Text-to-ESQ: A Two-Stage Controllable Approach for Efficient Retrieval of Vaccine Adverse Events from NoSQL Database

Wenlong Zhang[1], Kangping Zeng[2], Xinming Yang[1], Tian Shi, Ping Wang[1*]
[1]Dept. of Computer Science, Stevens Institute of Technology, Hoboken, NJ, USA
[2]School of Business, Stevens Institute of Technology, Hoboken, NJ, USA
{wzhang71,kzeng4,xyang70,ping.wang}@stevens.edu,researchtianshi@gmail.com

ABSTRACT

The Vaccine Adverse Event Reporting System (VAERS) contains detailed reports of adverse events following vaccine administration. However, efficiently and accurately searching for specific information from VAERS poses significant challenges, especially for medical experts. Natural language querying (NLQ) methods tackle the challenge by translating the input questions into executable queries, allowing for the exploration of complex databases with large amounts of information. Most existing studies focus on the relational database and solve the Text-to-SQL task. However, the capability of full-text for Text-to-SQL is greatly limited by the data structures and functionality of the SQL databases. In addition, the potential of natural language querying has not been comprehensively explored in the healthcare domain. To overcome these limitations, we investigate the potential of NoSQL databases, specifically Elasticsearch, and forge a new research direction for NLQ, which we refer to as **Text-to-ESQ generation**. This exploration requires us to re-design various aspects of NLQ, such as the target application and the advantages of NoSQL database. In our approach, we develop a two-stage controllable (TSC) framework consisting of a **question-to-question (Q2Q) translation** module and an **ESQ condition extraction (ECE)** module. These modules are carefully designed to efficiently retrieve information from the VEARS data stored in a NoSQL database. Additionally, we construct a dedicated question-ESQ pair dataset called VAERSESQ, to support the task in the healthcare domain. Extensive experiments were conducted on the VAERSESQ dataset to evaluate the proposed methods. The results, both quantitative and qualitative, demonstrate the accuracy and efficiency of our approach in generating queries for NoSQL databases, thus enabling efficient retrieval of VEARS data.

CCS CONCEPTS

• **Information systems** → **Question answering**; • **Computing methodologies** → **Information extraction**; **Machine translation**; • **Applied computing** → **Health informatics**.

*The first three authors contributed equally to this research.

BCB '23, September 3–6, 2023, Houston, TX, USA

ACM ISBN 979-8-4007-0126-9/23/09…$15.00
https://doi.org/10.1145/3584371.3613008

KEYWORDS

Natural language querying, question translation, Text-to-ESQ, VAERS, Elasticsearch query.

ACM Reference Format:
Wenlong Zhang[1], Kangping Zeng[2], Xinming Yang[1], Tian Shi, Ping Wang[1]. 2023. Text-to-ESQ: A Two-Stage Controllable Approach for Efficient Retrieval of Vaccine Adverse Events from NoSQL Database. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23), September 3–6, 2023, Houston, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3584371.3613008

1 INTRODUCTION

In recent years, querying and searching databases with natural language questions, which is often realized by translating questions into executable SQL queries, has become a promising research topic in both the natural language processing (NLP) community and database community [22, 25, 35, 38, 49]. This paradigm, known as Text-to-SQL query generation, provides an efficient tool for domain experts, such as doctors and scientists, to interact with relational databases without the need for database knowledge and without the help of database engineers. Text-to-SQL also allows people to freely explore different relational databases because queries are dynamically generated from natural language questions, rather than hard-coded. Several large-scale Text-to-SQL datasets in a variety of domains have been created [43, 45, 46, 49], and many state-of-the-art deep learning models have also been developed based on them [5, 12, 16, 37].

However, most studies in this field focus on solving a semantic parsing problem [16, 46] or a language generation problem [26, 37, 48] for the Text-to-SQL task in NLP. Despite there are many limitations of using SQL queries for searching, few attempts have been made beyond SQL query generation. In fact, Text-to-SQL capabilities are limited by the data structures and functionality of SQL databases, not to mention the technical difficulties in semantic parsing and language models [22]. For example, in the healthcare domain, Electronic Health Records (EHR) have a key component, known as clinical notes, which contain different types of information about patients (e.g., discharge summary and family history). However, due to the limited functionality of SQL server for full-text search, it is difficult to perform searching on clinical notes with SQL queries. Another limitation for SQL server is that it is difficult to incorporate external knowledge bases (KBs) into relational tables because the complex relationships between nodes in KBs will result in too many tables or table columns in the relational database. Therefore, these limitations of SQL databases motivate the authors to forge new research directions for natural language driven querying and searching on NoSQL databases.

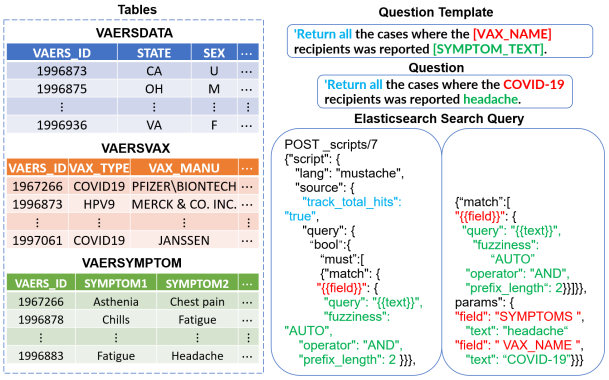


Figure 1: An example from VAERSESQ data. Colors show the correspondence between various components in the question template, NL/template questions, and query template. Here, "track_total _hits" indicates the query will retrieve all the relevant records. The "field" indicates a specific field to search, which takes variables provided in the "params" section to query. The sections "query" and "fuzziness" are designed for specific options to improve retrieval accuracy.

