Natural Language Interface for Patients' Electronic Health Records (Milestone)

Revanth Korrapolu
rrk69

Madhumitha Sivaraj
ms2407

1 Problem

A notable AHRQ-funded study (Minimizing Error, Maximizing Outcome) questioned 170,000 physicians and found that more than half of the physicians reported experiencing time pressures when conducting physical examinations. In 2012, the a bill passed to push hospitals to use electronic health records, as opposed to the existing paper based system. The hope was to alleviate some of the burden and free up time for doctors. Instead, this lead to more misdiagnoses, misperscriptions, and overall medical malpractice.

2 Goal

Our goal is to create a toy model which provides doctors a more intuitive way of interacting patient data. Our natural language interface consists of two parts: Natural Language Understanding (NLU) and Natural Language Generation (NLG).

2.1 Workflow

A typical workflow starts with a physician asking the interface a natural language question. This question is translated into a SQL query. Assuming that the information can be found in the database, the query will then be run against our sample patient database. The FHIR database will contain all patient information including the patient id, age, height, weight, symptoms, diagnosis, notes for check-ins and more. The resulting query response will then be formatted into a sentence using NLG and provided to the client.

3 Implementation

3.1 Datasets

We are utilizing a large-scale healthcare Question-to-SQL dataset, known as MIMICSQL, by using the publicly available real-world Medical Information Mart for Intensive Care III (MIMIC III) dataset. The MIMIC-III Clinical Database contains tables of clinical data relating to patients who stayed within the intensive care units at Beth Israel Deaconess Medical Center. A table is a data storage structure which is similar to a spreadsheet: each column contains consistent information (e.g., patient identifiers), and each row contains an instantiation of that information (e.g. a row could contain the integer 340 in the patient identifier column which would imply that the row's patient identifier is 340). MIMIC-SQL includes 10,000 Question-SQL pairs of doctors questions and EHR SQL queries.

3.2 NLG: Core-NLG

After exploring the existing NLG implementations, there are two notable open source libraries, RosaeNLG and Core-NLG. Core-NLG is written in python and seems to be the more convenient choice. However, RosaeNLG provide more sophisticated linguistic features likes anaphora, verbs, words and adjectives agreements. Both provide viable solutions.

3.3 Architecture

The interface consist of three layers as shown below in the figure 1.

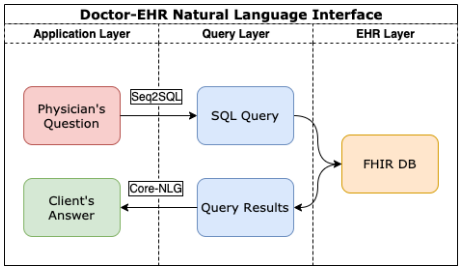

Figure 1: This architecture shows three layers which we will implement to simulate a realistic interaction between a doctor and EHR system.

1. **Application Layer (FINISHED)** - defines how the client (doctor) will interact with the interface. This may consist of a web app or chat bot in which the doctor can ask the interface questions about patient data in natural language.

2. **Query layer (IN PROGRESS)** - acts an intermediary layer that transposes Physicians' questions into SQL queries (using Seq2SQL) and query results into natural language (using Core-NLG).

3. **EHR layer (FINISHED)** - consists of our toy database holding patient data. Fast Healthcare Interoperability Resources (FHIR) specifies a new standard of data formatting for healthcare data. As such, we believe a FHIR-compliant database is the archetype of future EHR data systems.

4 Progress

We set up the environment with a sample FHIR database as well as a web app that shows patient data (see figure 2 below). Together, they demonstrate a sample EHR system. We have also explored Seq2SQL paper and played around with the pretrained Seq2SQL model.

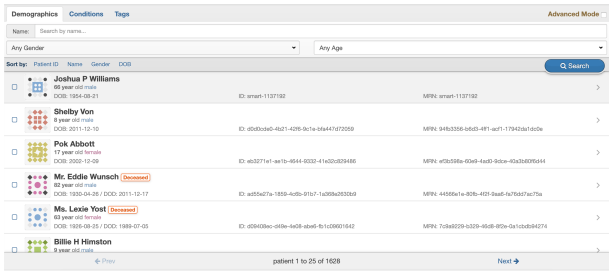

Figure 2: (a) The overall framework of the proposed TREQS model. [PH] represents the out of vocabulary words in condition values. (b) Illustration of dynamic and temporal attention mechanisms used in TREQS.

5 Next Steps

5.1 Seq2SQL

For the NLU we will be replicating TREQS. TREQS is a generative model that consists of three parts: (1) Translating an input question to a SQL query using a Seq2Seq based model, (2) Editing the generated query with attentive-copying mechanism, and (3) Further editing it with task-specific look-up tables. See figure 3 on the right.

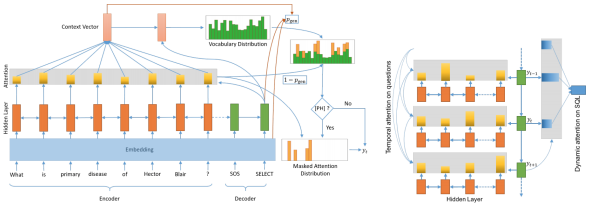

Figure 3: (a) The overall framework of the proposed TREQS model. [PH] represents the out of vocabulary words in condition values. (b) Illustration of dynamic and temporal attention mechanisms used in TREQS.

Our primary goal is to train Seq2SQL on the MIMICSQL dataset. Through this we hope to leverage the Seq2SQL model to transposes physicians' questions into SQL queries. Assuming that the information can be found in the database, the query will then be run against our sample patient database. The FHIR database will contain all patient information including the patient id, age, height, weight, symptoms, diagnosis, notes for check-ins and more. We aim to create transfer-learnable version of Seq2SQL, specifically for the medical domain.

By default, Seq2SQL uses vanilla GloVe embeddings. We may want to experiment with the newer medical embeddings (ie: BioBERT) to see if there are any performance gains. We also discovered DIALSQL-inspired Model-based Interactive Semantic Parsing framework which handles nested Q/A and boosts Seq2SQL. Our stretch goal is to explore these techniques as well.

5.2 NGL

After exploring existing NLG implementations, we decided to use CoreNLG, which is an open-source Python library for Natural Language Generation developed by Société Générale's Core-NLG. It provides developers with essential tools to structure and write NLG projects. We are using Core-NLG to help with parsing the resulting query response from Seq2SQL, formatting it into a sentence using natural language generation and providing to the client.