THE UNIVERSITY OF CAPE TOWN

MASTERS THESIS

Towards Answering Unanswerable Questions: Data Augmentation for Enhanced Medical Domain Question Answering

Author:
Natalie Bianca ALEXANDER

Supervisor:
Dr Jan BUYS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Data Science

in the

Department of Statistical Science and Department of Computer Science

June 1, 2025

Declaration of Authorship

I, Natalie Bianca ALEXANDER, declare that this thesis titled, "Towards Answering Unanswerable Questions: Data Augmentation for Enhanced Medical Domain Question Answering" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Signed by candidate

Date:

"Whatever you do, work at it with all your heart..."

Colossians 3:23

THE UNIVERSITY OF CAPE TOWN

Abstract

Faculty of Science
Department of Statistical Science and Department of Computer Science

Master of Science in Data Science

Towards Answering Unanswerable Questions: Data Augmentation for Enhanced Medical Domain Question Answering

by Natalie Bianca ALEXANDER

Hospitals store patient information in relational databases known as Electronic Health Records (EHRs). Existing EHRs have filter and search options on the front end that are converted to SQL queries at the back end. However, these search and filter options become cumbersome when querying the EHR. While users could write custom SQL queries to query the EHR directly, this approach requires database expertise. Recent advancements in medical question-answering leverage text-to-SQL parsing, which translates a user's natural language question into an executable SQL query, enabling information retrieval from a database. However, current medical text-to-SQL research only addresses a limited scope of questions, known as answerable questions. Questions that the system cannot reliably answer (unanswerable questions) result in inexecutable or incorrect SQL predictions which may return incorrect information that affects clinical decision-making. This limitation underscores the need for a medical text-to-SQL system that can reliably address both answerable and unanswerable questions. This project aims to expand the coverage of questions answered by medical text-to-SQL systems, by addressing unanswerable questions that are out-of-schema or require medical knowledge to simplify complex, medical jargon. More specifically, we focus on addressing real-world unanswerable questions related to diagnoses and medication. This research first explores methods for addressing out-of-schema questions by assessing how incorporating an unseen schema, during inference, enhances the performance of a sequence-to-sequence (T5) text-to-SQL model. We then compare this approach to the effectiveness of fine-tuning the model on a training dataset that includes these out-of-schema questions and their corresponding schema. Secondly, this research examines how external medical knowledge sources, related to diagnoses and medication, can be used in data augmentation (either through retrieval-augmented generation or SQL post-processing) to improve the answerability of unanswerable questions with complex medical jargon. In addition, we ensure model reliability by applying answer abstention when the text-to-SQL model cannot reliably answer a question, while also ensuring that the model does not deteriorate the answerability of the original answerable questions. As a result of these experiments, we find that out-of-schema questions are addressed by fine-tuning a T5-Base model on a training dataset that includes out-of-schema question representations, excluding additional schema information. In addition, we find that fine-tuning a T5-Large model with retrieval-augmented generation, which incorporates medical knowledge from the SNOMED CT and RxNorm medical vocabularies, improves the model's ability to answer unanswerable questions with complex medical jargon. We also find that an entropy-based uncertainty estimation method, which uses K-means clustering to establish the abstention threshold, is suitable for answer abstention. Finally, we find that our proposed models do not compromise the answerability of the original answerable questions.

Acknowledgements

I extend my deepest gratitude to my supervisor, Dr Jan Buys, and my external supervisors at IBM Research Johannesburg, Dr Joan Byamugisha and Dr Ndivhuwo Makondo, for their unwavering support and insightful critiques throughout my research journey. Their commitment to academic excellence and meticulous attention to detail have significantly shaped this dissertation.

Contents

List of Figures

List of Tables

List of Abbreviations

ADR: **A**dverse **D**rug **Reaction**
AST: **A**bstract **S**yntax **T**ree (AST)
BCE: **B**iomedical **C**oncept **E**xtraction
BiLSTM: **B**idirectional **L**ong **S**hort-**T**erm **M**emory
BIRD: **Bi**g Bench for la**R**ge-scale **D**atabases
CLM: **C**ausal **L**anguage **M**odelling
CM: **C**omponent **M**atching accuracy
CUI: **C**oncept **U**nique **I**dentifier
EDA: **E**xploratory **D**ata **A**nalysis
EHR: **E**lectronic **H**ealth **R**ecord
eICU: **E**lectronic **I**ntensive **C**are **U**nit
EM: **E**xact **M**atching accuracy
EX: **Ex**ecution accuracy
ICD-10: **I**nternational **C**lassification of **D**iseases, 10th Revision
ICD-10-CM: **I**nternational **C**lassification of **D**iseases, 10th Revision, **C**linical **M**odification
ICD10-PCS-10: textbfInternational **C**lassification of **D**iseases, 10th Revision, **P**rocedure **C**oding **S**ystem
ICL: **I**n-**C**ontext **L**earning
IDE: **I**ntegrated **D**evelopment **E**nvironment
HPC: **H**igh **P**erformance **C**omputing
JSON: **J**avaScript **O**bject **N**otation
LCS: **L**ongest **C**ommon **S**ubstring
LDA: **L**atent **D**irichlet **A**llocation
LM: **L**anguage **M**odel
LLM: **L**arge **L**anguage **M**odel
LLT: **L**owest **L**evel **T**erm
LSTM: **L**ong **S**hort-**T**erm **M**emory
MCE: **M**edical **C**oncept **E**xtraction
MeDRA: The **M**edical **D**ictionary for **R**egulatory **A**ctivities
MeSH: **Me**dical **S**ubject **H**eadings
MIMIC-III: **M**edical **I**nformation **M**art for **I**ntensive **C**are
MLM: **M**asked **L**anguage **M**odelling
NLM: **N**ational **L**ibrary of **M**edicine
NLP: **N**atural **L**anguage **P**rocessing
NLQ: **N**atural **L**anguage **Q**uestion
NLU: **N**atural **L**anguage **U**nderstanding
PyMedTermino: **M**edical **T**erminologies for **P**ython)
PLM: **Pre-trained Language Model**
QA: **Q**uestion **A**nswering
RAG: **R**etrieval **A**ugmented **G**eneration
RLHF: **R**einforcement **L**earning with **H**uman **F**eedback
RNN: **R**ecurrent **N**eural **N**etwork
RoPE: **R**otary **P**osition **E**mbedding
Seq2Seq: **S**equence-to-**S**equence
SFT: **Supervised Fine-Tuning**
SIDER: **S**ide **E**ffect **R**esource
SNOMED CT: **S**ystematized **No**menclature of **Med**icine - **C**linical **T**erms
SOTA: **S**tate **o**f **t**he **A**rt
SQL: **S**tructured **Q**uery **L**anguage
SVM: **S**upport **V**ector **M**achine

TF-IDF: **T**erm-**F**requency **I**nverse-**D**ocument **F**requency
TUI: **T**ype **U**nique **I**dentifier
QA: **Q**uestion **A**nswering
UMLS: The **U**nified **M**edical **L**anguage **S**ystem

This dissertation is dedicated to my late father, Allister Walter, whose encouragement inspired my pursuit of science. He instilled in me the importance of using my skills and knowledge for the greater good. I am also profoundly grateful to my mother, Margo, whose support and inspiration have been a guiding light throughout my journey. I would also like to extend my heartfelt thanks to my siblings, Audree, Kazwin and Nathan for their invaluable support and guidance, which have been instrumental in helping me reach this milestone. Finally, I would also like to express my gratitude to Joshua for inspiring me to pursue my academic goals with unwavering courage.

Chapter 1

Introduction

1.1 Background

Hospitals store longitudinal patient data in relational databases known as Electronic Health Records (EHRs) (Thakur et al., 2023). The data includes patient demographics, patient admission and discharge, laboratory results, diagnoses, and medical prescriptions, to name a few. The structured nature of the EHR makes the plethora of clinical information more manageable, facilitating efficient navigation of related entities through its relational design. This systematic organization enables medical professionals, including physicians, nurses and administrative personnel, to derive insights at both individual and group levels (Lee et al., 2022). These insights inform the decision-making of medical professionals, including decisions involving the verification of patient prescriptions and the accurate assignment of diagnostic codes in medical reports, among other critical decisions.

A challenge arises, however, where EHR users require specialized database knowledge to query the EHR (Lee et al., 2022). This includes proficiency in a database query language, such as the structured query language (SQL), along with schema familiarity (Li et al., 2024). Schema familiarity requires user awareness of schema items, including table names, column names, column values, table and column constraints, column data types and table relationships within a database. Certain hospitals bypass these challenges with the adoption of rule-based EHRs, relying on search and filter options (on the front end) that are translated to SQL queries (at the back end) (Wang et al., 2020). This approach, however, limits user queries to the available search and filter options. In addition, the search and filter options become cumbersome when querying the EHR for more complex queries.

These prevailing challenges, in EHR question answering (QA), have warranted the development of more sophisticated EHR QA systems. Recent research on EHR question answering have significantly been influenced by advancements in general domain text-to-SQL translation (Lee et al., 2022). Text-to-SQL translation is a semantic parsing task in natural language processing (NLP) that converts a user's natural language question (NLQ) into an executable SQL query (Li et al., 2023b; Yu et al., 2018c; Zhong et al., 2017). Text-to-SQL parsing expands database usability to non-database experts, enabling users to query databases with natural language alone (Katsogiannis-Meimarakis and Koutrika, 2023; Qin et al., 2022). Consequently, users can explore databases more freely without a requisite for schema and query language familiarity.

Text-to-SQL parsing, however, presents a set of new challenges. Here the text-to-SQL model should correctly interpret the meaning of the NLQ to generate a valid SQL query. This proves to be challenging as natural language is inherently complex (Katsogiannis-Meimarakis and Koutrika, 2023), including several factors such as synonymy (i.e., multiple words with the same meaning), paraphrasing (i.e., linguistic variations in the way sentences are expressed), ambiguity (i.e., expressions with more than one interpretation) and complex domain jargon that require simplification. A further challenge in natural language to SQL translation occurs where the model needs a comprehensive understanding of the SQL syntax and the ability to predict the correct schema items (tables, columns and values) in the SQL query that map to surface forms in the question (Li et al., 2023b). Question to schema mapping (also known as schema pruning or schema linking) proves to be challenging, especially when the question contains references to tables, columns and values not seen during initial model training.

Neural network architectures based on recurrent neural networks (RNNs) (Rumelhart et al., 1986) and transformers (Vaswani et al., 2017) were introduced to address natural language understanding challenges. The sequence-to-sequence (Seq2Seq) architecture, in particular, has proven to be successful in a wide range of NLP

tasks, including text-to-SQL parsing (Li et al., 2023b; Yu et al., 2018c; Zhong et al., 2017), machine translation (Conneau and Lample, 2019; Karita et al., 2019), question answering (Saxena et al., 2022), text classification (Yang, 2019), and coreference resolution (Joshi et al., 2020), among other NLP applications. To effectively train and evaluate the text-to-SQL model, it is essential to utilize high-quality text-to-SQL parsing datasets (also known as benchmarks) that reflect the complexities and nuances of natural language.

Numerous text-to-SQL semantic parsing datasets have been curated for text-to-SQL model training and evaluation. These benchmarks include a set of NLQ and SQL pairs defined over one or more databases. In contrast to the many general domain text-to-SQL benchmarks, medical text-to-SQL benchmarks are limited (Wang et al., 2020). In addition, many medical text-to-SQL parsing datasets do not reflect real-life hospital questions and often only include answerable questions, while excluding unanswerable questions (Lee et al., 2022). Questions that the text-to-SQL model can reliably answer are termed as *answerable* or *feasible*, whereas questions that the text-to-SQL model cannot confidently answer are termed as *unanswerable* or *infeasible* (Lee et al., 2024, 2022). Unanswerable questions are out-of-distribution because the model has not seen such a question and/or its related schema during initial model training. Unanswerable questions include questions with: (i) out-of-schema vocabulary that are unrelated to the training schema, (ii) complex domain jargon that require domain knowledge to simplify and relate to the existing schema, or (iii) ambiguous phrases that need user clarification (Chang et al., 2023; Gan et al., 2021a, 2021b; Lee et al., 2024, 2022). See **Figure 1.1** for a schematic representation of text-to-SQL questions.

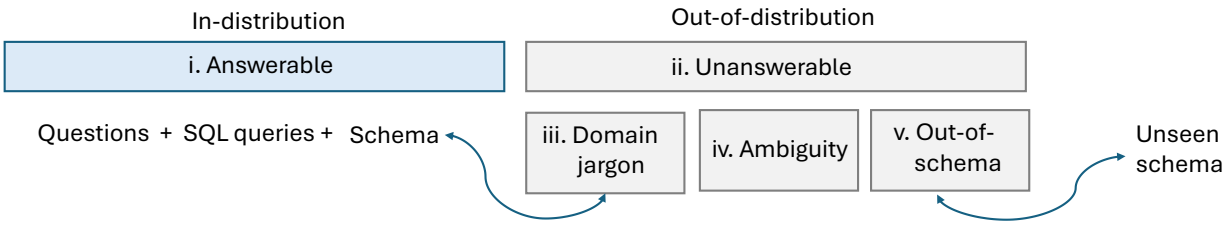


FIGURE 1.1: Schematic Representation of Real-Life Text-to-SQL Questions.

Real-life text-to-SQL models encounter both (i) answerable and (ii) unanswerable questions. (i) Answerable questions refer to questions that are in-distribution, meaning that these question types or their related schemas are seen during initial model training. (ii) Unanswerable questions are out-of-distribution because these question types are not seen during initial model training. Unanswerable questions include questions with (iii) complex, domain jargon that relates to the training schema but requires simplification for the model to understand and interpret its meaning, (iv) ambiguous phrases that require user clarification or (v) out-of-schema concepts that relate to an unseen schema, not seen during initial model training.

1.2 Problem Description

Existing EHR text-to-SQL models face a parsing problem when an unanswerable question is presented at inference (Chang et al., 2023; Gan et al., 2021a, 2021b; Lee et al., 2022; Li et al., 2023b). These unanswerable questions include questions with (i) out-of-schema vocabulary, (ii) complex medical jargon or (iii) ambiguous phrases. The model will either generate an inexecutable SQL query or an incorrect SQL query that returns inaccurate results. Consequently, medical professionals cannot reliably obtain answers from the EHR when the text-to-SQL model encounters unanswerable questions. This limits the usability and reliability of text-to-SQL models, as medical professionals require accurate answers for decision-making.

1.3 Motivation

The risk associated with inexecutable or incorrect text-to-SQL prediction in the medical field is incredibly high (Lee et al., 2024, 2022). For this reason, the text-to-SQL model should correctly answer answerable questions within the feasible region, while also addressing unanswerable questions in the infeasible region. In spite of this, many existing text-to-SQL models only address answerable questions. This thesis proposes that when

the EHR text-to-SQL model encounters an unanswerable medical question, it should have two options, either leveraging additional knowledge to answer the question (Li et al., 2023b) or abstain from answering the question (Lee et al., 2022).

1.4 Aims and Objectives

This research thesis aims to expand the coverage of questions answered by EHR text-to-SQL models. The scope of questions addressed includes questions previously identified as unanswerable by Lee et al. (2022), including important medical questions with out-of-schema concepts (not found in the initial training schema) and unanswerable questions with complex, medical jargon (that do not explicitly relate to the training schema). This thesis does not address ambiguous questions, as these questions require additional user clarification which may be solved with multi-turn user interactions and reinforcement learning with human feedback (RLHF) (Lee et al., 2024). This is beyond the scope of this thesis. More specifically, the objectives are to address important unanswerable questions related to diagnoses and medication, by augmenting the EHR, sequence-to-sequence models (T5) or the SQL post-processing steps with additional schema and medical knowledge. In addition to expanding text-to-SQL answerability, two constraints are put in place. First, the model should not deteriorate in performance when evaluated on the original, answerable questions. Moreover, model reliability is enforced by minimizing incorrect SQL predictions for both answerable and unanswerable questions. This is achieved using an answer abstention mechanism that refuses to answer questions with low model confidence. See **Figure 1.2** for an illustration of the text-to-SQL challenges, solutions and constraints. The success of the improved EHR text-to-SQL model is evaluated using the exact matching accuracy (EM), component matching accuracy (CM) and execution accuracy (EX). EM assesses if the predicted SQL query exactly matches the ground-truth SQL query. CM assesses if specific SQL components, such as the *SELECT*, *FROM* and *WHERE* SQL clauses, in the predicted SQL query match those found in the ground-truth SQL query. EX compares the execution values of the predicted SQL query to the values returned by executing the ground-truth SQL query against the EHR.

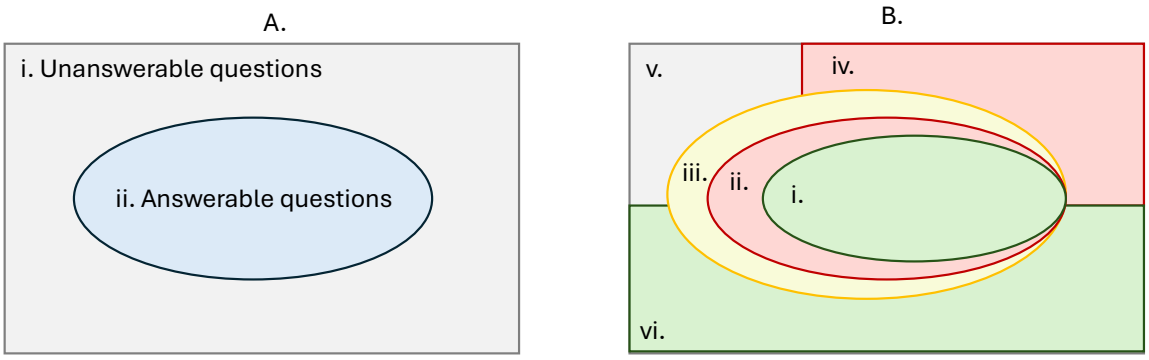


FIGURE 1.2: Text-to-SQL Challenges, Solutions and Constraints.

In frame (a), we see a real-life deployment scenario, where the text-to-SQL model encounters both (i) unanswerable questions in the infeasible region and (ii) answerable questions in the feasible region. In frame (b), we see that the text-to-SQL model can (i) predict the correct SQL queries for answerable questions, (ii) predict incorrect or inexecutable SQL queries for answerable questions, or (iii) incorrectly abstain from answering answerable questions - which does not pose a major risk. The model may also (iv) predict incorrect SQL queries for unanswerable questions, (v) correctly abstain from answering unanswerable questions or (vi) leverage additional knowledge to resolve unanswerable questions. Here green regions indicate correct SQL predictions, red regions indicate incorrect or inexecutable SQL predictions which pose a major risk, yellow regions indicate incorrect answer abstention which does not pose a major risk and grey regions indicate correct answer abstention.

1.5 Research Questions

The research questions addressed include:

1. ***RQ1***: How does the integration of an *unseen* schema during inference enhance the performance of a sequence-to-sequence text-to-SQL model in answering out-of-schema unanswerable questions, and how does this approach compare to the effectiveness of fine-tuning on a relevant training dataset that includes these out-of-schema questions and the *unseen* schema?

2. ***RQ2***: How can medical knowledge sources related to diagnoses and medication terminologies be utilized in data augmentation to simplify complex medical jargon in unanswerable questions?

1.6 Findings and Contributions

In short, this thesis makes the following contributions:

1. We identify specific categories, subcategories and types of real-life unanswerable medical questions within medical text-to-SQL parsing datasets.

2. We create a new EHR grounded with information from medical vocabularies, which we identified to address out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*).

3. We generate unanswerable questions and their associated SQL queries to improve the representation of real-life unanswerable questions in medical text-to-SQL parsing datasets. This includes synthetic reference questions and paraphrases for out-of-schema questions (*RQ1*) and questions with value-synonym replacement for unanswerable questions requiring medical knowledge (*RQ2*). The creation of question paraphrases and questions with value-synonym replacement simulates a real-life scenario where the model will encounter a diversity of questions with varying question structures and lexical variation.

4. We fine-tune a series of models to address out-of-schema questions and unanswerable questions requiring medical knowledge. We find that the T5-Base model—fine-tuned on MIMIC-III (Johnson et al., 2016), eICU (Pollard et al., 2018) and the unanswerable synthetic questions—addresses out-of-schema questions (*RQ1*). We also find that the T5-Large model that leverages retrieval augmented generation—incorporating medical knowledge from the SNOMED CT (Donnelly et al., 2006) and RxNorm (Le et al., 2024) medical vocabularies—simplifies medical jargon in unanswerable questions requiring medical knowledge (*RQ2*).

5. We find that the best models do not deteriorate the answerability of the original answerable questions in MIMIC-III and eICU. Rather, the best model in *RQ1* improves the answerability of the answerable questions and the best model in *RQ2* does not affect the answerability of the original answerable questions.

6. We find that K-means clustering is most effective in determining the abstention threshold, when using an entropy-based abstention method. This is compared to a one-class support vector machine (SVM) and a threshold based on existing literature.

7. We find that erroneous SQL queries are mainly inexecutable or contain incorrect tables and/or columns and/or values.

8. The code can be found here[1].

1.7 Thesis Overview

This thesis includes 7 chapters. In this chapter, Chapter 1, we present the introduction including the background, problem description, motivation, our aims and objectives, the research questions, our findings and contributions and the thesis overview. Chapter 2 provides a literature review that underpins the methods discussed in Chapter 3 and Chapter 4. Chapter 3 details the data sources, exploratory data analyses, data generation, data augmentation and data splitting steps. Chapter 4 discusses the experiments to address research questions *RQ1* and *RQ2*, including model fine-tuning and evaluation. Chapter 5 presents the results, after which Chapter 6 discusses these results in more detail. Finally, Chapter 7 concludes the thesis.

[1] github.com/NatalieAlexander/MSC_ehr_knowledge_augmentation

Chapter 2

Literature Review

2.1 Overview

In Chapter 1, we provide a background of medical text-to-SQL parsing, along with the problem description, motivation, aims and objectives, research questions and the findings and contributions of this research thesis. In this chapter, we review the transformer neural architecture which has demonstrated superior and consistent performance across various NLP tasks. This chapter highlights how transformer-based pre-trained language models (PLMs) with sequence-to-sequence (Seq2Seq) encoder-decoder architectures facilitate the translation of natural language questions (NLQs) into SQL queries. We then discuss the challenges that these models have in addressing unanswerable questions and discuss solutions to address unanswerable questions both in general domain and medical text-to-SQL parsing. These solutions include leveraging additional schema and domain knowledge, as well as answer abstention mechanisms. By examining the strengths and weaknesses of existing text-to-SQL research, this chapter aims to provide insight into the past, present and future of medical text-to-SQL parsing. This chapter first reviews transformers (see Section 2.2), after which we review pre-trained language models, with a particular focus on T5 (Raffel et al., 2020) (see Section 2.3). Finally, Section 2.4 introduces text-to-SQL parsing, where Section 2.5 reviews general domain text-to-SQL parsing and Section 2.6 reviews medical text-to-SQL parsing.

2.2 Transformers

Transformers (Vaswani et al., 2017) have become the state-of-the-art (SOTA) neural network architecture in natural language processing (NLP), surpassing traditional recurrent neural networks (RNNs) (Rumelhart et al., 1986) in most tasks (Rumelhart et al., 1986). Transformer-based language models excel at both natural language understanding (NLU) and natural language generation tasks. Moreover, transformers are capable of capturing long-range dependencies in sequential data, while scaling with increasing training data and model size, facilitating efficient parallel training. This is in contrast to traditional RNNs which suffer from memory decay over long sequences and sequentially process and predict tokens in a sequence. Three classes of transformers exist, encoder-only, decoder-only and encoder-decoder architectures. Examples of encoder-only transformers include BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and Longformer (Beltagy et al., 2020); while decoder-only transformers include Mistral (Jiang et al., 2023), GPT (Achiam et al., 2023; Brown et al., 2020; Radford et al., 2019), PaLM (Chowdhery et al., 2022), and Llama (Touvron et al., 2023), to name a few. Encoder-decoder transformer architectures include BART (Lewis et al., 2020), Pegasus (Zhang et al., 2020a), and T5 (Raffel et al., 2020).

This section introduces transformers with a specific focus on encoder-decoder architectures. We highlight their strengths in NLP tasks attributed to self-attention and positional encoding mechanisms.

Architecture

The transformer architecture, introduced by Vaswani et al. (2017), eliminates the recurrent connections characteristic of RNNs. In contrast to RNNs, which process input tokens sequentially and rely on internal hidden states to retain memory, transformers utilize self-attention to simultaneously capture inter-token relationships across the entire sequence through parallel processing. This enables transformers to capture both long-range and short-range inter-token dependencies without the limitations imposed by sequential processing. In addition, transformers employ positional encoding to retain sequence order during parallel processing. This architectural shift significantly enhances model performance.

Encoder and Decoder Stacks

When modeling a sequence (X), the transformer receives a sequence of tokens as input (Vaswani et al., 2017). Within the encoder-decoder architecture, the encoder maps the input sequence of tokens (X = X_1, ..., X_n) to a sequence of continuous representations (Z = Z_1, ..., Z_n). Subsequently, the decoder then processes Z as input and generates an output sequence of tokens (Y = Y_1, ..., Y_n). Each step is auto-regressive, leveraging the previous decoded output as additional input in subsequent generation steps. The transformer does this using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder.

Encoder: The original transformer (Vaswani et al., 2017) architecture includes an encoder with six identical stacked layers. Each layer includes two sub-layers, consisting of a multi-head self-attention mechanism, and a position-wise fully connected feed-forward network. Residual connections (He et al., 2016) surround each of the two sub-layers, followed by layer normalization (Ba, 2016).

Decoder: Similar to the encoder, the decoder also comprises six, identical (stacked) layers (as proposed in the original transformer architecture) (Vaswani et al., 2017). Each layer consists of three sub-layers, a multi-head self-attention mechanism, a position-wise fully connected feed-forward network (as in the encoder), as well as a third sub-layer with encoder-decoder attention. Akin to the encoder, the decoder employs residual connections around each of the sub-layers, followed by layer normalization. The decoder modifies the self-attention sub-layer by masking subsequent tokens. This ensures autoregressive behaviour, where token prediction is based on the previous decoded outputs.

Self-Attention

Traditional RNN (Rumelhart et al., 1986) architectures struggle to retain long-range dependencies given a long sequence of input, and so the self-attention mechanism in transformers (Vaswani et al., 2017) was introduced to address this challenge. More formally, self-attention calculates the weights of each token (embedding) in the sentence as it is related to every other token (embedding) in the sentence. This enables the model to predict the most likely tokens to be used in a sentence relative to the context in which it is found. During transformer training, self-attention is done in parallel. Consequently, the transformer model learns the grammar rules, leveraging statistical probabilities of how tokens are related and used in a given language. Mathematically, given a sequence of input tokens X_1, ..., X_n, its self-attention outputs a sequence of the same length, Y_1, ..., Y_n. **Equation 2.1** shows the scaled dot-product self-attention mechanism, where the three vectors represent the query (Q, current token), the keys (K, all other tokens), and the values (V, contextual information vectors). The attention score is computed by obtaining the dot product of the query Q with all keys K, scaling the result by the square root of d_k (representing the dimension of the key vectors), then normalizing the result by applying the softmax activation function, obtaining the attention weights. The attention weights are then used to obtain a weighted sum of the values, resulting in a context-aware representation for each query.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.1}$$

Positional Encoding

Self-attention by itself does not preserve sequence order (Vaswani et al., 2017). Positional encoding was introduced, conveying information about token position within a sequence. Unlike traditional RNN architectures, which maintain order by sequentially processing words, positional encoding enables parallel processing of tokens, while still preserving positional information. Positional encoding can either be learnt or fixed *a priori* (Gehring et al., 2017) and have the same dimension as the input token embeddings (X), enabling the two vectors to be summed.

Self-attention together with positional encoding enables transformers to learn the intricacies of language and understand the nuances of language form and structure. Section 2.3 below discusses how transformer-based language models are pre-trained to understand such language complexities.

2.3 Pre-Trained Language Models

Transformer-based pre-trained language models (PLMs) undergo initial pre-training on a vast, generic pre-training corpus using self-supervised learning methods (Devlin et al., 2018; Howard and Ruder, 2018; McCann et al., 2017; Peters et al., 2018). The LM retains extensive linguistic knowledge as it fits the distribution of the pre-training corpus. The LM is subsequently fine-tuned on a task-specific corpus, enabling task-specific inference (such as language translation or code translation). There are two prominent pre-training architectures, namely causal language modeling (CLM) and masked language modeling (MLM) (Micheletti et al., 2024). CLM predicts words sequentially in a unidirectional manner (from left to right), considering the prior text as context for a given token in the sequence. Decoder-only transformers that employ CLM include GPT (Achiam et al., 2023), PaLM (Chowdhery et al., 2023) and LLama (Touvron et al., 2023). However, this type of pre-training restricts learning to previous tokens only, not considering information preserved in the succeeding tokens of the sequence. MLM was developed to consider both preceding and succeeding tokens, enabling the prediction of a randomly masked segment of text within a bidirectional context. Models that employ MLM include BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2020).

2.3.1 Model Training

After model pre-training, the PLM is trained to perform task specific-inference (McCann et al., 2017). Three primary strategies have been introduced for PLM training, namely supervised fine-tuning (SFT) (Dong et al., 2024; Fan et al., 2023; Ohri and Kumar, 2024), zero-shot learning (Wang et al., 2023b) and in-context learning (ICL), also known as few-shot learning (Gao et al., 2024; Min et al., 2022; Pourreza and Rafiei, 2024). We focus on the dominant training strategies, namely, SFT and ICL. SFT involves training a language model (LM) on a large corpus of input-output pairs, where inputs include prompts or questions and outputs consist of answers (Dong et al., 2024; Fan et al., 2023; Ohri and Kumar, 2024). The effectiveness of SFT varies by task (Dong et al., 2024); for instance, tasks like code generation and mathematical reasoning benefit from larger training datasets, while more general tasks require fewer samples for similar performance. In addition, SFT faces challenges such as the risk of overfitting to a biased training dataset. Therefore, careful consideration must be given to data selection for model fine-tuning. In contrast to SFT, ICL does not need a large corpus of input-output training pairs. ICL requires a prompt template with a set of instructions to the LM on what task should be completed. The prompt template, constructed by means of prompt engineering, consists of three components: a context, an instruction, and an optional set of examples. Research indicates that providing examples enhances model performance in few-shot learning compared to zero-shot learning (Pourreza and Rafiei, 2024). Furthermore, strategies such as LLM role assignment (e.g., lawyer, mother, scientist, etc.) and positive prompt tone have shown improvements in ICL performance (Chen et al., 2024; Huang et al., 2024; Sun et al., 2024; Zheng et al., 2023). Despite these advancements, research suggests that text-to-SQL parsing trained using ICL still lag behind SFT (Pourreza and Rafiei, 2024). In Section 2.3.2, we introduce the T5 (Raffel et al., 2020) language model, an encoder-decoder Seq2Seq language model that is usually fine-tuned using SFT.

2.3.2 T5 Language Model

The T5 (Raffel et al., 2020) PLM has a Seq2Seq architecture that employs a bidirectional transformer-based encoder and decoder. T5 excels at numerous NLP tasks where the input and the output are both text. This text-to-text framework contrasts with BERT-style (Devlin et al., 2018) encoder-decoder LMs that generate either a single prediction for each input token or one prediction for an entire input sequence. For this reason, T5 models are more suited for generative tasks such as text-to-SQL parsing (Li et al., 2023b). The T5 architecture is similar to that of the original transformer architecture introduced by Vaswani et al. (2017), where the encoder and decoder each have identical stacked layers. Some modifications relative to the original transformer architecture include a simplified layer normalization placed outside of the residual path. In addition, T5 uses a relative position embedding scheme (Huang et al., 2019; Shaw et al., 2018) compared to the fixed embeddings used in the transformer architecture. T5 is pre-trained on the Colossal Clean Crawled Corpus (C4) (Raffel et al., 2020), including text and code scraped from the internet. This dataset is large, high in quality and diverse relative to its counterparts such as Wikipedia and Common Crawl web scrapes. Refer to Table A.1 in Appendix A for details on the T5 model series with varying parameter sizes. Take note that T5 has fewer model parameters relative to larger LMs such as GPT (Achiam et al., 2023). T5's smaller model size significantly lowers the computational cost required for fine-tuning.

2.4 Text-to-SQL Parsing

Text-to-SQL parsing (Zhong et al., 2017) is a semantic parsing task that is described as follows: Given a natural language question (NLQ) based on a relational database, generate a valid SQL query that matches both the user's intent and the database schema. **Equation 2.2** illustrates the inputs and outputs of the text-to-SQL parsing model (e.g., T5), where the parser interprets an NLQ and an optional schema to generate a valid SQL query. Existing text-to-SQL research mostly leverage neural text-to-SQL parsers such as the transformer architecture for direct NLQ to SQL conversion (Li et al., 2024; Li et al., 2023b; Wang et al., 2020; Xu et al., 2017; Yu et al., 2018c).

$$\text{SQL} = \text{Parser}(\text{NLQ}, \text{Schema}^{\text{optional}}) \tag{2.2}$$

In a manual setting, the user would need both SQL knowledge and schema familiarity in order to formulate valid SQL queries. By employing a text-to-SQL parsing model, the user can simply communicate with their database using natural language alone, without the need for database expertise (Yu et al., 2018c; Zhong et al., 2017). This motivation meets one of the goals of E. F. Codd who said: "If we are to satisfy the needs of casual users of databases we must break the barriers that presently prevent these users from freely employing their native language" (Brunner and Stockinger, 2021).

2.4.1 Evaluation

Following text-to-SQL translation, the text-to-SQL parsing model must be evaluated prior to its deployment in a real-world application. The body of literature uses various evaluation metrics to assess text-to-SQL model performance (Yu et al., 2018c). These metrics include the exact matching accuracy (EM), component matching accuracy (CM) and execution accuracy (EX), which we discuss below.