As one of the most important research directions in healthcare, vaccine administration is a vital response to the pandemic and other infectious diseases. Vaccine Adverse Event Reporting System (VAERS[1]) co-managed by the U.S. FDA and CDC is an important platform for reporting and analyzing side effects after getting vaccines. The VAERS data has been continuously updated since 1990, including structured information such as demographic information, vaccine details, and various coded symptoms, as well as narrative text descriptions. Currently, the VAERS data can be accessed via the CDC's WONDER system[2], which primarily relies on rule-based searching and filtering. For example, if we are interested in the total number of patients who had a *fever* after getting an *influenza vaccine*, we need to transform this question into two rules, one for the vaccine name and one for the symptom. However, there are several limitations to such a system. (1) *Complicated to use:* users need to follow detailed instructions to specify the rules in the front-end based on their interested information, therefore, domain experts need special training before using these systems. (2) *Inflexible to extend*: The adverse event information in VAERS is rapidly evolving with new reports over time, which may include new features and symptoms. It will require additional efforts to incorporate and update the new filtering functions in the current system. These limitations can be potentially addressed by Text-to-SQL, which aims to automatically translate natural language questions to SQL queries with different NLP techniques. However, Text-to-SQL is primarily designed for retrieving information from SQL databases with relational tables. As aforementioned, SQL databases have limited functionality for full-text search and it is difficult to incorporate external KB into relational tables. Therefore, we consider storing VAERS data with NoSQL databases in this paper.

Although there are some widely used databases and search engines in the industry, such as MongoDB and Elasticsearch, very little work has been devoted to developing NLQ tools for them and exploiting their potential. This paper aims to fill this gap and forge new research directions for NLQ on Elasticsearch by re-designing a variety of aspects of NLQ, including the target application, the benefit of NoSQL database, and the JSON data stored in the database, etc. We refer to the NLQ task with Elasticsearch queries as **Text-to-ESQ generation** task in this paper. Compared with Text-to-SQL, Text-to-ESQ has two primary advantages. (1) The document-like format of Elasticsearch database allows us to easily convert complex data formats into nested JSON objects, which allows us to integrate data from disparate sources, such as tables, texts, graphs, and other external knowledge bases. (2) Text-to-ESQ can incorporate reasoning processes into searching via both query clauses (e.g., Boolean relationships) and path to the target fields (e.g., *DATA.SYMPTOMS.SEVERITY*).

Since there are few studies in NLQ on NoSQL databases, we have to tackle the following challenges: (1) Problem formulation: Since NoSQL databases store data as nested JSON objects, the Text-to-ESQ task is substantially different from the Text-to-SQL task. For example, Text-to-SQL aims to retrieve the information from the table, while Text-to-ESQ focuses on information screening, i.e., retrieving the most relevant information. When dealing with complex questions, Text-to-SQL needs to generate queries that can perform joint searches on different relational tables, while Text-to-ESQ solves this problem by querying just one document with nested fields. (2) Datasets: To the best of our knowledge, there has been no such dataset used to train Text-to-ESQ models. To address these challenges, we create a large-scale VAERSESQ dataset with more than 20k question-query pairs based on the VAERS data. Figure 1 shows an example in the VAERSESQ data. We perform a detailed analysis of the datasets and further propose a model to generate ES queries based on the natural language questions. Our **major contributions** can be summarized as follows:

- Formally propose and formulate the Text-to-ESQ task to support NLQ on NoSQL database. To the best of our knowledge, this is the initial comprehensive investigation of NLQ on the NoSQL database.
- Propose a two-stage controllable (TSC) framework consisting of two modules for Text-to-ESQ: (1) *Question-to-question translation* module for translating natural language questions into the corresponding template questions, and (2) *ESQ condition extraction* module for parsing name entities about condition fields and values from template questions for further populating the query templates.
- Create a large-scale dataset VAERSESQ for Text-to-ESQ task for retrieving information from VAERS data. For each question, we include both template-based and natural language forms.
- Conduct an extensive experimental analysis of the VAERSESQ dataset and demonstrate the effectiveness of the proposed two-stage controllable method.

The rest of this paper is organized as follows. We describe the related work about natural language querying in Section 2. In Section 3, we introduce a detailed description of how we generate the VAERSESQ dataset. Section 4 provides a detailed discussion of the proposed two-stage controllable framework. In Section 5, we

[1]https://vaers.hhs.gov/data.html
[2]https://wonder.cdc.gov/VAERS.html

evaluate the performance of our proposed model with both quantitative and qualitative analysis. Finally, we conclude the discussion in Section 6.

2 RELATED WORK

2.1 Question Answering (QA)

The goal of QA is to automatically answer natural language questions about various data sources, such as databases [49], knowledge graphs [7], narrative texts [34], images [3], and multi-modal data types [11, 20]. Querying with natural language is a long-standing topic and has received great attention in the past few years [6, 8, 32]. Existing research activities often fall into one of two categories, i.e. rule-based approach [23, 38] and machine learning approach [9, 44]. Rule-based methods are designed based on pattern matching, grammar-related techniques, and intermediate representations. Machine learning methods grew rapidly due to the advances of deep learning and showed promising results.

QA in healthcare. QA in healthcare has not been fully explored due to the lack of high-quality annotated datasets and privacy concerns. Most of the existing work focuses on Electronic Heath Records (EHR) [18, 29, 43] to identify patient information from the heterogeneous medical history of patients. For example, the large-scale dataset MIMICSQL [43] and MIMICSQL*[30] are released for Text-to-SQL query generation tasks on EHR, which aims to automatically translate natural language questions about EHR to SQL queries. emrQA dataset [29] is created for machine reading comprehension (MRC) on clinical notes. MIMIC-SPARQL [30] and ClinicalKBQA [42] are created for Knowledge Graph-based Question Answering on EHR. Besides, several other datasets are also released for MRC tasks, such as PubMedQA on biomedical research texts [17], CliCR on clinical case reports [40], and BioASQ for semantic indexing and QA [41]. The topic of visual QA has also been explored in healthcare [2, 13].

