EC601 Project1
Li Cao (licao@bu.edu)
BU ID:U60827070

Topic: User Stories to Application launch without humans

Problem statement

The topic is User Stories to Application launch without humans, and this project was born because: there is a huge demand for software and data engineers right now. Therefore, it is desirable to automate common software engineering development tasks while leaving complex tasks to the experts.

On the one hand, it can free some software engineers from simple tasks. On the other hand, it also provides a simple platform for people without any foundation to initially fulfill their requirements.

Application

Even a very simple script or application is challenging to understand for people without basic knowledge. At present, various scenarios inevitably require some intelligent application or program design, and software engineers are needed in many places to complete the corresponding format. However, some functions' human input and capital expenditure are unnecessary. In addition to software design, we also have many areas worth exploring.

For enterprises, in many cases, communication with customers cannot be completed by technical personnel due to confidentiality. At this time, the realization of functions is explained by product requirements documents. However, product managers and other non-technical personnel spend time communicating with customers and conveying this step to technical personnel, which can easily lead to some misunderstandings in communication. People who are not programmers can automatically generate the general content framework required by simple descriptions without needing technology, which can significantly improve efficiency. For the parts that can be open-sourced, functional designs can be used directly, freeing many programmers and account managers from some elementary product requirements and saving unnecessary labor costs. For the part that requires technical confidentiality, at least it can save much time for professional and technical personnel in programming.

Therefore, the goal of this project is to automatically generate applications that meet the descriptive features by inputting natural language with descriptive features. The overall process includes the following two parts.

First: describe the requirements according to the specified format and convert the natural language into instructions that can be recognized by the machine through computer processing. Keyword tags, element extraction. can be used in natural language processing to obtain a simplified and convenient Machine understanding of demand instructions.

Second: modularize some basic functional components to generate business code more conveniently in the future. Using the demand instructions obtained in the first step, combine various basic functions according to the original logic, and get a front-end application can run normally, and the back-end application can use the Firebase platform provided by Google to process the business requests initiated by the front-end.

For example, to generate common e-commerce applications, basic functions can be realized through automation, including completing user login, logout, purchase history, shopping list. It can also be used to generate social software to realize account login and logout, display friend list, and send messages.

Google Firebaseis a mobile application development platform from Google with powerful features for developing, handling, and enhancing applications. Firebase is a backend platform for building web and mobile applications.

Firebase is fundamentally a collection of tools developers can rely on, creating applications and expanding them based on demand. Firebase aims to solve three main problems for developers: Build an app, fast ;Release and monitor an app with confidence ;Engage users. Developers relying on this platform get access to services that they would have to develop themselves, and it enables them to lay focus on delivering robust application experiences. Some of the Google Firebase platform's standout features include databases, authentication, push messages, analytics, file storage, and much more.

My Focus

The objective of my focus that interests me is to generate accurate and valid SQL queries after parsing natural language using open source tools and libraries. Users will be able to obtain SQL statement for the major 5 command words by passing in an English sentence or sentence fragment. We wish to do so in a way that progresses the current open source projects towards robustness and usability.

SQL is tool for manipulating data. To create an system which can generate a SQL query from natural language we need to make the system which can understand natural language. Most of the research done until now solve this problem by teaching a system to identify the parts of speech of a particular word in the natural language which is called tagging. After this the system is made to understand the meaning of the natural query when all the words are put together which is called parsing. When parsing is successfully done then the system generates a SQL query using proper syntax of MySQL.

Following corpus development which was helping computer identify tables and columns name.

The following might be an algorithm:

(1)Scanning the database: Here we will go through the database to get the table names, column names, primary and foreign keys.

(2)Input : We will take a sentence as a input from the user (using input.txt)

(3)Tokenize and Tag : We will tokenize the sentence and using POS tagging to tag the words

(4)Syntactic parsing : Here we will try to map the table name and column name with the given natural query. Also, we will try to identify different attributes of the query.

(5)Filtering Redundancy : Here we will try to eliminate redundancy like if while mapping we have create a join requirement and if they are not necessary then we remove the extra table.

(6)Query Formation : Here we will form a complete SQL query based on MySQL syntax.

(7)Query Execution : Here we will execute the query on database to get results

Text-to-sql different method

ATIS&GeoQuery

ATIS generates SQL statements from user questions and is a single-domain, context-sensitive dataset. GeoQuery is derived from US geography and includes 880 questions and SQL statements, and is a single-domain, context-independent dataset.

WikiSQL

The two datasets, ATIS & GeoQuery, suffer from small data sizes (less than a thousand SQL sentences) and simple annotation. So, in 2017, VictorZhong and other researchers annotated 80654 training data covering 26521 databases named WikiSQL. A series of excellent models were generated in the subsequent work, among the representative works: seq2sql, SQLNet, TypeSQL.

Spider

But WikiSQL also has problems. Each of its problems only involves a table and also only supports relatively simple SQL operations. This is not very consistent with our daily life scenarios. In real life, there are databases in various fields such as medical, ticketing, schools, transportation, etc., and each database has dozens or even hundreds of tables, and the tables have complex primary and foreign key links between them.