Exact Matching Accuracy

The exact matching accuracy (EM) (Yu et al., 2018c) measures whether the predicted SQL query as a whole exactly matches the ground-truth SQL query. EM is calculated using the formula in **Equation 2.3**. Here N is the number of predictions, Y_n is the n^{th} ground truth SQL query and $\hat{Y}_n$ is the n^{th} predicted SQL query. In this example, each Y_n maps to an associated $\hat{Y}_n$.

$$EM = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(Y_n, \hat{Y}_n) \tag{2.3}$$

Component Matching Accuracy

The component matching accuracy (CM) (Yu et al., 2018c) builds upon the exact matching accuracy, but it evaluates each SQL component separately. Here the components include SQL clauses such as the *SELECT*, *FROM* and *WHERE* clauses. For example, the SQL query, *SELECT column FROM table WHERE column = x* includes the following components:

1. SELECT clause: *SELECT column*
2. FROM clause: *FROM table*
3. WHERE clause: *WHERE column = x*

CM for the c^{th} component is calculated in **Equation 2.4**. Here N is the number of predictions, Y_{cn} is c^{th} component of the n^{th} ground truth SQL query and $\hat{Y}_{cn}$ is the c^{th} component of the n^{th} predicted SQL query. In this example, each Y_{cn} maps to an associated $\hat{Y}_{cn}$ and CM is computed for each component.

$$CM = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(Y_{cn}, \hat{Y}_{cn}) \quad \forall c \in C, \text{where C denotes the set of all SQL components.} \tag{2.4}$$

Execution Accuracy

The execution accuracy (EX) (Yu et al., 2018c) was proposed to address false negative evaluation as a result of the exact matching accuracy (EM). This relates to scenarios where the semantic parser predicts a correct

SQL query, which maintains the meaning of the natural language question, however the predicted SQL query differs from the ground-truth SQL query. EX is defined as the proportion of SQL queries in the evaluation set where the execution results of both predicted and ground-truth SQL queries are identical (Qin et al., 2022). EX is calculated in **Equation 2.5**, where N is the total number of predictions, V_n is the execution results of the n^{th} ground-truth SQL query (Y_n) and $\hat{V}_n$ is the execution results of the n^{th} predicted SQL query ($\hat{Y}_n$). In this example, each V_n maps to an associated $\hat{V}_n$.

$$EX = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}(V_n, \hat{V}_n) \tag{2.5}$$

2.5 General Domain Text-to-SQL Parsing

In this section, we discuss general domain text-to-SQL parsing models, benchmarks and challenges such as addressing unanswerable questions.

2.5.1 Models

Text-to-SQL models with encoder-decoder architectures may be categorized based on their encoder and decoder frameworks. Two primary categories exist for the encoding strategy, namely, text-to-SQL models which use: (i) LSTM-based encoders (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) and (ii) transformer-based encoders (Vaswani et al., 2017). For the decoder, there are three main categories, namely: (i) sketch-based decoders in slot-filling text-to-SQL methods (Dong and Lapata, 2018; Guo et al., 2019; Xu et al., 2017; Yavuz et al., 2018; Yu et al., 2018a, 2018b), (ii) sequence-based decoders in language generation methods (Wang et al., 2018; Zhang et al., 2019; Zhong et al., 2017), and (iii) grammar-based decoders that leverage SQL syntax in text-to-SQL generation (Choi et al., 2021; Dong and Lapata, 2018; Guo et al., 2019; Shi et al., 2018; Wang et al., 2019).

This section is partitioned into two, where we first introduce text-to-SQL models based on their encoding strategies, followed by their decoding strategies.

Encoder

The encoder first learns the representation of the input question and schema. We discuss two primary encoding strategies below, namely, LSTM-based and transformer-based encoders.

LSTM-based: LSTM-based encoders (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) learn contextualized representations of the input questions and, optionaly also, the schemas. Some LSTM-based encoders include TypeSQL (Yu et al., 2018a), Seq2SQL (Zhong et al., 2017) and SyntaxSQLNet (Yu et al., 2018b) that adopt the bidirectional LSTM (BiLSTM) to learn semantic representations of the input. Here the input is the concatenation of the input question and the column names. IRNet (Guo et al., 2019), on the other hand, encodes the question and schema using two separate BiLSTM encoders. Here the input of the two Bi-LSTM encoders are the token embeddings and the corresponding schema linking type embeddings, respectively. The schema linking type embeddings are obtained by n-gram string matching between the input question and schema, obtaining only relevant tables and columns for a given question.

Transformer-based: In recent works, transformer-based encoders (Vaswani et al., 2017) have demonstrated competitive performance in learning text representations, for a variety of NLP tasks. Text-to-SQL works that leverage transformer-based encoders include SQLova (Hwang et al., 2019) and SLSQL (Lei et al., 2020), which are extensions of BERT (Devlin et al., 2018). Similar to some LSTM-based encoders, transformer-based encoders can also simultaneously encode the question and the concatenated table and column names. In contrast to LSTM-based encoders, however, transformer-based encoders improve parsing accuracy by capturing long-range token dependencies within the input sequence.

Decoder

The decoder then receives the encoded representation of the question and the optional schema and decodes this representation into an SQL query. We discuss three main categories of decoders in text-to-SQL parsing, namely: sketch-based, sequence-based, and grammar-based decoders (Katsogiannis-Meimarakis and Koutrika, 2023;

Qin et al., 2022).

Sketch-based: Sketch-based decoding methods (Dong and Lapata, 2018; Guo et al., 2019; Xu et al., 2017; Yavuz et al., 2018; Yu et al., 2018a, 2018b) simplify the SQL generation task by predicting certain parts of the SQL query such as columns in the *SELECT* clause, tables in the *FROM* clause or values in the *WHERE* clause. In this method, a pre-defined SQL sketch (or template) with empty slots is provided and the sketch-based decoder fills in the empty slots. Sketch-based decoders ensure syntactically correct SQL queries due to these pre-defined SQL templates (Qin et al., 2022). However, the neural architecture may become overly complex because neural networks are required for each slot to be filled. In addition, sketch-based methods do not generalize well to complex SQL queries that differ from the pre-defined templates. Research that employs sketch-based decoders include SQLNet (Xu et al., 2017), SQLova (Hwang et al., 2019) and SDSQL (Hui et al., 2021).

Sequence-based: Sequence-based decoders (Cao et al., 2021; Guo et al., 2019; Huang et al., 2021; Wang et al., 2019) leverage sequence-to-sequence models to generate the entire (or a large part of the) SQL query. The decoder generates a sequence of tokens (comprising SQL keywords and schema items such as table names, column names and values). This enables the model to generate complex SQL queries, containing multiple tables and columns (Qin et al., 2022). However, there is less control over SQL generation. A challenge arises when the text-to-SQL translation results in an inexecutable or incorrect SQL query. This occurs when the translated SQL query contains typos, missing or incorrect keywords, columns, tables or values. Schema linking may improve question to schema mapping, while more advanced architectures such as transformer models, e.g., T5 (Raffel et al., 2020) may be used for the decoding step. Examples of language generation models that employ sequence-based decoders include Seq2SQL (Zhong et al., 2017).

Grammar-based: Grammar-based decoders (Choi et al., 2021; Dong and Lapata, 2018; Guo et al., 2019; Shi et al., 2018; Wang et al., 2019) are an extension of sequence-based approaches, that leverage grammar rules when decoding their output. The grammar rules (e.g., SQL syntax) govern the structure of the generated output. Common grammar-based decoders are usually LSTM-based. The grammar-based decoder first generates a structured intermediate query, e.g., an Abstract Syntax Tree (AST) (Yin and Neubig, 2017, 2018), before generating the final SQL query. This makes grammar-based decoders less prone to errors. However, advancements in sequence-based decoding strategies are challenging grammar-based decoders with methods in place to reduce erroneous predictions (e.g., answer abstention) (Lee et al., 2022).

The text-to-SQL models mentioned in this section need a text-to-SQL parsing dataset for model training and evaluation. Section 2.5.2 below introduces widely used benchmarks in general domain text-to-SQL parsing.

2.5.2 Benchmarks

Text-to-SQL datasets (otherwise known as benchmarks) include a set of natural language question (NLQ) and SQL pairs, including the database schemas (Zhong et al., 2017). Early text-to-SQL models were trained on small datasets. This restricted models to shallow training, since deep neural models require large and diverse samples. Later on, large-scale, cross-domain text-to-SQL parsing benchmarks were introduced. There have also been efforts to curate single-domain text-to-SQL benchmarks. Cross-domain benchmarks are curated for domain generalization, where the text-to-SQL model performs well across multiple domains. Single-domain text-to-SQL parsing benchmarks, however, prioritize real-life, complex NLQ/SQL representation in a single domain relative to simple NLQ/SQL pairs across multiple domains. In addition, research by Hazoom et al. (2021), suggests that most cross-domain text-to-SQL models do not generalize well to complex, single-domain datasets. This limitation of cross-domain text-to-SQL parsing emphasizes the need for real-world representative, domain-specific and complex benchmarks for improved text-to-SQL parsing. Widely used cross-domain benchmarks in the literature include WikiSQL (Zhong et al., 2017), Spider (Yu et al., 2018c), KaggleDBQA (Lee et al., 2021) and BIRD (Li et al., 2023b). Spider variants also exist namely, Spider-DK (Gan et al., 2021b), SpiderSyn (Gan et al., 2021a), and Dr Spider (Chang et al., 2023). Examples of domain-specific benchmarks include: ATIS for flight-booking (Dahl et al., 1994; Price, 1990), GeoQuery for US geography (Zelle and Mooney, 1996), and SEDE for Stack Exchange user-related questions (Hazoom et al., 2021). See **Table 2.1** for further details. Both cross-domain and single-domain benchmarks in general-domain text-to-SQL parsing, have an inherent limitation: the absence of unanswerable questions (Lee et al., 2022). Unanswerable questions are defined as questions that the text-to-SQL model cannot reliably answer. As a result, most text-to-SQL models are only trained and evaluated on answerable questions—those that the model can confidently

answer. Real-life text-to-SQL models, however, encounter both answerable and unanswerable questions. For this reason, the text-to-SQL benchmark must include both answerable and unanswerable questions to test model robustness in a real-life setting.

TABLE 2.1: Comparison of Text-to-SQL Parsing Benchmarks. The *Queries* column refers to the number of question and SQL pairs in a dataset. The *DBs* column represents the number of databases in a dataset. The *Tables/DB* column represents the average number of tables per database. The *Rows/Table* column is the average number of rows per table. The *Nesting* column denotes the average number of nesting levels per SQL query. The *UnANS* column indicates whether the dataset contains unanswerable questions or not. A '-' value represents unreported values.

Dataset	Year	Domain	Queries	DBs	Tables/DB	Rows/Table	Nesting	UnANS
ATIS[1–5]	1994	Flight-booking	275	1	25	-	1.39	No
GeoQuery[4–7]	1996	US Geography	525	1	7	-	2.03	No
WikiSQL[4,5,8,9]	2017	Wikipedia	80,654	26,521	1	-	1	No
Spider[5,9]	2018	Cross-domain	10,181	200	5.1	-	1.15	No
MIMICSQL[10]	2020	Medical	10,000	1	5	7,000	1.0	No
KaggleDBQA[3,4]	2021	Kaggle	272	8	2.25	280,000	1.0	No
SEDE[4,5,10,11]	2021	Stack Exchange	12,023	1	29	-	1.28	No
emrKBQA[12]	2021	Medical	940,000	1	9	-	-	No
EHRSQL[13]	2022	Medical	24,400	2	13.5	108,000	2.7	Yes
BIRD[9]	2023	Cross-domain	12,751	95	7.3	-	-	No

[1] Price, 1990 [2] Dahl et al., 1994 [3] Lee et al., 2021 [4] Katsogiannis-Meimarakis and Koutrika, 2023 [5] Finegan-Dollak et al., 2018 [6] Zelle and Mooney, 1996 [7]Lee et al., 2021 [8]Zhong et al., 2017 [9]Li et al., 2023b [10]Wang et al., 2020 [11]Hazoom et al., 2021 [12]Raghavan et al., 2021 [13]Lee et al., 2022

2.5.3 Unanswerable Questions

Unanswerable questions in text-to-SQL parsing are usually referred to as out-of-distribution or infeasible (Lee et al., 2024). However, past works also define unanswerable questions as having terminologies that are out-of-schema, out-of-domain, out-of-scope, requiring domain or world knowledge, containing domain jargon, ambiguous, vague, out-of-category, out-of-vocabulary and problematic (Chang et al., 2023; Lee et al., 2024; Wang et al., 2023a; Yu et al., 2018c). On the other hand, Gan et al. (2021b) refer to unanswerable questions as question paraphrases that contain unknown phrases to the model. Unanswerable questions are out-of-distribution because the model has not encountered such question types during model training. As a result, the text-to-SQL model will generate an incorrect or inexecutable SQL query given the unanswerable question. Text-to-SQL parsing benchmarks must include unanswerable questions to evaluate model robustness against unanswerable questions and develop strategies to address them. Generally, the proportion of unanswerable questions vary across datasets. A survey by Wang et al. (2023a) suggest that 20% of all users' questions are unanswerable[1], while Lee et al. (2022) suggest that 33% of users' questions are unanswerable. Regardless of the proportion of unanswerable questions, the text-to-SQL model should address both answerable and unanswerable questions to reflect real-life scenarios.

In this section, we define unanswerable questions by grouping them into three primary categories, namely out-of-schema questions, unanswerable questions requiring domain knowledge and ambiguous questions.

Out-of-Schema Questions

Recall that the text-to-SQL user has not seen the database schema. Out-of-schema questions are ascribed to the user's conceptual framework of the database, where the user refers to ideas, concepts and terms in the natural language question (NLQ) that do not exist as tables, columns or values in the existing database schema (Katsogiannis-Meimarakis and Koutrika, 2023; Lee et al., 2024; Qin et al., 2022). Refer to **Table 2.2** for examples of out-of-schema questions. Previous research by Zhang et al. (2020b) and Wang et al. (2023a), simulate out-of-schema scenarios by deleting and perturbing existing columns in the schema. Related work by Lee et al. (2024), create hypothetical columns. For example, Lee et al. (2024) refer to a hypothetical column as a column that is conceptually tied to the existing table but does not exist in the table (e.g., a hypothetical *duration* column in the *song* table). Other examples include hypothetical columns that relate to other columns from other databases

[1]Wang et al. (2023a) refer to unanswerable questions as problematic

(e.g., a hypothetical *product* category column in the *battles* database). Notably, most of these works focus on column perturbations and not table, value or entire schema perturbations.

TABLE 2.2: Examples of Out-of-Schema Questions in General Domain Text-to-SQL Datasets. Examples obtained from Price (1990). The red font refers to out-of-schema phrases. The blue font refers to related concepts (e.g., out-of-schema values expected to occur in a specific column)

Category	Example
Table Unanswerable	• **Question:** Show me all information related to hospital A. • **Tables:** Hospital B, Hospital C • **Note:** The phrase hospital A is out-of-schema because there is no table named hospital A in the database schema.
Column Unanswerable	• **Question:** Show me the model name by sales. • **Columns:** Brand, Sales, Year • **Note:** The phrase model name is out-of-schema because there is no column named model name in the database schema.
Value Unanswerable	• **Question:** Count the total of private hospitals. • **Columns:** NHHospitalCategory, State, Year, BenefitsPaid • **Note:** The phrase private hospitals is out-of-schema because there is no such value in the NHHospitalCategory column.

Unanswerable Questions Requiring Domain Knowledge

Unanswerable questions requiring domain knowledge, usually contain complex, domain jargon that refer to tables, columns and values in the existing schema (Lee et al., 2021; Lee et al., 2022; Li et al., 2023b; Suhr et al., 2020; Yu et al., 2018c). However, the text-to-SQL model cannot fully comprehend the complex domain jargon, limiting its ability to accurately map these terms to the existing schema. These questions require a domain expert to simplify the domain jargon found in the question. By simplifying the domain jargon, the text-to-SQL model can fully understand the user's intent (Lee et al., 2024, 2022). These questions are integral as they reflect those posed by domain experts in real-world scenarios. Gan et al. (2021a) simulate this scenario in the SpiderSyn benchmark by generating questions with synonyms for table and column names. This is known as synonym substitution. Research by Gan et al. (2021a) suggest that top-performing text-to-SQL models trained on the Spider dataset (Yu et al., 2018c) experience a significant decline in performance, ranging from 20% to 30%, when evaluated on the SpiderSyn dataset. Chang et al. (2023) refer to synonym-substitution as value-synonym replacement, replacing schema values in the question with their synonym form. Chang et al. (2023) find that value-synonym replacement is the most challenging perturbation for NLQs, where the most robust text-to-SQL model trained on Spider, has a 26.9% performance drop when values in the question are replaced with its synonym variant. Gan et al. (2021b), on the other hand, generate the Spider-DK dataset with question paraphrases that have unseen phrases relative to the training data. These questions require more complex reasoning to construct a valid SQL query. Similar to SpiderSyn, high-performing models trained on Spider experience a 20% to 30% performance drop when evaluated on Spider-DK. See **Table 2.3** for examples of unanswerable questions requiring domain knowledge.

TABLE 2.3: Examples of Unanswerable Questions Requiring Domain Knowledge in General Domain Text-to-SQL Datasets. Examples from Li et al. (2023b). The red font refers to phrases containing complex domain jargon that require domain knowledge to simplify. The blue font refers to the domain knowledge.

Examples
Question: Can you list the account IDs of those who choose the weekly issuance statement? **Domain Knowledge:** POPLATEK TYDNE stands for weekly issuance. **SQL:** SELECT account_id FROM account WHERE account.frequency = 'POPLATEK TYDNE';
Question: How many accounts are eligible for loans in New York City? **Domain Knowledge:** The condition of loans is that the type of the account should be OWNER. **SQL:** SELECT COUNT(*) FROM account WHERE account.type = 'OWNER' AND city = 'NY';

Ambiguous Questions

Natural language is inherently ambiguous, where questions can have multiple interpretations (Katsogiannis-Meimarakis and Koutrika, 2023; Qin et al., 2022). There are several categories of ambiguity such as lexical-, syntactic- and semantic-ambiguity, to name a few (Katsogiannis-Meimarakis and Koutrika, 2023). Lexical ambiguity (or polysemy) refers to words having multiple meanings. Syntactic ambiguity refers to an utterance having multiple interpretations based on its syntactic structure. Semantic ambiguity refers to a statement with multiple semantic interpretations. If a question is ambiguous because it does not contain the complete context needed for a system to fully understand it, the system may need to infer the implicit information based on the given context. Other times, additional context is required to disambiguate the question. In the context of text-to-SQL systems, ambiguous questions occur when the user refers to tables, columns or values without precise specification (Lee et al., 2024). As a result, the table, column or value can be mapped to multiple similar tables, columns or values, respectively (see **Table 2.4**). This results in an incorrect SQL query that does not meet the user's intent.

2.5.4 Resolving Unanswerable Questions with Knowledge Augmentation

This Section introduces various strategies to address unanswerable questions, namely out-of-schema questions, unanswerable questions requiring domain knowledge and ambiguous questions. Out-of-schema questions are addressed with schema augmentation. Unanswerable questions requiring domain knowledge are addressed by leveraging external knowledge sources. Ambiguous questions are addressed with additional user knowledge.

Out-of-Schema Questions

The literature proposes schema augmentation (Lee et al., 2022; Zhao et al., 2022) as a solution to addressing out-of-schema questions. Two primary methods are proposed, either providing the entire unseen schema to the text-to-SQL model or providing the model with the pruned unseen schema, including candidate schema items most relevant to the question.

Entire Schema: Recent studies suggest that providing the model with all table and column names from a schema reduces the risk of omitting important tables and columns—a common issue in schema pruning (Cao et al., 2024; Maamari et al., 2024; Yang et al., 2024). This schema augmentation strategy can be done at inference or both training and inference. Related work by Lee et al. (2022) use sequence-to-sequence models, T5-Base (Raffel et al., 2020) and T5-Base with schema serialization (Hemphill et al., 1990; Suhr et al., 2020), to compare model performance with the addition of a serialized schema. Schema serialization appends the entire schema, including all tables and columns in the form *Table name: column names, etc.*, to the end of the natural language question (NLQ).

TABLE 2.4: Examples of Ambiguous Unanswerable Questions in General Domain Text-to-SQL Datasets. Examples from Wang et al. (2023a). The red font refers to ambiguous phrases. The blue font refers to schema tables, columns or values to which the ambiguous phrase refers to.

Category	Example
Column Ambiguity	• **Question:** Can you show me the top rating movie? • **Columns:** Movie, IMDB Rating, Rotten Tomatoes Rating, Content Rating • **Note:** The token rating in the question is ambiguous because there are 3 column names containing rating. The model does not know which of these columns the ambiguous phrase is referring to.
Value ambiguity	• **Question:** For Jack, can you show me the date of license issued and license expired? • **Columns:** Engineer, Constructor, License issued, License expires • **Values:** Engineer column: Jack, Constructor column: Jack • **Note:** The token Jack in the question is ambiguous because both columns Engineer and Constructor have the value Jack. The model does not know which of these columns the ambiguous phrase is referring to.

Schema pruning: Other approaches explore schema pruning (otherwise known as schema linking), which aims to prune the schema to only the most relevant schema items (tables, columns and values) to a given NLQ (Katsogiannis-Meimarakis and Koutrika, 2023; Li et al., 2023a, 2024). Similar to schema serialization, the pruned schema is then appended to the end of the NLQ. Two approaches exist for schema pruning, either string matching-based (Brunner and Stockinger, 2021; Dou et al., 2022; Guo et al., 2019; Wang et al., 2019) or neural-based approaches (Bogin et al., 2019; Gu et al., 2023; Li et al., 2023a). Neural-based approaches work well when matching synonyms between NLQs and schema items. However, this approach may not always be practical for cases where complex domain jargon is used within either the NLQ or the schema. This is because existing language models (LM) undergo pre-training to understand basic language and not complex domain jargon. Li et al. (2024) leverage schema pruning to identify candidate column values for a given NLQ. First, a coarse search is employed, after which a fine-grained search is applied. The coarse search utilizes the BM25 algorithm (Robertson, Zaragoza, et al., 2009). This method retrieves hundreds of potential column value candidates, after which the longest-common substring (LCS) algorithm is employed to calculate the matching degrees between an NLQ and the coarsely filtered column values. This approach is ideal for text-to-SQL benchmarks with many database values. Providing all database values to the model is impractical due to its maximum context length and the risk of information loss. Notably, Li et al. (2024) only evaluate this approach for column values and not table names and column names. Li et al. (2023a) and Li et al. (2024) leverage a neural-based architecture for determining candidate tables and columns relevant to an NLQ. In this approach, a cross-encoder ranks the relevance of tables and columns to a given NLQ. It does this by calculating the embedding similarity between table names and the NLQ tokens, as well as between column names and the NLQ tokens. The resultant table and column names are ranked based on their probabilities between 0 and 1, where higher probabilities suggest higher relevance to the given NLQ.

Unanswerable Questions Requiring Domain Knowledge

Li et al., 2023b emphasize that text-to-SQL models need external knowledge for improved model understanding of the user's question. Li et al. (2023b) differentiate among four categories of external knowledge, namely domain knowledge (Dou et al., 2023; Zhao et al., 2022), synonym knowledge (Gan et al., 2021a), numeric reasoning knowledge and value illustration. Value illustration provides details about column values such as value types and categories. Li et al. (2023b) show that T5-Base achieves a +5.22% increase in execution accuracy (EX) and T5-Large achieves a +10.04% increase in EX when evaluated on the BIRD validation questions with external knowledge relative to the questions without external knowledge. Similar results are obtained for the test set, where T5-Base achieves a +5.83% increase in EX and T5-Large has a +10.56% increase in EX when evaluated on questions with external knowledge relative to questions without external knowledge. Despite this notable increase in performance, many text-to-SQL parsing datasets do not include external knowledge.

Ambiguous Questions

Text-to-SQL models may disambiguate an ambiguous question by using one of two methods, either inferring the missing information from the question itself (elliptical queries) or asking the user additional questions for context clarification (Katsogiannis-Meimarakis and Koutrika, 2023; Lee et al., 2024). For the former, elliptical queries use the existing context to disambiguate the question. For example, in the question *Who was the president before Trump?*, the model should infer that the user is referring to US presidents. For the latter, ambiguous questions are resolved through multi-turn user interactions, where the text-to-SQL model leverages the answers obtained from follow-up questions in reinforcement learning with human feedback (RLHF). Research by Arthur et al. (2015) paraphrase ambiguous questions using a language model, and leverage the paraphrase to disambiguate the generated meaning representation. This work is inspired by McKeown (1983) who paraphrase a question, and verify the correctness of the paraphrase by a user before the question is answered. Wang et al. (2023a) formulates this problem as a sequence labeling task. The model must identify the ambiguity in the question before asking the user a follow-up question. In this approach, a weakly supervised model, namely Detecting-Then-Explaining (DTE), classifies each word in a user's question as ambiguous or not. In this manner, the user knows precisely where the ambiguity lies.

2.5.5 Answer Abstention

For cases where unanswerable questions cannot be resolved with additional knowledge, the text-to-SQL model must abstain from answering the question. This greatly improves text-to-SQL reliability (Lee et al., 2024). Answer abstention requires the text-to-SQL model to first identify these unanswerable questions, before refusing to answer them. Existing unanswerable question detection methods are categorised into five primary categories: (i) heuristic-based methods, (ii) learning-based methods, (iii) distance-based methods, (iv) prompting-based methods and (v) uncertainty estimation methods (Lee et al., 2024; Wang et al., 2023a).

Heuristic-based: Heuristic-based methods use rules to detect and locate unanswerable phrases within a given question (Li et al., 2020; Sorokin and Gurevych, 2018; Wu et al., 2020). However, these methods are limited in that they require humans to do heavy feature engineering.

Learning-based: In contrast, learning-based methods rely on model training to identify unanswerable questions. Zhang et al. (2020b) leverage a neural model, namely, RoBERTa (Liu et al., 2019; Tan et al., 2022), to distinguish among four unanswerable question types. Related work by Zeng et al. (2020) view this challenge as a sequence labeling problem, by training a span index predictor to detect the location of unanswerable phrases within a question. This approach is limited, however, in that it only finds one unanswerable span per question. Wang et al. (2023a) then proposed a weakly supervised classification model to identify multiple unanswerable phrases within a question. This method leverages a BERT-based (Devlin et al., 2018) encoder and three task modules for unanswerable question detection, localization and explainability.

Distance-based: Distance-based methods classify questions as answerable or unanswerable based on the distribution of the training data. These methods compute metrics such as the Mahalanobis distance to compare unseen questions with questions in the training data (Lee et al., 2018; Ren et al., 2021, 2022; Zheng et al., 2020).

Prompting-based: Prompting-based strategies leverage few-shot demonstrations, by including examples of answerable and unanswerable questions and their corresponding SQL queries in the LLM prompt. Each example is labeled as answerable or unanswerable (Sun et al., 2021). The LLM is then prompted to identify unanswerable and answerable questions at inference. A stringent approach considers five iterations of LLM prompting. The SQL query is only considered answerable if all five outputs are answerable, otherwise the question is classified as unanswerable.

Uncertainty estimation: Uncertainty estimation methods leverage neural networks for out-of-domain detection (Lee et al., 2024, 2022; Liu et al., 2020). Two main categories of uncertainty estimation methods have been used in question answering research, namely, entropy-based and probability-based methods. Lee et al. (2024) finds that entropy-based and probability-based methods consistently outperform heuristic-based, learning-based, distance-based, and prompting-based answer abstention methods. Entropy-based approaches assign the maximum entropy of the generated SQL tokens to the predicted SQL query (Lee et al., 2022). Probability-based methods assign the lowest value of the highest token probabilities among the generated SQL tokens (Stengel-Eskin and Van Durme, 2023). This value is then compared to a pre-defined threshold. The threshold or decision boundary is calibrated to distinguish unanswerable questions from answerable questions (Lee et al., 2024, 2022). In this manner, the task is formulated as a question classification problem. A question is classified as unanswerable, if the decoded value (either obtained from entropy-based or probability-based methods) exceed the pre-defined threshold, otherwise it is classified as answerable.

2.6 Medical Text-to-SQL Parsing

Electronic health records (EHRs) often employ simple search and filter functions on the front end that are converted to executable SQL queries at the back end (Lee et al., 2022; Wang et al., 2020). These systems are cumbersome to query. For example, if a physician wants to know the number of patients with diabetes who are under the age of 40, they would need to create separate filters for age and disease, adding complexity to the query formulation process. Other approaches require medical professionals to manually formulate SQL queries for direct EHR querying. However, these systems were found to be overly complex and technical for non-expert users. Recent research on medical question answering (QA) has drawn inspiration from general domain text-to-SQL parsing (Bae et al., 2024; Bardhan et al., 2022; Lee et al., 2022; Wang et al., 2020). However, a problem arises when the text-to-SQL model encounters unanswerable questions (previously introduced in Section 2.5.3). Medical facilities, including hospitals, are high-risk environments where mistakes and missing

data can result in poor decision-making. This may lead to incorrect diagnoses or prescriptions, which can have detrimental implications (Lee et al., 2024). For this reason, medical text-to-SQL systems should be consistent and reliable. For consistency, knowledge augmentation improves the scope of questions answered, whereas, for reliability, answer abstention is put in place.

This section introduces various medical question-answering benchmarks, medical text-to-SQL models and re-introduces unanswerable questions in the context of medical question-answering. Finally, we also identify and discuss medical knowledge sources for resolving medical unanswerable questions, while also exploring answer abstention for cases where the unanswerable question cannot be resolved with medical knowledge.

2.6.1 Benchmarks

Text-to-SQL systems have demonstrated success across various domains, supported by numerous text-to-SQL parsing benchmarks. In contrast, medical text-to-SQL research and the availability of medical question-answering (QA) datasets are limited (Lee et al., 2022). Similar to single-domain QA datasets in general text-to-SQL parsing (Hazoom et al., 2021), medical-specific QA datasets are tailored to reflect real-life, medical questions, complex SQL queries and databases not present in cross-domain benchmarks such as Spider (Yu et al., 2018c) and WikiSQL (Zhong et al., 2017).

In this section, we introduce widely used medical QA datasets and discuss the limitations associated with each of them.

emrQA: The emrQA (Pampari et al., 2018) benchmark was introduced as the first accessible patient-specific QA dataset. The emrQA dataset includes questions, logical forms and answers based on physicians' clinical notes. However, emrQA is restricted to physicians' frequently asked questions, resulting in a lack of question diversity.

emrKBQA: The emrKBQA (Raghavan et al., 2021) QA dataset was then adapted from emrQA to expand the scope of medical questions beyond clinical notes (see **Table 2.1**). This dataset is based on MIMIC-III[2] (Johnson et al., 2016), a large and freely available EHR comprising de-identified patient data. However, emrKBQA is also limited in that it contains questions mostly targeting outpatient test results. In addition, emrKBQA includes questions related to only 1 database.

MIMICSQL: The MIMICSQL (Wang et al., 2020) dataset was introduced as a large-scale healthcare-specific text-to-SQL parsing dataset that is also based on MIMIC-III (Johnson et al., 2016). A limitation of MIMICSQL, however, is that it is non-representative of real-life hospital settings (Lee et al., 2022). MIMICSQL contains a limited scope of questions that are associated with simple, non-nested SQL queries. In addition, the questions are restricted to only 5 tables from one database (Lee et al., 2022) (see **Table 2.1**).

EHRSQL: More recently, EHRSQL (Lee et al., 2022) was introduced to address the limitations across medical QA datasets (see **Table 2.1**). EHRSQL contains realistic questions collected from a university hospital, complex SQL queries and two EHR sources, namely MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018). Experimental results indicate that in contrast to MIMICSQL (Wang et al., 2020), which is compatible with the grammar used in a cross-domain text-to-SQL parsing dataset (i.e., Spider) (Yu et al., 2018c), EHRSQL presents a more challenging scenario. When using the Spider-trained model, EHRSQL only has 7% (107 out of 1515) parsable queries among the set of answerable validation questions. This reinforces the need for text-to-SQL models trained on real-world, representative datasets tailored for specific domains, such as healthcare. Another advantage of EHRSQL is that it is the first medical text-to-SQL dataset to incorporate unanswerable questions. However, these unanswerable questions are mostly addressed with answer abstention and not resolved with knowledge augmentation as suggested by Li et al. (2023b).

2.6.2 Models

Most medical question-answering (QA) datasets are trained using sequence-to-sequence (Seq2Seq) (Sutskever et al., 2014) models. These models usually have an encoder-decoder architecture, comprising either: (i) recurrent neural networks (RNNs) such as long short-term memory units (LSTMs) or (ii) transformers. In this

[2]https://physionet.org/content/mimiciii/1.4/

section we introduce medical text-to-SQL parsing models, with a focus on sequence-to-sequence-based architectures. This section is divided into two, where each subsection relates to the encoder-decoder architecture, either LSTM-based or transformer-based.

LSTM-based

In this section, we discuss medical text-to-SQL translation models that leverage sequence-to-sequence (Sutskever et al., 2014) architectures, particularly those based on LSTM encoder-decoder networks. These models are used to train the medical QA datasets discussed in Section 2.6.1.

emrQA: Pampari et al. (2018) use a sequence-to-sequence model with Bahdanau attention when training on the emrQA dataset. The model translates the given question into a logical form. Similar to the original sequence-to-sequence architecture, both the encoder and decoder comprises LSTMs. This approach is compared to heuristic-based models, which perform biomedical concept extraction using Cliner (Boag et al., 2018). Biomedical concept extraction involves extracting biomedical-related concepts from text. The biomedical concepts are then transferred to pre-defined logical form templates using a slot-filling method. The sequence-to-sequence models, however, prove to be superior compared to the heuristic-based method.

emrKBQA: The emrKBQA dataset expands the emrQA dataset. Each question in the emrKBQA (Raghavan et al., 2021) dataset has a set of associated paraphrases. In one experiment, Raghavan et al. (2021) ensure that the paraphrases do not overlap between train and test splits, ensuring the model generalizes to different question forms. A sequence-to-sequence model, with an LSTM-based encoder and decoder, then translates the question or paraphrase into a logical form. Both the question and its paraphrases map to the same logical form. In a deterministic approach, the logical form is mapped to a pre-defined SQL template by computing string similarity between a logical form and SQL templates. The medical entities are then extracted from the logical form and slot-filled into the SQL template.

MIMICSQL: Wang et al. (2020) leverage a deep learning-based translate-edit model for question-to-SQL (TREQS) generation, trained on the MIMICSQL dataset. Unlike Pampari et al. (2018) and Raghavan et al. (2021) which translate emrQA and emrKBQA questions using slot-filling methods, TREQS leverages a language-generation translation method. TREQS adapts a sequence-to-sequence model to directly translate a question into an SQL query. The encoder comprises a single-layer bidirectional LSTM (Hochreiter and Schmidhuber, 1997), and the decoder includes a single-layer unidirectional LSTM. The sequence-based decoder uses the current input token and prior context to generate the next token in the SQL sequence. After decoding, the model performs SQL post-processing for schema linking and typo editing. Here schema linking involves matching question tokens to the most relevant columns, tables and values in the database.

Transformer-based

Lee et al. (2022) fine-tune a sequence-to-sequence model with a transformer-based encoder-decoder architecture on the EHRSQL dataset. These models include T5-Base (Raffel et al., 2020) and T5-Base with schema serialization (Hazoom et al., 2021; Suhr et al., 2020). Recent research suggests that transformer-based Seq2Seq language models, such as T5, surpass existing medical text-to-SQL models such as TREQS (Wang et al., 2020), which have LSTM-based Seq2Seq architectures. In addition, Lee et al. (2022) find that grammar-based decoders are only compatible with cross-domain datasets such as Spider (Yu et al., 2018c). These decoders do not perform well on medical datasets such as EHRSQL. For this reason, Lee et al. (2022) motivate the use of a sequence-based decoder, and answer abstention mechanisms to reduce errors generated by the sequence-based decoder.