2.2 Query Generation Based QA

There are research activities about translating natural language queries into machine-executable programs (i.e., queries) against different databases for question-answering tasks. Most of these studies focus on utilizing one of the two primary query types, including SQL and SPARQL, and have been successfully applied in various real-world applications, such as healthcare and customer services. Text-to-SQL generation aims to automatically translate the natural language questions into the corresponding SQL queries to retrieve information represented as the grid-like format of rows and columns in tables [33, 46, 49]. SPARQL query generation is designed to automatically query the knowledge base stored in RDF format with natural language questions [4, 39, 47]. The automatic generation of queries enables non-expert users to efficiently search for their interested information from large databases without knowing the query concepts and syntaxes and has become a promising research topic. Although there are many well-known, efficient, and scalable NoSQL databases and search engines, such as MongoDB and Elasticsearch, very little work has been devoted to exploring their potential. This paper aims to fill this gap and rethink various aspects of NLQ based on Elasticsearch, such as the benefit of NoSQL databases and the target applications.

In general, existing approaches for query generation-based QA fall into one of the following two paths: (1) *Language generation approaches:* These methods utilize the language generation models to directly generate the output queries or their logical forms. Therefore, they are able to flexibly handle input questions in any format and can be easily used to generate complex queries. One primary limitation of language generation approaches is the generated queries may not be executable due to the incorrectly generated tokens in the output queries. (2) *Semantic parsing approaches:* Based on the pre-defined templates of the queries, semantic parsing approaches aim to predict the name entities from the input questions to further populate the query templates. The limitation of such methods is that they highly rely on pre-defined templates and cannot efficiently handle natural language questions. In this paper, we combine the power of both language generation and semantic parsing approaches to solve the Text-to-ESQ task.

3 VAERSESQ DATASET

This section presents our work in generating a large-scale dataset for Text-to-ESQ on the Vaccine Adverse Event Reporting System (VAERS) data. To the best of our knowledge, there is no existing dataset for the Text-to-ESQ task so far.

3.1 VAERS Dataset

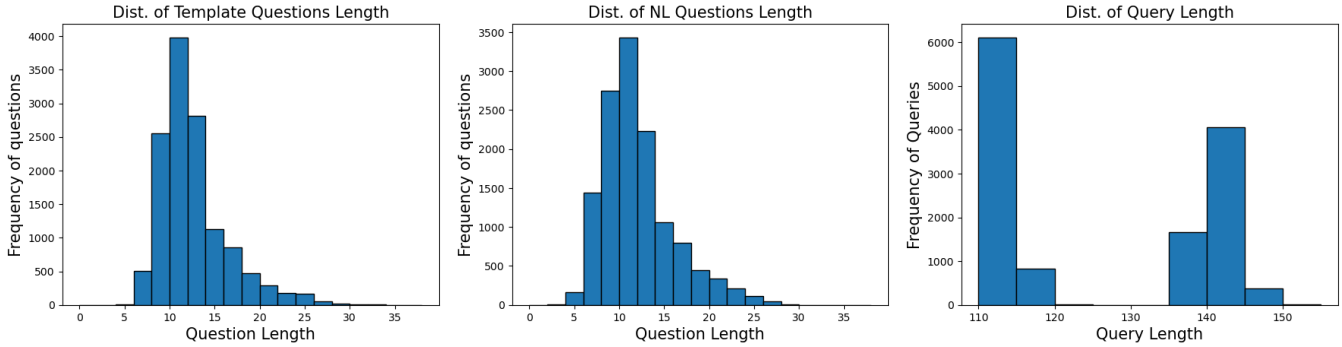
VAERS data are publically available at the national early warning system[3]. Three types of information are provided by VAERS, including VAERSDATA, VAERSVAX, and VAERSSYMPTOMS, where VAERSDATA contains demographic information, diagnosis, and more context of vaccine adverse events; VAERSVAX has the categorization and manufacturing information of the vaccines; VAERSSYMPTOMS gives a list of standardized symptoms. For each vaccine adverse event, we merge all three typing of data into one JSON object and index it into the Elasticsearch database.

3.2 VAERSESQ Dataset

We create a VAERSESQ dataset (see Figure 1) for the Text-to-ESQ task based on the VAERS data stored in Elasticsearch. This dataset consists of Question-Query pairs that can be used to train and evaluate Text-to-ESQ models. There are two questions associated with each Elasticsearch query. (1) *Template Question*: a question that is generated by rule-based methods (e.g., logic format and slot-filling-based techniques), when generating the corresponding query. (2) *Natural Language Question*: a question that is obtained by rephrasing a template question. In this section, we will present our methods of creating these question and query pairs.

3.2.1 Question Template Collection and Population. Based on the information provided in the VAERS data and people's interests in vaccine adverse events, we first summarized the possibly asked questions on VAERS data and then normalized them as question templates by identifying medical entities in questions and replacing them with generic holders, following the question generation method in [29]. These templates inherit two question types, retrieval and reasoning from previous work [19]. In this work, we

[3]https://vaers.hhs.gov/data.html



(a) Template questions. (b) Natural language questions. (c) Elasticsearch queries.

Figure 2: Distribution of questions and corresponding queries in VAERSESQ dataset.

focus on the retrieval questions, whose answers can be directly fetched from Elasticsearch.

Retrieval questions retrieve the relevant records directly from the VAERS data by focusing on certain fields of the database. The following are three examples of question templates for retrieval questions.

- How many people have [SYMPTOM] after vaccination?
- Give me all the patients who got [VAX_NAME_1] vaccine and [VAX_NAME_2] vaccine.
- Search all the patients who are diagnosed of [HISTORY].

As shown in the above examples, each question template has one or more capitalized text surrounded with square brackets, which represent the placeholder we used for further question population. Within each placeholder, the text indicates the condition name, and during population, we replace the placeholder with random values collected from the corresponding condition field. Since all fields' name within three tables is unique, here, we skip the table name and use the field name directly. With a total of 45 question templates, we automatically populate and collect over ten thousand template questions.

3.2.2 Natural Language Question Generation with Back-Translation. The machine generation based on question templates and slot-filling methods to a great extent allows us to collect a large set of questions efficiently. However, machine-generated questions tend to be less realistic and more rigid due to a fixed pattern. To alleviate this drawback, we utilized a state-of-the-art pre-trained language model, Many-to-Many multilingual translation model (M2M) [10], and proposed a back-translation strategy to synthetic a group of natural language questions. Different from standard machine translation which collects the translation by directly translating the source language to the target language, back-translations are collected when sentences are re-translated in a backward direction, from the target language to the source language.

