Knowledge Graph-based Question Answering with Electronic Health Records

Junwoo Park JUNWOO.PARK@KAIST.AC.KR
Youngwoo Cho CYW314@KAIST.AC.KR
KAIST / Daejeon, South Korea

Haneol Lee STUDYMODE@YONSEI.AC.KR
Yonsei University / Seoul, South Korea

Jaegul Choo JCHOO@KAIST.AC.KR
Edward Choi EDWARDCHOI@KAIST.AC.KR
KAIST / Daejeon, South Korea

Abstract

Question Answering (QA) on Electronic Health Records (EHR), namely EHR QA, can work as a crucial milestone towards developing an intelligent agent in healthcare. EHR data are typically stored in a relational database, which can also be converted to a Directed Acyclic Graph (DAG), allowing two approaches for EHR QA: Table-based QA and Knowledge Graph-based QA. We hypothesize that the graph-based approach is more suitable for EHR QA as graphs can represent relations between entities and values more naturally compared to tables, which essentially require JOIN operations. To validate our hypothesis, we first construct EHR QA datasets based on MIMIC-III, where the same question-answer pairs are represented in SQL (table-based) and SPARQL (graph-based), respectively. We then test a state-of-the-art EHR QA model on both datasets where the model demonstrated superior QA performance on the SPARQL version. Finally, we open-source[1] both `MIMICSQL*` and `MIMIC-SPARQL*` to encourage further EHR QA research in both directions.

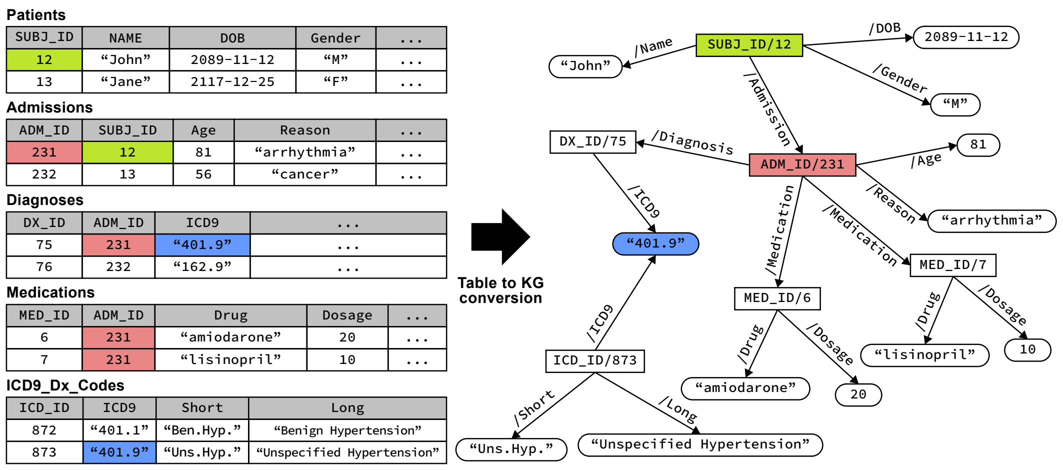
Keywords: Electronic Health Records, Knowledge Graph, Question Answering

1. Introduction

Electronic health records (EHR) consist of heterogeneous data (*e.g.,* demographics, diagnosis, medications, and labs) stored typically in a complex relational database. An agent that can understand EHR and perform complex reasoning is not only an advancement in AI research, but also has great potential in practical applications such as clinical decision support, hospital administration, and medical chatbots.

Recently, the Question-Answering (QA) task has played a vital role in developing and evaluating machine intelligence (Bordes et al., 2014; Antol et al., 2015; Seo et al., 2017; Zhong et al., 2017) , and we also aim to develop an intelligent EHR agent in a QA framework, namely EHR QA. Although EHR data are typically stored in a relational database (RDB), it can be seen as a Directed Acyclic Graph (DAG), or more specifically, a Knowledge Graph (KG) of clinical facts (see Figure 1), thus allowing two approaches for EHR QA: table-based QA and graph-based QA. We hypothesize that the graph-based approach is an advantageous choice for EHR QA as graphs can

1. https://github.com/junwoopark92/mimic-sparql



Patients

SUBJ_ID	NAME	DOB	Gender	...
12	"John"	2089-11-12	"M"	...
13	"Jane"	2117-12-25	"F"	...