2.6.3 Unanswerable Questions

Similar to general domain text-to-SQL benchmarks, most medical QA datasets do not include unanswerable questions (Bae et al., 2024; Pampari et al., 2018; Raghavan et al., 2021). As a result, unanswerable questions are not properly addressed at inference. This becomes a problem, especially in high-risk settings such as hospitals where missing or incorrect information can affect decision-making, resulting in detrimental implications. This section re-introduces unanswerable questions in the context of medical text-to-SQL parsing. We discuss three primary categories of unanswerable questions, namely, out-of-schema questions, unanswerable questions requiring medical knowledge and ambiguous questions.

Out-of-Schema Questions

Lee et al. (2022) refer to out-of-schema medical questions as questions that are beyond the database schema. The EHRSQL dataset mainly categorizes patient-specific questions as out-of-schema questions. However, the classification of out-of-schema questions is not always clear and distinct from unanswerable questions requiring medical knowledge. For this reason, unanswerable medical questions can be classified based on how they are resolved. Consider **Equation 2.6**, which models a valid SQL query given an unanswerable question (NLQ^{unans}). The predicted SQL query is incorrect because the model needs additional knowledge to understand the out-of-schema concept in the question.

$$Parser(NLQ^{unans}) = \text{ Incorrect SQL} \tag{2.6}$$

A question is out-of-schema if the unanswerable NLQ can be resolved with an additional schema. This means that the model is able to correctly map between the NLQ and the new schema. **Equation 2.7** models a correct SQL query given an unanswerable NLQ (NLQ^{unans}) and an additional schema ($schema^{+}$).

$$Parser(NLQ^{unans}, schema^{+}) = \text{ Correct SQL} \tag{2.7}$$

However, if the unanswerable NLQ is only resolved with additional medical knowledge, assuming a matching schema, then the unanswerable question is classified as requiring medical knowledge. **Equation 2.8** models a correct SQL query given an unanswerable NLQ (NLQ^{unans}), a matching schema ($schema^{+}$) and additional medical knowledge (K).

$$Parser(NLQ^{unans}, schema^{+}, K) = \text{ Correct SQL} \tag{2.8}$$

However, these classification scenarios have not been thoroughly investigated. Nonetheless, **Table 2.5** shows out-of-schema questions and scenarios.

TABLE 2.5: Examples of Out-of-Schema Questions in a Medical Text-to-SQL Dataset (Lee et al., 2022). The red font refers to out-of-schema phrases.

Category	Example
Diagnoses	• **Question:** Is there a drug that could be prescribed to treat bilat bb block nec? • **Note:** The phrase prescribed to treat is out-of-schema because the schema does not include a table with diagnoses to medication mappings (i.e., a treatment table).
Medication	• **Question:** What is the impact of long term use of furosemide? • **Note:** The phrase impact of long term use is out-of-schema because the schema does not include a medication side effects table.
Procedures	• **Question:** what are the precautions after the spinal canal explor nec procedure? • **Note:** The phrase precautions is out-of-schema because the database does not include a table with procedure precautions.

Unanswerable Questions Requiring Medical Knowledge

Lee et al. (2022) describe unanswerable questions requiring medical knowledge, in the context of medical text-to-SQL parsing, as medical questions that contain complex medical expressions. These expressions contain complex, medical jargon that require extensive medical knowledge to simplify and relate to the existing schema. This problem is modeled in **Equation 2.8**, by predicting a valid SQL query given an unanswerable NLQ (NLQ^{unans}), a matching schema ($schema^{+}$) and additional medical knowledge (K). **Table 2.6** shows examples of unanswerable questions requiring medical knowledge (Lee et al., 2022).

TABLE 2.6: Examples of Unanswerable Questions Requiring Medical Knowledge in a Medical Text-to-SQL Dataset (Lee et al., 2022). The red font refers to phrases with complex, medical jargon.

Category	Example
Diagnoses	• **Question:** Is there a drug that could be prescribed to treat valvular stenosis? • **Note:** The phrase valvular stenosis requires medical knowledge to simplify because its synonymous form exists in the diagnoses column.
Medication	• **Question:** What are the side effects of long term prednisolone acetate 0.12% ophth. susp. treatment? • **Note:** The phrase prednisolone acetate 0.12% ophth. susp. requires medical knowledge to simplify because its synonymous form exists in the medication column.
Procedures	• **Question:** What are the precautions after the systemic antibiotics - vancomycin procedure? • **Note:** The phrase systemic antibiotics - vancomycin procedure requires medical knowledge to simplify because its synonymous form exists in the medical procedures column.

Ambiguous Questions

Lee et al., 2022 do not address ambiguous questions, as these questions require additional user clarification. These questions generally refer to vague terms that relate to diagnoses, medication and medical procedures but do not specify the context to which the question is referring to. **Table 2.7** provides examples of ambiguous questions as defined by Lee et al. (2022).

TABLE 2.7: Examples of Ambiguous Unanswerable Questions in a Medical Text-to-SQL Dataset (Lee et al., 2022). The red font refers to ambiguous phrases.

Category	Example
Diagnoses	• **Question:** I am curious what the protocols for the drugs that work to treat cancer? • **Note:** The phrase cancer is ambiguous because the cancer type is not specified.
Medication	• **Question:** what is the reimbursement standard for drug? • **Note:** The phrase drug is ambiguous because the drug is not specified.
Procedures	• **Question:** What is the checklist you need to follow in advance of your drug levels procedure? • **Note:** The phrase drug levels procedure is ambiguous because the drug levels procedure is not specified.

2.6.4 Answer Abstention

Most medical text-to-SQL research only trains and evaluates their models on answerable questions, thus inflating results (Lee et al., 2022). These studies usually do not evaluate model robustness on unanswerable questions. Lee et al. (2022), on the other hand, proposes abstaining from answering unanswerable questions. They compare the maximum entropy values of the model's decoded output to a pre-defined threshold. Questions are classified as unanswerable when the maximum entropy of the decoded output exceeds the pre-defined threshold (Lee et al., 2022). Moreover, Lee et al. (2022) defines two strategies to determine the threshold, namely a clustering and a percentile-based approach. The clustering approach leverages the K-means algorithm (MacQueen, 2018) ($K = 2$ clusters) on the maximum entropy values of the SQL predictions in the validation set. The boundary between the two clusters is used as the threshold. The percentile-based approach sets the threshold to the n^{th} percentile of the maximum entropy values among SQL predictions in the validation set. Here n is computed as $n = 100 - u$, where u is the proportion of unanswerable questions in the validation set. For both these approaches, the unanswerable questions are fixed. This means that the percentage of unanswerable questions is always the same, because they are not resolved with additional knowledge. For this reason, the percentile-based approach works well because the n^{th} percentile of unanswerable questions is always the same. The K-means clustering method, however, identifies patterns in unanswerable questions and can be applied in more general contexts where the proportion of unanswerable questions change.

2.6.5 Resolving Unanswerable Questions Requiring Medical Knowledge

External knowledge has not been extensively used in resolving unanswerable questions within medical question-answering research. However, in this section, we identify and discuss medical knowledge sources that could be used to resolve unanswerable questions requiring medical knowledge. We discuss two medical knowledge sources, namely the Unified Medical Language System (UMLS) (Bodenreider, 2004) and the Side Effect Resource (SIDER) (Kuhn et al., 2016).

Unified Medical Language System

The Unified Medical Language System (UMLS) (Bodenreider, 2004) is the largest compendium of medical vocabularies developed by the United States (US) National Library of Medicine (NLM). The UMLS comprises three knowledge sources, namely the metathesaurus, the semantic network and the specialist lexicon. The

metathesaurus includes all medical vocabularies, hierarchies, and definitions of medical concepts. The semantic network groups medical concepts belonging to the same semantic type. Lastly, the specialist lexicon provides a general English lexicon with lexical information necessary for processing medical terms using the specialist NLP system. The UMLS is regularly updated, with quarterly updates each year. The 2023AA metathesaurus contains approximately 3.31 million concepts and 15.7 million distinct concept names from 185 medical vocabularies. Medical vocabularies integrated into the UMLS metathesaurus include the Systematized Nomenclature of Medicine–Clinical Terminology (SNOMED CT) (Donnelly et al., 2006), the US Edition, International Classification of Diseases and Related Health Problems, Tenth Revision (ICD-10) (Hirsch et al., 2016), and the Medical Subject Headings (MeSH) vocabulary (Lipscomb, 2000), to name a few. The medical terminologies for Python tool (PyMedTermino[3]) is a Python module that enables access to many of these medical vocabularies (also known as medical terminologies when accessed using PyMedTermino). These medical vocabularies can be used for biomedical concept extraction (BCE). The QuickUMLS[4] is a fast, unsupervised tool for BCE, that leverages medical vocabularies found in the UMLS, to map text, e.g., *unable to sleep*, to standardized medical concepts, e.g., *insomnia*. It employs word sense disambiguation to determine the correct meaning of terms in the given context. As a result, each normalized medical concept obtains a Concept Unique Identifier (CUI). The tool also enables users to constrain the search space by specifying semantic categories, using Type Unique Identifiers (TUIs). Below we describe commonly used medical vocabularies used in BCE.

SNOMED CT (Donnelly et al., 2006) is one of the most comprehensive and precise medical vocabularies that is adopted in health systems in over 50 countries. SNOMED CT is easy to use because medical concepts are organized in a hierarchical manner. SNOMED CT is commonly used in medical research (Chang and Mostafa, 2021; Lee et al., 2013, 2014), for tasks such as obtaining medical synonyms.

ICD-10-CM (Hirsch et al., 2016) is a standardized system that is used to code and classify medical diagnoses. ICD-10-CM is an extension of ICD-10, a system used to code and classify causes of death on death certificates. Clinical modification was needed to better describe patient diagnoses in ICD-10-CM. The codes are more precise than ICD-10. The *ICD-10-PCS* vocabulary, on the other hand, is also adapted from ICD-10 but is used to code and classify medical procedures.

MeSH (Lipscomb, 2000) is a medical vocabulary maintained by the NLM. MeSH is used for indexing articles on PubMed. MeSH includes information such as article headings, article descriptions, and medical definitions. MeSH comprises over 30,000 entries and is updated each year to incorporate changes in medical terminologies.

The *MedDra* vocabulary (Brown et al., 1999) is an international, clinically validated vocabulary used by regulatory authorities and the biopharmaceutical industry in the regulatory phase, clinical research and pharmacovigilance. MedDra includes information regarding a medication's safety information and adverse effects.

The *RxNorm* vocabulary (Nelson et al., 2011) provides normalized medication names, grouping all brand names of a medication to the same medical concept. RxNorm has been used in many pharmacy management and drug interaction software, such as First Databank, Micromedex, Multum, and the Gold Standard Drug Database, to name a few.

Side Effect Resource

The side effect resource (SIDER) (Kuhn et al., 2016) aims to aggregate information on medication, such as side effects and targets. The SIDER 4 database contains data on 1430 drugs, 5880 adverse drug reactions (ADRs) and 140,064 drug–ADR pairs. SIDER uses the MedDra (Brown et al., 1999) medical vocabulary to extract side effects from medication labels.

[3]https://pythonhosted.org/PyMedTermino/tuto_en.html
[4]https://github.com/Georgetown-IR-Lab/QuickUMLS

2.7 Summary

In this chapter, we reviewed the transformer architecture, which retains long-range dependencies in an input sequence. We then further discussed semantic parsing in the context of text-to-SQL translation. We highlighted several sequence-to-sequence neural models leveraged in general-domain and medical text-to-SQL translation research. We also discussed various text-to-SQL parsing benchmarks in general-domain and medical research. Finally, we introduced the concept of unanswerable questions, which includes out-of-schema questions, unanswerable questions requiring domain (or medical) knowledge and ambiguous questions. We discussed methods of addressing them, either through knowledge augmentation or answer abstention. Various knowledge augmentation methods exist, such as providing additional schema information to the text-to-SQL model to resolve out-of-schema questions, providing external domain (or medical) knowledge to resolve questions requiring domain (or medical) knowledge and additional user clarification to resolve ambiguous questions. We also identified and reviewed various medical knowledge sources that can resolve medical unanswerable questions, namely the UMLS and SIDER. With this comprehensive literature review in mind, the information serves to facilitate understanding of the methods discussed in Chapter 3 and Chapter 4. Chapter 3 provides information on the data sources, exploratory data analyses, synthetic data generation, data splitting and data augmentation. In Chapter 4 we discuss how this data is used for model fine-tuning and evaluation.

Chapter 3

Data Generation Methods

3.1 Overview

In Chapter 2, we review the transformer architecture, particularly the sequence-to-sequence encoder-decoder architecture, such as T5, and its application in text-to-SQL parsing. We also discuss current approaches of addressing unanswerable questions in text-to-SQL research, including knowledge augmentation and answer abstention. This chapter discusses the data generation steps, including EHR augmentation and synthetic unanswerable question generation. First, we perform exploratory data analysis (EDA) on the unanswerable categories, namely, out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*) to identify important unanswerable question subcategories and types. We then augment the EHR grounded with information to answer these unanswerable questions. Subsequently, we synthesize additional unanswerable questions to improve the unanswerable question representation in the dataset. Finally, we discuss the data splitting of questions into the training, validation, and test sets. **Figure 3.1** illustrates the data generation methods followed in this chapter, including the experimental methods in Chapter 4. In this chapter, Section 3.2 introduces the data sources. Section 3.3 discusses how EDA is conducted. Section 3.4 discusses the experimental setup. Section 3.5 discusses EHR augmentation. Section 3.6 discusses how we synthetically generate unanswerable questions and ground-truth SQL queries. Section 3.7 outlines the data splitting into training, validation, and test sets.

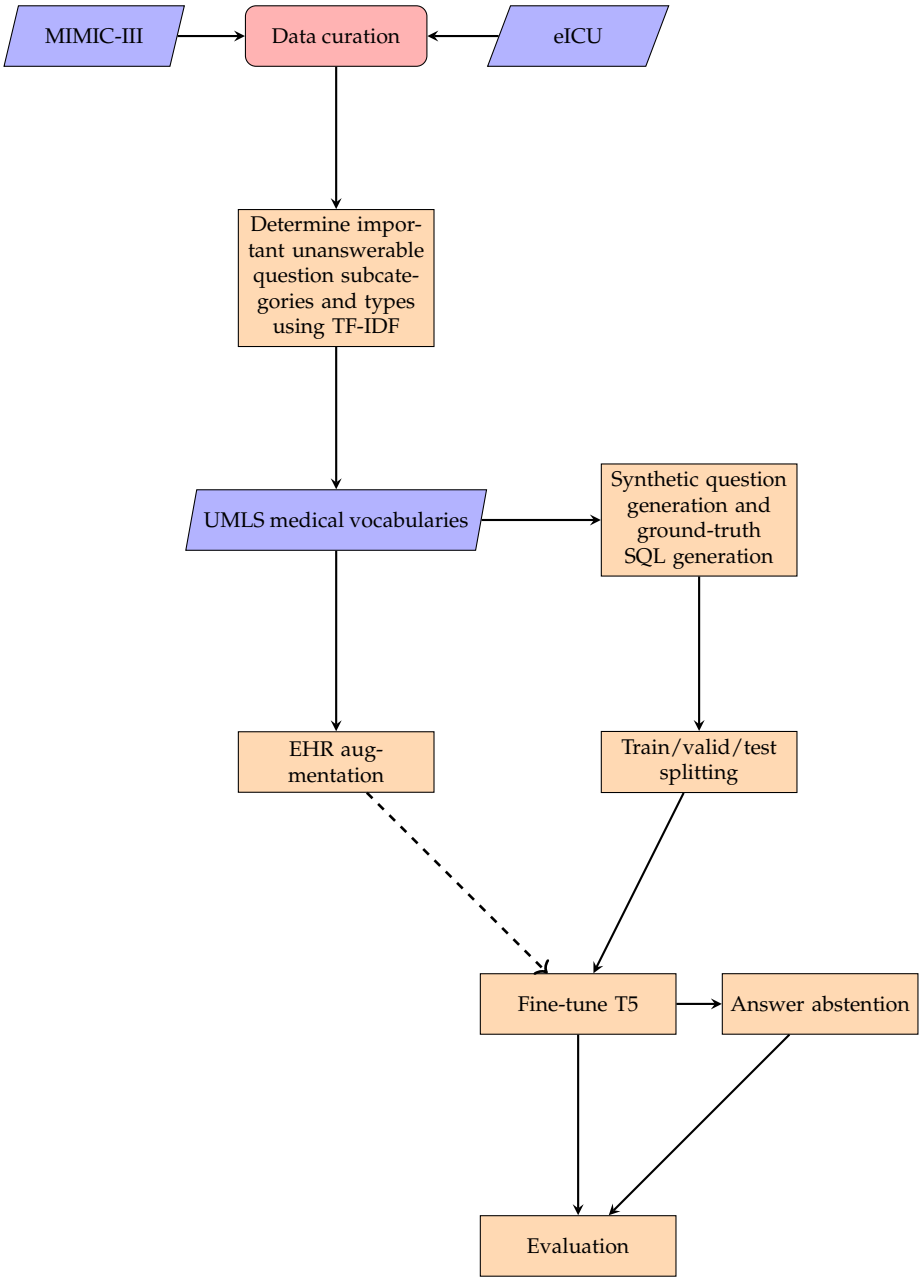


FIGURE 3.1: Schematic Representation of the Methods.

This flow-diagram shows the methods to follow in Chapter 3 and Chapter 4. The pink rectangle shows the start of the methods to follow. The purple parallelograms represent the data inputs. The orange squares and rectangles represent the steps in the methods to follow. Solid lines with arrows illustrate the flow of information from one step to another. Dotted lines illustrate optional inputs.

3.2 Data

This thesis uses the EHRSQL (Lee et al., 2022) text-to-SQL parsing benchmark for text-to-SQL fine-tuning. EHRSQL is a multi-source dataset, including natural language question (NLQ) and SQL pairs defined over the MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018) electronic health record (EHR) databases. The MIMIC-III and eICU EHRs can be accessed upon credentialed request on PhysioNet (Moody, 2022). EHRSQL was chosen as the dataset for this research thesis due to its patient anonymity. In addition, it contains realistic medical queries, including unanswerable questions and complex, SQL queries. See the sample sizes of EHRSQL in **Table 3.1**.

TABLE 3.1: EHRSQL Question Partitions. The row names show the question source and the column names show the data split. The first row of values for a given question source represents the total questions, the second row represents the count of unanswerable questions and the third row represents the count of answerable questions.

Question Source	Train	Valid	Test
MIMIC III	**9318**	**1122**	**1786**
	0	362	588
	9318	760	1198
eICU	**9270**	**1114**	**1791**
	0	362	588
	9720	752	1203

The EHRSQL dataset includes 13.5 tables on average, and 24.4K high-quality, realistic questions collected from 222 hospital staff at the Konyang South Korean University hospital, including 1.9K unanswerable questions, 22.5K answerable questions, and complex SQL queries with a 2.7 average nesting per SQL query (see **Table 3.2**). This is in contrast to other medical question-answering datasets, e.g., MIMICSQL (Wang et al., 2020), emrQA (Pampari et al., 2018), and emrKBQA (Raghavan et al., 2021), that either contain a few tables in only one database, a limited scope of questions (excluding unanswerable questions), or simple SQL queries with no nesting. The EHRSQL dataset defines 3 categories of unanswerable questions, namely: (i) out-of-schema questions, (ii) unanswerable questions requiring medical knowledge and (iii) ambiguous questions, of which we focus on categories (i) and (ii). Ambiguous questions are not addressed in this research thesis because these questions require multi-turn user interactions to disambiguate (Lee et al., 2024, 2022). This is out of the scope of this research thesis.

TABLE 3.2: Comparison of Medical Text-to-SQL Parsing Benchmarks. The *Queries* column refers to the number of question and SQL pairs in the dataset. The *DBs* column represents the number of databases in a dataset. The *Tables/DB* column represents the average number of tables per database. The *Rows/Table* column is the average number of rows per table. The *Nesting* column denotes the average number of nesting levels per SQL query. The *UnANS* column indicates whether the dataset contains unanswerable questions or not. A '-' value represents unreported values.

Dataset	Year	Queries	DBs	Tables/DB	Rows/Table	Nesting	UnANS
EHRSQL [1]	2022	24,400	2	13.5	108,000	2.7	Yes
MIMICSQL [2]	2020	10,000	1	5	7000	1.0	No
emrKBQA [3]	2021	940,000	1	9	-	-	No

[1] Lee et al., 2022
[2] Wang et al., 2020
[3] Raghavan et al., 2021

3.3 Exploratory Data Analysis

This research thesis aims to address two primary categories of unanswerable questions, namely out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*) (see Section 1.5). In this section, we employ exploratory data analysis (EDA) to identify important unanswerable question subcategories and fine-grained unanswerable question types in the EHRSQL dataset (Lee et al., 2022). Identifying important unanswerable question subcategories and types enables us to properly represent these unanswerable question subcategories and types in the training and evaluation sets. It is important to effectively represent these questions in the datasets, as the goal is to address real-life unanswerable questions. Since unanswerable questions are only available in the validation and test sets (see **Table 3.1**), we conduct EDA only on the validation unanswerable questions. This strategy ensures that the test set remains unseen. To identify the most important unanswerable question subcategories and unanswerable question types, we first determine the important n-grams among the unanswerable validation questions. We do this by computing the term-frequency inverse-document frequency (TF-IDF) scores (Das, Alphonse, et al., 2023; Gomes et al., 2023; Paulsen et al., 2023) of n-grams in the unanswerable validation questions. Important n-grams represent the important unanswerable question subcategories and types. These unanswerable question subcategories and types are used in downstream tasks such as EHR augmentation (see Section 3.5) and synthetic question and ground-truth SQL generation (see Section 3.6) that are later leveraged in text-to-SQL fine-tuning (see Chapter 4). See Section B.1 for the detailed EDA steps. We also considered using latent Dirichlet allocation (LDA) topic modelling (Blei et al., 2009) over TF-IDF. However, we find that TF-IDF provides a more granular analysis of medical concepts. **Figure B.1** in Section B.1.5 shows the word cloud of the top-200 most important medical concepts among the validation, unanswerable questions, while **Table 3.3** shows the top-10 n-grams and their TF-IDF scores. The full list of medical concepts and associated TF-IDF scores can be seen in **Table B.1** of Section B.1.6 in Appendix B. From these results, it is clear that most unanswerable questions either belong to diagnoses, medication or medical procedure unanswerable question subcategories. These three unanswerable question subcategories are further sub-categorized into unanswerable question types using the TF-IDF results. However, we focus on the diagnoses and medication unanswerable question subcategories to adhere to the thesis aims (see Section 1.4). The unanswerable question types belonging to the diagnoses subcategory include: ICD-10-CM diagnoses codes, diagnoses-to-medication treatment mappings and diagnoses-to-medication prevention mappings (see **Table B.2** in Section B.1.7 of Appendix B). The medication subcategory includes unanswerable question types such as medication side-effects, tradenames, medication-to-diagnoses treatment mappings, and medication-to-diagnoses prevention mappings (see **Table B.3** in Section B.1.7 of Appendix B).

TABLE 3.3: Top 10 Medical Concepts Ranked According to TF-IDF Scores.

Concept	TF-IDF
side	11.27
effect	11.19
influenza	10.17
side effect	9.71
blood	9.64
procedure	9.63
prescription	9.33
prescribed	9.17
please	9.15
code	9.08

3.4 Experimental Setup

We use Python 3.10[1] for all methods in Chapter 3. In Section 3.3 we identify unanswerable question subcategories (related to diagnoses and medication) and fine-grained unanswerable question types based on the unanswerable validation questions associated with the MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018) EHRs. More specifically, EDA in Section 3.3 assists us in identifying what information is required to augment the EHR so that important unanswerable questions can be answered. Refer to Section 3.5 for details on EHR augmentation.

In addition, we reduce question bias and improve unanswerable question representation in the training and evaluation datasets by synthetically generating the unanswerable questions identified in EDA. We generate out-of-schema, unanswerable questions for *RQ1* by synthetically generating reference questions (see Section 3.6.1) and their paraphrases (see Section 3.6.2). We then generate unanswerable questions requiring medical knowledge for *RQ2* by performing value-synonym replacement of medical jargon in the synthetic reference questions (see Section 3.6.3). **Figure 3.2** shows the unanswerable question hierarchy. Level 1 shows the primary unanswerable categories, namely unanswerable questions that are out-of-schema (*RQ1*) or require medical knowledge (*RQ2*). Level 2 shows the unanswerable groups, where out-of-schema questions comprise the synthetic reference questions and paraphrases, while the unanswerable questions that require medical knowledge comprise the value-synonym replacement questions. Level 3 shows the unanswerable subcategories, where each unanswerable group comprises questions related to diagnoses and medication. Level 4 shows the unanswerable question types, where the diagnoses subcategory comprises unanswerable questions related to ICD-10-CM codes, diagnoses-to-medication treatment mappings, and diagnoses-to-medication prevention mappings. The medication subcategory comprises unanswerable questions related to medication side effects, tradenames, medication-to-diagnoses treatment mappings and medication-to-diagnoses prevention mappings.

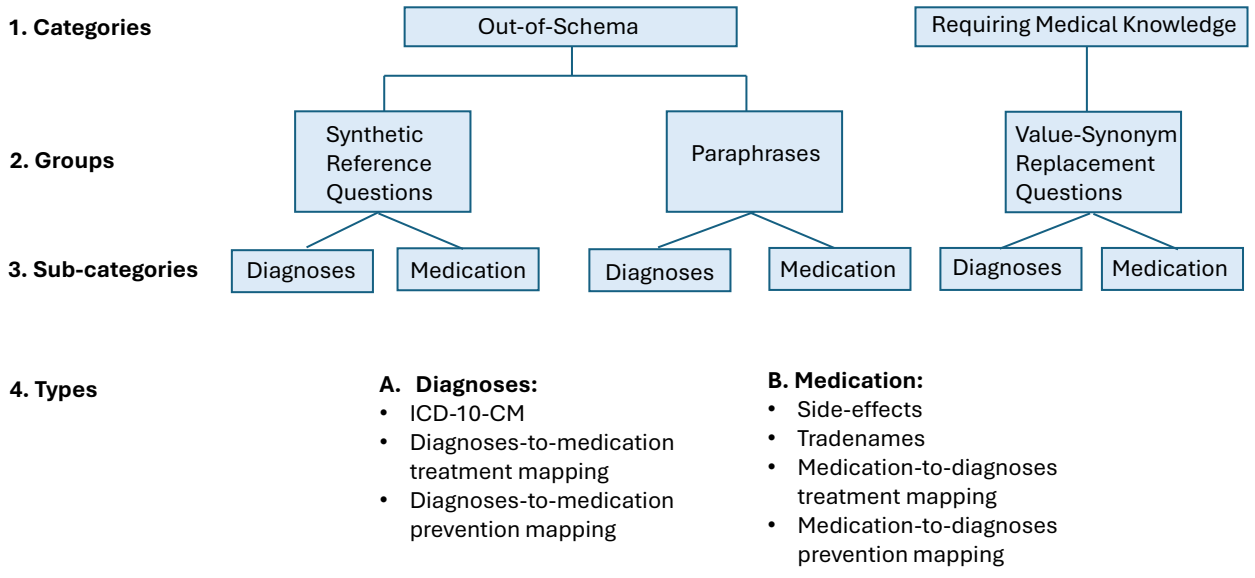


FIGURE 3.2: Unanswerable Question Hierarchy.

[1]https://www.python.org/downloads/release/python-3100/

3.5 EHR Augmentation

The existing EHRs, MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018), do not answer all questions. The knowledge gap in the existing EHRs result in an inexecutable or incorrect SQL query when the text-to-SQL model encounters an unanswerable question. Consequently, medical professionals cannot retrieve valuable information that informs their decision-making when they pose an unanswerable question to the text-to-SQL model. As a solution to this knowledge gap, we follow a similar approach to Lee et al. (2024), Pi et al. (2022), Wang et al. (2023a), and Zhang et al. (2020b). However, instead of adding or s columns of an existing table, we create an entirely new EHR database schema with curated medical knowledge. This approach creates a new EHR resource that facilitates reproducible research. The new EHR database aims to answer unanswerable questions that are out-of-schema (*RQ1*) or unanswerable questions requiring medical knowledge (*RQ2*). We curate medical knowledge from the UMLS (Bodenreider, 2004) and SIDER (Kuhn et al., 2016) knowledge sources to populate the new EHR. We are interested in populating the new EHR with information related to diagnoses and medication, as identified as the most important, unanswerable question subcategories in Section 3.3. We first perform biomedical concept extraction (BCE) to identify all medical concepts related to diagnoses and medication in the MIMIC-III and eICU EHRs. We then normalize the medical concepts. Finally, we search for information related to the normalized medical concepts by identifying relevant UMLS medical vocabularies that house this information.

This section describes the EHR augmentation task, with the following steps in place: (i) schema creation (see Section 3.5.1), (ii) biomedical concept extraction (BCE) and concept normalization (see Section 3.5.2), (iii) vocabulary mapping (see Section 3.5.3), and (iv) database augmentation (see Section 3.5.4).

3.5.1 Schema Creation

We use the sqlite3[2] module in Python[3] to create an additional EHR schema. This schema is built to house the additional medical information intended to answer unanswerable questions that are out-of-schema or require medical knowledge. More specifically, the EHR is intended to answer unanswerable question subcategories and types identified in Section 3.3. We create a new EHR schema with 6 tables shown in **Figure B.2** (see Appendix B). Take note that we have a table for medical procedures, ICD10PCS_PROCEDURE, however, we do not perform downstream tasks related to medical procedures. This is because we were unable to identify sufficient medical vocabularies to fully address unanswerable questions related to medical procedures. However, we include this table as an additional resource for future work. In agreement with our research aims, we leverage the tables related to diagnoses and medication in downstream tasks.

3.5.2 Biomedical Concept Extraction and Concept Normalization

We obtain diagnosis and medication descriptions from the MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018) EHRs. This ensures that we are representing a real-life distribution of medical concepts. Refer to **Table B.4** for examples of diagnoses and **Table B.5** for examples of medication, both sourced from the MIMIC-III and eICU EHRs. We then perform biomedical concept extraction (BCE) and normalization of medical concepts related to diagnoses and medication descriptions. This is done by querying the medical concept (known as the query term) against the UMLS (Bodenreider, 2004). We use the UMLS because it is the largest compendium of medical vocabularies developed by the United States (US) National Library of Medicine (NLM). We query the UMLS by leveraging the QuickUMLS[4] tool in Python. The QuickUMLS is dependent on the UMLS metathesaurus, for which we use the 2023AA metathesuarus throughout this research thesis for consistency. The 2023AA metathesaurus was chosen because it was the most recent version available when commencing the thesis. To constrain the search space to diagnoses and medication names, we identify and specify the semantic types for the QuickUMLS, denoted by their type unique identifiers (TUIs). **Table B.6** (in Section B.2.3 of Appendix B) contains the TUIs we identified for this research task. As a result, QuickUMLS returns the normalized medical concept along with the concept unique identifier (CUI) for each medical concept queried against the UMLS. Refer to Section 2.6.5 for more information on the UMLS and related concepts. See Section B.2.4 in Appendix B for further details on how the QuickUMLS was used in this research task.

[2]https://docs.python.org/3/library/sqlite3.html
[3]https://www.python.org/
[4]https://github.com/Georgetown-IR-Lab/QuickUMLS

3.5.3 Vocabulary Mapping

We then leverage the PyMedTermino[5] module to access UMLS vocabularies in Python. The following UMLS vocabularies are considered: ICD-10-CM (Hirsch et al., 2016), MeSH (Lipscomb, 2000), MedDra (Brown et al., 1999) and RxNorm (Nelson et al., 2011). These vocabularies are trusted within the medical domain, and are actively utilized in medical research, clinical practice, and the development of medical software. In addition, they are regularly updated and include sufficient information on diagnoses and medication. Refer to Section 2.6.5 for more details on these medical vocabularies. The CUIs obtained in Section 3.5.2, are used in vocabulary mapping to map the CUI to their unique identifier in a relevant medical vocabulary. This mapping enables us to obtain the vocabulary identifier. The vocabulary identifier facilitates vocabulary navigation and retrieval of information linked to the medical concept associated with it. We utilize the ICD-10-CM (Hirsch et al., 2016) vocabulary to map diagnosis CUI codes to ICD-10-CM codes, while also employing RxNorm (Nelson et al., 2011) to map CUI codes to medication identifiers, which assists us in identifying tradenames and associated treatable and preventable diagnoses. In addition, we use the MeSH (Lipscomb, 2000) vocabulary to map medical concept CUI codes to MeSH identifiers, enabling us to retrieve medical definitions for synonym retrieval tasks (see Section 5.3.2). In addition, we utilize the SIDER (Kuhn et al., 2016) database to identify medication side effects by mapping medication names from SIDER to those in the RxNorm outputs, retaining only the lowest level term (LLT) labels for side effects. The resultant mapping of this research task is all medication names, their CUIs and a set of side effects.

3.5.4 Database Augmentation

Finally, we leverage sqlite3 in Python to populate the new EHR schema (see **Figure B.2**) with diagnosis and medication information. This is done by executing *INSERT* statements against the schema. In Section 3.6, we then synthetically generate unanswerable questions that can be answered with the information in this new EHR schema.

3.6 Synthetic Question and Ground-Truth SQL Generation

In Section 3.3, we identify unanswerable question subcategories (i.e., diagnoses and medication) and unanswerable question types related to the primary unanswerable categories, namely out-of-schema questions (*RQ1*) and questions requiring medical knowledge (*RQ2*). In this section, our objective is to generate unanswerable questions related to these unanswerable subcategories and types identified in a real-world medical QA dataset. By synthetically generating these unanswerable questions, we improve the representation of unanswerable questions in the training and evaluation question sets. We generate reference questions and paraphrases for the out-of-schema questions (*RQ1*) and we generate unanswerable questions with value-synonym replacements for the unanswerable questions requiring medical knowledge (*RQ2*). We perform four tasks, namely: (i) reference question generation from base questions in the EHRSQL QA dataset (see Section 3.6.1), (ii) question paraphrasing (see Section 3.6.2), (iii) value-synonym replacement of medical concepts in the reference questions (see Section 3.6.3) and (iv) ground-truth SQL template generation (see Section 3.6.4). We repeat these sub-tasks for each unanswerable question subcategory and their unanswerable question types.

3.6.1 Reference Question Generation

Synthetic reference questions are generated for each unanswerable question subcategory and type. In this section, we discuss the prompt engineering strategy for reference question generation, the large language model (LLM) used, the LLM hyperparameters, and the post-processing steps and quality checks for these questions.