In our task, we have a large set of machine-generated template questions, and there are no available natural language questions to support the proposed Text-to-ESQ task. We utilize the back-translation strategy to collect the corresponding natural language questions. Specifically, using the pre-trained M2M model, we set

Table 1: Basic statistics of VAERSESQ dataset. [a]The three tables include VAERSDATA, VAERSSYMPTOMS, and VAERSVAX.

Data	Value
# of tables	3
# of fields/columns in tables[a]	35/8/11
Number of template/natural questions	13,040
Average template question length (in words)	12.13
Average NL question length (in words)	11.52
Average query length (including template keywords)	167.65

the source language to be Chinese and first translate the existing template questions from English to Chinese, and further translate them back to English questions presented in a natural way. The M2M model we used includes 1.2B parameters and is pre-trained on 7.5B sentences from 100 languages, which greatly supports our translation between Chinese and English. The back-translation strategy with M2M allows us to efficiently collect a large set of natural language questions that are accurate and fluent, by leveraging the translation process between target and source. Finally, the generated VAERSESQ dataset is enriched with the natural language question for each of the template questions generated in Section 3.2.1.

3.2.3 Elasticsearch Query Generation. Elasticsearch has several **key properties** to support the templated search. (1) The query templates only require users to pass input as parameters for a search template to query the database without knowing any query syntax. (2) One query template can respond to multiple question templates. This enables us to use a less generic query template to handle various questions. (3) One specific query can respond to multiple questions since the query results (defined as *hits*) provide comprehensive information, such as general information (e.g., the total number of hits, maximum matching score) and individual hit (e.g., metadata, matching value, and a JSON object with specific information from the original document). Therefore, the query template provided by Elasticsearch makes it practical to generate the queries on a large scale [1]. In general, each search template

consists of two components, including a query template and its corresponding parameters. This template structure allows it to respond to various questions without changing the query template to a large extent. While the parameters in the templates are usually placeholders for populating specific condition values. Therefore, when generating the template questions, the corresponding Elasticsearch queries are generated at the same time by populating the placeholders in the query templates. Figure 1 shows an example in the VEARSESQ dataset, with the data stored in the Elasticsearch database, the input question along with the question template, and the Elasticsearch query template.

3.3 Basic Statistics of VAERSESQ

The VAERSESQ dataset is publicly available at[4]. VAERSESQ contains 13,040 template/natural question-query pairs generated based on VAERS dataset in 2022. Table 1 and Figure 2 jointly show the basic statistics of the VAERSESQ dataset. The template questions and natural language questions have 12.13 and 11.52 words on average, respectively. The average length of the queries (including the template keywords) is 167.65 words. Figure 2(a) and Figure 2(b) show the detailed distribution of question length in words for template questions and natural language questions, respectively. Figure 2(c) shows the distribution of Elasticsearch query length in words.

4 METHODOLOGY

In this section, we present a practical two-stage controllable (TSC) framework for the Text-to-ESQ task.

4.1 Problem Formulation

As aforementioned in the introduction, Text-to-ESQ is based on Elasticsearch and aims at information screening, i.e., getting a list of the most relevant candidates, instead of retrieving an exact answer from a database, which is substantially different from Text-to-SQL. In other words, Text-to-ESQ focuses more on condition fields and values. One of the straightforward solutions for Text-to-ESQ is to leverage language generation models (such as sequence-to-sequence models) to directly translate natural language questions to ES queries. However, language generation models require a lot of diversified high-quality question-query pairs, which is challenging for domain-specific applications, such as electronic health records or VAERS data. Therefore, in this paper, we considered some practical and controllable solutions and benchmark them on our VAERSESQ data.

When constructing the VAERSESQ dataset, we observed that it is simple to use rule-based methods to generate template question-query pairs. By fine-tuning large language model-based sequence labeling models on datasets composed of template question-query pairs, we were able to achieve near-perfect accuracy in detecting condition fields and values from a template question based on our initial test (see Table 4). Thus, in practice, we can first rephrase a natural language question to a template question and then apply sequence-labeling models on the template question to obtain condition fields and values, which can be used to compose ES queries via rule-based methods.

[4]https://github.com/LEAF-Lab-Stevens/Text2ESQ

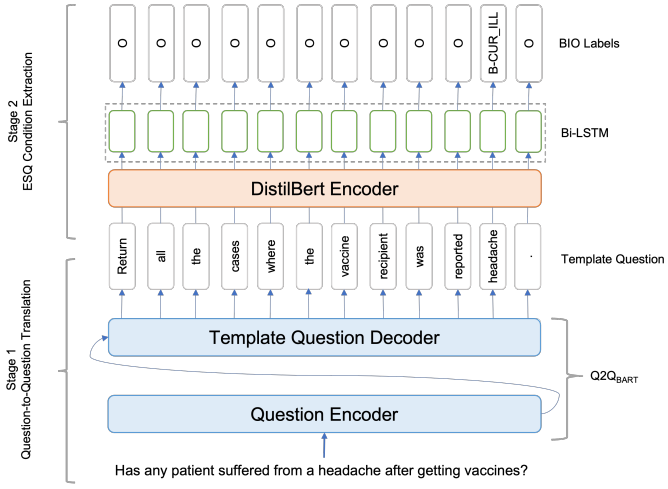


Figure 3: The overall framework of the proposed two-stage controllable Text-to-ESQ method.

4.2 The Proposed Method

Figure 3 shows the architecture of the proposed two-stage controllable Text-to-ESQ method, including a question-to-question translation and ESQ condition extraction module.

4.2.1 Question-to-Question (Q2Q) Translation Module. The goal of this module is to translate a natural language question into a template question. In practice, domain experts (such as healthcare practitioners) may not have expertise in databases and table structures. Therefore, in most cases, their questions may not contain terms that exactly match the condition fields or values. Thus, with the help of sequence-to-sequence models, we can translate questions asked by domain experts to template questions, which contain these standardized terms used in the database.