Admissions

ADM_ID	SUBJ_ID	Age	Reason	...
231	12	81	"arrhythmia"	...
232	13	56	"cancer"	...

Diagnoses

DX_ID	ADM_ID	ICD9	...
75	231	"401.9"	...
76	232	"162.9"	...

Medications

MED_ID	ADM_ID	Drug	Dosage	...
6	231	"amiodarone"	20	...
7	231	"lisinopril"	10	...

ICD9_Dx_Codes

ICD_ID	ICD9	Short	Long
872	"401.1"	"Ben.Hyp."	"Benign Hypertension"
873	"401.9"	"Uns.Hyp."	"Unspecified Hypertension"

Figure 1: A subset of tables and their corresponding knowledge graph (best viewed in color). Rectangles and ellipse in the graph indicate entities and literal values, respectively. Those keys linking two tables are color-coded and denoted in the graph.

represent relations between entities and values more naturally compared to tables, which require *JOIN* operations.

The goal and contribution of this work include the following. First, for an empirical comparison between the two, we construct both a table-based and a graph-based datasets based on MIMICSQL (Wang et al., 2020), an existing table-based EHR QA dataset. We modify MIMICSQL to follow the original schema of MIMIC-III (Johnson et al., 2016), thus building MIMICSQL⋆ and its graph-based counterpart MIMIC-SPARQL⋆. Second, we test the state-of-the-art EHR QA model called TREQS (Wang et al., 2020), on both MIMICSQL⋆ and MIMIC-SPARQL⋆ where TREQS gained a 5.1% boost in predicting the correct relations between entities, and a 3.6% boost in predicting the correct answer. To the best of our knowledge, this is the first work to propose and empirically demonstrate the graph-based EHR QA. Finally, we open-source both MIMICSQL⋆ and MIMIC-SPARQL⋆ to encourage further EHR QA research in both directions.

1.1. Related Work

QA is an active research area, including machine reading comprehension (MRC) (Rajpurkar et al., 2016; Nguyen et al., 2016), visual question answering (VQA) (Antol et al., 2015; Johnson et al., 2017), table-based QA (Neelakantan et al., 2015; Zhong et al., 2017), and graph-based QA (Berant et al., 2013; Yih et al., 2015). EHR QA is a research area with great potential for real-world impact, as it provides an interface where both medical experts and ordinary users can easily query clinical facts and statistics via natural language. Previously, (Pampari et al., 2018) constructed emrQA, an MRC dataset based on clinical notes. emrQA, however, focuses only on the free-text notes, thus not being able to be used to train machines to leverage the structural properties of EHR. Recently, (Wang et al., 2020) used MIMIC-III (Johnson et al., 2016), a publicly available EHR dataset, to construct MIMICSQL, a table-based (*i.e.*, text-to-SQL) QA dataset. However, while constructing MIMICSQL, they preprocessed the tables of MIMIC-III such that the original schema was heavily modified. Therefore the final dataset (questions and tables) does not reflect the complex, yet interest-

	MIMIC SQL⋆	MIMIC-SPARQL⋆
# of patients	100	
# of Question-Query pairs	10,000	
Avg. NL question length	16.45	
Avg. SQL/SPARQL length	44,68 (21.04)	28.98 (27.28)
Max # of *INNER JOIN*s	5 (2)	-
Max depth of graph	-	5 (3)
# of triples	-	173,096 (257,187)

Table 1: Basic statistics of MIMICSQL⋆ and MIMIC-SPARQL⋆. The values in the parentheses are from the original MIMICSQL and its counterpart MIMIC-SPARQL.

ing structure of MIMIC-III, thus losing real-world applicability and the value as a tool for fostering machine intelligence.