Prompt Engineering

We prompt a Large Language Model (LLM) using in-context Learning (ICL) (Min et al., 2022), enabling the LLM to perform a new task by following the instruction and imitating the input and output example pairs in the prompt. This approach is more feasible than a supervised fine-tuning (SFT) approach which requires training a large corpus of input and output examples. **Figure B.3** illustrates the general prompt template used for synthetic reference question generation. The prompt includes a context, an instruction and a set of

[5]https://owlready2.readthedocs.io/en/latest/pymedtermino2.html

examples. The context in the prompt provides the background that the LLM is a doctor's assistant who is speaking to a doctor about a certain medical concept. We assign the LLM the role of doctor's assistant because research indicates that assigning roles to an LLM enhances model performance (Chen et al., 2024; Huang et al., 2024; Zheng et al., 2023). The prompt instruction states that the assistant must recall the doctor's words and seek help from the nurse. We record the assistant's question posed to the nurse as the synthetic reference question. To enable the assistant to perform this task, we provide 2 unique, examples that are randomly sampled for each unanswerable question type. The examples include an input and an output, where the input is a medical concept, either a diagnosis or medication name and the output is a question containing the medical concept. See **Table B.7** and **Table B.8** in Section B.4.2 of Appendix B for the prompt examples. We choose a few-shot learning approach (d = 2 examples) as opposed to a zero-shot learning approach due to evidence that few-shot learning boosts model performance over zero-shot learning (Pourreza et al., 2024; Pourreza and Rafiei, 2024). Finally, we include a positive tone in the prompt, concluding the prompt with *..take a deep breath....* This is aligned with recent research that suggests positive tone in the prompt improves model performance relative to neutral and negative tones (Sun et al., 2024).

Large Language Model

We use the Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) LLM for this research task because it is an open-source LLM that is instruction fine-tuned (i.e., follows prompt instructions), and at the time of writing surpassed the Llama 2 13B – chat (Touvron et al., 2023) model on human and automated benchmarks. This is significant because Llama 2 13B – chat has more parameters relative to Mistral-7B-Instruct-v0.2, yet Mistral-7B-Instruct-v0.2 still outperforms Llama 2 13B – chat. Smaller LLMs with good performance are ideal because they require less computational resources. We utilize the Mistral-7B-Instruct-v0.2 model in Python through the Hugging Face library (Jain, 2022) to synthetically generate reference questions across all unanswerable question types. Mistral-7B-Instruct-v0.2 is an instruct fine-tuned version of Mistral-7B-v0.2. Refer to Section B.3 in Appendix B for further details.

Quality Checks

Quality checks are done to obtain only high-quality synthetic reference questions. A high-quality question is characterized as one that closely resembles the prompt examples (otherwise known as the demonstrations) provided in the LLM prompt, while still remaining distinct. Similar to Li et al. (2024), we use a contrastive learning approach to filter out low-quality synthetic reference questions. Refer to Section B.4.3 in Appendix B for a detailed description of the quality checks. See **Table 3.4** for the filtering statistics of this task.

TABLE 3.4: Synthetic Reference Question Filtering. The *High Quality* column represents the sample size after filtering. The *Low Quality* column represents the sample size of the discarded samples. The *Total* column represents the total size of the samples before filtering, including high-quality and low-quality samples. The grey headings show the unanswerable subcategory, namely, (a) diagnoses and (b) medication.

Unanswerable Question Type	High Quality	Low Quality	Total
a. Diagnoses			
ICD-10-CM	1337	80	1417
Diagnoses-to-medication treatment mapping	797	72	869
Diagnoses-to-medication prevention mapping	755	114	869
b. Medication			
Side-effects	145	14	159
Tradenames	1372	257	1629
Medication-to-diagnoses treatment mapping	1468	161	1629
Medication-to-diagnoses prevention mapping	1182	447	1629

3.6.2 Paraphrasing

This task aims to paraphrase synthetic reference questions to assess if the text-to-SQL model generalizes well to unseen question structures and variations in lexical terms. Paraphrases, together with the synthetic reference questions form part of the out-of-schema questions (*RQ1*). In this section, we prompt an LLM to paraphrase the synthetic reference questions generated in Section 3.6.1. In addition, we also up-sample the minority question class, i.e., medication side-effects which is severely underrepresented in the dataset (n = 145 samples).

Paraphrasing

All unanswerable question types, except for those related to medication side effects, undergo a paraphrasing step. To ensure consistency, paraphrasing is performed using the same language model (LLM) and hyper-parameters as the synthetic reference questions. Following Witteveen and Andrews (2019) and Wahle et al. (2022), we adopt LLM-based paraphrasing, supported by research indicating that human annotators have rated LLM paraphrases comparable relative to their reference questions. This is in contrast to template-based paraphrasing approaches that simply slot-fill predefined paraphrase templates (Zhang et al., 2023). Template-based approaches require considerable manual effort to generate a template for each reference sentence and also restricts the diversity of paraphrases to these pre-defined templates. We aim to generate 500 high-quality paraphrases for each unanswerable question type due to computational constraints. The LLM is prompted given the context that it should behave as a doctor's assistant. The instruction is to paraphrase the synthetic reference question, given an example of an input-output pair. Only those paraphrases with a cosine similarity score between 0.6 and 0.9 relative to their reference questions are retained, ensuring they maintain meaning while avoiding excessive similarity. In addition, any duplicates or irrelevant outputs are filtered out. See prompt template in **Figure B.4**. In **Table 3.5** we provide the mean similarity of high-quality paraphrases relative to their synthetic reference questions and in **Table 3.6** we provide examples of the generated paraphrases.

TABLE 3.5: LLM Paraphrasing Statistics. The column *Average similarity* shows the average cosine similarity of a paraphrase relative to its synthetic reference question among the pruned paraphrases, where n = 500 paraphrases.

Unanswerable Question Type	Average similarity
ICD10-CM	0.860
Tradenames	0.845
Medication-to-diagnoses treatment mapping	0.883
Diagnoses-to-medication treatment mapping	0.882
Medication-to-diagnoses prevention mapping	0.840
Diagnoses-to-medication prevention mapping	0.877

TABLE 3.6: Examples of Generated Paraphrases.

Synthetic Reference Question	Paraphrase
Could you please inform me about the medication that's used to treat aortic coarctation?	What is the medication utilized for treating aortic coarctation?
Can you tell me the commercial brand names of the medication Pantoprazole sodium?	Which brands offer Pantoprazole sodium medication?
What is the ICD-10 code for vitreomacular adhesion?	Please find me the specific ICD-10 code that corresponds to vitreomacular adhesion diagnosis.

Up-sample Minority Question Class

Using the LLM discussed in Section 3.6.1, we perform targeted up-sampling of the minority class i.e., questions related to medication side effects (n = 145 questions). We randomly sample with replacement, medical concepts (i.e., medication names) and their associated synthetic reference questions from the minority class. We sample with replacement, rather than without replacement because the minority class (i.e., medication side effects) only has a few samples. Using the same paraphrasing and filtering strategy as above, we generate an average of 7 paraphrases per synthetic reference question. This is similar to Raghavan et al. (2021), which generates 7.5 paraphrases per question. As a result, we obtain a total of n = 1007 paraphrases.

3.6.3 Value-Synonym Replacement

In this task, we synthetically generate unanswerable questions requiring medical knowledge (refer to *RQ2* in Section 1.5). This is done by applying value-synonym replacement (Chang et al., 2023) to the synthetic reference questions generated in Section 3.6.1. See detailed methods in Section B.6. Value-synonym replacement, in this context, introduces linguistic richness by replacing medical jargon in each synthetic reference question with appropriate medical synonyms. This is in contrast to Gan et al. (2021a), which replaces common nouns and verbs. Value-synonym replacement is necessary to evaluate the text-to-SQL model's robustness in handling synonym substitutions of medical jargon within natural language questions (NLQs), where the synonym does not explicitly match values in the database columns (Chang et al., 2023; Gan et al., 2021a). For medication names, we leverage the RxNorm vocabulary (Nelson et al., 2011) to obtain synonyms (also known as tradenames). We choose RxNorm due to its abundance of medication tradenames (see Section 2.6.5). For diagnoses, we leverage the SNOMED CT vocabulary (Donnelly et al., 2006). The SNOMED CT vocabulary was chosen due to its credible reputation in medical science research. We also considered WordNet (Fellbaum, 2010) as a synonym source (see Section B.6.1 in Appendix B). However, our analysis suggests that SNOMED CT is more suitable for this task. We also considered training a synonym substitution model using the SpiderSyn (Gan et al., 2021a) text-to-SQL parsing dataset. However, we found that this dataset contains limited medical NLQs, with synonym substitutions primarily occurring for common nouns and verbs, not complex medical jargon (see **Figure B.5** of Appendix B). Alternatively, creating an additional training dataset would require additional computational overhead. We aim for greater than n = 500 value-synonym replacement questions for each unanswerable question type since this category (in *RQ2*) only includes the value-synonym replacement

unanswerable subcategory. See **Table 3.7** for examples of value-synonym replacement questions and **Table 3.8** for the sample sizes of the value-synonym replacement questions according to the unanswerable question type.

TABLE 3.7: Examples of Value-Synonym Replacement Questions. Value-Synonym replacement questions are generated using SNOMED CT for diagnoses and RxNorm for medication names. The blue font is the medical jargon in the synthetic reference question, while the green font is the medical synonym in the value-synonym replacement question

Diagnoses	**Medication**
Which medication can I use to prevent the diagnosis Bronchial Spasm? Which medication can I use to prevent the diagnosis bronchospasm (finding)	What are the drug tradenames of Iloperidone? What are the drug tradenames of Fanapt?
Which medication can I use to prevent the diagnosis of eclampsia? Which medication can I use to prevent the diagnosis of eclamptic toxemia	Could you please inform me of the tradenames for vitamin K1? Could you please inform me of the tradenames for Phytonadione
Which medication can I use to prevent hydatidiform mole? Which medication can I use to prevent invasive mole?	What are the drug tradenames of Docetaxel? What are the drug tradenames of Docefrez?

TABLE 3.8: Sample Sizes of the Questions with Value-Synonym Replacement.

Unanswerable Question Type	**Total**
ICD-10-CM	551
Diagnoses-to-medication treatment mapping	618
Diagnoses-to-medication prevention mapping	587
Side-effects	886
Tradenames	616
Medication-to-diagnoses treatment mapping	616
Medication-to-diagnoses prevention mapping	626

3.6.4 Ground Truth SQL Template Generation

At this stage, we have a dataset comprising triples, where each triple consists of a synthetic reference question, and in some cases its paraphrase and value-synonym replacement. The synthetic reference questions and paraphrases form part of the out-of-schema questions that are tested in *RQ1*, while the value-synonym replacement questions form part of the questions requiring medical knowledge in *RQ2* (see Section 1.5). In this task, we generate the ground-truth SQL query for each question in the synthetic reference questions, paraphrases and value-synonym replacement questions. We generate one ground-truth SQL query for each question. This step is important as we perform supervised fine-tuning that requires ground-truth labels (i.e., ground-truth SQL queries). The LLM leverages these ground-truth SQL queries to learn how to translate an NLQ into a correct SQL query. To ensure consistency, we create SQL templates for each unanswerable question type. See **Figure B.7** for diagnoses SQL templates and **Figure B.6** for medication SQL templates in Section B.7 of Appendix B. This enables control over the generation of the ground-truth SQL queries, ensuring that each unanswerable question type has the same SQL form, including the same tables and columns. The ground-truth SQL queries must be correct, as the text-to-SQL model will learn mappings from the NLQ to the SQL query. Each unanswerable question type maps to tables and columns that must be correct in the SQL template. The SQL templates have empty slots for conditional values in the *WHERE* clause. We slot-fill the *WHERE* clause with their respective question values, i.e., either diagnoses or medication names. Since each value-synonym replacement question and paraphrase map to a synthetic reference question, the value-synonym replacement question and paraphrase get the same ground-truth SQL query as their synthetic reference question. We do this so that when we evaluate the exact matching (EM) accuracy, we test if the text-to-SQL model can still predict the correct ground-truth SQL query, regardless of perturbations to the synthetic reference question.

3.7 Data Splitting

In this section, we perform data splitting of the synthetic NLQ/SQL pairs generated in Section 3.6 into training, validation and test splits and merge it with the MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018) data splits.

3.7.1 MIMIC-III and eICU

EHRSQL (Lee et al., 2022), comprises the MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018) data sources. In each of EHRSQL's validation and test splits, 33% of questions are unanswerable. At a granular level, MIMIC-III and eICU, also have approximately 33% of their total questions as unanswerable within each of the validation and test splits. See **Table 3.9** for the sample sizes of each EHR source.

TABLE 3.9: Question Partitions for all Data Sources. The *Unans.* row name represents the count of unanswerable questions in a given data source and data split, while the *Ans.* row name represents the count of answerable questions in a given data source and data split. The *Totals* heading shows the total question count, including unanswerable and answerable questions for a specific data split (train, validation and test splits).

	Train	Valid	Test
MIMIC-III			
Unans.	0	362	588
Ans.	9318	760	1198
eICU			
Unans.	0	362	588
Ans.	9270	752	1203
Synthetic Reference Questions			
Unans.	4828	1607	1615
Paraphrases			
Unans.	1784	600	616
Value-Synonym Replacement Questions			
Unans.	2627	955	837
Totals			
Totals	27,827	5398	6645

3.7.2 Synthetic Data

See **Figure 3.3** for a schematic representation of the synthetic data splitting. Each unanswerable question group, namely, synthetic reference questions, paraphrases and value-synonym replacement questions, comprises unanswerable question types identified in Section 3.3. For each unanswerable question type, we apply a 6:2:2 ratio to split unanswerable questions into train, validation and test splits (see **Figure 3.3**, label 3). We then combine the respective splits across these unanswerable question types for a given unanswerable group (see **Figure 3.3**, label 4). As a result, we obtain train, validation and test splits for each unanswerable question group. The respective splits are then combined across unanswerable question groups (see **Figure 3.3**, label 5). Consequently, we obtain a single train, validation and test split that comprises questions from each unanswerable group. In agreement with Li et al. (2023b), we then look for data leakage across the splits (train, validation and test) to check that a question only appears in one split and we remove instances of duplicates across splits. Moreover, we ensure that a triple (synthetic reference question, paraphrase and value-synonym replacement) only occurs in one split. By including a triple in only one split we ensure that the evaluation questions in the validation and test sets (regardless of their form) remain unseen during training. As a result, the LLM learns various representations of a question, rather than memorizing questions. We then also look for duplicates within a single split and remove them. Finally, the original MIMIC-III and eICU splits are combined with the synthetic data. See **Table 3.9** for the final training, validation and test split sizes. The training data has a sample size = 27,827, the validation data has a sample size of n = 5398 and the test set has a sample size of n = 6645.

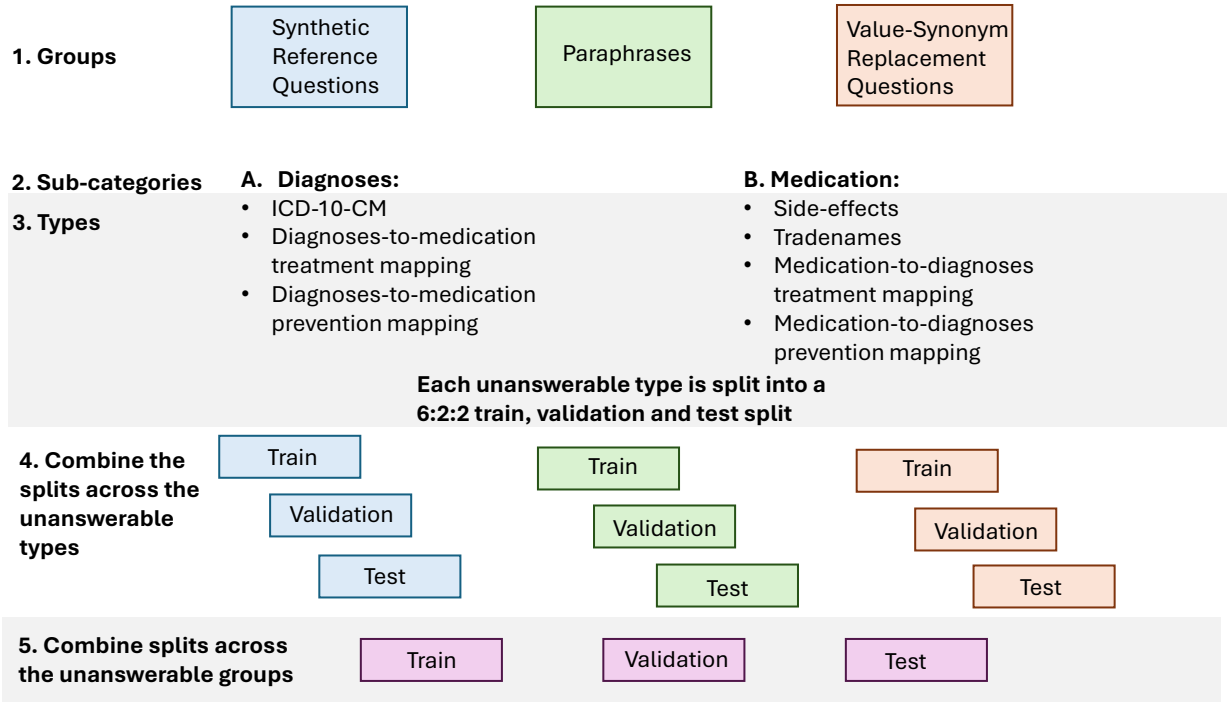


FIGURE 3.3: Schematic Representation of Data Splitting.

Label 1 shows the unanswerable groups, where the synthetic reference questions and paraphrases form part of the out-of-schema questions and the value-synonym replacement questions form part of the unanswerable questions requiring medical knowledge. Label 2 shows the 2 unanswerable subcategories, where each unanswerable group comprises of diagnoses and medication subcategories of questions. Label 3 shows that we create 6:2:2 train, validation and test splits for each unanswerable question type in a given unanswerable question group. Label 4 shows that we then combine the train, validation and test splits across the unanswerable types to form a single train, validation and test split for each unanswerable group. Finally, label 5 shows how we combine splits across the unanswerable groups into a single train, validation and test split.

3.8 Summary

In this chapter, we discuss the data used for text-to-SQL fine-tuning. We also discuss the data augmentation, including EHR augmentation and synthetic question generation to address the primary unanswerable question categories, namely out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*). In Section 3.3, we perform EDA to determine the most important unanswerable question subcategories and question types. We find that two primary unanswerable subcategories exist, namely, questions related to diagnoses and medication. Each unanswerable subcategory also consists of unanswerable question types. The diagnoses subcategory includes the following unanswerable types: ICD-10-CM codes, diagnoses-to-medication treatment mappings, and diagnoses-to-medication prevention mappings. The medication subcategory includes the following unanswerable types: medication side effects, tradenames, medication-to-diagnoses treatment mappings and medication-to-diagnoses prevention mappings. See **Figure 3.2** to review the unanswerable question hierarchy. In Section 3.5, we then leverage these unanswerable subcategories and types to create an additional EHR schema, grounded with information from medical knowledge sources, including medical vocabularies in the UMLS (Bodenreider, 2004) and the SIDER (Kuhn et al., 2016) knowledge sources. In Section 3.6, we then synthetically generate the unanswerable question subcategories and question types, and their respective ground-truth SQL queries. This process involves synthetically generating three unanswerable question groups, namely synthetic reference questions, paraphrases and value-synonym replacement questions. The synthetic reference questions and paraphrases form part of the out-of-schema questions, while the value-synonym replacement questions form part of the unanswerable questions requiring medical knowledge. Finally, in Section 3.7, we perform data splitting into training, validation and test splits. We then merge the original MIMIC-III and eICU data splits with the synthetic question splits and employ filtering to remove duplicate samples across splits and within splits. In Chapter 4 we discuss how this data is used in text-to-SQL fine-tuning to address out-of-schema questions and unanswerable questions requiring medical knowledge.

Chapter 4

Experimental Method

4.1 Overview

In Chapter 3, we discuss the data sources, and exploratory data analyses (EDA) to identify important unanswerable question subcategories and types that belong to the primary unanswerable question categories, namely, out-of-schema questions and unanswerable questions requiring medical knowledge. We then use these unanswerable subcategories and types to synthetically generate unanswerable questions and a new EHR database that answers these questions. We conclude the chapter by splitting the data to obtain the training, validation, and test sets. In this chapter, we discuss how the synthetic data is used in experiments to address our research questions, *RQ1* and *RQ2*, which focus on out-of-schema questions and unanswerable questions requiring medical knowledge, respectively (see Section 1.5). Refer to **Figure 3.1** to review the methods to follow. Section 4.2 discusses the experimental setup. Section 4.3 discusses the choice of the transformer model used for text-to-SQL translation. Section 4.4 discusses the text-to-SQL model hyperparameters. In Section 4.5, we discuss the experiments—model fine-tuning to find the best models when evaluated on the validation questions. Section 4.5.1 details the experiments for the out-of-schema questions (*RQ1*) and Section 4.5.2 details the experiments for unanswerable questions requiring medical knowledge (*RQ2*). Section 4.6 discusses model evaluation of the best models on the test questions. We also evaluate if our model constraints were adhered to, by assessing model reliability with the implementation of answer abstention and evaluating if the best models maintain answerability on the original answerable questions. Finally, we also perform error analyses to identify the types of questions the best models fail to answer.

4.2 Experimental Setup

The experiments are conducted using Python 3.7[1] in a conda[2] environment. Computations were performed using facilities provided by the University of Cape Town's ICTS High Performance Computing (HPC[3]) team. The HPC employs a Linux CENTOS 7.7-1908 operating system and a 200 GB storage pool. All experiments are run with a NVIDIA GeForce RTX 4090 graphics card, with 24 GB of G6X memory. We use the synthetic questions and the additional schema generated in Chapter 3 in subsequent experiments to address the unanswerable questions in *RQ1* and *RQ2* (see Section 1.5). In these scenarios, the additional schema is seen or unseen during training. First, we address out-of-schema questions (*RQ1*), as seen in **Figure 4.1** (a). We simulate a scenario where the out-of-schema question includes concepts from an unseen schema, not found in the training schemas (e.g., *drug tradenames*). For the second scenario, we address unanswerable questions requiring medical knowledge, as seen in **Figure 4.1** (b). Here the unanswerable question contains complex medical jargon that does not explicitly relate to concepts in the training schemas. For example, the question in the figure contains the synonym, *metazolv*, of the base medical concept, *metaclopramide*, found in one of the schemas. As a result of these challenges, the text-to-SQL model predicts inaccurate or inexecutable SQL queries when confronted with out-of-schema concepts or complex medical terminology in the input question.

[1] https://www.python.org/downloads/release/python-370/
[2] https://anaconda.org/anaconda/conda
[3] hpc.uct.ac.za

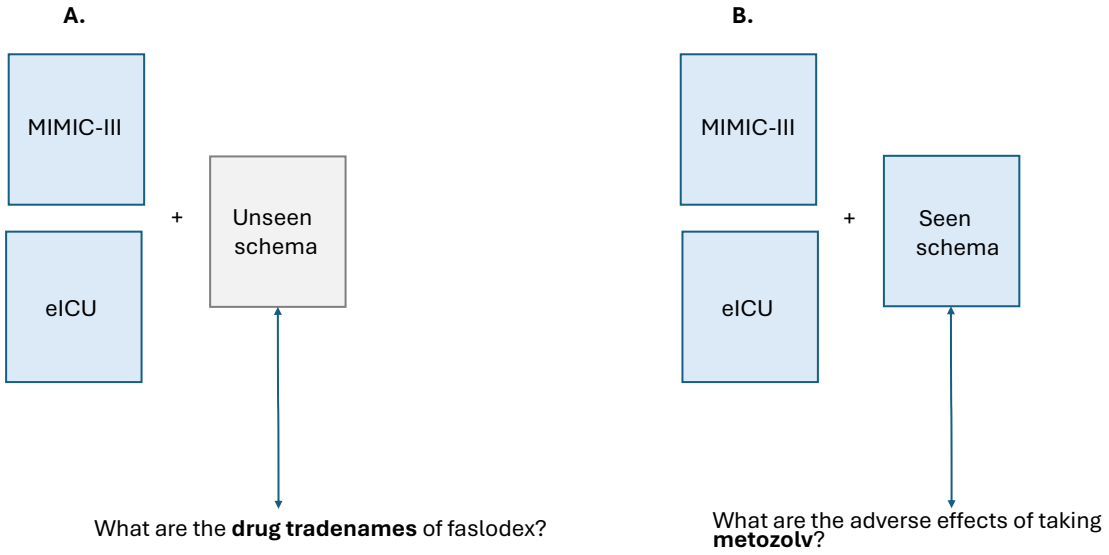


FIGURE 4.1: Experimental Setup.

Blue squares are the seen schemas. The grey square is an unseen schema. The bold font in the question suggests unanswerability. Here (a) represents an unanswerable question with an out-of-schema concept (*drug tradenames*) found in the unseen schema that is not found in the original, training schemas. The diagram (b) represents an unanswerable question requiring medical knowledge, where the question maps to a seen schema. However, the medical jargon in the question (*metazolv*) does not explicitly match terms in the seen schema.

4.3 Text-to-SQL Model

We choose a sequence-to-sequence (Seq2Seq) transformer-based large language model (LLM), T5 (T5-Base and T5-Large), for text-to-SQL fine-tuning (Raffel et al., 2020; Vaswani et al., 2017). This is in accordance with Lee et al. (2022) and Lee et al. (2024), which fine-tune T5 models on medical question-answering data. Lee et al. (2022), in particular, show that transfer learning from pre-trained transformer-based language models, such as T5, surpasses healthcare-specific text-to-SQL models with LSTM-based architectures (Bae et al., 2021). In addition, we choose T5 for this SQL generation task due to its wide use in selective generation (Ren et al., 2022) and calibration (Stengel-Eskin and Van Durme, 2023) in NLP. See Section 2.3.2 for more details on the T5 model. We also considered the GPT-4 (Achiam et al., 2023) series and its variants, as implemented by Li et al. (2023b). However, we strongly recommend T5 for two reasons: (1) it has relatively few model parameters, which reduces computational constraints during training while still delivering satisfactory performance, and (2) the T5 series is the standard in existing text-to-SQL research. We then perform supervised fine-tuning (SFT) as opposed to in-context learning (ICL) because research suggests that ICL still lags behind SFT in text-to-SQL translation tasks (Pourreza and Rafiei, 2024). Similar to Hazoom et al. (2021), we motivate that the creation of a large, high-quality medical text-to-SQL parsing dataset enables the effective fine-tuning of a domain-specific text-to-SQL model.

4.4 Hyperparameter Tuning

A targeted hyperparameter search is employed using T5-Base to identify and train the downstream models with optimal hyperparameters. We train the model on the training dataset obtained in Chapter 3, comprising the MIMIC-III, eICU and synthetic unanswerable questions (n^{train} = 27,827 and n^{valid} = 5398). For this task, the Weights & Biases (W&B[4]) AI developer platform is used to log the training and validation loss (i.e., the cross-entropy loss). The hyperparameters can be seen in **Table 4.1**, where changes to the training batch size, learning rate and weight decay hyperparameters are assessed in two different experiments, denoted by *model 1* and *model 2*, respectively. We discuss these modeling choices below.

[4]https://github.com/wandb/wandb

A smaller *training batch size* encourages model generalization by introducing diverse samples in the training process, however, a larger training batch size encourages faster model training (Lee et al., 2022). In accordance with Lee et al. (2022), a relatively small training batch size (train batch size = 4) is assessed in model 1 and a larger training batch size is assessed in model 2 (train batch size = 20).

Similar to Lee et al. (2022), *learning rates* with a magnitude of 10^-4 and 10^-3 are assessed, with a fixed learning rate throughout training. A relatively small learning rate (r = 4 x 10^{-4}) is chosen in model 1, enabling slow learning, while in model 2 we assess a relatively larger learning rate (r = 1 x 10^{-3}) enabling the model to converge more quickly.

We also assess the effect of L2 regularization, namely *weight decay* (Krogh and Hertz, 1991), which penalizes large model weights and encourages weights to take on small values. In model 1, we assess a relaxed weight decay penalty factor (L_2 = 1 x 10^{-1}) and in model 2 we assess a stricter weight decay penalty factor (L_2 = 1.0).

The optimal model is determined by selecting hyperparameters that correspond to the local minima of both the training and validation loss. In **Table 4.2**, we see the training and validation loss for model 1 and model 2. We observe that model 1 has the smallest training and validation loss relative to model 2. For this reason, we employ the hyperparameters of model 1 in downstream model fine-tuning and inference. Take note that the validation loss is higher than the training loss because the validation dataset includes unseen, unanswerable questions that are not present in the training dataset.

TABLE 4.1: T5-Base Training and Evaluation Hyperparameters.

Training		
	Model 1	**Model 2**
Total training steps	100000	100000
Batch size	4	20
Max length	512	512
Optimizer	Adam[1]	Adam
Learning rate	0.0004	0.001
Learning rate scheduler	Fixed	Fixed
Accumulation steps	8	8
Warm-up steps	0	0
Weight decay	0.1	1.0
Evaluation		
	Model 1	**Model 2**
Number of beams	5	5
Repetition penalty	1.0	1.0
Length penalty	1.0	1.0

TABLE 4.2: Experimental Results Showing the Training and Validation Loss.

Model ID	**Training Loss**	**Validation Loss**
Model 1	0.0000916	1.733
Model 2	0.0001064	1.789

[1]Kingma (2014)

4.5 Experiments

In this section, we discuss the experiments for each research question, *RQ1* and *RQ2*, to address out-of-schema questions (see Section 4.5.1) and unanswerable questions requiring medical knowledge (see Section 4.5.2), respectively. Similar to Yu et al. (2018c), we compute the exact matching (EM) accuracy, the component matching (CM) accuracy and the execution (EX) accuracy during evaluation, when reporting the validation and test results. However, during model training, we focus solely on computing the EM accuracy for optimization purposes. Section 2.4.1 in the literature review defines the evaluation metrics and provides the equations for their computation.

4.5.1 Out-of-Schema Questions

In this section, we describe the experiments for *RQ1*, where we investigate strategies to resolve and answer out-of-schema questions. See **Table 4.3** for the experiment details. The models are evaluated on the out-of-schema validation questions, comprising the synthetic reference questions and paraphrases (n^{valid} = 2207). However, in Experiments 2, 4, and 5 we evaluate the models on the validation questions and its serialized schema, while for Experiment 6 we evaluate the model on the validation questions and the pruned schema. In each experiment, T5-Base with the optimal hyperparameters in **Table 4.1** (model 1) are used in model fine-tuning. In addition, we also fine-tune T5-Large in Experiments 1 to 5. We did not fine-tune the model in Experiment 6 using T5-Large, as we saw minimal improvements with T5-Large coupled with schema serialization (see Section 5.3.1). The same reason applies to why we did not fine-tune the model in Experiment 6 with schema pruning at inference only, where we see minimal performance for schema serialization at inference only.

TABLE 4.3: Experiments to Resolve Out-of-Schema Questions. Each model in each experiment is evaluated on the out-of-schema, validation questions (n^{valid} = 2207), comprising the synthetic reference questions and paraphrases (see **Table 3.9**).

Experiment ID	Model	Train Size	Training Data	Augmentation
Experiment 1	T5-Base, T5-Large	18,588	MIMIC-III, eICU	None
Experiment 2	T5-Base, T5-Large	18,588	MIMIC-III, eICU	Schema serialization at inference
Experiment 3	T5-Base, T5-Large	27,827	MIMIC-III, eICU, synthetic questions	None
Experiment 4	T5-Base, T5-Large	27,827	MIMIC-III, eICU, synthetic questions	Schema serialization at inference
Experiment 5	T5-Base, T5-Large	27,827	MIMIC-III, eICU, synthetic questions	Schema serialization at inference & training
Experiment 6	T5-Base	27,827	MIMIC-III, eICU, synthetic questions	Schema pruning at inference & training

Experiment 1: MIMIC-III + eICU

Similar to Lee et al. (2022), in this experiment, we train the text-to-SQL model only on the original, answerable questions in the MIMIC-III (Johnson et al., 2016) and eICU (Pollard et al., 2018) datasets. However, unlike Lee et al. (2022), we do not train the text-to-SQL model on a single dataset. Instead, we train the text-to-SQL model on the combined MIMIC-III and eICU datasets. The reason for combining the two datasets is to develop a text-to-SQL model capable of addressing questions from multiple EHR sources. This approach aims to improve the model's generalizability for medical text-to-SQL translation, enabling the model to perform effectively across diverse medical datasets with different EHR schemas and questions (Li et al., 2023b; Yu et al., 2018c). We train T5-Base and T5-Large (Raffel et al., 2020) on the MIMIC-III and eICU training data (n^{train} = 18,588), after which we evaluate the model on the out-of-schema validation questions (n^{valid} = 2207). This experiment serves as a baseline, as we do not expect to achieve good model performance when the model is evaluated on the out-of-schema, validation questions. However, we use the results of this experiment as additional motivation that existing text-to-SQL models do not properly address out-of-schema questions (Lee et al., 2022). This warrants strategies to address out-of-schema questions, which we investigate in downstream experiments.

Experiment 2: MIMIC-III + eICU + Schema Serialization at Inference

The hypothesis for this experiment is that the text-to-SQL model learns text-to-SQL patterns in the training data and only needs to see the new, unseen schema at inference to make valid SQL predictions of out-of-schema questions (Gupta et al., 2024). This experiment leverages the model fine-tuned in Experiment 1, however, we provide the model with the complete serialized schema at inference. Similar to Lee et al. (2022) and Lee et al. (2024), the serialized schema includes all table and column names appended to the end of the out-of-schema validation question. Refer to **Figure 4.2** for an example. The serialized schema includes the schema items from the augmented EHR created in Section 3.5 and has the format *Table name: column names, etc.*. Once the out-of-schema validation questions are prepared in the format NLQ + serialized schema, we provide the input to the model and evaluate the results. The language model can then infer associations between tokens in the NLQ and tables and columns in the serialized schema.

FIGURE 4.2: Natural Language Question with Serialized Schema. Black font indicates the question. Red font indicates table names. Green font indicates column names and a bar character separates tables.

NLQ + Serialized Schema:
Which medication can I use to treat the patient's dysentery? | ICD10CM_DIAGNOSIS : ROW_ID, ICD10CM_CODE, CUI_CODE, DIAGNOSIS_SHORT_TITLE, DIAGNOSIS_LONG_TITLE | ICD10PCS_PROCEDURE : ROW_ID, ICD10PCS_CODE, CUI_CODE, PROCEDURE_SHORT_TITLE, PROCEDURE_LONG_TITLE | DRUG_TRADENAMES : ROW_ID, CUI_CODE, DRUG_NAME, TRADENAMES, SNOMED_CODE, MESH_CODE, RXNORM_CODE | MEDICATION_TO_DIAGNOSIS : ROW_ID, CUI_CODE, DRUG_NAME, TREATMENT_OF, PREVENTION_OF, DRUG_INFORMATION | DRUG_SIDE_EFFECTS : ROW_ID, CUI_CODE, DRUG_NAME, SIDE_EFFECTS | DIAGNOSIS_TO_MEDICATION : ROW_ID, DIAGNOSIS, MEDICATION_TREATMENT, MEDICATION_PREVENTION

Number of Tokens: 312

Experiment 3: MIMIC-III + eICU + Synthetic Data

Similar to Xie et al. (n.d.), who employ continual pre-training on an open-source LLM for domain adaptation, we hypothesize that to address out-of-schema questions, one needs to iteratively identify the users' out-of-schema questions and incorporate these questions and their corresponding SQL queries in the model's training data. In this experiment, we do not provide any additional schema information to the model input. Instead, we train the model on the MIMIC-III, eICU and the synthetic unanswerable questions (n^{train} = 27,827). The assumption is that the model does not need additional schema information, however, the model needs sufficient NLQ and SQL training pairs to learn how the out-of-schema questions relate to the new schema.