To achieve this goal, we adopt a large pre-trained transformer-based sequence-to-sequence model, namely BART [24], as the base model and fine-tune it on natural language question and template question pairs. Formally, given an input natural language question $x = (x_1, x_2, ..., x_I)$, we feed it into the question-to-question (Q2Q) translation module to obtain the template questions x' with

$$x' = (x'_1, x'_2, ..., x'_{I'}) = Q2Q_{BART}(x) \tag{1}$$

Here, I and I' are the lengths of the original question and template question, respectively. The loss function used to fine-tune the BART model is a cross-entropy loss defined by

$$\mathcal{L}_{\text{Q2Q}} = -\sum_{t=1}^{I'} \log p(x'_t | x'_{<t}, x) \tag{2}$$

where, $p(x'_t | x'_{<t}, x)$ is known as likelihood, which is the conditional probability of the next token x'_t given all previous ones denoted by $x'_{<t}(x'_1, x'_2, ..., x'_{t-1})$. Finally, the generated question x' will serve as the input for the subsequent sequence-labeling module.

4.2.2 ESQ Condition Extraction (ECE) Module. This module is designed to identify condition fields and values from template questions. One strategy is to extract condition fields and values independently, but it is challenging to align condition fields and values if there are multiple fields. Therefore, we formulate the ESQ condition extraction problem as a sequence labeling task to overcome this problem. Conditions values are regarded as relevant tokens, with condition fields being the corresponding labels (i.e., categories).

Formally, given a template question denoted by a sequence of tokens $x' = (x'_1, x'_2, ..., x'_{I'})$, our goal is to learn a sequence of labels $l = (l_1, l_2, ..., l_{I'})$, which classify each token as a contextual token or a condition value. We adopt the widely used BIO-tagging scheme [15], where B, I, and O represent the beginning, inside, and outside of a condition value, respectively. Condition field names are treated as categories, and encoded into the beginning and inside labels. Specifically, consider $F = \{f_i\}_{i=1}^{N}$ as the set of all field names in the database, the set of all candidate labels can be represented as $L = \bigcup_{f_i \in F}\{B-f_i, I-f_i, O\}$. After obtaining the predicted sequence of BIO labels, it is straightforward to populate the query template with the required condition fields and values, since fields are explicitly included in the BIO labels, and condition values can be easily mapped based on the predicted BIO labels.

We adopt a pre-trained language model, namely DistilBERT [36] as the encoder, and use a feature-based approach (similar to [21]) to predict the BIO labels. Formally, DistilBERT encodes a list of tokens as a sequence of hidden states $h = (h_1, h_2, ..., h_{I'})$, also known as contextual embeddings. Then, these contextual embeddings are used as input to a randomly initialized Bi-LSTM before the classification layer. Finally, we fix the DistilBERT parameters and fine-tune the rest parameters on the VAEARSESQ dataset. The loss function used in the fine-tuning is a token-level cross-entropy loss

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{I'} \log p(l_t, x') \tag{3}$$

where $p(l_t, x')$ is the likelihood.

This two-stage controllable method can leverage a limited amount of natural questions. Results based on this method will significantly outperform the baseline method in which we directly generate Elasticsearch query (ESQ) from natural language questions. The benefit: not a lot of natural language questions, this solution allows us to get better results.

5 EXPERIMENTAL RESULTS

In this section, we address the following research questions (RQ) through experimental analysis.

- **RQ1**: Is the Question-to-Question (Q2Q) translation module able to map the input natural language questions into the template questions?
- **RQ2**: Is the ESQ condition extraction (ECE) module able to identify correct condition fields and values from the template questions?
- **RQ3**: What is the performance of the proposed two-step controllable method on natural language questions?
- **RQ4**: What is the break-down performance on both the condition name and value components in Elasticsearch query?
- **RQ5**: How to interpret the output obtained from the ECE module?

5.1 Experimental Settings

5.1.1 Dataset Description. We use the template and natural language questions in VAERSESQ dataset (described in Section 3) along with their Elasticsearch queries for evaluation. We randomly split the dataset into the training, development, and testing sets in the ratio of 0.8/0.1/0.1. The BIO labels of the template questions are generated during the question generation stage. Besides, to compare our proposed TSC method with the methods that directly generate the logical forms for the input questions, we also automatically generated the logical forms of the Elasticsearch queries for template questions, following several pre-defined templates.

5.1.2 Comparison Methods. We evaluate the performance of the proposed Question-to-Question translation module by comparing it with two baseline methods.

- **M2M [10]:** a multilingual encoder-decoder model specifically developed for the task of translation and can be used to translate the natural language questions into the template questions. In this work, we use M2M 418 MB parameters version.
- **Seq2Seq [14]:** a type of recurrent neural network (RNN) designed for handling the vanishing gradient problem of vanilla RNN.

To evaluate the performance of the ESQ condition extraction module, we compare it with the following methods. The first Seq2Seq method is used to directly generate the logical forms of the queries, which can be easily converted to the ESQ with rule-based methods. While the remaining ones are used to perform the BIO label prediction task for ESQ condition extraction.

- **Seq2Seq [28]:** sequence-to-sequence (Seq2Seq) includes a bidirectional LSTM encoder and an LSTM decoder.
- **RoBERTa [27]:** a replication study of BERT [21] pretraining with an improved training recipe by investigating the impact of key parameters and training size.
- **RoBERTa+Bi-LSTM:** on the top of RoBERTa, it integrates a Bidirectional LSTM layer and a linear layer.
- **DistilBERT [36]:** a light version of BERT with 40 percent fewer parameters by leveraging knowledge distillation during the pretraining phase.

5.1.3 Implementation Details. All the experiments were performed using NVIDIA Quadro RTX 5000 GPUs. The proposed TSC model is implemented with PyTorch [31]. We adopt the SGD with momentum optimizer during the training of the model parameters. The learning rate is set to 0.01. The experiments for all the models are obtained by running 16 epochs with the mini-batch size 32. The development set is used to select the best model. The implementation of the TSC model is made publically available at[5].

5.1.4 Evaluation Metrics. To evaluate the performance of the Q2Q translation module, we use the string-matching metric ROUGE-1 to measure the similarity of the overall original natural language questions and the generated template questions. Besides, we also obtain the specific evaluation on condition values, which is the most challenging task in query generation.