2. Dataset

To the best of our knowledge, there is no existing dataset for graph-based EHR QA (*i.e.* text-to-SPARQL). In this section, we address the schema issue of MIMICSQL and create MIMICSQL⋆ that preserves the true structure of MIMIC-III.

2.1. MIMICSQL to MIMICSQL*

Wang et al. (2020) chose a subset of information from nine tables of MIMIC-III (*i.e.* Patients, Admissions, Diagnoses, Prescriptions, Procedures, Labs, Diagnosis Descriptions, Procedure Descriptions, and Lab Descriptions), then simplified their structure by merging them into five tables: Demographics, Diagnoses, Procedures, Prescriptions, and Labs. By doing so, they compromised database normalization, such as *JOIN*ing Diagnoses with Diagnosis Descriptions, thus causing duplicate column values in multiple rows.

Although MIMICSQL can serve as a basic clinical QA dataset, their tables are unnormalized and simpler than the tables used in actual hospitals. In other words, a machine trained on MIMICSQL would not generalize well to an actual hospital environment, since it was trained in an unrealistic problem space. Therefore, to preserve the schema of MIMIC-III, we normalize MIMICSQL tables back to the original nine tables, following the original schema.

We also modified the SQL part of MIMICSQL's question-SQL pairs, such that the new tables and columns were correctly reflected. For example, in MIMICSQL, a patient's name and age are stored in the Demographic table. However, the original schema requires a JOIN operation to find a patient's age by their names, since they are stored in Admissions and Patients, respectively. This modification increased the average number of tokens of SQL from 21.04 to 44.68, due to the added JOIN operations. We denote this modified dataset as MIMICSQL⋆. Appendix A.2 and Appendix A.3 describe the methods for converting the tables to the Knowledge Graph and the SQL queries to the SPARQL queries, respectively.

Method	Dataset	# of Tables	Development			Testing		
			Acc_{LF}	Acc_{EX}	Acc_{ST}	Acc_{LF}	Acc_{EX}	Acc_{ST}
Seq2Seq	MIMICSQL (Paper)	5	0.076	0.112	-	0.091	0.131	-
	MIMICSQL	5	0.229	**0.429**	0.494	**0.304**	**0.440**	0.540
	MIMIC-SPARQL	5	**0.243**	0.334	**0.665**	0.298	0.400	**0.596**
	MIMICSQL⋆	9	0.151	0.302	0.305	0.209	0.313	0.334
	MIMIC-SPARQL⋆	9	**0.244**	**0.345**	**0.631**	**0.285**	**0.387**	**0.588**
TREQS	MIMICSQL (Paper)	5	0.451	0.511	-	0.486	0.556	-
	MIMICSQL	5	**0.583**	0.705	0.833	**0.658**	**0.727**	**0.815**
	MIMIC-SPARQL	5	0.576	**0.709**	**0.834**	0.646	0.722	0.809
	MIMICSQL⋆	9	0.581	0.672	0.742	0.621	0.695	0.753
	MIMIC-SPARQL⋆	9	**0.603**	**0.741**	**0.833**	**0.657**	**0.731**	**0.826**

Table 2: Medical QA performance evaluated with logic form accuracy (Acc_{LF}), execution accuracy (Acc_{EX}) and the structural accuracy (Acc_{ST}). In addition to MIMICSQL, we include MIMICSQL (Paper), the performance reported in Wang et al. (2020).

3. Experiments

3.1. Experiments Setting

Methods. We employed two sequence-to-sequence-based approaches: Seq2Seq model (Luong et al., 2015) and TREQS (Wang et al., 2020). Seq2Seq uses a bidirectional LSTM encoder and an LSTM decoder with an attention mechanism, but it cannot handle out-of-vocabulary (OOV) tokens that frequently occur in medical data. Proposed with MIMICSQL, TREQS is a state-of-the-art question-to-SQL translation model for clinical questions, equipped with the copying-mechanism (See et al., 2017) to address the OOV issue, and the attention mechanism (Bahdanau et al., 2015) to help SQL token generation.

