, drop).

  This project is of course meaningful. Since SQL queries are not easy for everyone to learn, this interface will be an excellent tool for everyone to query data from relational databases. For example, users of a searching engine are very likely type in a question like "Which river is the longest river in the world?", and the searching engine should transform this question into SQL queries and execute it against its huge database. If the searching engine is not able to understand the meaning of the question correctly, it will fail to provide users with correct answer.
  People have been working on this for decades, but it is really more about the understanding of natural language rather than the generating of SQL queries. Because SQL queries have well defined grammars and all the tricky ambiguities lie in natural languages. Thankfully, nowadays we have high accuracy dependency syntax parser to make NLIDB really feasible. In this course project, we are both interested in natural language processing and its application in database application.

- (Briefly) What previous approaches have done.

  In the past, many natural language interfaces to databases (NLIDBs) has been built to better translate natural language into database queries. NLIDBs have many advantages over other widely used accepted query interfaces, but they have not been adopted widely. This is because understanding natural language is hard. It is important for NLIDBs to improve its parsing and transformation systems, and interact with users more effectively. [2]

- Overview of what you plan to do (in the proposal) or have done (midterm/final report).

  We intend to closely follow the paper *Fei Li 2014*. The main steps of translating a natural language to an SQL query are as follows:

  – Parse the natural language using a dependency parser (Stanford NLP Parser).

  – Map the parse tree nodes to SQL keywords and table and attribute names.

  – Adjust the structure of the parse tree to make it follow the structure of an SQL query. The result of this procedure is then called a query tree.

  – Translate the query tree to an SQL query.

  – During step 2 and 3 there is interactive communication with the user to let the user choose the desired mappings or structures out of ranked choices.

Steps 2 - 4 are the most complicated parts, because we need to code up the grammatical rules that the parse trees should follow to get translated to SQL queries, which requires deep understanding of the SQL language. **Each member of our group will focus on one step among steps 2 - 4.** Also there will be a user interface in our system coded with Java.
We plan to use *Microsoft Academic Search Database* for testing as in *Fei Li 2014*, but PostgreSQL as our database (since we've learned to use it in CS516).

Possible obstacles we might encounter are:

- The system is really complicated, and except for the available Standford NLP Parser, we need to code the user interface, parser tree node mapper, parser tree structure adjuster and the query tree translator by ourselves.

- We are not very familiar with NLP, and have no experience in translating natural language into SQL language, so we should study related knowledge by ourselves.

2 Related Work

- A survey of related work on all aspects of the problem.

NLIDBs which we will study belongs to QA systems. Modern QA systems have developed into many categories, like natural language, visual questions, and AI ability test. Here we focus on the development of natural language QA systems. These systems can develop on structured data like databases and semi-structured or unstructured data like web information. [1]
Early work in natural language database systems is usually based on small scale database. These databases are of simple schema and small numbers of entities, thus can only answer a small set of questions. Moreover, their natural language parsing systems can only support ad-hoc methods and semantic parsing. In other words, early work will produce ambiguity if database is scalable and natural language queries are "open-domain". And without the help of machine learning methods, early QA systems cannot update their parsing methods as they accumulate more data.
As research goes deeper, natural language database systems can support larger databases and more complicated query formats. The mainstream approach is semantic parsing of questions. It can map natural language questions to logic forms or structured queries, and produce accurate answers when the query is complete and clear. And there exist several challenges in this method. The accuracy of answering questions will decline if the input language is ambiguous, if the logic relationship of the query is complicated, or if the database is very large. In recent years, a huge amount of literature has developed many innovative methods to reduce ambiguity during interpretation of natural language, transformation from natural language into SQL queries, and searching into database. These attempts have achieved satisfactory improvements in output accuracy. But ambiguity due to complexity in query languages and datasets is still the major challenge in this field, and researchers are still working on new methods to improve QA systems' performance.

- How your problem and the solutions (would) differ from them.

Our problem differs from previous literature in that, we mainly focus on how to transform parsed natural language into query trees and SQL queries, and we will use a third-party package to finish parsing tasks. But at present, research focuses on how to increase parsing accuracy with cutting-edge methods in machine learning, and how to increase accuracy when the query structure is complicated and the dataset is huge. This is because our major purpose is to get hands-on experience in how to establish such a large system and how the system operates. We can focus on improving system efficiency in future

work.

3 Problem Definition

- Explain all the notations you will be using.

  NLIDBS - Natural Language Interfaces to Databases, a computer human interface that is capable of understanding varieties of ambiguous input and executing corresponding actions on database.
  Parse tree structure adjustor - It is used to adjust the sequence of the SQL keywords that are parsed from mapper to correctly understand and execute the intended query. Once the natural language keywords have been matched with SQL keywords, it will inserted some keywords if missing to better interpret the input and generate valid SQL query.
  Parse tree node mapper - This is a mapper that maps keywords in natural language to SQL keywords. The mapper will try to output the most accurate match between natural language keyword and SQL keyword. If no match is found, it will ignore the word. Our current strategy is to mapping keywords based on its neighbour words (i.e. if two words are adjacent, they are considered as relevant and it will affect the interpret of words that have multiple interpretations).
  Dependency parser - Dependencies that is used to parse natural language query linguistically. We will be using Stanford Parser to perform the task.
  NV node - Every node in the parse tree, which has one or more candidate map is an NV node.

- Include reviews of background material.

  In Fei Li's research, he states several methods that he used in building the NLIDBS. First of all, he defined different types of nodes. Every node type has a specified pool of corresponding natural language words. With the definition of these types, keywords in natural language can be organized more accurately. Second, he proposed that in order for mapping to work more accurately, database schema should be meaningful and human-legible. When similar meaning words present, they will be more likely to be matched with corresponding SQL keywords correctly. Third, after all keywords have been translated into SQL keywords, there will be more than one candidate maps returned for user to choose. Once user chooses the one that fits user's intended query the best, the candidate map is finalized. Li's paper proposed procedures for handling and parsing natural language into SQL. [2] Follow the idea, we will implement our own algorithm for solving similar problems.

- Give a formal definition of your problem.

  Currently, there is not many easy-to-use NLIDBS available. Our goal is to create an interface that allows user to input natural language and process it into SQL. Users can type in query in natural language in the text box. Parser will validate the input and execute corresponding query.

4 Algorithms/Our Results

midterm: include
final: update

- **Technical contributions of your project.**
- **Include theorems, lemmas, algorithms, proofs, running time discussion.**

5 Implementation or System

midterm: include
final: update

- **Description of your system or prototype (unless your project is a theoretical one).**
- **Describe the architecture with a diagram.**
- **Explain the components.**

6 Experiments

midterm: include
final: update

- **Detailed experiments (unless your project is a theoretical one).**
- **The machine configuration where you ran your experiments.**
- **Datasets, no. of tuples, no. of attributes, size, what they store (add references).**
- **Performance (vary all parameters).**
- **Qualitative Evaluation (whether the system returns meaningful results).**
- **Comparison with other prior approaches.**

7 Observations

final: include

- **Observations from your project.**

8 Future Work and Conclusions

final: include

- **Scope of future work for your project.**

9 Contributions of Project Members

midterm: include
final: update

- **Write down the contributions of each member here.**

1. **Author-1**
   - **Midterm:**
   - **Final:**
2. **Author-2**
   - **Midterm:**
   - **Final:**
3. **Author-3**
   - **Midterm:**
   - **Final:**

References

[1] Question and answer tutorial. https://github.com/scottyih/Slides/blob/master/QA%20Tutorial.pdf. Accessed: 25-09-2016.

[2] F. Li and H. Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.