To evaluate the performance of the ESQ condition extraction module, we adopt the commonly used string matching metric **logic**

[5] https://github.com/LEAF-Lab-Stevens/Text2ESQ

Table 2: Performance (measured by ROUGE-1) of different methods on translating natural language questions into template questions. We evaluate the performance on both the overall questions and condition values.

Methods	Development		Testing	
	Overall	Value	Overall	Value
Seq2Seq	0.73	0.35	0.70	0.36
M2M	0.92	0.60	0.90	0.63
Q2Q	0.88	0.65	0.85	0.63

form accuracy defined as $Acc_{f+v} = N_{f+v}/N$ to evaluate the overall performance on both condition fields and values. Here, N_{f+v} denotes the number of questions that are identified with both correct condition fields and values, and N denotes the total number of Question-ESQ pairs in VAERSESQ dataset. Besides, we also evaluate the breakdown string matching performance on the condition fields and values separately by calculating $Acc_f = N_f/N$ and $Acc_v = N_v/N$, respectively. Here, N_f (N_v) denotes the number of questions that are identified with correct condition fields (values).

5.2 Experimental Results

*5.2.1 Performance of Q2Q Translation Module (**RQ1**).* This section evaluates if the proposed Question-to-Question (Q2Q) translation module is able to map the input natural language questions into the template questions. Table 2 provides the performance on the overall questions measured by ROUGE-1, which reflects the similarity between the generated template questions and the ground truth template questions. During our analysis, we found that the condition value is the most challenging component during the generation, and although all models provide a good ROUGE score, there are still some errors in the generated condition values, such as vaccine serial numbers and complex symptom names. This aligns with the findings in previous studies about Text-to-SQL [43, 49]. Therefore, we also provide a specific evaluation of the condition value component.

We compare the performance of the proposed Q2Q module with both M2M and Seq2Seq. We observe from Table 2 that Seq2Seq leads to the worst performance on both the development and testing sets due to its poor performance in translating the condition values. M2M achieved the best results on the overall evaluation of both development and test sets. Compared with M2M, the proposed Q2Q module provides competitive performance on the overall evaluation, and superior performance in generating the condition values.

In addition to the quantitative analysis, we have also conducted an extensive qualitative analysis to compare the generated template questions produced by various models and the ground truth questions. Table 3 provides one representative example of the generated template questions from all three methods. It can be observed that all three models can capture the correct question type and the components of question templates. Seq2Seq did not generate the correct condition value *COVID-19 Vaccine*. M2M is able to partly recover the condition value *COVID Vaccine* from the input natural language question. The proposed Q2Q module is able to correctly map the input natural language question into the corresponding ground truth template question. Therefore, both the qualitative and quantitative results demonstrate the importance of successfully generating the correct keywords of condition values for generating the correct template questions. The Q2Q module is able to address this challenge better compared with the other two methods.

Table 3: An example of translating the natural language questions (NLQ) into template questions (TQ) with the Q2Q module.

Methods	Question
NLQ	Which type of reaction is most common after a COVID vaccine?
Ground Truth TQ	Which symptom is most common after a COVID-19 vaccine?
Seq2Seq	Which symptom is most common after a ?
M2M	Which symptom is most common after a COVID vaccine?
Q2Q	Which symptom is most common after a COVID-19 vaccine?

*5.2.2 Performance of ESQ Condition Extraction (ECE) Module (**RQ2**).* This section evaluates if the proposed ESQ condition extraction module is able to identify correct condition fields and values from the input template questions. Table 4 provides the quantitative results on both template questions and natural language questions. The best performance for template questions and natural language questions are highlighted in bold, respectively. The first section of Table 4 shows the results of the template questions, which require only the ESQ condition extraction module and do not require applying the Q2Q module for translation. Specifically, the goal of the ESQ condition extraction module is to extract the correct condition fields and values from the input questions.

In the baseline methods, Seq2Seq directly generates the logic form of the query, which can be easily converted to the corresponding Elasticsearch query. Compared with other entity extraction methods, Seq2Seq provides the worst performance due to the errors in the generated logic forms. The remaining four methods overcome this poor generation by formulating the task as a sequence labeling task and further provide good performance over 95%. Compared with all other methods, our proposed ECE module leads to the best overall performance on both development and testing sets. The ECE module also shows a strong ability when we evaluate the identified condition fields and values separately (more details are discussed in Section 5.2.4).

These findings indicate that the ECE module is able to extract correct condition fields and values for template questions. The better performance of the ECE module is due to the rich long-range contextual information captured by combining DistillBERT with Bi-LSTM layers, which enables a powerful contextual representation learning and the ability to capture sequential dependencies.

*5.2.3 Performance on Natural Language Questions (**RQ3**).* This section evaluates the performance of the proposed two-stage controllable framework on the natural language questions. The second

Table 4: Performance evaluation with accuracy on *template questions* and *natural language questions*. We bold the best results on template questions and natural language questions separately.

Type	Method	Development			Testing		
		Overall	Field	Value	Overall	Field	Value
Template	Seq2Seq	0.515	0.646	0.316	0.690	0.740	0.640
	RoBERTa	0.959	0.986	0.991	0.956	0.979	0.986
	RoBERTa+Bi-LSTM	0.967	0.982	0.992	0.967	0.982	0.992
	DistilBERT	0.981	**0.993**	0.995	0.975	**0.989**	0.992
	ECE	**0.982**	0.992	**0.998**	**0.983**	**0.989**	**0.999**
Natural language	Seq2Seq+Seq2Seq	0.351	0.350	0.231	0.301	0.324	0.287
	Seq2Seq+RoBERTa	0.355	0.358	0.357	0.360	0.366	0.362
	Seq2Seq+RoBERTa+Bi-LSTM	0.352	0.357	0.354	0.358	0.360	0.359
	Seq2Seq+DistilBERT	0.343	0.346	0.349	0.342	0.347	0.347
	Seq2Seq+**ECE**	0.343	0.348	0.349	0.348	0.350	0.350
	M2M+Seq2Seq	0.389	0.374	0.291	0.351	0.404	0.307
	M2M+RoBERTa	0.544	0.551	0.552	0.471	0.476	0.477
	M2M+RoBERTa+Bi-LSTM	0.547	0.551	0.551	0.477	0.478	0.479
	M2M+DistillBERT	0.552	0.554	0.554	0.475	0.479	0.478
	M2M+**ECE**	0.553	0.553	0.554	0.476	0.478	0.479
	Q2Q+Seq2Seq	0.469	0.588	0.288	0.473	0.537	0.304
	Q2Q+RoBERTa	0.599	0.612	0.609	0.593	0.601	0.602
	Q2Q+RoBERTa+Bi-LSTM	0.606	0.612	0.610	0.596	0.602	0.604
	Q2Q+DistilBERT	**0.609**	**0.613**	**0.612**	0.598	0.604	0.603
	Q2Q+**ECE**	0.601	0.612	**0.612**	**0.601**	**0.605**	**0.605**