Datasets. To demonstrate the performance between SPARQL and SQL, we compare MIMICSQL against MIMIC-SPARQL, and MIMICSQL⋆ against MIMIC-SPARQL⋆. Additionally, we found an improved method to tokenize SQLs of MIMICSQL (detailes described in Appendix A.1)

Metrics. The execution accuracy Acc_{EX} is calculated based on the correctness of the answer retrieved by querying the tables/KG with the generated SQL/SPARQL. The logic form accuracy (Zhong et al., 2017) Acc_{LF} is calculated by comparing the generated SQL/SPARQL with the truth SQL/SPARQL token-by-token. Additionally, to evaluate how well the model understands the relations between columns/entities, we use the structural accuracy Acc_{ST}, which is equivalent to Acc_{LF} except that the condition value tokens (*e.g.*, numeric values or string values) are ignored.

3.2. Experiments Results

As shown in Table 2, the SQL dataset (MIMICSQL) shows competitive performance to its SPARQL counterpart (MIMIC-SPARQL) for the simplified schema with five tables. For the original MIMIC-III schema, however, the graph-based approach (MIMIC-SPARQL⋆) consistently outperforms the table-based approach (MIMICSQL⋆). It is noteworthy that the Acc_{ST} gap between SQL and SPARQL dramatically increases when moving from the simple schema to the original. Furthermore, while there was a significant performance drop when moving from MIMICSQL to MIMICSQL⋆, espe-

Method	**Dataset**	**# of *INNER JOIN*s**					
		0	1	2	3	4	5
Seq2Seq	`MIMICSQL⋆`	37	134	21	17	0	0
Seq2Seq	`MIMIC-SPARQL⋆`	29	180	43	31	2	0
TREQS	`MIMICSQL⋆`	79	276	145	93	2	0
TREQS	`MIMIC-SPARQL⋆`	79	289	163	111	1	0

Table 3: The number of correctly generated queries for test questions binned by the number of *JOIN*s in the SQLs of `MIMICSQL⋆`. The SPARQLs are classified without *JOIN*s because both queries are derived from the same question

cially when using a simpler model (*i.e.* Seq2Seq), there was either marginal decrease or even slight increase in performance when moving from `MIMIC-SPARQL` to `MIMIC-SPARQL⋆`, indicating the effectiveness of the graph-based approach as the knowledge structure becomes more complex. These results support our hypothesis that the graph-based approach is a better choice for EHR QA as graphs can represent relations between entities and values more concisely compared to tables, which require *JOIN* operations. Specifically, based on the statistics of the dataset, Table 1 shows that the query length of SQL is nearly twice as long as that of SPARQL. This is because a single JOIN in SQL requires 11 tokens (including '=' and '.'), while a single hop in SPARQL requires only 3 (i.e. subject, predicate, object). Such syntactic difference between SQL and SPARQL originates from the intrinsic difference between the relational tables and a knowledge graph in terms of how to connect information (joining tables and hopping triples). This leads to the superior performance of the graph-based approach over the table-based one especially in Acc_{ST}, which only evaluates the structural accuracy (i.e. disregarding the correctness of condition values). Table 3 also supports our analysis by showing that the graph-based approach does not reduce the number of correctly generated queries compared to table-based for varying query complexities.

4. Conclusion

In this work, we constructed a pair of EHR QA datasets, one table-based (`MIMICSQL⋆`) and another graph-based (`MIMIC-SPARQL⋆`). With extensive experiments, we empirically demonstrated the superior performance of the graph-based approach. To the best of our knowledge, this is the first work to propose and successfully demonstrate the graph-based EHR QA. As future work, we plan to extend both our datasets (`MIMICSQL⋆` and `MIMIC-SPARQL⋆`) to cover the entire MIMIC-III.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proc. of the IEEE international conference on computer vision (ICCV)*, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. the International Conference on Learning Representations (ICLR)*, 2015.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Nilesh Chakraborty, Denis Lukovnikov, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann, and Asja Fischer. Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*, 2019.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. In *Proc. the International Conference on Learning Representations (ICLR)*, 2015.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: a human-generated machine reading comprehension dataset. 2016.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. emrqa: A large corpus for question answering on electronic medical records. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