Experiment 4: MIMIC-III + eICU + Synthetic Data + Schema Serialization at Inference

In this experiment, we hypothesize that in addition to the synthetic training data (including out-of-schema questions and unanswerable questions requiring medical knowledge) and the MIMIC-III and eICU training data (n^{train} = 27,827), the model also needs to *see* the schema at inference (Gupta et al., 2024). Using the model fine-tuned in Experiment 3, we follow a similar rationale to Experiment 2, by appending the serialized schema to the out-of-schema validation questions at inference. This results in a model input with the NLQ + serialized schema. Refer to **Figure 4.2** for an example. The language model can then infer associations between tokens in the NLQ and tables and columns in the serialized schema.

Experiment 5: MIMIC-III + eICU + Synthetic Data + Schema Serialization

This experiment builds upon Experiment 4 by incorporating the serialized schema not only during inference but also during model training (Lee et al., 2024, 2022). Refer to **Figure 4.2** for an example. The hypothesis is that by incorporating the serialized schema during training, the model learns how to effectively use the additional schema information at inference time.

Experiment 6: MIMIC-III + eICU + Synthetic Data + Schema Pruning

We use a similar approach in this experiment relative to Experiment 5, where a serialized schema is provided to the model both at training and inference. We do not evaluate the model only at inference, as our experimental results (see Chapter 5) suggest that models in the other experiments, leveraging schema serialization at both training and inference, outperform those models that only use schema serialization at inference. However, in this experiment, we do not provide the entire serialized schema to the model. Rather we employ schema pruning to find the most relevant tables and their columns to a given question. We leverage the BM25 algorithm (Robertson, Zaragoza, et al., 2009) for table pruning. This is in contrast to Li et al. (2024), which applies the BM25 algorithm only for column value pruning. In this approach, we select the top three most relevant tables for a given question, along with all their columns. The rationale is that by providing only relevant schema items, the input length of the LLM will be shorter, thus reducing the risk that the model will forget information in the input sequence. We also considered pruning the columns, however, this resulted in suboptimal performance. We hypothesize that this approach will improve text-to-SQL performance on out-of-schema questions because the model only sees relevant schema items. Unlike Li et al. (2023b), however, we do not subsequently employ the longest-common substring (LCS) algorithm to further reduce the pruned schema items. This is because the number of tables and columns in our schemas are considerably fewer than that of the BIRD schema. Refer to **Figure 4.3** for an example of an NLQ with a pruned schema.

FIGURE 4.3: Natural Language Question with Pruned Serialized Schema. Black font indicates the question. Red font indicates table names. Green font indicates column names and a bar character separates tables.

NLQ + Pruned Serialized Schema:
What are the drug tradenames of chlorothiazide sodium? | DRUG_TRADENAMES: CUI_CODE, ROW_ID, SNOMED_CODE, TRADENAMES, MESH_CODE, RXNORM_CODE, DRUG_NAME | DRUG_SIDE_EFFECTS: CUI_CODE, SIDE_EFFECTS, DRUG_NAME, ROW_ID | MEDICATION_TO_DIAGNOSIS: CUI_CODE, ROW_ID, DRUG_INFORMATION, PREVENTION_OF, TREATMENT_OF, DRUG_NAME

Number of Tokens: 160

4.5.2 Unanswerable Questions Requiring Medical Knowledge

In this section, we describe the experiments for *RQ2*, where we investigate strategies to resolve and answer unanswerable questions requiring medical knowledge. See **Table 4.4** for the experiment details. The models are evaluated on the unanswerable validation questions that require medical knowledge, consisting of the questions with value-synonym replacement of their medical jargon (n^{valid} = 955). In each experiment, T5-Base with the optimal hyperparameters in **Table 4.1** (model 1) are used for model fine-tuning. In addition, we also fine-tune T5-Large in Experiments 7 to 10.

TABLE 4.4: Experiments to Resolve Unanswerable Questions Requiring Medical Knowledge. Each model is fine-tuned on n^{train} = 27,827 question and SQL pairs and evaluated on the validation questions (n^{valid} = 955), where the validation questions comprise the unanswerable questions with value-synonym replacement of medical jargon (see **Table 3.9**). Each model configuration is fine-tuned on T5-Base and in another experiment, T5-Large. Note that Experiments 8 and 9 use SQL post-processing at inference, while Experiment 10 uses RAG at both fine-tuning and inference.

Experiment ID	Training Data	Augmentation
Experiment 7	MIMIC-III, eICU, synthetic questions	None
Experiment 8	MIMIC-III, eICU, synthetic questions	Synonyms at inference (obtained from MeSH definitions)
Experiment 9	MIMIC-III, eICU, synthetic questions	Synonyms at inference (obtained from SNOMED CT and RxNorm)
Experiment 10	MIMIC-III, eICU, synthetic questions	RAG using synonyms obtained from SNOMED CT and RxNorm

Experiment 7: Vanilla

The vanilla model serves as the baseline and includes the model fine-tuned on the MIMIC-III, eICU and the synthetic training NLQ and SQL pairs (n^{train} = 27,827). The synthetic training data includes both unanswerable questions requiring medical knowledge and out-of-schema questions. Similar to Gan et al. (2021a) and Chang et al. (2023), we do not expect this model to perform well, since the unanswerable questions requiring medical knowledge include an NLQ perturbation, namely a value-synonym replacement. In a value-synonym replacement, the medical jargon in the question is replaced with a medical synonym not found in the predicted column's values within the EHR. However, the model *sees* the tables and columns during model training by learning mappings between an NLQ and SQL query. As a result, the model should do relatively well in predicting columns in the *SELECT* clause and tables in the *FROM* clause of the SQL query, however, it will not do as well when predicting the values in the *WHERE* clause.

Experiment 8: SQL Post-Processing with Synonyms Obtained from MeSH Definitions

In this experiment, we obtain the SQL predictions of the questions requiring medical knowledge from the model fine-tuned in Experiment 7. Similar to Wang et al. (2020), we apply an SQL post-processing step to the model predictions. During this post-processing step, all valid synonyms of the medical jargon in a given question are included in the SQL *WHERE* clause. Here valid synonyms are obtained by matching the MeSH (Lipscomb, 2000) definition of a medical concept to all its synonyms with the same MeSH definition (refer to Section 3.5.3). As a result, the WHERE clause is modified from *WHERE col = x* to *WHERE col in (x, synonym1, synonym2..)*. The hypothesis is that at least one synonym in the *WHERE* clause will be the base medical concept found in the predicted column. Refer to **Figure 4.4** for an example.

FIGURE 4.4: SQL Post-Processing with Synonyms Obtained from MeSH Definitions. Here the *WHERE* clause of the predicted SQL is modified to include synonyms related to the conditional value, Zofran ODT.

Predicted SQL:
SELECT treatment_of FROM medication_to_diagnosis WHERE drug_name = 'Zofran ODT' COLLATE NOCASE

Modified SQL:
SELECT treatment_of FROM medication_to_diagnosis WHERE drug_name COLLATE NOCASE IN ('Zuplenz', 'Zofran', 'ondansetron')

Experiment 9: SQL Post-Processing with Synonyms Obtained from SNOMED CT and RxNorm

Similar to Experiment 8, the predicted SQL queries for unanswerable questions requiring medical knowledge are derived from the model in Experiment 7. The SQL predictions then undergo an SQL post-processing step. The *WHERE* clause is modified to include all possible medical synonyms of the medical concept found in the given question. However, instead of using MeSH (Lipscomb, 2000) definitions to find medical synonyms, we utilize the SNOMED CT (Donnelly et al., 2006) vocabulary for diagnoses and the RxNorm (Nelson et al., 2011) vocabulary for medication when finding synonyms for medical concepts in a given question. Refer to Section B.6.3 for more details on how querying is done against the medical vocabulary. As a result, the WHERE clause is modified from *WHERE col = x* to *WHERE col in (x, synonym1, synonym2..)*. We test the same hypothesis as in Experiment 8, where at least one synonym will be the base medical concept found in the predicted column. Refer to **Figure 4.5** for an example.

FIGURE 4.5: SQL Post-Processing with Synonyms Obtained from RxNorm. Here the *WHERE* clause of the predicted SQL is modified to include synonyms related to the conditional value, fluorouracil.

Predicted SQL:
SELECT tradenames FROM drug_tradenames WHERE drug_name = 'fluorouracil' COLLATE NOCASE

Modified SQL:
SELECT tradenames FROM drug_tradenames WHERE drug_name COLLATE NOCASE IN ('tolak', 'carac', 'fluoroplex', 'adrucil', 'efudex', 'fluorouracil')

Experiment 10: RAG

This experiment leverages retrieval augmented generation (RAG) to answer unanswerable questions requiring medical knowledge. The medical jargon in a question is simplified by giving the model additional context. This context includes the medical synonyms obtained in Experiment 9, where diagnoses synonyms are obtained from the SNOMED CT (Donnelly et al., 2006) medical vocabulary and medication synonyms are obtained from the RxNorm (Nelson et al., 2011) medical vocabulary. We provide this additional context both during training and at inference so that the model can learn how to leverage the additional knowledge. See an example of the prompt template in **Figure 4.6**.

FIGURE 4.6: Retrieval-Augmented Generation (RAG). Here the unanswerable question requiring medical knowledge is appended with additional synonym context.

Question:
Translate the following question into an SQL query - What are the drug tradenames of Angiomax?

Context:
Incorporate the following medical synonym(s) into the query - Bivalirudin

4.6 Testing

We choose the two best-performing models (based on the validation performance) to address out-of-schema unanswerable questions and the unanswerable questions requiring medical knowledge, respectively. We then evaluate the performance of these models on the test questions, where n^{test} = 6645 questions. This evaluation is important to assess that the best-performing models generalize to unseen questions. For these experiments, the out-of-schema test questions have n^{test} = 2231 questions; and the unanswerable test questions that require medical knowledge have n^{test} = 837 questions. The out-of-schema test questions consist of the synthetic reference questions (n^{test} = 1615) and the paraphrases (n^{test} = 616). The unanswerable test questions that require medical knowledge include the unanswerable questions with value-synonym replacement. We then evaluate the best models on the model constraints. Finally, we also perform error analysis of the best models to evaluate where their predictions are incorrect. Below we provide more details on the model constraints.

4.6.1 Model Constraints

Similar to Lee et al. (2022) and Lee et al. (2024), we employ answer abstention to improve text-to-SQL model reliability. Answer abstention refuses to answer a given question when the text-to-SQL model is not confident in its translated SQL query. In addition, we also assess that the model performs well on the original answerable questions. This ensures a well-rounded text-to-SQL model that resolves unanswerable questions, ensures the reliability of its outputs, and also maintains model performance on the original answerable questions. Below we discuss various answer abstention strategies.

Answer Abstention

This section aims to improve text-to-SQL model trustworthiness by ensuring the model only answers questions it can reliably answer. We fine-tune a T5-Base model on the MIMIC-III, eICU and synthetic reference questions (n^{train} = 27,827). We then use the model to make SQL predictions on the validation questions (n^{valid} = 5398). The decoded predictions are used for the answer abstention analysis. Similar to Lee et al. (2022) and Lee et al. (2024) we employ a simple uncertainty estimation method to determine when the text-to-SQL model should refuse to answer a question. We employ an entropy-based strategy, where the model will refuse to answer a question if the maximum entropy of the decoded SQL tokens are greater than a pre-defined threshold. Research by Lee et al. (2024) suggest that entropy-based abstention outperforms other strategies, including heuristic-based, learning-based, distance-based, and prompting-based answer abstention approaches. See Section 2.5.5 for further details. We obtain the classification threshold using K-means clustering (MacQueen, 2018; Steinhaus, 1957), a one-class support vector machine (SVM) (Shin et al., 2005) or a literature-based threshold (Lee et al., 2022). The chosen threshold should correctly differentiate answerable from unanswerable questions, including scenarios where an unanswerable question becomes answerable with the addition of schema or medical knowledge. This section discusses how we establish the question classification threshold using an entropy-based uncertainty estimation method, where we investigate K-means clustering, a one-class support vector machine, or a literature-derived threshold.

K-Means Clustering: In this experiment, we determine the answer abstention threshold using the K-means clustering algorithm (MacQueen, 2018; Steinhaus, 1957) on the maximum decoded entropy values (n^{valid} = 5398). We do this using the scikit-learn[5] module in Python. Here K = 2 clusters, namely answerable and unanswerable questions. The SQL predictions of the unanswerable questions will have relatively higher entropy values relative to answerable questions and so two clusters should form. We determine the threshold as the decision boundary between these two clusters of answerable and unanswerable questions. To determine the threshold, we determine the maximum and minimum values of both the first and second clusters. The threshold is set as the average of either the maximum of the first cluster and the minimum of the second cluster, or the minimum of the first cluster and the maximum of the second cluster, depending on which is higher. We find that the threshold is p = 1.32. The binary classifier predicts a question as unanswerable if the maximum decoded entropy value is greater than p = 1.32, otherwise it is classified as answerable.

One-Class Support Vector Machine: In this experiment, we determine the threshold by training the majority class of the validation questions using a one-class support vector machine (SVM) (Shin et al., 2005). We do this using the scikit-learn module in Python. The majority class is the synthetic unanswerable questions (n^{valid} = 3162). At inference, the one-class SVM isolates normal data points (majority class) from anomalous points (minority class). The threshold is determined in a similar manner to that of the K-means clustering method above. We find that p = 0.05. The classifier predicts a question as unanswerable if the maximum decoded entropy value is greater than p = 0.05, otherwise, it is classified as answerable.

Literature-based: In this experiment we use the threshold employed by Lee et al. (2022), where p = 0.15. Lee et al. (2022) obtain this threshold using a percentile-based approach, where the threshold is chosen as the 67^{th} percentile of the maximum entropy values among the validation decoded values. The 67^{th} percentile is chosen based on the fact that their validation data contains a static proportion of unanswerable questions, i.e., 33%. The percentile is computed as 100 - unanswerable percentage. The unanswerable questions are static because they remain unanswerable and are not resolved with schema or medical knowledge augmentation. The binary classifier predicts a question as unanswerable if the maximum decoded entropy value is greater than p = 0.15, otherwise, it is classified as answerable.

[5]https://scikit-learn.org

Answer Abstention Accuracies: Using the answer abstention thresholds obtained by the K-means clustering algorithm (p = 1.32), the one-class support vector machine (p = 0.05) and the literature-based threshold (p = 0.15), we calculate the accuracies of each classifier when predicting answerable questions, unanswerable questions and unanswerable questions that have been resolved with additional knowledge. The accuracies are seen in **Table 4.5**, where the answer abstention classifier utilizing K-means clustering to determine the threshold outperforms all other models. We see that this method achieves a 0.99 accuracy on the answerable questions and a 0.94 accuracy on the resolved unanswerable questions. In addition, this classifier achieves an accuracy of 0.21 on the MIMIC-III and eICU unanswerable questions, which is expected because a large proportion of the original unanswerable questions become answerable in this subset of questions. As a result, the model chooses to answer these unanswerable questions.

TABLE 4.5: Abstention Classifier: Answer Abstention Accuracies. The columns refer to the method used to compute the abstention threshold. The rows represent the questions on which the classifier is evaluated on.

Metric	**K-Means**	**One-Class SVM**	**Literature-based**
MIMIC-III and eICU Answerable	0.99	0.66	0.71
MIMIC-III and eICU Unanswerable	0.21	0.87	0.4
Resolved Unanswerable	0.94	0.5	0.5

Answerable Questions

We then investigate whether the best text-to-SQL models affect the answerability of the original, answerable questions in the MIMIC-III and eICU datasets. The model must generate the correct SQL queries for the original answerable questions to be considered successful. We compute the EM and EX accuracies as we did before for the unanswerable questions.

4.7 Summary

This chapter provides a comprehensive overview of our experimental methods and model development. We begin by detailing the experimental setup and text-to-SQL model used for fine-tuning, followed by an exploration of hyperparameter tuning to identify optimal model configurations. The chapter further presents our evaluation metrics and discusses a series of experiments designed to assess model performance on the validation questions, focusing specifically on addressing out-of-schema questions and unanswerable questions requiring medical knowledge. We then assess model reliability by evaluating answer abstention classifiers, where we find that the K-means clustering algorithm outperforms all other methods when determining the answer abstention threshold. We then also assess model performance on the original answerable questions, ensuring answerability is not lost. Finally, we perform error analysis to assess where the best models fail. In Chapter 5, we discuss the results of the experiments detailed in this chapter.

Chapter 5

Results

5.1 Overview

In Chapter 3 we discuss the theoretical framework supporting the data generation methods. In Chapter 4 we then discuss how the data is used for text-to-SQL fine-tuning to address unanswerable questions, namely, out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*). We also discuss our model constraints, including answer abstention for model reliability and maintaining answerability for the original answerable questions. This chapter reports the findings of model fine-tuning, including evaluation of the augmentation strategies in *RQ1* and *RQ2*. Section 5.2 re-introduces the research task, while the experimental results (for models evaluated on the validation questions) are detailed in Section 5.3. More specifically, Section 5.3.1 presents the findings for research question *RQ1*, while Section 5.3.2 presents the results for research question *RQ2*. The best models are then evaluated on the test questions in Section 5.4. Section 5.4.1 presents the test results for the best model in *RQ1* and Section 5.4.2 presents the test results for the best model in *RQ2*. In this section, we also report whether the model constraints are adhered to and perform error analyses on the final models' predictions.

5.2 Task

This research thesis aims to expand the coverage of answerable questions, ultimately using knowledge augmentation to transform unanswerable questions into answerable questions. The first objective leverages additional schema knowledge at inference or both training and inference to resolve out-of-schema questions (*RQ1*). The second objective is to augment additional medical knowledge to the model input or model output to simplify the meaning of medical jargon embedded within unanswerable questions requiring medical knowledge (*RQ2*). Two constraints are investigated. First, the text-to-SQL model should only answer questions it is confident in answering, and abstain from answering questions it is not confident in answering. Second, the model's performance on the original answerable questions should not deteriorate relative to the baseline model (without the proposed augmentation strategies). We evaluate the performance of the text-to-SQL models by computing the exact matching accuracy (EM), the execution accuracy (EX) and the component matching accuracy (CM). The model's performance on the validation set is used to identify the best model, whereas the performance on the test set evaluates model generalization on an unseen dataset. Refer to Section 2.4.1 for a more detailed explanation of the evaluation metrics.

5.3 Validation

This section investigates and identifies the best model to address out-of-schema questions in *RQ1* (see Section 5.3.1) and unanswerable questions requiring medical knowledge in *RQ2* (see Section 5.3.2).

5.3.1 RQ1: Out-of-Schema Questions

This section investigates the research question *RQ1*, as seen below.

RQ1: How does the integration of an *unseen* schema during inference enhance the performance of a sequence-to-sequence text-to-SQL model in answering out-of-schema unanswerable questions, and how does this approach compare to the effectiveness of fine-tuning on a relevant training dataset that includes these out-of-schema questions and the *unseen* schema?

Details of the experiments can be seen in Section 4.5.1.

Main Findings

Table 5.1 shows the performance of the T5 models (T5-Base and T5-Large) when evaluated on the out-of-schema validation questions (n^{valid} = 2207), namely, the synthetic reference questions (n^{valid} = 1607) and their paraphrases (n^{valid} = 600). Consequently, the integration of an unseen schema at inference (Experiments 2a and 2b) does not improve the model performance for both T5-Base and T5-Large. This is in comparison to the baseline models in Experiment 1a and 1b, without any augmentation strategy. In contrast, we see a significant increase in model performance (for both T5-Base and T5-Large) when fine-tuning with out-of-schema questions during training (Experiments 3a and 3b). This suggests that including structural representation of out-of-schema questions and SQL queries in the training set enables the model to answer these questions effectively. Take note that questions are distinct across the data splits. Moreover, the results show that any further augmentation strategy (either schema serialization or schema pruning) to the optimal model in Experiment 3a does not improve performance.

TABLE 5.1: Text-to-SQL Model Performance on the Out-of-Schema Validation Questions. The exact matching (EM) and execution accuracies (EX) are reported for T5-Base and T5-Large models with various schema augmentation strategies. The best EM and EX values are highlighted in blue.

Model ID	n^{train}	EM (%)	EX (%)
A. T5-Base			
Experiment 1a: MIMIC-III + eICU	18,588	0	0
Experiment 2a: MIMIC-III + eICU + schema serialization at inference	18,588	0	0
Experiment 3a: MIMIC-III + eICU + synthetic data	27,827	89.65	93.46
Experiment 4a: MIMIC-III + eICU + synthetic data + schema serialization at inference	27,827	62.81	70.1
Experiment 5a: MIMIC-III + eICU + synthetic data + schema serialization	27,827	85.73	91.40
Experiment 6a: MIMIC-III + eICU + synthetic data + schema pruning	27,827	0	0
B. T5-Large			
Experiment 1b: MIMIC-III + eICU	18,588	0	0
Experiment 2b: MIMIC-III + eICU + schema serialization at inference	18,588	0	0
Experiment 3b: MIMIC-III + eICU + synthetic data	27,827	87.37	92.15
Experiment 4b: MIMIC-III + eICU + synthetic data + schema serialization at inference	27,827	71.44	75.83
Experiment 5b: MIMIC-III + eICU + synthetic data + schema serialization	27,827	85.95	91.09

Analysis

Experiment 1: The results of this experiment serve as a sanity check and baseline, demonstrating that the models trained on MIMIC-III and eICU produce only incorrect SQL queries for out-of-schema questions in the validation set. In **Table 5.1** (a) and (b), we observe that the models, T5-Base and T5-Large, achieve 0% for both EM and EX on the out-of-schema, validation questions. These results emphasize the need for a text-to-SQL model that can correctly answer the out-of-schema, unanswerable questions.

Experiment 2: This evaluation shows that the models, T5-Base and T5-Large, do not effectively leverage additional schema information at inference. The results of this model in **Table 5.1** (a) and (b), show that both the EM and EX are 0%, when evaluated on the out-of-schema questions in the validation set. This suggests that the models are unable to generalize to out-of-schema questions, even when the unseen schema is incorporated at inference.

Experiment 3: **Table 5.1** (a) shows that the T5-Base variant of this model outperforms all other models (including the T5-Base and T5-Large models) when evaluated on the out-of-schema validation questions. The results show that there is a large increase in EM (EM = 89.65%) and EX (EX = 93.46%) performance relative to the models in Experiment 1a and Experiment 2a (each with an EM and EX of 0%). Although questions are distinct across splits, the model learns how the unseen schema and SQL queries are associated with the out-of-schema questions. Consistent with the findings of Experiment 3a, in **Table 5.1** (b), we observe that this model has superior performance among the T5-Large models, achieving an EM = 87.37% and an EX = 92.15%. However, these results still lag behind the best model fined-tuned with T5-Base (Experiment 3a). This approach has a marginal decline of 2.28% in EM and a 1.31% decline in EX relative to the best model in Experiment 3a. This suggests that using a larger T5 model does not improve the performance under the same settings.

Experiment 4: Building upon Experiment 3, this experiment suggests that the additional schema context given at inference deteriorates the model performance rather than improving it. In **Table 5.1** (a), we see that this model achieves an EM = 62.81% and an EX = 70.1%. The results show that there is a 26.84% performance drop in EM and a 23.36% performance drop in EX relative to the best model in Experiment 3a. In **Table 5.1** (b), we observe that this model achieves an EM = 71.44% and an EX = 75.83%. The results show that there is a 15.93% performance drop in EM and a 16.32% performance drop in EX relative to the model in Experiment 3b, which does not include any schema items during model training or inference. In addition, this model still lags behind the best model fine-tuned with T5-Base in Experiment 3a. This model has an 18.21% drop in EM and a 17.63% drop in EX compared to the best model evaluated in Experiment 3a. However, this model does relatively better than T5-Base under the same settings, in Experiment 4a. This model achieves an 8.63% increase in EM and a 5.73% increase in EX relative to the model in Experiment 4a.

Experiment 5: The results of this experiment suggest that the T5 model does not need the serialized schema at inference or both training and inference when it has already seen some representation of out-of-schema questions and their associated SQL queries during training. However, these models perform better than the models that only incorporate schema serialization at inference in Experiment 4. **Table 5.1** (a) shows that this model achieves an EM = 85.73% and an EX = 91.40%. The results show that this approach does not improve the performance of the best model in Experiment 3a. There is a marginal 3.92% decline in EM and a 2.06% drop in EX relative to the best model in Experiment 3a, which does not incorporate any schema items during training or inference. In contrast, this model, in Experiment 5a, does relatively better than the model in Experiment 4a, which only sees schema items at inference. The results show that there is a 22.92% increase in EM and a 21.3% increase in EX relative to the model in Experiment 4a. In **Table 5.1** (b), we observe that this model achieves an EM = 85.95% and an EX = 91.09%. The results show that this approach does not improve the performance relative to the model in Experiment 3b or the best overall model in Experiment 3a. However, this model improves over Experiment 4b, and achieves comparable performance to the model in Experiment 5a. The model obtains a marginal 1.42% drop in EM and a 1.06% drop in EX relative to Experiment 3b. This model also has a 3.7% performance drop in EM and a 2.37% performance drop in EX relative to the best model in Experiment 3a. In contrast, the model achieves a 14.51% increase in EM and a 15.26% increase in EX relative to Experiment 4b. Finally, this model achieves comparable performance relative to Experiment 5a, with a 0.22% increase in EM and a 0.31% decline in EX relative to Experiment 5a.

Experiment 6: The results of this experiment suggest that the schema pruning strategy does considerably worse than schema serialization introduced at inference or both training and inference in Experiment 4a and

Experiment 5a, respectively. **Table 5.1** (a) shows that this model achieves an EM = 0% and an EX = 0%. Examples of the NLQ + pruned schema are shown below. In these examples, we see that the pruned schema contains relevant table and column names, however, the model is still unable to use this information effectively. Manual inspection of the SQL predictions, show that the model predicts incorrect table names and column names.

- What are the drug tradenames of chlorothiazide sodium?
  DRUG_TRADENAMES: CUI_CODE, ROW_ID, SNOMED_CODE, TRADENAMES, MESH_CODE, RXNORM_CODE, DRUG_NAME
  DRUG_SIDE_EFFECTS: CUI_CODE, SIDE_EFFECTS, DRUG_NAME, ROW_ID
  MEDICATION_TO_DIAGNOSIS: CUI_CODE, ROW_ID, DRUG_INFORMATION, PREVENTION_OF, TREATMENT_OF, DRUG_NAME
- What medication is used to treat the condition referred to as purpura, thrombocytopenic, idiopathic?
  DIAGNOSIS_TO_MEDICATION: DIAGNOSIS, ROW_ID, MEDICATION_TREATMENT, MEDICATION_PREVENTION ICD10CM_DIAGNOSIS: CUI_CODE, ROW_ID, ICD10CM_CODE, DIAGNOSIS_LONG_TITLE, DIAGNOSIS_SHORT_TITLE
  DRUG_TRADENAMES: CUI_CODE, ROW_ID, SNOMED_CODE, TRADENAMES, MESH_CODE, RXNORM_CODE, DRUG_NAME
- What is the ICD-10 code for the diagnosis injury axillary nerve?
  ICD10CM_DIAGNOSIS: CUI_CODE, ROW_ID, ICD10CM_CODE, DIAGNOSIS_LONG_TITLE, DIAGNOSIS_SHORT_TITLE
  DRUG_SIDE_EFFECTS: CUI_CODE, SIDE_EFFECTS, DRUG_NAME, ROW_ID
  DRUG_TRADENAMES: CUI_CODE, ROW_ID, SNOMED_CODE, TRADENAMES, MESH_CODE, RXNORM_CODE, DRUG_NAME

Additional Analysis

Fine-Grain Analysis of EM and EX: **Table C.1** in Appendix C shows the fine-grain EM and EX results for the models evaluated on the synthetic reference questions (n^{valid} = 1607) and the paraphrases (n^{valid} = 600), respectively. Similar to the model performance reported in **Table 5.1**, the model trained in Experiment 3a outperforms all other models when evaluated on the synthetic reference questions and paraphrases.

EM and EX Analysis for Model Generalization: We assess model generalization of the best model (in Experiment 3a) on the paraphrases relative to the synthetic reference questions. The exact matching accuracies (EM) and execution accuracies (EX) are reported in **Table 5.2**. Our evaluation is based on a sample of validation pairs (n^{valid} = 302 samples), where each sample consists of a synthetic reference question and its associated paraphrase. The results show that the model generalizes well on the question paraphrases, when considering EX. The model obtains an EX = 98.01% when evaluated on the synthetic reference questions and an EX = 96.03% when evaluated on the paraphrases. In contrast, there is a marginal decline for the EM of the paraphrases, where EM = 87.09% for the paraphrases relative to the synthetic reference questions with an EM = 94.7%. These results suggest that the model in Experiment 3a accurately predicts SQL queries for reference questions and paraphrases in most cases.

TABLE 5.2: Out-of-Schema Model Generalization: Exact Matching (EM) Accuracies and Execution (EX) Accuracies Evaluated on Synthetic Reference Question and Paraphrase Pairs (n^{valid} = 302 pairs).

Out-of-Schema Question Group	EM (%)	EX (%)
Synthetic Reference Questions	94.7	98.01
Paraphrases	87.09	96.03

CM Analysis for Model Generalization: We assess model generalization by comparing the component matching accuracies (CM) of the model when evaluated on the synthetic reference questions relative to that of the paraphrases (see **Table 5.3**). This analysis suggests that the model generalizes to paraphrases when making table predictions (in the *SELECT* clause) and column predictions (in the *FROM* clause) in the predicted SQL query. In addition, the model has satisfactory performance when predicting values (in the *WHERE* clause), given a question paraphrase. Our evaluation is based on the sample of validation pairs (n^{valid} = 302 samples), where each sample consists of a synthetic reference question and paraphrase pair. We use the best model in Experiment 3a to make SQL predictions. The results show that the model has similar performance for the synthetic reference questions and paraphrases when considering the *SELECT* and *FROM* clauses, respectively. The model achieves the following results on the synthetic reference questions: CM^{SELECT} = 100% and CM^{FROM} = 100%. The model achieves the following results on the paraphrases: CM^{SELECT} = 97.35% and CM^{FROM} = 99.67%. However, the CM of the *WHERE* clause has a marginal decline for the paraphrases (CM^{WHERE} = 89.40%) relative to the synthetic reference questions (CM^{WHERE} = 94.7%). The overall comparison of CM, EM and EX collectively suggests that the best model (in Experiment 3a) achieves generalization when encountering an out-of-schema question.

TABLE 5.3: Out-of-Schema Model Generalization: Component Matching (CM) Accuracies Evaluated on Synthetic Reference Question and Paraphrase Pairs (n^{valid} = 302 pairs). The *SELECT*, *FROM* and *WHERE* columns represent the CM percentages for the *SELECT*, *FROM* and *WHERE* clauses, respectively.

Out-of-Schema Question Group	SELECT (%)	FROM (%)	WHERE (%)
Synthetic Reference Questions	100	100	94.7
Paraphrases	97.35	99.67	89.40

5.3.2 RQ2: Unanswerable Questions Requiring Medical Knowledge

This section investigates the research question *RQ2*, as seen below.

RQ2: How can medical knowledge sources related to diagnoses and medication terminologies be utilized in data augmentation to simplify complex medical jargon in unanswerable questions?

Details of the experiments can be seen in Section 4.5.2.

Main Findings

Table 5.4 shows the performance of the T5 models (T5-Base and T5-Large) when evaluated on the validation unanswerable questions requiring medical knowledge (n^{valid} = 955). These questions include the value-synonym replacement questions which contain complex medical jargon that do not explicitly match to column values in the schema. The results show that the medical knowledge sources (namely, SNOMED CT and RxNorm) related to diagnoses and medication improve (T5) model performance relative to the vanilla models without medical knowledge augmentation. This trend is observed for both the RAG and SQL post-processing augmentation strategies when compared to the model that employs SQL post-processing with synonyms derived from MeSH definitions. More specifically, we find that the T5-Large with RAG model outperforms all other models.

TABLE 5.4: Text-to-SQL Model Performance on the Unanswerable Validation Questions Requiring Medical Knowledge. The exact matching (EM) and execution (EX) accuracies are reported for T5-Base and T5-Large models with various data augmentation strategies. The best EM and EX values are highlighted in blue.

Model ID	EM (%)	EX (%)
A. T5-Base		
Experiment 7a: Vanilla	43.93	60.12
Experiment 8a: SQL Post-Processing with synonyms obtained from MeSH definitions	34.72	50.13
Experiment 9a: SQL Post-Processing with synonyms obtained from SNOMED CT and RxNorm	21.23	66.38
Experiment 10a: RAG	45.64	65.4
B. T5-Large		
Experiment 7b: Vanilla	23.07	43.44
Experiment 8b: SQL Post-Processing with synonyms obtained from MeSH definitions	19.39	39.51
Experiment 9b: SQL Post-Processing with synonyms obtained from SNOMED CT and RxNorm	11.66	60.61
Experiment 10b: RAG	49.2	67.61

Analysis

Experiment 7: The vanilla models serve as a baseline for models fine-tuned in subsequent experiments. **Table 5.4** (a) shows that this model achieves an EM = 43.93% and an EX = 60.12%, when evaluated on the unanswerable validation questions requiring medical knowledge. **Table 5.4** (b) shows that this model achieves an EM = 23.07% and an EX = 43.44%, when evaluated on the unanswerable validation questions requiring medical knowledge. The vanilla model serves as a baseline for subsequent models fine-tuned using T5-Large. The model obtains a 20.86% decline in EM and a 16.68% decline in EX relative to the T5-Base vanilla model in Experiment 7a. This suggests that a larger T5 model, does not improve the baseline performance.

Experiment 8: **Table 5.4** (a) shows that this model achieves an EM = 34.72% and an EX = 50.13%, when evaluated on the unanswerable validation questions requiring medical knowledge. This model experiences a 9.21% decline in EM and a 9.99% decline in EX when compared to the vanilla model in Experiment 7a. The decline in EM is expected for SQL post-processing, which modifies the predicted SQL queries, deviating from the ground-truth SQL queries. The decline in EX, however, suggests that the synonyms obtained by matching MeSH definitions are not adequate for improving the answerability of unanswerable questions containing complex, medical jargon. **Table 5.4** (b) shows that this model achieves an EM = 19.39% and an EX = 39.51%, when evaluated on the unanswerable validation questions requiring medical knowledge. Similar to Experiment 8a, the model has a performance decline relative to the vanilla model (T5-Base in Experiment 7a). The model obtains a 3.68% decline in EM and a 3.93% decline in EX relative to the vanilla model in Experiment 7b. The decline in EX supports the findings of Experiment 8a that MeSH definitions are unsuitable for this task.