section in Table 4 shows the results of the natural language questions, which requires applying both the Q2Q translation module and the ESQ condition extraction module. Therefore, the results for the natural language questions are for evaluating the performance of the whole two-stage controllable (TSC) method for Text-to-ESQ generation.

Based on the results, it can be observed that, when we utilize the Seq2Seq model for translating in the first stage, the overall performance on natural language questions is relatively lower compared with M2M and the proposed Q2Q. This is due to the poor generation behavior of Seq2Seq when generating template questions in the first stage and further, the errors are propagated to the second stage. In general, when we utilize M2M for translation in the first stage, it provides better performance compared with Seq2Seq. However, the performance is still relatively low compared with the results obtained when adopting the proposed Q2Q module. We can observe that if we adopt Q2Q in the first stage, the performance of all methods in the second stage is significantly improved compared with the results obtained by Seq2Seq and M2M. The proposed TSC framework, with Q2Q and ECE modules, obtains competitive results on the development set and outperforms all other methods on the testing set. It also outperforms other models in terms of the condition name, which means it can capture the condition fields and values more accurately. As for the condition field and value extraction task in the second stage, it should also be noted that, after fine-tuning our specific task, all the pre-trained models have a notable advantage compared with the Seq2Seq model for generating the logic form.

These findings indicate that the quality of translated template questions obtained in stage one will significantly affect the quality of the extracted condition field and values in the second stage. Therefore, we can consider the first stage of Q2Q translation as the process of acquiring standardized input questions, which will significantly enhance the performance of the overall performance. For example, in Table 3, the term "symptom" used in the ground truth template question exactly aligns with the corresponding field name in the database, which makes it easy for the model to automatically identify. For the term "reaction" in the natural language question, it is not directly utilized as the field name in the database. However, the translation module will automatically map it to the corresponding template question and further provides the correct condition field.

*5.2.4 Break-down Generation Performance (**RQ4**).* Besides the overall evaluation of both template questions and natural language questions, in this section, we also evaluate the breakdown performance

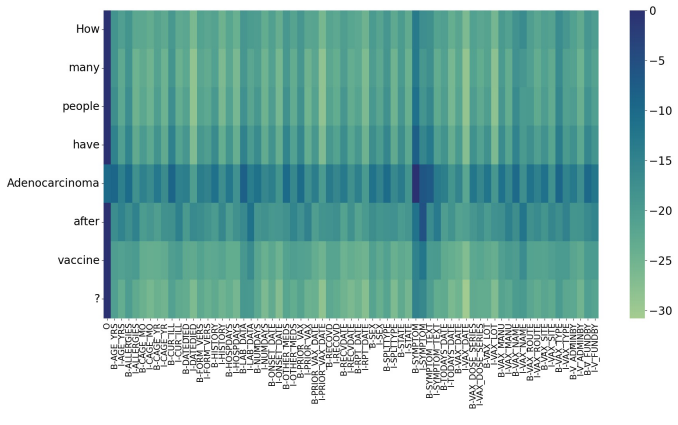


Figure 4: Visualization of the weight for each token in a template question example over each candidate label for sequence labeling. Intense color in the heatmap indicates a higher probability.

of extracting the condition fields and values separately. The results are shown in the *Field* and *Value* columns in Table 4.

In general, the breakdown results align with the overall performance that the sequence labeling-based methods by fine-tuning the pre-trained models outperform the generation-based Seq2Seq model. More specifically, we find that for Seq2Seq, the performance on generating condition values is significantly lower compared to that on condition fields. This is because condition value is the most challenging component during generation, similarly in the Text-to-SQL task [43, 49]. Differently, for all the sequence labeling-based methods by fine-tuning the pre-trained models are able to achieve a higher and balanced performance on condition field and value. The proposed TSC framework, by combining Q2Q and ECE modules, provides the best performance on both condition field and value. This is due to two important advantages of these models, (1) the sequence labeling of each token in the input question with classification task provides more accurate results compared with generation-based methods, and (2) the pre-training and fine-tuning paradigm allows the model to capture more informative contextual relationships.

*5.2.5 Interpreting Model Outputs with Visualization (**RQ5**).* To interpret the model outputs and help explain the experimental results, we visualize the predicted score (after taking the logarithm) obtained from the ESQ condition extraction (ECE) module for condition field and value extraction. In Figure 4, we provide the visualization for all tokens in a template question example to illustrate the output of our proposed ESQ condition extraction module. In the figure, the y-axis represents each token in the question, and the x-axis represents the candidate labels for sequence labeling (i.e. condition name). The data depicted by colors in the heatmap represents the predicted probability for labeling the specific condition field for each token in the input question. The probability is obtained by taking the logarithm of the output values from the top layer. Intense color in the heatmap indicates a higher probability. The goal of this visualization is to investigate if the proposed ECE module is able to successfully provide the label for each token in the input question.

For this example, the input template question is "*How many people have Adenocarcinoma after vaccine?*". The model provides the highest score for labeling the token "*Adenocarcinoma*" as "B-SYMPTOM", which indicates that "*Adenocarcinoma*" is predicated as the beginning token for the condition field SYMPTOM. At the same time, the model provides the highest score for labeling all the remaining tokens as "O", which indicates that all other tokens in the question are outside and not related to any conditions in the database. With the identified condition field and value, it is easy for us to obtain the final Elasticsearch query for getting the corresponding data information in the database. Therefore, the findings indicate that the proposed models can correctly predict the corresponding condition name of each token, where the correct label exceeds other candidates by a large margin.