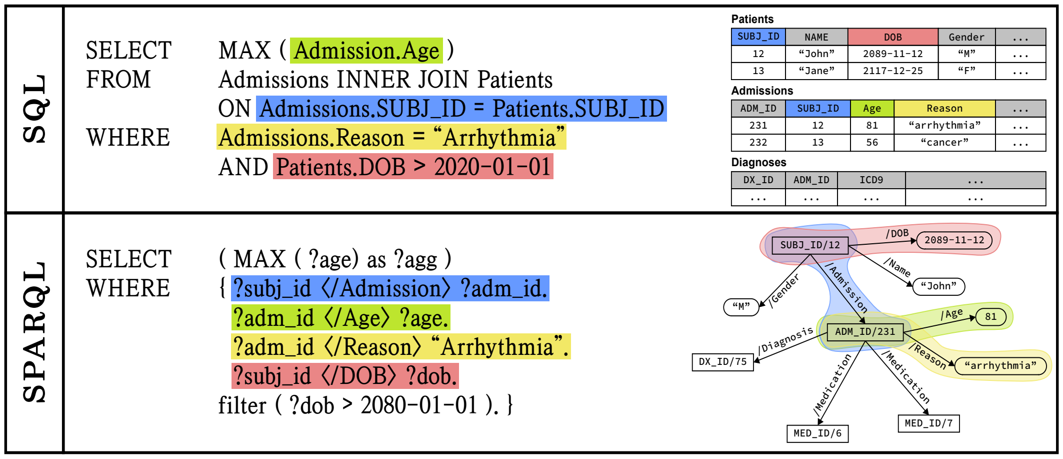
Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proc. the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *Proc. the International Conference on Learning Representations (ICLR)*, 2017.

Ping Wang, Tian Shi, and Chandan K Reddy. Text-to-sql generation for question answering on electronic medical records. In *Proc. the International Conference on World Wide Web (WWW)*, 2020.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 2015.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.



```
SELECT    MAX ( Admission.Age )
FROM      Admissions INNER JOIN Patients
          ON Admissions.SUBJ_ID = Patients.SUBJ_ID
WHERE     Admissions.Reason = "Arrhythmia"
          AND Patients.DOB > 2020-01-01
```

Patients

SUBJ_ID	NAME	DOB	Gender	...
12	"John"	2089-11-12	"M"	...
13	"Jane"	2117-12-25	"F"	...

Admissions

ADM_ID	SUBJ_ID	Age	Reason	...
231	12	81	"arrhythmia"	...
232	13	56	"cancer"	...

Diagnoses

DX_ID	ADM_ID	ICD9	...
...	...	...	...

```
SELECT    ( MAX ( ?age) as ?agg )
WHERE     { ?subj_id ⟨/Admission⟩ ?adm_id.
            ?adm_id ⟨/Age⟩ ?age.
            ?adm_id ⟨/Reason⟩ "Arrhythmia".
            ?subj_id ⟨/DOB⟩ ?dob.
          filter ( ?dob > 2080-01-01 ). }
```

Figure 2: SQL & SPARQL Pair: The highlighted parts in SQL and SPARQL correspond to each other. SQL requires explicit JOIN operations when linking information across tables, whereas SPARQL just hops between nodes.

Appendix A. Dataset Detail

This section provides the detailed explanation about datasets and methods that convert the tables into KG.

A.1. Performance Gap between MIMICSQL (paper) and MIMICSQL

In the queries provided by MIMICSQL, the table name and the column name are not separated and are considered as a single token. This makes it difficult for the model to learn the relationship between the *SELECT* clause and *FROM* clause. Thus, we separate the table name from the column name and report the effectiveness of this tokenization in the MIMICSQL. Table 2 shows that training for MIMICSQL increases performance for all metrics compared with MIMICSQL (paper). By simply changing tokenization method, the execution accuracy increased by 58% and 48% in the Seq2Seq and TREQS, respectively.