Experiment 9: **Table 5.4** (a) shows that this model achieves an EM = 21.23% and an EX = 66.38%, when evaluated on the unanswerable validation questions requiring medical knowledge. This model achieves the best EX relative to all other T5-Base models evaluated on the unanswerable validation questions that require additional medical knowledge. The model obtains a 22.7% decline in EM and a 6.26% increase in EX, relative to the vanilla model in Experiment 7a. In addition, the model obtains a 13.49% decline in EM and a 16.25% increase in EX, relative to the alternate SQL post-processing strategy in Experiment 8a. As discussed in the results of Experiment 8a, the decline in EM is expected for SQL post-processing. However, the increase in EX relative to Experiment 7a and Experiment 8a, suggests that SQL post-processing with medical synonyms found in the SNOMED CT and RxNorm medical vocabularies improves the answerability of questions containing complex medical jargon. **Table 5.4** (b) shows that this model achieves an EM = 11.66% and an EX = 60.61%, when evaluated on the unanswerable validation questions requiring medical knowledge. The model obtains a 11.41% decline in EM and a 17.17% increase in EX relative to the vanilla model in Experiment 7b. In addition, the model obtains a 7.73% decline in EM and a 21.1% increase in EX relative to Experiment 8b. We also see that this model achieves a marginally greater EX (a 0.49% increase in EX) relative to the T5-Base vanilla model in Experiment 7a. However, the model still lags behind Experiment 9a, which fine-tunes a smaller model (T5-Base) under the same settings. The model obtains a 9.57% decline in EM and a 5.77% decline in EX relative to Experiment 9a. As mentioned before in Experiment 8b, the decline in EM is expected for an SQL post-processing strategy that modifies the predicted SQL queries, deviating from the ground-truth SQL queries. These results suggest that while this augmentation strategy improves EX relative to the prior T5-Large models, the performance still lags behind a smaller model under the same settings.

Experiment 10: The results of this experiment suggest that RAG enhances model performance compared to the vanilla models, while a larger T5 model further improves performance when coupled with RAG. **Table 5.4** (a) shows that this model achieves an EM = 45.64% and an EX = 65.4%, when evaluated on the unanswerable validation questions requiring medical knowledge. This model achieves the best EM compared to all other T5-Base models evaluated on the unanswerable validation questions. The model obtains a 1.71% increase in EM and a 5.28% increase in EX, relative to the vanilla model in Experiment 7a. In addition, the model obtains a 10.92% increase in EM and a 15.27% increase in EX, relative to Experiment 8a. The model also obtains a 24.41% increase in EM and a 0.98% decrease in EX, relative to Experiment 9a. These results suggest a consistent increase in EM and an increase in EX relative to most T5-Base models. **Table 5.4** (b) shows that this model achieves an EM = 49.2% and an EX = 67.61%, when evaluated on the unanswerable validation questions requiring medical knowledge. This model achieves the best EM and EX results among all models, including T5-Base and T5-Large models. In addition, this model achieves a 5.27% and a 26.13% increase in EM relative to the vanilla models in Experiment 7a (T5-Base) and Experiment 7b (T5-Large), respectively. In addition, the model achieves a 7.49% and a 24.17% increase in EX relative to the vanilla models in Experiment 7a (T5-Base) and Experiment 7b (T5-Large), respectively. This model also achieves an increase of 29.81% in EM and a 28.1%

increase in EX compared to Experiment 8b, as well as a 37.54% increase in EM and a 7% increase in EX relative to Experiment 9b. Finally, this model surpasses the model in Experiment 10a, with a smaller model (T5-Base), fine-tuned under the same settings. Here the model obtains a 3.56% increase in EM and a 2.21% increase in EX relative to Experiment 10a.

Additional Analysis

We then assess the component matching accuracies (CM) of the various models evaluated on the unanswerable validation questions requiring medical knowledge. See **Table 5.5** for the CM results for the *SELECT*, *FROM* and *WHERE* clauses, respectively. This analysis confirms that the increase in model performance is attributed to the additional knowledge given to the model. Consistent with the results in **Table 5.4**, we see that Experiment 10b (T5-Large + RAG) has the best CM performance for all components (the *SELECT*, *FROM* and *WHERE* clauses) across all models. The models in the experiments have similar CM for the *SELECT* and *FROM* clauses, suggesting that these augmentation strategies have minimal effect on the model's prediction of the table and column names. However, the CM for the *WHERE* clause varies across the models. More specifically, we see that for the SQL post-processing experiments, Experiments 8a, 9a, 8b and 9b, have poor CM for the *WHERE* clause relative to all other models. This is expected for SQL post-processing strategies which modifies the *WHERE* clause of the predicted SQL query, deviating from the ground-truth SQL query. On the other hand, Experiment 10b has the best CM overall, with a CM for the *WHERE* clause of $CM^{WHERE} = 57.55\%$. Experiment 10b achieves a 32.4% increase in CM^{WHERE} relative to the vanilla model (T5-Large) in Experiment 7b ($CM^{WHERE} = 25.15$), and a 6.75% increase in CM^{WHERE} relative to the vanilla model (T5-Base) in Experiment 7a ($CM^{WHERE} = 50.80\%$). In addition, the model in Experiment 10b has a 2.58% increase in CM^{WHERE} relative to T5-Base, under the same settings (Experiment 10a, with a $CM^{WHERE} = 54.97\%$).

TABLE 5.5: Text-to-SQL Model Performance on the Unanswerable Validation Questions Requiring Medical Knowledge. The component matching (CM) accuracies are reported for T5-Base and T5-Large models with various data augmentation strategies. The *SELECT*, *FROM* and *WHERE* columns represent the CM percentages for the *SELECT*, *FROM* and *WHERE* clauses, respectively. The best CM values are highlighted in blue.

Model ID	SELECT (%)	FROM (%)	WHERE (%)
A. T5-Base			
Experiment 7a: Vanilla	86.01	86.01	50.80
Experiment 8a: SQL Post-Processing with synonyms obtained from MeSH definitions	86.01	86.01	39.88
Experiment 9a: SQL Post-Processing with synonyms obtained from SNOMED CT and RxNorm	86.01	86.01	22.94
Experiment 10a: RAG	84.66	84.66	54.97
B. T5-Large			
Experiment 7b: Vanilla	85.89	85.89	25.15
Experiment 8b: SQL Post-Processing with synonyms obtained from MeSH definitions	85.89	85.89	20.61
Experiment 9b: Post-Processing with synonyms obtained from SNOMED CT and RxNorm	85.89	85.89	11.66
Experiment 10b: RAG	87.61	87.61	57.55

5.4 Testing

In this section, we evaluate the best models on the test questions (n^{test} = 6645). Section 5.4.1 presents the results for the best model (in Experiment 3a) evaluated on the out-of-schema test questions (*RQ1*) and Section 5.4.2 presents the results for the best model (in Experiment 10b) evaluated on the unanswerable test questions requiring medical knowledge (*RQ1*). For each research question, we report the results for the unanswerable groups (reference questions, paraphrases or value-synonym replacement questions). In addition, we also evaluate if the model constraints are maintained. We report the performance when evaluated on the original answerable questions in MIMIC-III and eICU. We also report the results before and after answer abstention is applied. Here we employ the abstention threshold obtained from the K-means clustering algorithm, where the classification threshold is p = 1.32. Finally we perform error analyses of the predictions after abstention is applied.

5.4.1 RQ1

Out-of-Schema Questions

Table 5.6 presents the results for the best model fine-tuned in Experiment 3a, when evaluated on the out-of-schema test questions (n^{test} = 2231), including the synthetic reference questions (n^{test} = 1615) and the paraphrases (n^{test} = 616). **Table 5.6** shows (a) the results before answer abstention, and (b) the results after abstention is applied. The values in brackets show the increase in performance relative to the baseline model in Experiment 1a. The results in the table present three trends: (i) the model fine-tuned in Experiment 3a shows significant improvements relative to that of the baseline model, (ii) there is a minimal performance difference when the model is evaluated on the reference questions and paraphrases in the test set, suggesting model generalization and (iii) there is a minimal decline in performance after answer abstention is applied, suggesting that the abstention model usually makes the correct decision in answering out-of-schema questions (that have been resolved with additional knowledge). More specifically, the model in Experiment 3a achieves the following improvements relative to the baseline model, before answer abstention in (a): EM = +88.9% and EX = +93.65% when evaluated on the out-of-schema test questions, with an EM = +92.24% and EX = +95.58% for the reference questions and an EM = +85.55% and EX = +91.72% for the paraphrases. The model achieves the following improvements relative to the baseline model, after answer abstention in (b): EM = +88.15% and EX = +91.8% when evaluated on the out-of-schema test questions, with an EM = +91.88% and EX = +93.18% for the reference questions and EM = +84.42% and EX = +90.42% for the paraphrases.

Answerable Questions

Table 5.6 also presents the results for the best model, in Experiment 3a, when evaluated on the answerable test questions (n^{test} = 2401), including the MIMIC-III answerable test questions (n^{test} = 1198) and the eICU answerable test questions (n^{test} = 1203). Similar to the out-of-schema questions, we observe significant improvements relative to the baseline model, when the model in Experiment 3a is evaluated on the answerable test questions. In contrast, we see a marginal decline in performance after answer abstention is applied. More specifically, the model in Experiment 3a achieves the following improvements relative to the baseline model, before answer abstention in (a): EM = +55.37% and EX = +53.98% when evaluated on the answerable test questions, with an EM = +53.53% and EX = +54.26% for MIMIC-III and EM = +57.21% and EX = 53.7% for eICU. The model then achieves the following improvements relative to the baseline model, after answer abstention in (b): EM = +59.52% and EX = +61.81% when evaluated on the answerable test questions, with an EM = +56.76% and EX = +61.68% for MIMIC-III and EM = +62.27% and EX = +61.93% for eICU. As a further note, the baseline performance is low because the model is evaluated on an unseen dataset, with a different distribution to the training data. However, it is the change in performance we are interested in.

TABLE 5.6: Text-to-SQL Model Performance on the Out-of-Schema Test Questions. The exact matching (EM) and execution (EX) accuracies are reported for the model in Experiment 3a. The grey headings in the table indicate the metrics (a) before answer abstention is applied to the predictions and (b) after answer abstention is applied. The green metrics report the increase in performance relative to the baseline model in Experiment 1a. The bold font *Out-of-Schema* represents the average performance of the model evaluated on the reference questions and paraphrases. The bold font *Answerable* represents the average performance of the model evaluated on the answerable questions in MIMIC-III and eICU.

Question Group	n^{test}	EM (%)	EX (%)
A. Before Abstention			
Out-of-Schema	**2231**	**88.9 (+88.9)**	**93.65 (+93.65)**
Reference Questions	1615	92.24 (+92.24)	95.58 (+95.58)
Paraphrases	616	85.55 (+85.55)	91.72 (+91.72)
Answerable	**2401**	**75.71 (+55.37)**	**84.58 (+53.98)**
MIMIC-III Answerable	1198	68.61 (+53.53)	78.05 (+54.26)
eICU Answerable	1203	82.81 (+57.21)	91.11 (+53.7)
B. After Abstention			
Out-of-Schema	**2231**	**88.15 (+88.15)**	**91.8 (+91.8)**
Reference Questions	1615	91.88 (+91.88)	93.18 (+93.18)
Paraphrases	616	84.42 (+84.42)	90.42 (+90.42)
Answerable	**2401**	**75.46 (+59.52)**	**84.16 (+61.81)**
MIMIC-III Answerable	1198	68.53 (+56.76)	77.71 (+61.68)
eICU Answerable	1203	82.39 (+62.27)	90.61 (+61.93)

Additional Analysis

We also report the CM accuracies to determine which components of the SQL query the model accurately predicts, while inferring the components it does not do so well in predicting. We only report the CM results before answer abstention, which is the same as after abstention. **Table 5.7** shows that the model does relatively well in predicting all SQL components, including column predictions for the *SELECT* clause, table predictions for the *FROM* clause and value predictions for the *WHERE* clause.

TABLE 5.7: Component Matching Accuracies Evaluated on the Out-of-Schema Test Questions, Using the Model in Experiment 3a. The *SELECT*, *FROM* and *WHERE* columns represent the CM percentages for the *SELECT*, *FROM* and *WHERE* clauses, respectively. The first row in bold, represents the average CM for the out-of-schema questions.

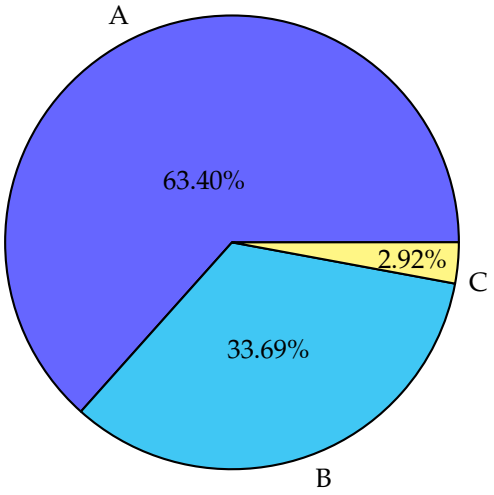
Out-of-Schema Question Group	n^{test}	SELECT (%)	FROM (%)	WHERE (%)
Out-of-Schema Questions	**2231**	**98.97**	**99.86**	**90.82**
Synthetic Reference Questions	1615	99.88	99.88	94.30
Paraphrases	616	98.05	99.84	87.34

Error Analysis

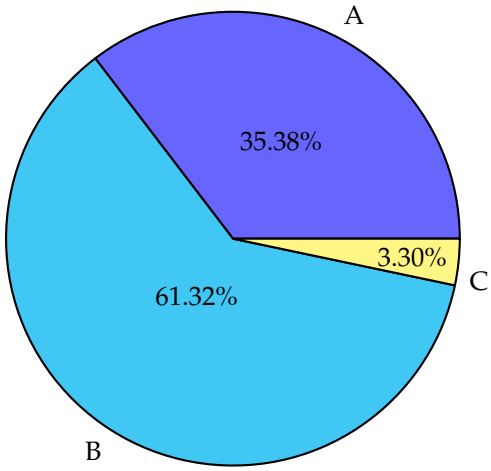
In **Figure 5.1**, we perform error analysis on the SQL predictions, derived from the best model in Experiment 3a (after answer abstention). We assess the percentage of incorrect SQL predictions for each dataset. The SQL predictions are categorized into three categories of errors: (a) inexecutable SQL predictions, (b) SQL predictions with incorrect table names and/or column names and/or values, and (c) SQL predictions that are incorrectly abstained, i.e., incorrectly classified as unanswerable. Below we list some examples of SQL prediction errors. We find that for inexecutable SQL queries, the SQL queries have incorrect SQL syntax, and non-exist tables or columns. See examples below.

1. **Inexecutable SQL Prediction** (non-existent column names):
   (a) **Question**: What is the cost for the procedure oth transureth prostatec?
   (b) **Ground-Truth SQL**:
   SELECT DISTINCT cost.cost FROM cost WHERE cost.event_type = 'procedures_icd' AND cost.event_id IN (SELECT procedures_icd.row_id FROM procedures_icd WHERE procedures_icd.icd9_code = (SELECT d_icd_procedures.icd9_code FROM d_icd_procedures WHERE d_icd_procedures.short_title = 'oth transureth prostatec'))
   (c) **Predicted SQL**:
   SELECT cost.cost FROM cost WHERE cost.eventtype = 'procedures_icd' AND cost.eventid IN (SELECT procedures_icd.row_id FROM procedures_icd WHERE procedures_icd.icd9_code = (SELECT d_icd_procedures.icd9_code FROM d_icd_procedures WHERE d_icd_procedures.short_title = 'oth transureth prostatec'))
2. **Incorrect Tables and/or Columns and/or Values**:
   (a) **Question**: What are the drug tradenames for carrington dermal wound?
   (b) **Ground-Truth SQL**:
   SELECT tradenames FROM drug_tradenames WHERE drug_name = "carrington dermal wound" COLLATE NOCASE
   (c) **Predicted SQL**:
   SELECT treatment_of FROM medication_to_diagnosis WHERE drug_name = "carrington dermal wound" COLLATE NOCASE
3. **Incorrect Answer Abstention**:
   (a) **Question**: What are the trade names for the medication eplerenone?
   (b) **Ground-Truth SQL**: SELECT tradenames FROM drug_tradenames WHERE drug_name = "eplerenone"
   (c) **Prediction**: NULL

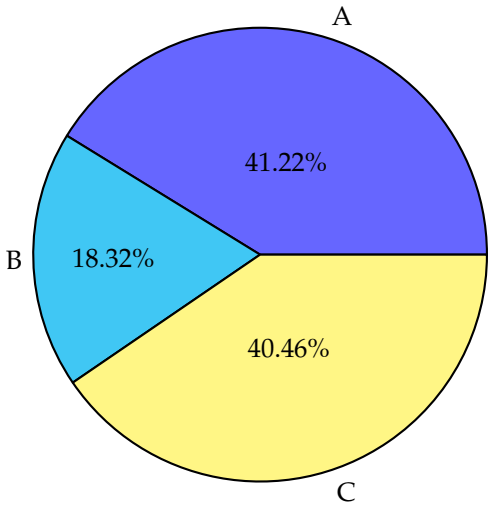
i. MIMIC-III Answerable
($n^{errors} = 377$)



ii. eICU Answerable
($n^{errors} = 212$)



iii. Resolved Out-of-Schema Questions: Synthetic Reference Questions
($n^{errors} = 131$)



iv. Resolved Out-of-Schema Questions: Paraphrases
($n^{errors} = 96$)

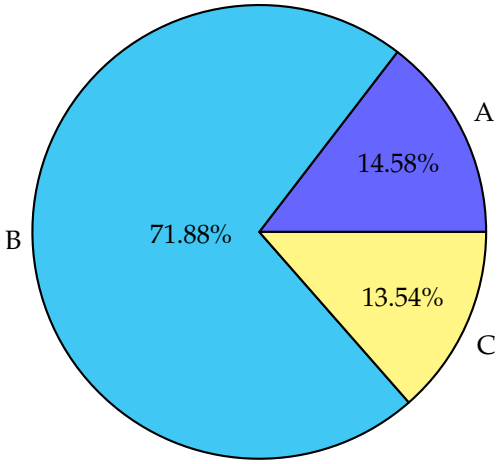


FIGURE 5.1: Distribution of SQL Prediction Errors Across the Different Datasets. SQL predictions are obtained from the model in Experiment 3a. We show the following error groups: (a) inexecutable SQL predictions, (b) incorrect table names and/or column names and/or values, and (c) SQL predictions that are incorrectly abstained.

5.4.2 RQ2

Unanswerable Questions Requiring Medical Knowledge

Table 5.8 presents the results for the best model fine-tuned in Experiment 10b (T5-Large + RAG), when evaluated on the unanswerable test questions requiring medical knowledge (n^{test} = 837). These questions include questions with value-synonym replacement. The **Table 5.8** shows (a) the results without answer abstention and (b) the results after abstention is applied. The values in brackets show the increase in performance relative to the baseline model in Experiment 7b, where no value in brackets suggest no change in performance. Red downward pointing arrows suggest a significant decrease in the performance after abstention. The results in the table present two trends: (i) T5-Large + RAG improves model performance (relative to the baseline in Experiment 7b) when evaluated on unanswerable questions containing complex medical jargon, and (ii) in some cases, the answer abstention method incorrectly abstains from answering unanswerable questions which have been resolved with medical knowledge. This is seen by the decrease in performance after answer abstention is applied. More specifically, the model in Experiment 10b achieves the following improvements relative to the baseline model: (a) before answer abstention, EM = +9.44% and EX = +9.68%, and (b) after answer abstention, EM = +7.64% and EX = +7.41% when evaluated on the unanswerable test questions that require additional medical knowledge.

TABLE 5.8: Text-to-SQL Model Performance on the Unanswerable Test Questions Requiring Medical Knowledge. The exact matching (EM) and execution (EX) accuracies are reported for the model in Experiment 10b. The grey headings in the table indicate the metrics (a) before answer abstention is applied to the predictions and (b) after answer abstention is applied. The green metrics in brackets report the increase in performance relative to the baseline model in Experiment 7b. No value in brackets suggest no change in performance. Red downward pointing arrows suggest a significant decrease in the performance after abstention. The bold font *Answerable* represents the average performance of the model evaluated on the answerable questions in MIMIC-III and eICU.

Question Group	n^{test}	EM (%)	EX (%)
A. Before Abstention			
Value-Synonym Replacement	837	32.5 (+9.44)	52.33 (+9.68)
Answerable	2401	35.51	32.37
MIMIC-III Answerable	1198	26.46	35.31
eICU Answerable	1203	44.56	61.43
B. After Abstention			
Value-Synonym Replacement	837	26.28 (+7.64) ↓	38.83 (+7.41) ↓
Answerable	2401	29.56 ↓	37.17 ↓
MIMIC-III Answerable	1198	20.62 ↓	26.29 ↓
eICU Answerable	1203	38.49↓	48.05 ↓

Answerable Questions

Table 5.8 also presents the results for the best model fine-tuned in Experiment 10b (T5-Large + RAG), when evaluated on the answerable test questions (n^{test} = 2401). These questions include answerable questions in the MIMIC-III (n^{test} = 1198) and eICU (n^{test} = 1203) datasets. The results in the table present two trends: (i) fine-tuning with RAG does not affect model predictions of answerable questions (gain of 0%) relative to the baseline in Experiment 7b, and (ii) similar to the unanswerable questions, the answer abstention method sometimes incorrectly abstains from answering answerable questions (shown by the red downward pointing arrows in part (b) of the table). As a further note, similar to *RQ1*, the baseline performance is low because the model is evaluated on an unseen dataset, with a different distribution to the training data. However, it is the change in performance we are interested in.

Additional Analysis

We also report the CM accuracies to determine which components of the SQL query the model accurately predicts, while inferring the components it does not do so well in predicting. We only report the CM results before answer abstention, which is the same for after answer abstention. **Table 5.9** shows that the model does relatively well in predicting columns for the *SELECT* clause and predicting tables for the *FROM* clause. However, the model does not do so well in predicting values in the *WHERE* clause. However, this result is encouraging, as we provide the results based on RAG with an exhaustive list of medical concepts and their related synonyms. This suggests that the results may improve with additional medical knowledge.

TABLE 5.9: Component Matching Accuracies Evaluated on Unanswerable Test Questions Requiring Medical Knowledge. We use the model in Experiment 10b for SQL predictions. The *SELECT*, *FROM* and *WHERE* columns represent the CM percentages for the *SELECT*, *FROM* and *WHERE* clauses, respectively.

Unanswerable Question Group	SELECT (%)	FROM (%)	WHERE (%)
Value-Synonym Replacement	85.09	86.02	36.80

Error Analysis

In **Figure 5.2**, we perform error analysis on the SQL predictions, derived from the best model in Experiment 10b (after answer abstention). We assess the percentage of incorrect SQL predictions for each dataset. The SQL predictions are categorized into three categories of errors: (a) inexecutable SQL predictions, (b) SQL predictions with incorrect table names and/or column names and/or values, and (c) SQL predictions that are incorrectly abstained, i.e., incorrectly classified as unanswerable. Below we list some examples of SQL prediction errors. We find that for inexecutable SQL queries, the SQL queries have incorrect SQL syntax and non-exist tables or columns. See examples below.

1. **Inexecutable SQL Prediction** (non-existent tables and columns):
   (a) **Question**: Tell me the method of fluconazole intake?
   (b) **Ground-Truth SQL**:
   SELECT DISTINCT prescriptions.route FROM prescriptions WHEREprescriptions.drug = 'fluconazole'
   (c) **Prediction**: SELECT DISTINCT medication.routeadmin FROM medication WHERE medication.drugname = 'fluconazole'",
2. **Incorrect Tables and/or Columns and/or Values**:
   (a) **Question**: Which medication can I use to treat the patient's congenital tricuspid atresia and stenosis (disorder)?
   (b) **Ground-Truth SQL**:
   SELECT medication_treatment FROM diagnosis_to_medication WHERE diagnosis = "Tricuspid Atresia" COLLATE NOCASE
   (c) **Predicted SQL**:
   SELECT medication_treatment FROM diagnosis_to_medication WHERE diagnosis = "tricuspid atresia and stenosis" COLLATE NOCASE
3. **Incorrect Answer Abstention**:
   (a) **Question**: What are the side effects of taking antirobe?
   (b) **Ground-Truth SQL**:
   SELECT side_effects FROM drug_side_effects WHERE drug_name = "clindamycin" COLLATE NOCASE
   (c) **Prediction**: NULL

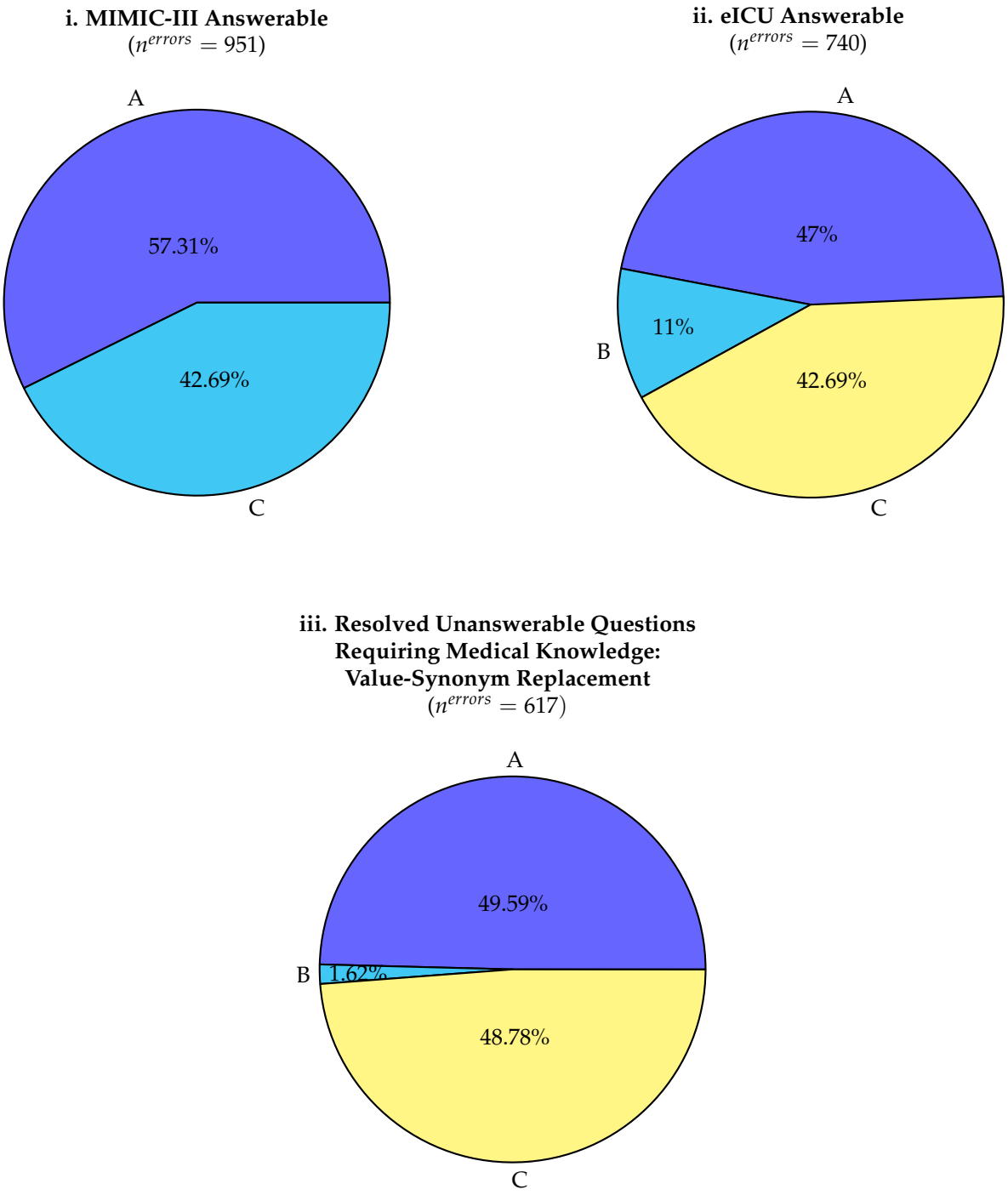


FIGURE 5.2: Distribution of SQL Prediction Errors Across the Different Datasets. SQL predictions are obtained from the model in Experiment 10b. We show the following error groups: (a) inexecutable SQL predictions, (b) incorrect table names and/or column names and/or values, and (c) SQL predictions that are incorrectly abstained.

5.5 Summary

In this section we evaluate the research questions *RQ1* and *RQ2* to find the best models for resolving out-of-schema questions, for the former, and unanswerable questions requiring medical knowledge, for the latter. This evaluation is done on the validation questions. For *RQ1*, we find that the best model, T5-Base (in Experiment 3a), leverages an augmentation strategy that involves: (i) identifying unanswerable questions, (ii) augmenting an additional schema and (iii) including these out-of-schema question and SQL pairs in model training. The model then learns about the new schema from the question to SQL mappings. For *RQ2*, we find that the best model, T5-Large (Experiment 10b), leverages RAG with additional medical knowledge from SNOMED CT (for diagnoses) and RxNorm (for medication). The model is provided with additional context during training and inference. We then evaluate these models on the test questions and find that the models generalize well to unseen questions. In addition, we find that these models either improve in model performance when evaluated on the original answerable questions (for *RQ1*) or have no effect on their answerability (for *RQ2*). We then find that the answer abstention strategy usually correctly classifies answerable questions as answerable, as well as unanswerable questions that have been resolved with medical knowledge. However, there are some cases where the model incorrectly classifies answerable and resolved unanswerable questions as unanswerable. Finally, error analyses suggest that both models either predict: (i) inexecutable SQL queries, (ii) executable SQL queries with incorrect table names and/or column names and/or values or (iii) incorrectly abstains from answering a question. We discuss these results in more detail in Chapter 6.

Chapter 6

Discussion

6.1 Overview

In Chapter 5, we present the findings from our experiments focused on addressing out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*). These experiments use knowledge augmentation strategies that draw on medical knowledge from reliable sources. In this chapter, we explore the key findings of the experiments, including model generalization, model constraints, error analysis, as well as limitations and directions for future research.

6.2 Context and Approach

Current research on medical text-to-SQL parsing present a parsing challenge, where the model generates an incorrect or inexecutable SQL query when an unanswerable question is presented to the model (Chang et al., 2023; Gan et al., 2021a, 2021b; Lee et al., 2022; Li et al., 2023b). These unanswerable questions usually contain out-of-schema concepts not present in the existing schema, or complex medical concepts that need to be simplified so the model can map them to concepts in the existing schema. As a result of these challenges, medical professionals cannot fully rely on text-to-SQL systems to gain insights that inform their decision-making.

This research thesis investigates knowledge augmentation strategies to address these challenges. We focus on addressing two categories of unanswerable questions, including out-of-schema questions (which consist of reference questions and paraphrases) (*RQ1*) and unanswerable questions that require medical knowledge (which include questions with value-synonym replacement) (*RQ2*). In doing so, we expand the coverage of questions answered by the text-to-SQL model. We also implement two model constraints. First, we improve model reliability by allowing the model to only answer questions it is confident in answering. Second, we prevent the model from becoming biased toward unanswerable questions, ensuring reliable performance when it encounters one of the original answerable questions. Based on these objectives, we discuss the findings for *RQ1* and *RQ2* below.

6.3 RQ1: Out-of-Schema Questions

RQ1: How does the integration of an *unseen* schema during inference enhance the performance of a sequence-to-sequence text-to-SQL model in answering out-of-schema unanswerable questions, and how does this approach compare to the effectiveness of fine-tuning on a relevant training dataset that includes these out-of-schema questions and the *unseen* schema?

6.3.1 Key Findings

In addressing the research question *RQ1*, which focuses on out-of-schema questions, we summarize three key findings based on the evaluation of the models on the validation set:

1. *Training on Out-of-Schema Representations*: The optimal T5 model for predicting correct SQL queries (given out-of-schema questions) requires training on out-of-schema question representations and their corresponding SQL queries. In contrast, incorporating the entire or pruned serialized schema in the model input (either at inference or both inference and training) deteriorates performance.

2. *Impact of Schema Serialization*: While schema serialization at both training and inference yields suboptimal performance relative to the best model (which does not include schema serialization), we see notable improvements compared to models that only use schema serialization during inference and those that apply schema pruning during both training and inference.

3. *Impact of Model Size*: T5-Base is sufficient for resolving out-of-schema questions, and a larger model (T5-Large) does not provide significant advantages in performance.

Refer to **Table 5.1** for the validation results of the models evaluated on out-of-schema questions, and **Table C.1** for detailed results on the synthetic reference questions and their paraphrases.

6.3.2 Findings Interpretation

We dissect the results by addressing why T5-Base in Experiment 3a outperforms other models. These other models include baseline models (Experiments 1a and 1b) which exclude out-of-schema questions during training, including other suboptimal models that include out-of-schema questions in their training process. In addition, these models either use the entire serialized schema or a pruned schema in their input, either during inference only or both training and inference. The difference in validation performance resulting from these model configurations are attributed to: (i) the sequence-based decoder (Wang et al., 2018; Zhang et al., 2019; Zhong et al., 2017), (ii) the model's inherent limitations (e.g., context length and model scale) (Raffel et al., 2020), (iii) the limitations of schema pruning (otherwise known as schema linking) (Maamari et al., 2024) and (iv) the training data size. Below we discuss how these factors relate to the key findings.

Training on Out-of-Schema Representations

Our experiments suggest that training with out-of-schema questions improves the model's ability to generalize to new out-of-schema questions. The model does not need additional schema information at training or inference. Similar to Zhong et al. (2017), the performance of the best model is attributed to the *sequence-based decoder*. The sequence-based decoder employs one of the simplest decoding strategies relative to other methods such as the grammar-based decoder and the template-based decoder (Katsogiannis-Meimarakis and Koutrika, 2023). The sequence-based decoder predicts an SQL query, in a generative manner, as a sequence of tokens based on their co-occurrence probabilities in the training data. As a result of this decoding strategy, our T5 models perform best when the inputs at inference closely resemble the inputs and associated SQL queries in the training distribution. This explains the poor results of the models in Experiments 2a and 2b (comparable to the baseline models with EM = 0% and EX = 0%), which exclude out-of-schema questions during training but incorporate the serialized schema at inference. At inference, these models do not effectively leverage the new table and column names that are not seen during model training. In contrast, the models in Experiment 3a (EM = 89.65% and EX = 93.46%) and Experiment 3b (EM = 87.37% and EX = 92.15%) leverage prior exposure to out-of-schema questions during model training, where questions are both in their original and paraphrased forms. This strategy enables the model to learn associations between question tokens and SQL queries (including SQL keywords, table names, column names, and values). While some research suggests that sequence-based decoders may generate syntactically incorrect SQL queries, due to their non-constrained generative approach (Katsogiannis-Meimarakis and Koutrika, 2023), the optimal model in Experiment 3a presents favourable results attributed to the high-quality SQL queries in the training data, which the model learns from.