6 CONCLUSIONS

The Vaccine Adverse Event Reporting System (VAERS) contains a large number of adverse events that are reported after vaccine administration. Efficient and accurate information retrieval from VEARS data can benefit doctors' decision-making. Natural language querying methods are popularly adopted to convert the natural language questions regarding the database into executable queries, which enables the exploration of complex databases. Most existing work focuses on the Text-to-SQL query generation task on relational databases. Due to the limitation of the full-text search for Text-to-SQL, in this paper, we extend the NLQ task to NoSQL databases and propose the Text-to-ESQ task on Elasticsearch. We create the VEARSESQ dataset for the Text-to-ESQ task on VEARS data. Further, we develop a two-stage controllable framework consisting of a question-to-question translation module and an ESQ condition extraction module. The experimental results demonstrate the effectiveness of the proposed approach in generating Elasticsearch queries.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation grant IIS-2245907 and the startup funding from Stevens Institute of Technology.

REFERENCES

[1] 2000. Resource Description Framework (RDF). https://www.elastic.co/cn/.
[2] Asma Ben Abacha, Sadid A Hasan, Vivek V Datla, Joey Liu, Dina Demner-Fushman, and Henning Müller. 2019. VQA-Med: Overview of the medical visual question answering task at ImageCLEF 2019. *CLEF (Working Notes)* 2, 6 (2019).
[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*. 2425–2433.
[4] Asma Ben Abacha and Pierre Zweigenbaum. 2012. Medical question answering: translating medical questions into sparql queries. In *Proceedings of the 2nd ACM SIGHIT international health informatics symposium*. 41–50.
[5] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2541–2555.
[6] James Clifford. 1988. Natural language querying of historical databases. *Computational Linguistics* 14, 4 (1988), 10–34.
[7] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2019. KBQA: learning question answering over QA corpora and

knowledge bases. In *Proceedings of the VLDB Endowment*. 565–576.
[8] Sasa M Dekleva. 1994. Is natural language querying practical? *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 25, 2 (1994), 24–36.
[9] Li Dong and Mirella Lapata. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 731–742.
[10] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, et al. 2021. Beyond english-centric multilingual machine translation. *The Journal of Machine Learning Research* 22, 1 (2021), 4839–4886.
[11] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 457–468.
[12] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2030–2042.
[13] Sadid A Hasan, Yuan Ling, Oladimeji Farri, Joey Liu, Henning Müller, and Matthew P Lungren. 2018. Overview of ImageCLEF 2018 Medical Domain Visual Question Answering Task.. In *CLEF (Working Notes)*.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[15] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).
[16] Binyuan Hui, Ruiying Geng, Lihan Wang, Bowen Qin, Yanyang Li, Bowen Li, Jian Sun, and Yongbin Li. 2022. S2SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers. In *Findings of the Association for Computational Linguistics: ACL 2022*. 1254–1262.
[17] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A Dataset for Biomedical Research Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2567–2577.
[18] Qiao Jin, Zheng Yuan, Guangzhi Xiong, Qianlan Yu, Huaiyuan Ying, Chuanqi Tan, Mosha Chen, Songfang Huang, Xiaozhong Liu, and Sheng Yu. 2022. Biomedical question answering: A survey of approaches and challenges. *ACM Computing Surveys (CSUR)* 55, 2 (2022), 1–36.
[19] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5648–5656.
[20] Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition*. 4999–5007.
[21] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.
[22] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to SQL: where are we today? *Proceedings of the VLDB Endowment* 13, 10 (2020), 1737–1750.
[23] Chuan Lei, Fatma Özcan, Abdul Quamar, Ashish R Mittal, Jaydeep Sen, Diptikalyan Saha, and Karthik Sankaranarayanan. 2018. Ontology-based natural language query interfaces for data exploration. (2018).
[24] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
[25] Fei Li and Hosagrahar V Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment* 8, 1 (2014), 73–84.
[26] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 4870–4888.
[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).
[28] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.
[29] Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records. *arXiv preprint arXiv:1809.00732* (2018).
[30] Junwoo Park, Youngwoo Cho, Haneol Lee, Jaegul Choo, and Edward Choi. 2021. Knowledge graph-based question answering with electronic health records. In *Machine Learning for Healthcare Conference*. PMLR, 36–53.
[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).
[32] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. 149–157.
[33] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. *arXiv preprint arXiv:2208.13629* (2022).
[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2383–2392.
[35] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Özcan. 2016. ATHENA: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1209–1220.
[36] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).
[37] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 9895–9901.
[38] Jaydeep Sen, Fatma Ozcan, Abdul Quamar, Greg Stager, Ashish Mittal, Manasa Jammi, Chuan Lei, Diptikalyan Saha, and Karthik Sankaranarayanan. 2019. Natural language querying of complex business intelligence queries. In *Proceedings of the 2019 International Conference on Management of Data*. 1997–2000.
[39] Saeedeh Shekarpour, Soren Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler. 2011. Keyword-driven sparql query generation leveraging background knowledge. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Vol. 1. IEEE, 203–210.
[40] Simon Suster and Walter Daelemans. 2018. CliCR: a Dataset of Clinical Case Reports for Machine Reading Comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 1551–1563.
[41] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics* 16, 1 (2015), 138.
[42] Ping Wang, Tian Shi, Khushbu Agarwal, Sutanay Choudhury, and Chandan K Reddy. 2022. Attention-based aspect reasoning for knowledge base question answering on clinical notes. In *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. 1–6.
[43] Ping Wang, Tian Shi, and Chandan K Reddy. 2020. Text-to-SQL generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*. 350–361.
[44] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R Radev, Richard Socher, and Caiming Xiong. 2021. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. In *ICLR*.
[45] Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 1962–1979.
[46] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3911–3921.
[47] Hamid Zafar, Giulio Napolitano, and Jens Lehmann. 2018. Formal query generation for question answering over knowledge bases. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 714–728.
[48] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 5338–5349.
[49] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* (2017).