A.2. Table to Knowledge Graph

Knowledge Graphs (KG) are a set of triples that describe the relationship between two entities or between an entity and a literal value (Chakraborty et al., 2019). To extract triples from the nine tables of MIMICSQL⋆, we defined a simple algorithm to map each column to either an entity, a literal, or a relation. The primary or foreign key columns of a table are defined as entities, and non-key columns (*i.e.* property columns) are defined as literals. For example, in the Patients table of Figure 1, the values of SUBJ_ID are defined as entities that can have child nodes. The values of Name, DOB (Date-of-Birth), and Gender are defined as literals that cannot have child nodes. The column names of the non-key columns become relations, *e.g.* /Name links between SUBJ_ID/12 and "John". Lastly, the relation between a primary key and a foreign key is derived from the table name (*e.g.* /Admission links between SUBJ_ID/12 and ADM_ID/231). The final KG consists of 173,096 triples and has a max depth of 5.

A.3. SQL to SPARQL

In order to create question-SPARQL pairs from question-SQL pairs, we treat the *SELECT* clause and the *WHERE* clause of SPARQL separately. First, we copy the aggregation function (*e.g.* MAX, AVERAGE) and the columns from the *SELECT* part of SQL, and fill the SPARQL template. Then we extract the condition columns, comparison operator, and condition values from the *WHERE* part of SQL. In order to fill the *WHERE* clause of SPARQL, which consists of triples between variables, we find the shortest distance between the variable in the KG[2]. As for combining comparison operators (*e.g.* $>$, $<$, $=$) with triples, equal operators are handled by replacing the variable with values, whereas all other operators are handled by adding filter statements. An example of SQL-to-SPARQL conversion is depicted by Figure 2. After the conversion, we checked the semantic equivalence of SQLs and SPARQLs by querying them to the tables and the KG, where we confirmed a 100% match. The KG and SPARQL queries comprise the graph-based EHR QA dataset `MIMIC-SPARQL⋆`. Note that we also constructed `MIMIC-SPARQL` based on the five tables of `MIMICSQL` for further empirical analysis. Table 1 describes the basic statistics of `MIMICSQL⋆`, `MIMIC-SPARQL⋆`, `MIMICSQL`, and `MIMIC-SPARQL`.

Appendix B. Experiment Detail

This section describes the detailed experimental environment and hyperparameters of the model and includes qualitative results generated from baseline models.

B.1. Implementation Detail

In the experiments section, we evaluate two baseline methods based on Sequence-to-Sequence framework. We used Pytorch to implement the Seq2Seq model. In order to carefully compare the grammatical difference between two query languages without modifying the model architecture, we utilized the official code[3] of TREQS and hyperparameters described in (Wang et al., 2020). We only tuned hyperparameters, which are irrelevant to the model size with the same methods. The number of trainable parameters of TREQS is $2,795,907$ and Seq2Seq is $3,844,096$ throughout all experiments. As shown in Table 4, the hyperparameters are chosen to optimize the model for the dataset without increasing the model size and sampled from uniform distribution.

We train all baselines from scratch to 20 epochs. The development set is utilized to select the best configuration. In the test time, we employ a beam search algorithm to generate the output query sentences and the beam size is set to be 5. Training of both models require an average of one hour on a TITAN-Xp GPU.

B.2. Qualitative Results

This section demonstrates the qualitative results to reveal differences between the datasets and differences between two methods. For a NL question with the same meaning, we generate a query for four cases shown in Figure 3. In terms of methods, TREQS handles OOV correctly by copying the condition value in the NL question. However, Seq2Seq generates <UNK> tokens for condition value token on both `MIMICSQL⋆` and `MIMIC-SPARQL⋆` datasets because there exist several abbreviations and low-frequency words in EHR. As we mentioned in Section 3.2, TREQS that learns

2. Since we know the schema of MIMIC-III, we can create a relation graph and apply Dijkstra's algorithm.
3. https://github.com/wangpinggl/TREQS