Impact of Schema Serialization

We find that for the T5 model to utilize the additional schema context effectively, the model must see the schema during training to leverage the schema information at inference. Experiment 5a (EM = 85.73% and EX = 91.4%) and Experiment 5b (EM = 85.95% and EX = 91.09%) demonstrate that models fine-tuned with out-of-schema questions and serialized schemas outperform the models in Experiment 4a (EM = 62.81% and EX = 70.1%) and Experiment 4b (EM = 71.44% and EX = 75.83%), respectively, also fine-tuned with out-of-schema questions, but only include serialized schemas at inference. This result is similar to that of Rangan and Yin (2024), which shows that model performance improves with additional context given at training. Despite these improvements, however, the performance of the models in Experiments 5a and 5b still lag behind the optimal model that does not leverage schema serialization. This is in contrast to Lee et al. (2022), which show that schema serialization does not significantly affect performance compared to the model without it. The discrepancy in performance is a result of the T5 model which requires a *larger training set* for each question group. In addition to the original questions related to MIMIC-III and eICU, our experiments also include questions

related to the new schema. The model requires better representation of these question groups during training to effectively learn how these questions relate to the serialized schemas. In addition, the discrepancy in results are also attributed to the *long context length*, where the input includes the question and the entire serialized schema, including all tables and columns from the MIMIC-III (Johnson et al., 2016), eICU (Pollard et al., 2018) or the augmented schemas. This introduces a challenge when the input sequence exceeds the maximum context length of T5 (512 tokens) (Raffel et al., 2020). Where this is not the case, the serialized schema includes relevant schema items (e.g., tables and columns) to a given question, as well as irrelevant schema items, resulting in incorrect SQL predictions with incorrect table and column names as a result of the irrelevant schema items (Maamari et al., 2024).

Schema pruning was introduced to include the most relevant schema items to a given question in the model input, thereby reducing the context length. Contrary to the findings by Li et al. (2023a) which show that schema pruning can enhance text-to-SQL performance (specifically for T5 models), our results indicate that T5 models with pruned schemas in the input (Experiment 6a, EM = 0% and EX = 0%) perform relatively worse than those with the entire serialized schemas in the input—see Experiments 4a, 4b, 5a and 5b. According to Maamari et al. (2024), one hypothesis is that schema pruning risks the inclusion of irrelevant schema items in the input sequence or the exclusion of relevant schema items. Inspection of the NLQ and pruned schema, however, shows that this is typically not the case. Our analysis shows that the model predicts incorrect table and column names, despite having relevant schema items in the input sequence. We hypothesize that by providing the entire serialized schema during training, the model expects the same serialized schema for each input question at inference. The model learns what the relevant tables and columns are for a given question during training. Conversely, for schema pruning, different question types get different pruned schemas appended to the input question. As a result, the model cannot learn consistent patterns in the training data. These results are in agreement with empirical evidence by Maamari et al. (2024), which shows that the text-to-SQL model does not need schema pruning, but rather the model is capable of identifying relevant schema items given the entire schema as input. Notably, while Maamari et al. (2024) employs larger language models such as GPT (Achiam et al., 2023), our study observes a similar finding but utilizes a smaller language model, namely, T5.

Impact of Model Size

Among the optimal models, we find that T5-Base in Experiment 3a outperforms T5-Large in Experiment 3b when evaluated on the out-of-schema validation questions. This is in contrast to the results reported by Raffel et al. (2020), which introduced the T5 architecture. Raffel et al. (2020) generally demonstrates an increase in model performance with increasing T5 model size. Although unexpected, our finding is observed in some NLP literature. Aoki et al. (2023), for example, show that T5-Base outperforms T5-Large in question-answering reasoning tasks that generate an answer either all-at-once or step-by-step. Other NLP research by Hanif (2023) shows that T5 outperforms larger language models such as GPT-2 in summarization tasks. The reason for this unexpected result is attributed to T5-Large which requires a *larger training dataset* to effectively update model weights of the entire model. T5-Base, however, is advantageous, as a smaller model with fewer model parameters and a relatively smaller training dataset, achieves optimal performance, with reduced computational time and resources.

6.4 RQ2: Unanswerable Questions Requiring Medical Knowledge

RQ2: How can medical knowledge sources related to diagnoses and medication terminologies be utilized in data augmentation to simplify complex medical jargon in unanswerable questions?

6.4.1 Key Findings

In addressing the research question *RQ2*, which focuses on unanswerable questions requiring medical knowledge, we summarize three key findings based on the evaluation of the models on the validation set:

1. *Impact of RAG on Medical Jargon Simplification*: The T5-Large with RAG model significantly simplifies complex medical jargon in unanswerable questions, achieving a 26.13% increase in EM and a 24.17% increase in EX compared to the vanilla T5-Large model. This improvement is attributed to the integration of medical vocabularies, namely, SNOMED CT which simplifies complex diagnoses concepts, while RxNorm simplifies medication names in unanswerable questions.

2. *Impact of Knowledge Sources for SQL Post-Processing Techniques*: Although SQL post-processing yields sub-optimal performance, the models that utilize the SNOMED CT and RxNorm medical vocabularies are more effective than those relying on the MeSH vocabulary.

3. *Impact of Model Size:* The T5-Large with RAG model resolves unanswerable questions requiring medical knowledge, achieving a 3.56% increase in EM and a 2.21% increase in EX relative to the T5-Base with RAG model.

Refer to **Table 5.4** for the validation results of the models evaluated on the unanswerable questions requiring medical knowledge.

6.4.2 Findings Interpretation

In this section we interpret the main findings, highlighting why T5-Large with RAG (in Experiment 10b) outperforms all other models, largely attributed to the selected knowledge sources.

Impact of RAG on Medical Jargon Simplification

The findings of these experiments are similar to the experiments in *RQ1*, where the T5 model learns how to effectively use additional knowledge during training. As a result, the model simplifies medical jargon in unanswerable questions by utilizing synonyms from the SNOMED CT vocabulary for diagnoses concepts and the RxNorm vocabulary for medication terms. The results of our experiments indicate that combining RAG with T5-Large (Experiment 10b) yields optimal performance (EM = 49.2% and EX = 67.61%). The model achieves a 26.13% increase in EM and a 24.17% increase in EX relative to the T5-Large vanilla model (EM = 23.07% and EX = 43.44%); and a 5.27% increase in EM and a 7.49% increase in EX relative to the T5-Base vanilla model (EM = 43.93% and EX = 60.12%). The performance gain when leveraging external knowledge is in accordance with the findings of Li et al. (2023b), which indicate that T5-Large with external knowledge achieves a 10.04% improvement in EX accuracy compared to T5-Large without external knowledge (n^{valid} = 1534). Similar research by Lee et al. (2021) shows that their text-to-SQL model improves by 13.2% in EM accuracy when incorporating additional knowledge to simplify complex column names (n^{valid} = 272).

We verify that our best model leverages the additional knowledge to simplify complex medical jargon. We do this by evaluating the CM accuracies for each model (see **Table 5.5**). We find that the CM (on the validation samples) for the *SELECT* and *FROM* clauses are relatively similar across the models, suggesting that the additional knowledge has no significant effect on the model's column and table predictions, respectively. For the CM of the *WHERE* clause, however, we find that the T5-Large with RAG model achieves a 32.4% increase in CM^{WHERE} compared to the T5-Large vanilla model, as well as a 6.75% increase in CM^{WHERE} compared to the T5-Base vanilla model, and a 2.58% increase in CM^{WHERE} compared to the T5-Base with RAG model. These results suggest that the T5-Large with RAG model correctly interprets and simplifies the medical jargon by predicting the correct values in the SQL query, i.e., the base form of a medical concept found in the predicted column's values. This analysis is in contrast to Li et al. (2023b), which does not provide the CM accuracies, thereby limiting our understanding of how the model utilizes the additional knowledge and which SQL components it influences.

Impact of Knowledge Sources for SQL Post-Processing Techniques

We also find that the SNOMED CT and RxNorm vocabularies leveraged in SQL post-processing (Experiments 8a and 8b) are more suitable for obtaining synonyms when compared to using MeSH definitions (Experiments 9a and 9b). Here Experiments 8a and 9a use T5-Base and Experiments 8b and 9b use T5-Large. More specifically, Experiment 9a (EM = 21.23% and EX = 66.38%) achieves a 16.25% increase in EX relative to the model in Experiment 8a (EM = 34.72% and EX = 50.13%). Likewise, for the models that use SQL post-processing, when fine-tuned with T5-Large, Experiment 9b (EM = 11.66% and EX = 60.61%) achieves a 21.1% increase in EX relative to Experiment 8b (EM = 19.39% and EX = 39.51%). We do not consider the change in EM, as SQL post-processing deviates the predicted SQL query from the ground-truth SQL query. Nonetheless, the EX improvement is supported by a review by Chang and Sung (2024), which shows that 89% (17/19) of articles reported an increase in performance when SNOMED CT was integrated. In contrast, research by Kimura et al. (2024) shows that RAG which employs RxNorm for drug mapping improves model performance relative to traditional string-matching and vector similarity methods (as in SQL post-processing).

Impact of Model Size

The T5-Large with RAG model (Experiment 10b) resolves unanswerable questions requiring medical knowledge, achieving a 3.56% increase in EM and a 2.21% increase in EX relative to the T5-Base with RAG model (Experiment 10a). This result is similar to Xia et al. (2024), which suggests that T5-Large with additional context is superior to T5-Base with additional context, but the improvement is marginal. Similar to Xia et al. (2024) and our findings in *RQ1*, the modest improvement of T5-Large with RAG relative to T5-Base with RAG is attributed to T5-Large which requires a larger dataset to effectively learn how to leverage the additional knowledge. However, our finding is still significant because it suggests that a larger training dataset, or a more expansive knowledge resource can further improve the performance of T5-Large with RAG.

6.5 Generalization

In this section, we present the findings of the best models in Experiments 3a (*RQ1*) and 10b (*RQ2*), highlighting their performance on the test questions. We also provide additional validation results to support our claims of generalization.

6.5.1 RQ1

Test Performance

Evaluation of the best model in Experiment 3a, when evaluated on the test questions (see **Table 5.6**), achieves similar performance to that of the validation questions (see **Table 5.1**). This suggests model generalization on an unseen dataset. We see that the model achieves an average EM = 88.9% and an EX = 93.65% when evaluated on the out-of-schema test questions. This includes an average EM = 92.24% and an EX = 95.58% when evaluated on the reference test questions; and an EM = 85.55% and EX = 91.72% on the test paraphrases. These results support the finding that T5-Base when fine-tuned on out-of-schema questions, improves the model's ability to generalize to new out-of-schema questions.

Generalization of Model on Out-of-Schema Questions

We also observe similar performance when evaluating the best model in Experiment 3a on a subset of out-of-schema validation questions, including n^{valid} = 302 pairs of synthetic reference questions and their corresponding paraphrases (see **Table 5.2**). The model achieves an EM = 94.7% and an EX = 98.01% when evaluated on the synthetic reference questions and an EM = 87.09% and an EX = 96.03% when evaluated on the paraphrases. These results suggest that the model not only performs well on questions with similar structural representations relative to that found in the training data but also on paraphrases with lexical and structural variation. **Table 5.3** further reports the model CM accuracies. The model achieves a CM^{SELECT} = 100%, CM^{FROM} = 100% and CM^{WHERE} = 94.7% on the synthetic reference questions; and a CM^{SELECT} = 97.35%, CM^{FROM} = 99.67% and CM^{WHERE} = 89.4% when evaluated on the paraphrases. In contrast to Gan et al. (2021b) and Gan et al. (2021a), these results suggest model robustness when predicting columns, tables and values in the *SELECT*, *FROM* and *WHERE* clauses, respectively, regardless of the structural and lexical variation in the question.

6.5.2 RQ2

Test Performance

Evaluation of the best model in Experiment 10b, when evaluated on the test questions (see **Table 5.8**) achieves similar performance to that of the validation questions (see **Table 5.4**). The model achieves a 9.44% increase in EM and a 9.68% increase in EX relative to the baseline model in Experiment 1b. These results support the finding that T5-Large with RAG improves the model's ability to leverage additional medical knowledge when simplifying complex medical jargon in questions. In addition, we find that the model achieves a CM^{SELECT} = 85.09%, CM^{FROM} = 86.02% and CM^{WHERE} = 36.8% when evaluated on the test questions (see **Table 5.9**). These results suggest that the model adequately predicts the correct column and table names, with encouraging performance when predicting the correct values in the SQL query. While additional knowledge improves the model's understanding of complex medical jargon, a more expansive list of medical synonyms is required to further improve model performance.

6.6 Model Constraints

6.6.1 Answerable Questions

One of our model objectives was to ensure that the answerability of the original answerable questions (in the MIMIC-III and eICU datasets) is not negatively affected by the fine-tuned models. For the model proposed in *RQ1* (Experiment 3a), when addressing out-of-schema questions, we see that the model improves in performance when evaluated on the answerable questions relative to the baseline model in Experiment 1a (see **Table 5.6**). These results suggest that a richer dataset, with more schemas and a greater diversity of question types, significantly improves model performance and generalization (Zhu et al., 2024). For the model proposed in *RQ2* (Experiment 10b), when addressing unanswerable questions requiring medical knowledge, we find no significant effect of the T5-Large with RAG model on the answerable questions (see **Table 5.8**). This indicates that the T5-Large model is effective in adhering to prompt instructions, where our prompt specifically instructs the model to utilize the additional synonym context for unanswerable questions. As a result of these findings, we meet our model constraints.

6.6.2 Answer Abstention

A second objective was to ensure model reliability by incorporating an answer abstention method that correctly abstains from answering unanswerable questions, while correctly choosing to answer answerable questions and unanswerable questions that are resolved with additional knowledge. For the best model evaluated in *RQ1* (Experiment 3a), the performance after abstention is similar to that of the performance before abstention (for both out-of-schema and answerable questions). This suggests that the model usually correctly predicts answerable questions and resolves unanswerable questions as answerable (see **Table 5.6**). For the best model evaluated in *RQ2* (Experiment 10b), however, we see a marginal decline in performance after answer abstention is applied (see **Table 5.8**). This does not pose a major risk. However it highlights a trade-off. Similar to Lee et al. (2024), we see that with strict answer abstention (i.e., one that favours abstaining from answering unanswerable questions), we achieve model reliability and trustworthiness, however, the model's utility is reduced when encountering a question that should be answered.

6.7 Error Analysis

For the best models (in Experiments 3a and 10b), each incorporating the abstention classifier, we find that erroneous SQL predictions fall into three primary categories, namely, (i) inexecutable SQL predictions, (ii) SQL predictions with incorrect tables and/or columns and/or values, and (iii) incorrect abstention, where the abstention model incorrectly chooses not to answer a question. For both models, we find that the errors mainly comprise inexecutable SQL queries or SQL predictions with incorrect table names and/or column names and/or values. This suggests that the answer abstention model is satisfactory, while more rigorous strategies should be implemented to reduce the SQL prediction errors.

6.8 Limitations and Future Work

This research thesis has several limitations. First, while it addresses unanswerable questions related to diagnoses and medication, it does not address unanswerable questions related to medical procedures, as identified as an important unanswerable question subcategory in our exploratory data analyses (see Section 3.3). Future work should leverage the UMLS (Bodenreider, 2004) to resolve unanswerable questions related to medical procedures. Second, our research is based on three sources of EHRs, namely MIMIC-III (Johnson et al., 2016), eICU (Pollard et al., 2018) and a third EHR with information related to these two EHRs. Having only three EHRs limits the model's ability to generalize to questions related to EHRs with varying complexities. This is in contrast to models that are fine-tuned with question and SQL pairs from cross-domain text-to-SQL datasets, such as Spider (Yu et al., 2018c), which includes 200 databases. Lastly, for the implementation of RAG, future work should expand the lexicon to medical concepts found in EHRs beyond our existing ones. Alternatively, one should enable real-time RAG by querying a knowledge base such as the UMLS (Bodenreider, 2004) at inference. Finally, error analysis indicates that SQL errors primarily include inexecutable SQL queries or SQL predictions with incorrect table names, and/or column names, and/or values. We intended to implement direct policy optimization (DPO) to mitigate these errors. This is done by using erroneous SQL predictions as

feedback in model fine-tuning. However, due to time and resource constraints, the experiments were still in progress at the time of time of completion of this thesis. Our future work should explore this approach further.

6.9 Summary

This chapter re-introduced the research problem and the methodological approach, while also highlighting and interpreting the key findings for *RQ1* and *RQ2*. We also discuss the additional findings, such as model generalization on the test set, the model constraints and error analysis. Finally, we conclude with the limitations and future work. The model constraints include the model's ability to maintain answerability on the original answerable questions and answer abstention when the model is not confident in answering a question. In summary, we find that for *RQ1* the best model in Experiment 3a, includes fine-tuning a T5-Base model with the original data (MIMIC-III and eICU) and the unanswerable questions, including out-of-schema question representations. The model outperforms all other models, including those that incorporate schema serialization. For *RQ2*, we find that the best model in Experiment 10b, includes fine-tuning T5-Large with RAG, where the input includes additional synonym context for complex medical jargon simplification. Here we find that SNOMED CT (for diagnoses) and RxNorm (for medication) are suitable medical vocabularies for our RAG implementation. In addition, we find that the best models either have a performance gain (*RQ1*) or no effect (*RQ2*) on the original answerable questions. For answer abstention in *RQ1*, we find that the model typically correctly classifies answerable and resolved out-of-schema questions as answerable, while for *RQ2*, we find a marginal performance decline after answer abstention is applied. This suggests a trade-off where a strict abstention model, which favours abstaining unanswerable questions, provides improved model reliability and trustworthiness at the potential expense of reduced model utility when the model encounters a question that should be answered. Finally, error analysis indicates that SQL errors primarily include inexecutable SQL queries or SQL predictions with incorrect table names, and/or column names, and/or values. In Chapter 7, we conclude the thesis by reiterating the key findings, the overall contributions and the significance of this work.

Chapter 7

Conclusion

This thesis presents a dual text-to-SQL model approach to address two distinct challenges in medical question answering, namely resolving out-of-schema questions (*RQ1*) and unanswerable questions requiring medical knowledge (*RQ2*).

7.1 Key Findings

In our investigation to find the optimal model to address out-of-schema questions (*RQ1*) and unanswerable questions that require medical knowledge (*RQ2*), we have identified three key findings for each. For *RQ1* we find that: (i) the T5-Base model (in Experiment 3a) requires training on out-of-schema question representations, excluding schema serialization during training or inference, (ii) while schema serialization underperforms compared to the prior strategy, the model learns during training how to effectively utilize schema information at inference, making this approach preferable to strategies that use schema serialization only at inference or schema pruning more generally, and (iii) T5-Base is sufficient for resolving out-of-schema questions, as T5-Large does not significantly improve model performance. For *RQ2* we find that: (i) the T5-Large with RAG model (in Experiment 10b) effectively simplifies complex medical jargon in unanswerable questions, leveraging the SNOMED CT and RxNorm medical vocabularies for synonym context, (ii) while SQL post-processing strategies underperform compared to the prior strategy, they also benefit from medical synonyms derived from SNOMED CT and RxNorm relative to those derived from matching MeSH definitions, and (iii) while T5-Large with RAG achieves superior performance, the improvement over T5-Base with RAG is marginal.

7.2 Contributions

We find that the best model for addressing out-of-schema questions (*RQ1*) involves identifying unanswerable questions, augmenting the database schema with relevant medical knowledge, and fine-tuning a T5-Base transformer model on data that includes these out-of-schema question representations. Our findings indicate that T5-Base effectively learns out-of-schema question-to-SQL mappings. For unanswerable questions requiring medical knowledge (*RQ2*), T5-Large with RAG simplifies complex medical jargon by leveraging synonym knowledge obtained from the SNOMED CT and RxNorm medical vocabularies. For both research questions, the proposed models consistently improve in exact matching accuracy (EM), execution accuracy (EX) and component matching accuracy (CM) across the validation and test sets compared to the baseline models, indicating strong generalization capabilities for unseen unanswerable questions. In addition, our models correctly abstain from answering questions where the model is uncertain in its predictions, while correctly choosing to answer answerable questions in MIMIC-III and eICU, as well as unanswerable questions that have become answerable with additional knowledge. Furthermore, the additional knowledge has either positive effects (*RQ1*) or no effect (*RQ2*) on the model's answerability of the original answerable questions. This ensures the model expands coverage to unanswerable questions rather than model bias towards unanswerable questions. Finally, error analysis indicates that SQL errors primarily stem from inexecutable SQL queries or SQL predictions with incorrect table names, and/or column names, and/or values. Overall, this thesis achieves its aims and objectives, by expanding the coverage of questions answered by the text-to-SQL models, ensuring model reliability and identifying areas of improvement.

7.3 Significance

In conclusion, our results support the hypothesis that additional knowledge sources are crucial for sequence-to-sequence text-to-SQL models, such as T5, to accurately interpret the meaning of unanswerable questions, either with out-of-schema concepts or complex medical jargon that needs simplification. As a result, the model expands its answering capabilities to both answerable and unanswerable questions, with model reliability in place. This advancement enables healthcare professionals (regardless of technical expertise) to effectively communicate with electronic health records (EHRs), facilitating access to insights that support informed decision-making and promote actionable patient care in real-world hospital environments. Beyond the medical domain, these findings contribute to the broader field of explainable AI, demonstrating how large language models can be leveraged to interpret and clarify expert-level or jargon-heavy queries. By leveraging large language models' capabilities to incorporate external knowledge, our work aligns with ongoing efforts in explainable AI that aim to improve interpretability and trustworthiness of AI systems in handling complex, domain-specific queries.

Appendix A

Literature Review

A.1 T5

TABLE A.1: T5 Language Model Configurations and Their Respective Parameters. The *Parameters* column provides the number of model parameters for a given model. The # *layers* column provides the number of layers in the encoder, which is the same as the number of layers found in the decoder. The d_{model} column provides the dimension of the embedding vectors. The d_{ff} column provides the dimension of the feedforward network within the encoder and also the decoder layers. The d_{kv} column provides the dimension of the key and value vectors used in the self-attention mechanism. The # *heads* column provides the number of attention heads in each attention block.

Model	**Parameters**	**# layers**	d_{model}	d_{ff}	d_{kv}	**# heads**
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B (XL)	3B	24	1024	16384	128	32
11B (XXL)	11B	24	1024	65536	128	128

Appendix B

Data Generation Methods

B.1 Exploratory Data Analysis

Below we discuss the EDA steps, where Section B.1.1 introduces data pre-processing and Section B.1.2 introduces the TF-IDF computation. The subsequent sections provide further details related to the topics covered in these two sections.

B.1.1 Data Pre-Processing

We generate high-quality n-grams from the unanswerable validation questions in the data pre-processing step. This includes removing non-ASCIII characters from questions, tokenization of questions into unigrams and bigrams, and further data cleaning such as word lemmatization, stripping the text of its punctuation and removing English stop words.

Removing Non-ASCII Characters

All non-ASCII characters are removed from each validation unanswerable question. This ensures standardized English output, comprising only ASCII characters (Gaur et al., 2021; Muftie and Haris, 2023). Standardizing the questions is essential for data consistency, where certain processing tools and parsers expect only ASCII input. Examples of non-ASCII characters that are removed include letters with accents (e.g., *ö* in Afrikaans and *é* in French), non-Latin alphabets (e.g., Greek letters such as α and β), mathematical symbols (e.g., the summation symbol $\sum$ and the infinity symbol ∞) and emoticon symbols, among others. ASCII characters included in the output consist of English alphabetic characters irrespective of letter case (i.e., A-Z and a-z), numeric characters (i.e., 0-9), punctuation (e.g., *!?,.*), and a set of control characters (e.g., tab and newline characters).

Tokenization and Additional Data Cleaning

Each unanswerable question is then tokenized into n-grams, either unigrams or bigrams since medical concepts may be found as single words or phrases (here we focus on phrases consisting of two words), where words can carry important conceptual meaning together (Yu et al., 2013). We use unigrams and bigrams instead of higher-order n-grams due to data sparsity, where the count of n-grams decreases in a dataset as the number of n tokens increases in an n-gram. While higher-order n-grams offer richer context, they tend to overfit and generalize poorly across datasets. The NLTK[1] tool in Python is then used to lemmatize (or standardize) each word in the n-gram to its root form, known as the lemma. Lemmatization is necessary to reduce noise by standardizing the inflected form of words (Balakrishnan and Lloyd-Yemoh, 2014; Khyani et al., 2021; Toporkov and Agerri, 2024). Examples of inflected forms of a word include plurals (e.g., *drugs* as opposed to the root word *drug*), past tense (e.g., *found* as opposed to the root word *find*), and the present participle (e.g., *operating* as opposed to the root word *operate*), among others. Subsequently, punctuation is removed from n-grams to further reduce noise (e.g., the word *drug:* becomes *drug*). English stop words (Nothman et al., 2018) are then removed using the NLTK tool. Removing stop words reduces noise by removing common English words. This ensures that n-grams only contain important medical terms. Examples of stop words include articles such as *a*, *an* and *the*, prepositions such as *on*, *of* and *from* and pronouns such as *he*, *she*, and *her*, among others. See Section B.1.3 in Appendix B for the full list of stop words.

[1] https://www.nltk.org/

B.1.2 Term-Frequency Inverse-Document Frequency

The n-gram importance is computed using the term-frequency inverse-document frequency (TF-IDF) (Das, Alphonse, et al., 2023; Gomes et al., 2023; Paulsen et al., 2023). TF-IDF measures the importance of an n-gram within a document, relative to the corpus in which it is found. Mathematically, the TF-IDF of an n-gram (see **Equation B.1**) is computed as the product of the frequency of the n-gram in a specific document (known as the term frequency, or TF; see **Equation B.2**) and the inverse frequency of that n-gram across other documents (known as the inverse document frequency, or IDF; see **Equation B.3**). Here, D represents a specific document and D_n represents the corpus of documents.

$$\text{TF-IDF(n-gram)} = TF \cdot IDF \tag{B.1}$$

$$\text{TF(n-gram)} = \frac{\text{Total count of an n-gram in D}}{\text{Total count of all n-grams in D}} \tag{B.2}$$

$$\text{IDF(n-gram)} = \log_e \left(\frac{\text{Total count of } D_n}{\text{Total count of } D_n \text{ containing an n-gram}} \right) \tag{B.3}$$

If an n-gram appears frequently in a document relative to other documents, the n-gram is important to that document and will have a relatively large TF-IDF score. Conversely, if the n-gram occurs frequently across other documents as well, then the n-gram is not as significant and will have a relatively small TF-IDF score. In this context, the document is simply an unanswerable question and the corpus is the set of unanswerable questions in the validation set. We compute the TF-IDF score for each n-gram in the unanswerable questions and aggregate these scores to obtain a single TF-IDF score for each n-gram. This implementation follows the documentation of scikit-learn (Pedregosa et al., 2011), where we leverage the TF-IDF Vectorizer[2] module in Python to compute the TF-IDF scores. We also considered using latent Dirichlet allocation (LDA) topic modelling (Blei et al., 2009) over TF-IDF. However, TF-IDF provides a more granular analysis of medical concepts, relative to LDA which groups similar concepts. We employ L2 normalization to the TF-IDF scores to normalize the resulting feature vectors to have a unit norm. This ensures that the length of the document (the number of n-grams in a question) does not affect the vectorized representation, thereby allowing for more meaningful comparisons between questions based on their content rather than their length. The top 200 n-grams with the largest TF-IDF scores are plotted using a word cloud. Preliminary results show that the term *patient* has the highest TF-IDF score of 64.83. This is expected for EHR questions which have an abundance of patient-specific questions. Other n-grams that also pertain to patient-specific questions include *appointment* (*TF-IDF* = 20.19), *number* (*TF-IDF* = 17.41), and *address* (*TF-IDF* = 16.75), to name a few. As a further experiment, we investigate the top n-grams that are non-patient-specific. This is because we are interested in unanswerable question types which may be answered with knowledge augmentation. Patient-specific unanswerable questions require additional patient screening to properly answer, which we do not have access to. We redo the experiment, including patient-specific n-grams in the stop words list (see Section B.1.4 in Appendix B). **Figure B.1** in Section B.1.5 of Appendix B shows the word cloud of the top-200 most important medical concepts among the validation, unanswerable questions. The full list of medical concepts and associated TF-IDF scores can be seen in **Table B.1** of Section B.1.6 in Appendix B.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

B.1.3 English Stop Words

- ourselves
- hers
- between
- yourself
- but
- again
- there
- about
- once
- during
- out
- very
- having
- with
- they
- own
- an
- be
- some
- for
- do
- its
- yours
- such
- into
- of
- most
- itself
- other
- off
- is
- s
- am
- or
- who
- as
- from
- him
- each
- the
- themselves
- until
- below
- are
- we
- these
- your
- his
- through
- don
- nor
- me
- were
- her
- more
- himself
- this
- down
- should
- our
- their
- while
- above
- both
- up
- to
- ours
- had
- she
- all
- no
- when
- at
- any
- before
- them
- same
- and
- been
- have
- in
- will
- on
- does
- yourselves
- then
- that
- because
- what
- over
- why
- so
- can
- did
- not
- now
- under
- he
- you
- herself
- has
- just
- where
- too

- only
- myself
- which
- those
- i
- after
- few
- whom
- t
- being
- if
- theirs
- my
- against
- a
- by
- doing
- it
- how
- further
- was
- here
- than

B.1.4 Patient-Specific Stop Words

- patient
- tell
- appointment
- number
- address
- address patient
- patient appointment
- number patient
- phone
- phone number
- history
- travel
- overseas

B.1.5 Word Cloud

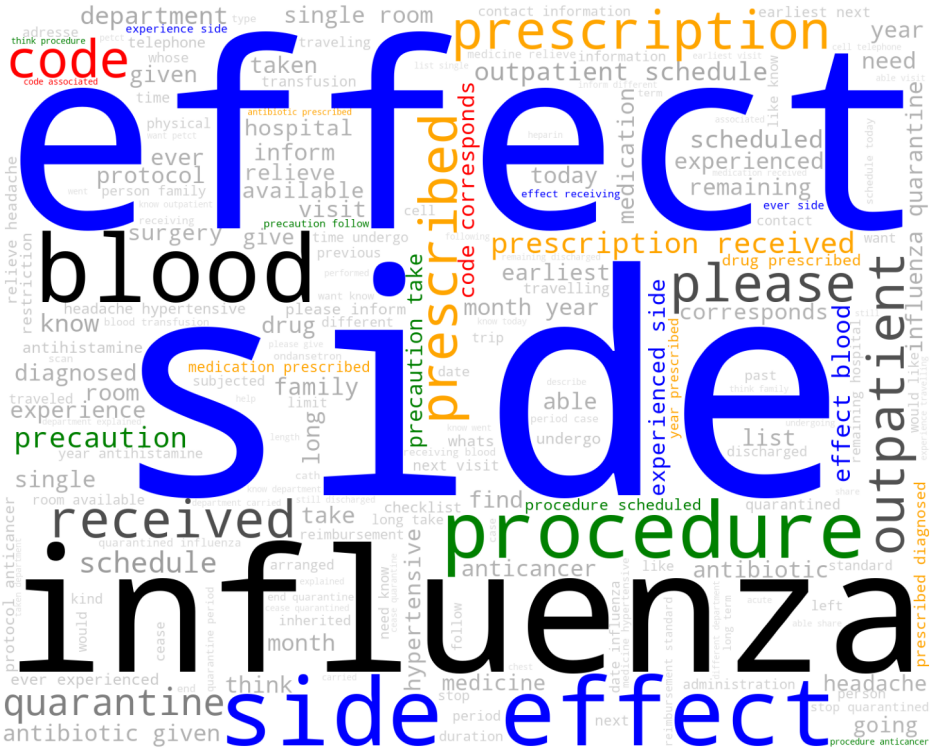


FIGURE B.1: Word cloud Showing the Top 200 Most Important Medical Concepts Among the Validation, Unanswerable Questions. Importance is measured using TF-IDF scores. Large letter sizes suggest higher importance. Grouped concepts are colour-coded.

B.1.6 TF-IDF scores

TABLE B.1: Important Medical Concepts in the EHRSQL Text-to-SQL Parsing Dataset, Obtained Using TF-IDF Scores Ranked in Descending Order of TF-IDF Scores.