So, in 2018, researchers at Yale University introduced the Spider dataset, which is the most complex Text-to-SQL dataset available. It has the following features: 1) the domain is richer, with more than 200 databases from 138 domains, each corresponding to 5.1 tables on average, and the databases appearing in the training set and test set does not overlap. 2) the SQL statements are more complex, containing orderBy, union, except, groupBy, intersect limit, having keywords, nested queries, etc.

Sparc

The Yale University research team followed up with SParC, a context-sensitive version of Spider. The database is based on Spider and simulates users performing the database.

Pointer Network

The word list used in the decoder part of the traditional seq2seq model is fixed, i.e., the words are selected from a fixed word list in the generated sequence. However, Text-to-SQL is different from the general seq2seq task in that: a) words in the Question; b) SQL keywords; c) table names and column names in the corresponding database may appear in the generated sequence.

Pointer Network solves this problem well, and the word list used in its output is variable with the input. This is done by using the attention mechanism to select words directly from the input sequence as output.

In the Text-to-SQL task, consider the user's Question and other words that may appear in the target SQL statement as the input sequence (column name word sequence; keyword list of SQL; word sequence of the Question), and use PointerNetwork to directly select words from the input sequence as the output. At each step of the decoder, the attention score is calculated with each implicit state of the encoder, and the maximum value is taken as the current output as well as the input for the next step.

Seq2SQL

Although Pointer Network solves the problem to some extent, it does not take advantage of the syntactic structure inherent in SQL statements. seq2SQL divides the generated SQL statement into three parts: aggregation operations: (SUM, COUNT, MIN, MAX, etc.), SELECT: selected columns, and WHERE: query conditions. Each part uses a different method for calculation.

SELECT and aggregation operations both use an attention mechanism for classification. The WHERE clause can be trained using the Pointer Network introduced earlier, but for many queries, the WHERE clause is not written in a unique way. This may lead to an otherwise correct output being judged as wrong. So the authors propose to use reinforcement learning to optimize based on the query results. In the decoder part, the possible outputs are sampled to produce several SQL statements, each denoted as y = [y1, y2 ... yT], and each sentence is scored using the scoring function.

SQLNet

In order to solve the problem that Seq2SQL is not effective in using reinforcement learning, SQLNet divides the SQL statement into two parts, SELECT and WHERE, sets several slots in each part, and only fills the slots with the corresponding symbols.

The SELECT clause part is similar to Seq2SQL. The difference lies in the WHERE clause, which uses a sequence-to-set (generating a set from a sequence) mechanism for selecting the columns that are likely to appear in the WHERE clause of the target SQL statement. A probability is given for each column in the table. After that, the number of conditions k in the WHERE clause is calculated, and then the top k columns with the highest probability are selected. Finally, the operator and condition values are sorted by the attention mechanism.

TypeSQL

The model is based on SQLNet and uses a template-filling approach to generate SQL statements. To better model the rare entities and numbers that appear in the text, TypeSQL explicitly assigns each word type.

SQLNet sets a separate model for each component in the template; TypeSQL improves on this by better modeling similar components, such as SELECT_COL and COND_COL and #COND (conditional numbers), for which there are dependencies between the information, by combining them into a single model.

IRNet

Similar to SyntaxSQLNet, IRNet defines a series of CFG grammars that forward SQL into a syntax tree structure. It can be seen as an intermediate representation between natural language and SQL statements (called SemQL by the authors), and the whole parsing process is performed for SemQL. The whole parsing process is also done for SemQL.

Global-GNN

To make better use of the structural information of relational databases, BenBogin et al. proposed to use graph networks to model table names and column names. As shown in the figure below: the bolded nodes of the circles represent tables, and the unbolded nodes represent column names; the bi-directional edges represent the subordinate relationships between tables and column names; the red dashed edges, and blue dashed edges represent the primary and foreign key relationships. The orange nodes represent the results related to the problem, and the light color is irrelevant.

RAT-SQL

This work can be seen as a follow-up to the graph network GNN, where the authors define more edges between Table, Column, and Question.

Global-GNN only models the connection between Table and Column, RAT-SQL adds nodes of Question to this graph, and the types of edges are enriched by using string matching.

Open Source

[1] Seq2sql: Generating structured queries fromnatural language using reinforcement learning (Victor Zhong, Caiming Xiong,Richard Socher. CoRR2017)
[2] Towards Complex Text-to-SQL in Cross-DomainDatabase with Intermediate Representation (Jiaqi Guo, Zecheng Zhan, Yan Gao,Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. ACL2019)
[3] SQLNet: Generating Structured Queries FromNatural Language Without Reinforcement Learning (Xiaojun Xu, Chang Liu, DawnSong. ICLR2018)
[4] TypeSQL: Knowledge-based Type-Aware NeuralText-to-SQL Generation (Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, DragomirRadev. NAACL2018)

https://firebase.google.com/

https://github.com/salesforce/TabularSemanticParsing

https://github.com/microsoft/ContextualSP

https://towardsdatascience.com/text-to-sql-learning-to-query-tables-with-natural-language-7d714e60a70d

https://github.com/FerreroJeremy/ln2sql

https://gitlab.com/osadey/gpt-3-dash-playground