Method	**Dataset**	**Hyperparameters**		
		Parameter	Search space	Selected value
Seq2Seq	MIMICSQL⋆	Batch Size	$16, 32, 48$	32
		Step decay	$[0.01, 0.8]$	0.1
		Step size	$1, 2, 5, 10$	2
		Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	0.001
	MIMIC-SPARQL⋆	Batch Size	$16, 32, 48$	16
		Step decay	$[0.01, 0.8]$	0.8
		Step size	$1, 2, 5, 10$	2
		Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	0.001
TREQS	MIMICSQL⋆	Batch Size	$16, 32, 48$	16
		Step decay	$[0.01, 0.8]$	0.8
		Step size	$1, 2, 5, 10$	2
		Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	0.0005
	MIMIC-SPARQL⋆	Batch Size	$16, 32, 48$	48
		Step decay	$[0.01, 0.8]$	0.1
		Step size	$1, 2, 5, 10$	2
		Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	0.0005

Table 4: Description of hyperparameters

from MIMICSQL⋆ shows the poor results to predict the exact relationship between tables. On the other hand, the model, which learns from MIMIC-SPARQL⋆ predicts all entity relationships from the corresponding NL questions. Additionally, the SPARQL query can be executed only when the data type of the condition value is correctly generated as a token such as "*xmlschema/string*" for "*other shock without mention of trauma*".

NL	What number of married patients suffered from other convulsions?
GT MIMICSQL*	select count (distinct patients . "subject_id") from patients inner join admissions on patients . "subject_id" = admissions . "subject_id" inner join diagnoses on admissions . "hadm_id" = diagnoses . "hadm_id" inner join d_icd_diagnoses on diagnoses . "icd9_code" = d_icd_diagnoses . "icd9_code" where d_icd_diagnoses . "long_title" = "other shock without mention of trauma"
Seq2Seq MIMICSQL*	select count (distinct patients . "subject_id") from patients inner join admissions on patients . "subject_id" = admissions . "subject_id" inner join diagnoses on admissions . "hadm_id" = diagnoses . "hadm_id" inner join d_icd_diagnoses on diagnoses . "icd9_code" = d_icd_procedures . "icd9_code" where d_icd_diagnoses . "long_title" = <UNK> of <UNK> of <UNK> and <UNK>
TREQS MIMICSQL*	select count (distinct patients . "subject_id") from patients inner join admissions on patients . "subject_id" = admissions . "subject_id" inner join procedures on admissions . "hadm_id" = procedures . "hadm_id" inner join d_icd_procedures on procedures . "icd9_code" = d_icd_procedures . "icd9_code" where d_icd_procedures . "long_title" = "other shock without mention of trauma"
GT MIMIC-SPARQL*	select (count (distinct ?subject_id) as ?agg) where { ?subject_id </hadm_id> ?hadm_id. ?hadm_id </diagnoses> ?diagnoses. ?diagnoses </diagnoses_icd9_code> ?diagnoses_icd9_code. ?diagnoses_icd9_code </diagnoses_long_title> "other shock without mention of trauma ^^<http://www.w3.org/2001/xmlschema#string>. }
Seq2Seq MIMIC-SPARQL*	select (count (distinct ?subject_id) as ?agg) where { ?subject_id </hadm_id> ?hadm_id. ?hadm_id </procedures> ?procedures. ?procedures </procedures_icd9_code> ?procedures_icd9_code. ?procedures_icd9_code </procedures_long_title> "other shock without mention of <UNK> ^^<http://www.w3.org/2001/xmlschema#string>. }
TREQS MIMIC-SPARQL*	select (count (distinct ?subject_id) as ?agg) where { ?subject_id </hadm_id> ?hadm_id. ?hadm_id </diagnoses> ?diagnoses. ?diagnoses </diagnoses_icd9_code> ?diagnoses_icd9_code. ?diagnoses_icd9_code </diagnoses_long_title> "other shock without mention of trauma ^^<http://www.w3.org/2001/xmlschema#string>. }

Figure 3: Generated examples