Medical Concept	**TF-IDF**
side	11.26959777
effect	11.18830307
influenza	10.17187278
side effect	9.710100405
blood	9.641874428
procedure	9.633420509
prescription	9.329277211
prescribed	9.170631595
please	9.152703375
code	9.080883333
received	8.966553481
outpatient	8.780048897
quarantine	8.367975842
prescription received	8.171720384
schedule	8.126297243
precaution	7.912418611
outpatient schedule	7.881951834
department	7.664493809
scheduled	7.513395368
know	7.510583923
effect blood	7.505295301
family	6.860915557
antibiotic	6.526679109
long	6.380829475
single	6.272518371
take	6.213313134
given	6.208736052
drug	6.18392192
inform	6.083777788
list	6.003686996
room	5.898645093
ever	5.818629702
taken	5.710017203
earliest	5.602003896
anticancer	5.548027669
protocol	5.404215101
single room	5.358652605
find	5.335744062
remaining	5.231164936
antibiotic given	5.188444033
think	5.187299607
able	5.12829059
medication	5.06221153
month	4.956428191
year	4.956428191
month year	4.956428191
experienced	4.875273
need	4.847431374
influenza quarantine	4.758778132
visit	4.653673423
experience	4.628663266

Medical Concept	**TF-IDF**
today	4.55879335
diagnosed	4.511762561
experienced side	4.37062806
corresponds	4.345430889
code corresponds	4.345430889
surgery	4.319758223
available	4.304886158
hospital	4.29542139
hypertensive	4.208676881
precaution take	4.064737715
relieve	4.026429829
headache	4.008623416
going	4
give	3.903231235
medicine	3.895526108
procedure scheduled	3.826815271
drug prescribed	3.7653405
relieve headache	3.749753875
protocol anticancer	3.702220403
antihistamine	3.674335516
ever experienced	3.619834361
whats	3.619079204
prescribed diagnosed	3.535047585
restriction	3.527063017
undergo	3.420447662
please inform	3.34698269
travelling	3.156359001
headache hypertensive	3.154400891
telephone	3.13213346
checklist	3.116252074
medication prescribed	2.985004873
time	2.906790102
quarantined	2.863739101
transfusion	2.789929565
next	2.771780512
earliest next	2.771780512
next visit	2.771780512
year antihistamine	2.738325875
kind	2.621520293
need know	2.586395152
reimbursement	2.580385146
want	2.543536594
limit	2.541791906
date	2.501004105
date influenza	2.501004105
remaining hospital	2.435372579
room available	2.414855998
administration	2.387586614
different	2.354586152
cease	2.334947379
contact	2.309401077
information	2.309401077
contact information	2.309401077
discharged	2.246767452
time undergo	2.218653416
would	2.175121882

Medical Concept	TF-IDF
standard	2.137267001
period	2.104097965
person	2.090793252
person family	2.090793252
cell	2.077996496
follow	2.074329385
trip	2
previous	2
physical	2
arranged	2
subjected	2
inherited	2
left	2
traveled	2
adresse	2
traveling	2
past	2
whose	2
duration	1.963634233
long take	1.92494111
year prescribed	1.922353006
like	1.915476823
would like	1.915476823
like know	1.915476823
stop	1.89696742
stop quarantined	1.89696742
quarantined influenza	1.89696742
ever side	1.800923691
reimbursement standard	1.793175846
experience side	1.765637571
medicine relieve	1.741478045
precaution follow	1.622018732
schedule today	1.603128446
scan	1.589936428
want know	1.589653027
end	1.567237612
case	1.567237612
end quarantine	1.567237612
quarantine period	1.567237612
period case	1.567237612
blood transfusion	1.558418207
type	1.535442595
receiving	1.52427223
effect receiving	1.52427223
receiving blood	1.52427223
medicine hypertensive	1.517098258
ondansetron	1.494287554
term	1.490878642
long term	1.490878642
cath	1.480752907
antibiotic prescribed	1.428879297
medication received	1.405344999
associated	1.402844782
code associated	1.402844782
know department	1.396495908
procedure anticancer	1.388641482

Medical Concept	**TF-IDF**
help	1.378835718
think procedure	1.373616799
undergoing	1.362020497
performed	1.361407508
following	1.35556233
know today	1.345838242
taken department	1.335438766
earliest visit	1.328348105
please give	1.328084237
think family	1.321355556
able visit	1.321355556
describe	1.314640192
length	1.305389975
cease quarantine	1.297706312
different department	1.289857557
know outpatient	1.277054671
inform different	1.27015166
went	1.266738777
know went	1.266738777
remaining discharged	1.257244165
experience travelling	1.257244165
carried	1.250420167
department carried	1.250420167
explained	1.250420167
department explained	1.250420167
cease quarantined	1.240950771
list single	1.238097009
cell telephone	1.230905096
acute	1.229431821
share	1.226851574
able share	1.226851574
heparin	1.215520742
petct	1.205712363
want petct	1.205712363
still	1.185539837
still discharged	1.185539837
chest	1.175387085
detail	1.172086293
detail performed	1.172086293
completed	1.172086293
completed scan	1.172086293
thing	1.163306991
list thing	1.163306991
limit drug	1.156420497
disorder	1.142351822
spinal	1.134703342
canal	1.134703342
explor	1.134703342
spinal canal	1.134703342
canal explor	1.134703342
long influenza	1.130337203
step	1.118707268
tube	1.115293261
aware	1.110575117
aware effect	1.110575117
describe prescription	1.092501534

Medical Concept	TF-IDF
provide	1.078381212
provide prescription	1.078381212
kind antibiotic	1.074335888
prescription given	1.058518044
given department	1.058518044
packed	1.051241379
packed cell	1.051241379
seizure	1.030434171
able find	1.02535838
find procedure	1.02535838
carvedilol	1.004236279
ready	1
morphine	0.985157003
neurology	0.98360721
effect administration	0.975688373
hypothyroidism	0.975662416
heart	0.969085897
health	0.961653439
family health	0.961653439
inform prescription	0.959926808
treat	0.959653682
sodium	0.956343667
likely	0.944973119
type likely	0.944973119
likely undergo	0.944973119
serum	0.937658252
serum transfusion	0.937658252
vaccine	0.93603503
influenza vaccine	0.93603503
respiratory	0.932826916
kind room	0.931209556
prescribed antihistamine	0.92966726
tobacco	0.923835591
pain	0.918713538
followed	0.91719398
precaution followed	0.91719398
followed procedure	0.91719398
closed	0.914975886
biopsy	0.914975886
know antibiotic	0.909064357
knowing	0.908532374
guardian	0.908532374
help knowing	0.908532374
knowing guardian	0.908532374
work	0.90317617
cancer	0.899781019
exact	0.898392428
please exact	0.898392428
exact prescription	0.898392428
hypertensive relieve	0.895298064
impact	0.886965659
impact long	0.886965659
right	0.869838396
give side	0.867355333
commonly	0.866673338
anticonvulsant	0.866650349

Medical Concept	TF-IDF
cont	0.866545457
coronar	0.857675753
match	0.857122692
code match	0.857122692
suffered	0.854122118
ever suffered	0.854122118
suffered side	0.854122118
suppose	0.852072074
suppose experience	0.852072074
transfusion ever	0.848139757
effect experienced	0.846240791
experienced blood	0.846240791
prescription medication	0.844888756
medication diagnosed	0.844888756
vasopressor	0.839235001
complication	0.834441859
ever complication	0.834441859
complication undergoing	0.834441859
undergoing blood	0.834441859
longterm	0.831398487
effect longterm	0.831398487
longterm administration	0.831398487
precaution spinal	0.819288223
know limit	0.819276055
know prescription	0.812026694
cerebral	0.803438945
please medicine	0.799370416
endotracheal	0.7967784
endotracheal tube	0.7967784
long undergo	0.793311404
length time	0.786877401
consultation	0.777332521
diagnosed acute	0.77483695
receive	0.771358996
long receive	0.771358996
taking	0.757522843
esoph	0.756597774
appropriate	0.751927269
someone	0.751927269
appropriate drug	0.751927269
prescribed someone	0.751927269
someone diagnosed	0.751927269
cell transfusion	0.739785441
effect long	0.733982914
medicine given	0.732078037
precaution taken	0.731236152
sinusitis	0.731090939
radiographically	0.731090939
sinusitis radiographically	0.731090939
type medication	0.724427337
long period	0.723176531
vial	0.722800116
soln	0.714481732
refers	0.711405902
code refers	0.711405902
prescribe	0.700347775

Medical Concept	**TF-IDF**
kind drug	0.700347775
drug prescribe	0.700347775
mitral	0.691988714
valve	0.691988714
mitral valve	0.691988714
prepare	0.677485477
prior	0.677485477
checklist prepare	0.677485477
prepare prior	0.677485477
carvedilol side	0.675631754
agent	0.675039503
could	0.663569882
drug could	0.663569882
could prescribed	0.663569882
prescribed treat	0.663569882
inhibitor	0.663047167
pneumonia	0.655983562
code pneumonia	0.655983562
jaund	0.655983562
code jaund	0.655983562
hyposmolality	0.651731372
prescribed hyposmolality	0.651731372
acute respiratory	0.647254703
incurred	0.646406495
precaution incurred	0.646406495
incurred undergoing	0.646406495
repair	0.646123819
therapy	0.639377435
syndrome	0.63646773
restriction heparin	0.636289213
tracheostomy	0.632517252
total	0.632399768
total length	0.632399768
antiarrhythmic	0.63131518
advance	0.629277065
checklist need	0.629277065
need follow	0.629277065
follow advance	0.629277065
proceed	0.627356361
proceed anticancer	0.627356361
restriction sodium	0.626464864
limit morphine	0.626464864
take precaution	0.626118161
done	0.623047728
stay	0.623047728
safe	0.623047728
done stay	0.623047728
stay safe	0.623047728
myectomy	0.619566775
myectomy time	0.619566775
assigned	0.614494286
referring	0.614494286
occurs	0.614494286
code assigned	0.614494286
assigned referring	0.614494286
lidocaine	0.612654263

Medical Concept	TF-IDF
restriction lidocaine	0.612654263
dextrose	0.612654263
restriction dextrose	0.612654263
aspirin	0.612654263
restriction aspirin	0.612654263
model	0.608373534
reimbursement model	0.608373534
using	0.605234858
whats code	0.604419322
code work	0.604419322
head	0.603758066
neck	0.603758066
head neck	0.603758066
step neurology	0.601096137
quick	0.600609969
look	0.600609969
quick look	0.600609969
precautionary	0.593114687
measure	0.593114687
precautionary measure	0.593114687
measure following	0.593114687
anyone	0.590598224
determine	0.590598224
anyone help	0.590598224
help determine	0.590598224
determine code	0.590598224
show	0.588981374
find code	0.588981374
code show	0.588981374
cystoscopy	0.588864426
step cystoscopy	0.588864426
analgesic	0.586239043
anticancer drug	0.579171478
failure	0.571568942
take chest	0.569401412
oral	0.55736309
corresponds pain	0.554288565
corresponds hypothyroidism	0.554288565
mtrl	0.552276081
take mtrl	0.552276081
malnutrition	0.547154452
prescribed malnutrition	0.547154452
hypoxemia	0.547154452
prescribed hypoxemia	0.547154452
whats protocol	0.546641544
delirium	0.54465384
corresponds delirium	0.54465384
thrombocytopenia	0.54465384
corresponds thrombocytopenia	0.54465384
depression	0.54465384
corresponds depression	0.54465384
distl	0.54465384
corresponds distl	0.54465384
brunner	0.543649779
schedule brunner	0.543649779
swain	0.543649779

Medical Concept	**TF-IDF**
schedule swain	0.543649779
glaucoma	0.539201046
prescribed glaucoma	0.539201046
iron	0.533135481
precaution influenza	0.531559805
suffering	0.53155092
headache suffering	0.53155092
currently	0.529092681
room currently	0.529092681
glucose	0.528831002
undergo scan	0.524507732
follow cont	0.523422216
undergo anticonvulsant	0.518963807
drug ondansetron	0.518350266
ventilation	0.515286253
follow ventilation	0.515286253
aphasia	0.511695706
associated aphasia	0.511695706
tacrolimus	0.50932327
tacrolimus give	0.50932327
phenylephrine	0.507151291
think phenylephrine	0.507151291
phenylephrine side	0.507151291
match seizure	0.506545719
treatment	0.50615301
outpatient treatment	0.50615301
treatment schedule	0.50615301
whats serum	0.499674568
code tobacco	0.499160759
tobacco disorder	0.499160759
duration serum	0.497705509
circumcision	0.495502698
duration circumcision	0.495502698
circumcision procedure	0.495502698
used	0.494976083
protocol used	0.494976083
used anticancer	0.494976083
antihistamine prescribed	0.492474966
empyema	0.485749878
prescribed empyema	0.485749878
empyema right	0.485749878
diagnosed tobacco	0.483516257
diagnosed hypothyroidism	0.483516257
proccedural	0.481687963
proccedural drug	0.481687963
drug cancer	0.481687963
klebsiella	0.480909193
pneumoniae	0.480909193
code klebsiella	0.480909193
klebsiella pneumoniae	0.480909193
renal	0.480909193
ureteral	0.480909193
code renal	0.480909193
renal ureteral	0.480909193
candidias	0.480909193
urogenital	0.480909193

Medical Concept	TF-IDF
code candidias	0.480909193
candidias urogenital	0.480909193
culture	0.479226219
urine	0.479226219
precaution culture	0.479226219
culture urine	0.479226219
following coronar	0.477863128
coronar cath	0.477863128
laxative	0.477103771
lactulose	0.477103771
long laxative	0.477103771
laxative lactulose	0.477103771
anti	0.475402384
protocol anti	0.475402384
anti cancer	0.475402384
amount	0.474071287
standard amount	0.474071287
amount reimbursement	0.474071287
dopamine	0.471985724
checklist vasopressor	0.471985724
vasopressor dopamine	0.471985724
atrial	0.471781114
diagnosed atrial	0.471781114
pulmonary	0.471781114
diagnosed pulmonary	0.471781114
year prescription	0.471122774
prescription antihistamine	0.471122774
therapeu	0.465994347
plasmapheresis	0.465994347
checklist therapeu	0.465994347
therapeu plasmapheresis	0.465994347
take influenza	0.464093662
antihistamine commonly	0.463604464
tracheal	0.458943716
suctioning	0.458943716
following tracheal	0.458943716
tracheal suctioning	0.458943716
year commonly	0.458269491
commonly prescribed	0.458269491
undergo influenza	0.452778624
take neurology	0.445159504
neurology consultation	0.445159504
good	0.442846013
good medicine	0.442846013
medicine headache	0.442846013
antihistamins	0.442605486
year think	0.442605486
think antihistamins	0.442605486
traffic	0.440998196
diagnosed traffic	0.440998196
limit ondansetron	0.440945921
dysthymic	0.440823427
prescribed dysthymic	0.440823427
dysthymic disorder	0.440823427
available right	0.439490724
venous	0.438286679

Medical Concept	TF-IDF
take venous	0.438286679
venous cath	0.438286679
varice	0.437516869
prescribed esoph	0.437516869
esoph varice	0.437516869
edema	0.436235205
corresponds cerebral	0.436235205
cerebral edema	0.436235205
exploratory	0.435252129
laparotomy	0.435252129
take exploratory	0.435252129
exploratory laparotomy	0.435252129
antihypertensive	0.435252129
labetalol	0.435252129
take antihypertensive	0.435252129
antihypertensive labetalol	0.435252129
procedure coronar	0.434440163
stomatits	0.431491644
mucosits	0.431491644
corresponds stomatits	0.431491644
stomatits mucosits	0.431491644
silibinin	0.430511898
limit silibinin	0.430511898
undergo endotracheal	0.429924124
ultrasonography	0.428035953
abdomen	0.428035953
take ultrasonography	0.428035953
ultrasonography abdomen	0.428035953
carbapenem	0.423852007
duration antibiotic	0.423852007
antibiotic carbapenem	0.423852007
carbapenem procedure	0.423852007
plcmt	0.42345305
undergo cath	0.42345305
cath plcmt	0.42345305
given chest	0.422940162
chest pain	0.422940162
term morphine	0.421439279
angiocardiogram	0.420741107
taken heart	0.420741107
heart angiocardiogram	0.420741107
norepinephrine	0.420702276
follow vasopressor	0.420702276
vasopressor norepinephrine	0.420702276
metoprolol	0.420474673
term metoprolol	0.420474673
angiography	0.418376799
undergo angiography	0.418376799
angiography cerebral	0.418376799
kind medication	0.418137929
medication relieve	0.418137929
receive endotracheal	0.417603107
nonop	0.416442754
exam	0.416442754
follow nonop	0.416442754
nonop exam	0.416442754

Medical Concept	TF-IDF
proximal	0.415447316
gastrectomy	0.415447316
long proximal	0.415447316
proximal gastrectomy	0.415447316
gastrectomy take	0.415447316
helicobacter	0.414540558
pylorus	0.414540558
associated helicobacter	0.414540558
helicobacter pylorus	0.414540558
recommend	0.414074211
recommend medicine	0.414074211
best	0.414074211
best medicine	0.414074211
contr	0.413915288
cerebr	0.413915288
undergo contr	0.413915288
contr cerebr	0.413915288
systemic	0.413529407
vancomycin	0.413529407
precaution systemic	0.413529407
systemic antibiotic	0.413529407
antibiotic vancomycin	0.413529407
albumin	0.413276118
human	0.413276118
albumin human	0.413276118
human give	0.413276118
potassium	0.411160297
effect potassium	0.411160297
potassium long	0.411160297
demonstrated	0.409088937
code sinusitis	0.409088937
radiographically demonstrated	0.409088937
incision	0.407023673
mediastinum	0.407023673
thing incision	0.407023673
incision mediastinum	0.407023673
take packed	0.405524905
male	0.405169284
inflam	0.405169284
match male	0.405169284
male inflam	0.405169284
liver	0.405169284
transplant	0.405169284
refers liver	0.405169284
liver transplant	0.405169284
administration vial	0.40416231
vial ondansetron	0.40416231
obst	0.404083331
diagnosed obst	0.404083331
sedative	0.403553736
prior sedative	0.403553736
diagnosed mitral	0.403425821
knee	0.402885694
amputat	0.402885694
receive knee	0.402885694
knee amputat	0.402885694

Medical Concept	TF-IDF
temporary	0.402885694
long temporary	0.402885694
temporary tracheostomy	0.402885694
tracheostomy taking	0.402885694
valproate	0.402885694
long anticonvulsant	0.402885694
anticonvulsant valproate	0.402885694
valproate taking	0.402885694
rent	0.401996516
available rent	0.401996516
insulin	0.401911704
aspart	0.401911704
insulin aspart	0.401911704
aspart soln	0.401911704
soln side	0.401911704
abdominal	0.401360182
compartment	0.401360182
code abdominal	0.401360182
abdominal compartment	0.401360182
compartment syndrome	0.401360182
hydralazine	0.399741642
administration hydralazine	0.399741642
safe cont	0.398315712
creat	0.396703441
esophagastr	0.396703441
sphinc	0.396703441
precaution creat	0.396703441
creat esophagastr	0.396703441
esophagastr sphinc	0.396703441
entral	0.395496958
infus	0.395496958
nutrit	0.395496958
long entral	0.395496958
entral infus	0.395496958
infus nutrit	0.395496958
insertion	0.394950136
thoracostomy	0.394950136
insertion thoracostomy	0.394950136
thoracostomy tube	0.394950136
tube time	0.394950136
hypertension	0.393337285
medicine take	0.393337285
take relieve	0.393337285
headache hypertension	0.393337285
take spinal	0.392630143
beta	0.392566873
blocker	0.392566873
checklist beta	0.392566873
beta blocker	0.392566873
blocker carvedilol	0.392566873
syringe	0.39079075
enoxaparin	0.39079075
limit syringe	0.39079075
syringe enoxaparin	0.39079075
enoxaparin sodium	0.39079075
undergo packed	0.384216988

Medical Concept	TF-IDF
would please	0.382451421
plastic	0.381683291
take plastic	0.381683291
plastic surgery	0.381683291
surgery consultation	0.381683291
duration packed	0.381379341
transfusion procedure	0.381379341
uterine	0.380966994
take closed	0.380966994
closed uterine	0.380966994
uterine biopsy	0.380966994
saline	0.380388949
solution	0.380388949
take administration	0.380388949
administration saline	0.380388949
saline solution	0.380388949
chemistry	0.376387664
prescribe blood	0.376387664
blood chemistry	0.376387664
valvular	0.374727611
treat valvular	0.374727611
pro	0.373654374
atrium	0.373654374
take pro	0.373654374
pro repair	0.373654374
repair atrium	0.373654374
neuromuscular	0.373654374
blocking	0.373654374
take neuromuscular	0.373654374
neuromuscular blocking	0.373654374
blocking agent	0.373654374
card	0.372877688
precaution heart	0.372877688
heart card	0.372877688
card cath	0.372877688
coagulopathy	0.371200082
associated coagulopathy	0.371200082
coagulopathy heparin	0.371200082
heparin administration	0.371200082
prescribe sinusitis	0.368567032
intracranial	0.368329726
injury	0.368329726
intracerebral	0.368329726
corresponds intracranial	0.368329726
intracranial injury	0.368329726
injury intracerebral	0.368329726
lower	0.36727053
work lower	0.36727053
lower esoph	0.36727053
injection	0.364674773
think ondansetron	0.364674773
ondansetron injection	0.364674773
injection vial	0.364674773
vial side	0.364674773
curious	0.363254874
curious protocol	0.363254874

Medical Concept	TF-IDF
protocol drug	0.363254874
drug work	0.363254874
work treat	0.363254874
antiretrovirals	0.360898216
protease	0.360898216
long antiretrovirals	0.360898216
antiretrovirals protease	0.360898216
protease inhibitor	0.360898216
inhibitor take	0.360898216
legal	0.35937242
referring legal	0.35937242
legal occurs	0.35937242
prednisolone	0.359292862
acetate	0.359292862
term prednisolone	0.359292862
prednisolone acetate	0.359292862
transureth	0.359130305
prostatec	0.359130305
undergoing transureth	0.359130305
transureth prostatec	0.359130305
levofloxacin	0.358077176
effect levofloxacin	0.358077176
levofloxacin soln	0.358077176
soln long	0.358077176
trying	0.357317212
figure	0.357317212
trying figure	0.357317212
figure medicine	0.357317212
fiberoptic	0.357069325
colonoscopy	0.357069325
chemocautery	0.357069325
taken fiberoptic	0.357069325
fiberoptic colonoscopy	0.357069325
colonoscopy chemocautery	0.357069325
status	0.355765776
cardiac	0.355765776
pacemaker	0.355765776
given status	0.355765776
status cardiac	0.355765776
cardiac pacemaker	0.355765776
foley	0.355243519
catheter	0.355243519
length foley	0.355243519
foley catheter	0.355243519
catheter procedure	0.355243519
explain	0.354198057
explain medication	0.354198057
medication using	0.354198057
using relieve	0.354198057
endo	0.352757083
polpectomy	0.352757083
lrge	0.352757083
thing endo	0.352757083
endo polpectomy	0.352757083
polpectomy lrge	0.352757083
senile	0.351547858

Medical Concept	TF-IDF
dementia	0.351547858
uncomp	0.351547858
refers senile	0.351547858
senile dementia	0.351547858
dementia uncomp	0.351547858
upper	0.350723213
obstruction	0.350723213
tumor	0.350723213
code upper	0.350723213
upper respiratory	0.350723213
respiratory obstruction	0.350723213
obstruction tumor	0.350723213
generalized	0.347134965
tonic	0.347134965
diagnosed seizure	0.347134965
seizure generalized	0.347134965
generalized tonic	0.347134965
prophylaxis	0.346645141
conventional	0.346645141
whats prophylaxis	0.346645141
prophylaxis conventional	0.346645141
conventional heparin	0.346645141
heparin therapy	0.346645141
antihyperlipidemic	0.344380102
reductase	0.344380102
duration antihyperlipidemic	0.344380102
antihyperlipidemic agent	0.344380102
agent reductase	0.344380102
reductase inhibitor	0.344380102
preterm	0.342236682
corresponds preterm	0.342236682
level	0.336416278
advance drug	0.336416278
drug level	0.336416278
antifungal	0.333455859
look antifungal	0.333455859
antifungal therapy	0.333455859
therapy long	0.333455859
advance closed	0.332941034
closed biopsy	0.332941034
show mitral	0.332637404
valve disorder	0.332637404
bilat	0.331106715
block	0.331106715
treat bilat	0.331106715
bilat block	0.331106715
electrolyte	0.328447454
undergoing electrolyte	0.328447454
electrolyte administration	0.328447454
administration oral	0.328447454
abscess	0.324943988
mastoiditis	0.324943988
associated abscess	0.324943988
abscess head	0.324943988
neck mastoiditis	0.324943988
neurological	0.321995377

Medical Concept	TF-IDF
process	0.321995377
code acute	0.321995377
respiratory failure	0.321995377
failure neurological	0.321995377
neurological process	0.321995377
excise	0.317435389
diaphragm	0.317435389
lesion	0.317435389
length excise	0.317435389
excise diaphragm	0.317435389
diaphragm lesion	0.317435389
lesion procedure	0.317435389
endarter	0.317268956
following head	0.317268956
neck endarter	0.317268956
destruct	0.317082504
wall	0.317082504
prior destruct	0.317082504
destruct chest	0.317082504
chest wall	0.317082504
class	0.31565759
thing antiarrhythmic	0.31565759
antiarrhythmic class	0.31565759
class antiarrhythmic	0.31565759
aneurysm	0.313622705
resection	0.313622705
following aneurysm	0.313622705
aneurysm resection	0.313622705
resection repair	0.313622705
bronchial	0.305408477
look closed	0.305408477
closed bronchial	0.305408477
bronchial biopsy	0.305408477
biopsy long	0.305408477
might	0.302471399
please describe	0.302471399
describe medication	0.302471399
medication might	0.302471399
might able	0.302471399
able relieve	0.302471399
partial	0.294260562
complex	0.294260562
referring seizure	0.294260562
seizure partial	0.294260562
partial complex	0.294260562
complex occurs	0.294260562
chronic	0.293857683
kidney	0.293857683
disease	0.293857683
stage	0.293857683
show chronic	0.293857683
chronic kidney	0.293857683
kidney disease	0.293857683
disease stage	0.293857683
bolus	0.293119522
parenteral	0.293119522

Medical Concept	**TF-IDF**
precaution analgesic	0.293119522
analgesic bolus	0.293119522
bolus parenteral	0.293119522
parenteral analgesic	0.293119522
using anticancer	0.289585739
drug protocol	0.289585739
congestive	0.285978236
corresponds congestive	0.285978236
congestive heart	0.285978236
heart failure	0.285978236
coronary	0.275645787
unstable	0.275645787
angina	0.275645787
work acute	0.275645787
acute coronary	0.275645787
coronary syndrome	0.275645787
syndrome unstable	0.275645787
unstable angina	0.275645787
current	0.269918181
admission	0.269918181
ventilatory	0.269918181
support	0.269918181
thing tracheostomy	0.269918181
tracheostomy performed	0.269918181
performed current	0.269918181
current admission	0.269918181
admission ventilatory	0.269918181
ventilatory support	0.269918181
ferrous	0.26656774
compound	0.26656774
sucrose	0.26656774
duration ferrous	0.26656774
ferrous iron	0.26656774
iron compound	0.26656774
compound iron	0.26656774
iron sucrose	0.26656774
sucrose procedure	0.26656774
juice	0.264415501
safe glucose	0.264415501
glucose juice	0.264415501
juice oral	0.264415501
oral glucose	0.264415501

B.1.7 Unanswerable Subcategories and Types

TABLE B.2: Examples of Unanswerable Question Types Belonging to the Diagnoses Unanswerable Question Subcategory.

Unanswerable Question Type	Question
ICD-10-CM	• What is the ICD-10 code for Vulvar vestibulitis? • Can you tell me the ICD-10 code associated with Ulcerative blepharitis?
Diagnoses-to-Medication Treatment Mapping	• Which medication can I use to treat the patient's constipation? • Can you please tell me the medication used for treating vitamin A deficiency?
Diagnoses-to-Medication Prevention Mapping	• Which medication can I use to prevent hypertension? • Can you please tell me the medication used for preventing fever?

TABLE B.3: Examples of Unanswerable Question Types Belonging to the Medication Question Subcategory.

Unanswerable Question Type	Question
Side-effects	• What are the side effects of heparin? • Can you tell me the side effects associated with warfarin?
Tradenames	• What are the drug tradenames of heparin? • Can you tell me the commercial brand names of the medication warfarin?
Medication-to-Diagnoses Treatment Mapping	• What diagnoses may be treated with insulin glargine? • Which medical conditions does iron treat?
Medication-to-Diagnoses Prevention Mapping	• Which diagnoses does vitamin E assist in preventing? • What medical conditions does Calcium citrate help prevent?

B.2 EHR Augmentation

B.2.1 New Schema

FIGURE B.2: Additional EHR Schema Used to Answer Unanswerable Questions.

B.2.2 Biomedical Concepts

TABLE B.4: Diagnosis Descriptions Extracted from the MIMIC-III and eICU Electronic Healthcare Records.

Dataset	**Diagnosis**	
	Short Title	**Long Title**
MIMIC-III	TB pneumothorax-micro dx	Tuberculous pneumothorax, tubercle bacilli found (in sputum) by microscopy
	Human immuno virus dis	Human immunodeficiency virus [HIV] disease
	Coxsackie virus mening	Meningitis due to coxsackie virus
eICU	None	endocrine | glucose metabolism | diabetes mellitus
	None	cardiovascular | chest pain / ASHD | coronary artery disease | known
	None	neurologic | disorders of vasculature | stroke

TABLE B.5: Medication Descriptions Extracted from the MIMIC-III and eICU Electronic Healthcare Records.

Dataset	Medication
MIMIC-III	Tacrolimus
	Heparin Sodium
	Metoclopramide
eICU	Morphine Inj
	Metoprolol Tartrate 25 mg PO Tabs
	Bisacodyl 10 mg RE Supp

B.2.3 TUIs

TABLE B.6: The Unified Medical Language System (UMLS) Semantic Types. Semantic types are denoted by their type unique identifier (TUI). This table presents the TUIs relevant to diagnoses and medication, which are the focus of this research thesis.

TUI	Description
	A. Diagnoses
T019	Congenital Abnormality
T020	Acquired Abnormality
T033	Finding
T034	Laboratory or Test Result
T037	Injury or Poisoning
T047	Disease or Syndrome
T048	Mental or Behavioral Dysfunction
T049	Cell or Molecular Dysfunction
T184	Sign or Symptom
T190	Anatomical Abnormality
	B. Medication
T0195	Antibiotic
T123	Biologically Active Substance
T103	Chemical
T200	Clinical Drug
T126	Enzyme
T125	Hormone
T121	Pharmacologic Substance
T127	Vitamin

B.2.4 QuickUMLS

When using the QuickUMLS[3] tool, we find that the long-titles of diagnoses in MIMIC-III (Johnson et al., 2016) return too many fragmented normalized medical concepts, also referred to as hits. As a solution, we use the short titles of diagnoses in MIMIC-III as query terms to reduce the number of fragmented hits. On the other hand, the MIMIC-III medication descriptions and eICU medical descriptions are directly used as query terms. Once normalized medical concepts are obtained, we filter for high-quality hits. The QuickUMLS uses the Jaccard similarity (Murphy, 1996) by default. The Jaccard similarity measures the ratio of the intersection to the union of two sets of words, specifically comparing the overlap between words in the hit and query term. Generally hits with a Jaccard similarity of 0.9 or greater relative to the query term are considered good hits. We choose a strict threshold to maintain a high level of integrity since medical concepts may sound similar but have completely different meanings given a single alteration such as an addition, deletion or substitution of a single word e.g., *Head Injury, Minor (C0555305)* vs *Major head injury (C1282312)*. Here the medical concepts are lexically similar, however, they have different CUI codes as they infer different meanings.

[3] https://github.com/Georgetown-IR-Lab/QuickUMLS

B.3 Mistral

Mistral-7B-v0.2 has the following changes relative to Mistral-7B-v0.1: (i) a 32k context window (vs 8k context in v0.1), (ii) a RoPE-theta = $1x10^6$, and (iii) no sliding-window attention. The context window refers to the number of tokens the LLM can consider when making a prediction. Rotary Position Embedding (RoPE) is used to assist the LLM in understanding inter-token relationships, by encoding the position of tokens within a sentence. Sliding window attention is used by transformer models to limit attention to a pre-specified number of tokens surrounding a target token, reducing computational complexity and improving model efficiency.

B.3.1 Synthetic Reference Questions: LLM Hyperparameters and Post-Processing

We set the padding token to be the same as the end-of-sequence token (<*eos*>). Effective handling of the padding token encourages model simplification (Kim et al., 2022). In this scenario, the model only needs to handle one special token, an end-of-sequence token. We then set the number of maximum new tokens to n = 500. This is the maximum number of tokens the LLM can generate, excluding the number of tokens in the prompt. We restrict the number of tokens the LLM can generate to enable concise and relevant outputs (Niculescu et al., 2022; Ragazzi et al., 2024). We enable a sampling-based approach to predict the next token. Sampling improves the diversity of the text generated (Song et al., 2024). This approach is most appropriate for tasks that require diverse outputs, as in this use case. This is in contrast to a deterministic approach such as greedy decoding (Dijkstra, 2022) that chooses the most probable next token at each step. After the decoding step, we strip any trailing white spaces, hyphens, single-character quotations and stop characters (<*/s*>) from the synthetic reference questions. We then save the questions to a JSON[4] file with *medical concepts* as keys and *synthetic reference questions* as values.

[4]https://www.json.org/json-en.html

B.4 Synthetic Reference Question Generation

B.4.1 Prompt Template

FIGURE B.3: LLM Prompt Template for Synthetic Reference Question Generation. This prompt template is specific to the ICD-10-CM unanswerable question type. All other unanswerable question types follow a similar prompt structure. The *{diagnosis}* placeholder is replaced with a medical concept at inference.

Context:
You are the doctor's assistant. The doctor tells you to find the ICD-10 code of the diagnosis *{diagnosis}*.

Instruction:
Your task is to ask the nurse what the ICD-10 code is for the diagnosis *{diagnosis}*.
Do not attempt to answer the question, just provide the question asked to the nurse as you said it out loud to the nurse. Do not attempt to shorten or simplify the medical description. Ask the question in a natural, human-like manner, but do not change the diagnosis *{diagnosis}*.

Examples:
Here is an example:
Diagnosis Input: Vulvar vestibulitis
Question Output: What is the ICD-10 code for Vulvar vestibulitis?

Here is another example:
Diagnosis Input: Ulcerative blepharitis
Question Output: Can you tell me the ICD-10 code associated with Ulcerative blepharitis?

Now take a deep breath, listen carefully to the doctor and go ask the nurse what the ICD-10 code is for the given diagnosis *{diagnosis}*.

B.4.2 Prompt Examples

TABLE B.7: Diagnoses Unanswerable Question Subcategory: Prompt Examples for Synthetic Reference Question Generation.

Unanswerable Question Type	Input	Output
ICD-10-CM		
	• Vulvar vestibulitis	• What is the ICD-10 code for Vulvar vestibulitis?
	• Ulcerative blepharitis	• Can you tell me the ICD-10 code associated with Ulcerative blepharitis?
Diagnoses-to-Medication Treatment Mapping		
	• Constipation	• Which medication can I use to treat the patient's constipation?
	• Vitamin A deficiency	• Can you please tell me the medication used for treating vitamin A deficiency?
Diagnoses-to-Medication Prevention Mapping		
	• Hypertension	• Which medication can I use to prevent hypertension?
	• Fever	• Can you please tell me the medication used for preventing fever?

TABLE B.8: Medication Unanswerable Question Subcategory: Prompt Examples for Synthetic Reference Question Generation.

Unanswerable Question Type	Input	Output
Side Effects		
	• Heparin	• What are the side effects of heparin?
	• Warfarin	• Can you tell me the side effects associated with warfarin?
Tradenames		
	• Heparin	• What are the drug tradenames of heparin?
	• Warfarin	• Can you tell me the commercial brand names of the medication warfarin?
Medication-to-Diagnoses Treatment Mapping		
	• Glargine	• What diagnoses may be treated with insulin glargine?
	• Iron	• Which medical conditions does iron treat?
Medication-to-Diagnoses Prevention Mapping		
	• Vitamin E	• Which diagnoses does vitamin E assist in preventing?
	• Calcium citrate	• What medical conditions does Calcium citrate help prevent?

B.4.3 Quality Checks

Question Template Generation

Given a synthetic reference question and the prompt examples ($d = 2$ examples), the medical entities are extracted to obtain question templates. Template comparison is suitable when one is interested in the structural similarity between two sentences, disregarding named entities that are similar between the two. In contrast to Li et al. (2024), we do not use the NLTK[5] tool to remove all nouns and verbs in a sentence; rather we remove the medical entities by means of BCE (see Section 3.5.2). We obtain a question template by replacing the medical concept in the question with a placeholder (i.e., an underscore).

Compute Embeddings and Cosine Similarity

The template and question embeddings are computed using Princeton NLP's SimCSE[6] model. The SimCSE model is based on RoBERTa (Liu et al., 2019; Tan et al., 2022). Subsequently, the pairwise cosine similarity is computed between embeddings of: (i) a synthetic reference question template and each of the prompt example templates, and (ii) a synthetic reference question and each of the prompt examples.

Similarity Comparison

The maximum similarity of each comparison m_i and m_{ii} is determined to obtain the most representative example template and example for each question. The former represents the most representative example of the given question based on structural similarity, while the latter represents the example with the most representative named entity and structural similarity relative to the given question. We then compare m_i and m_{ii} to obtain the maximum similarity overall (m_{max}). This m_{max} value represents the similarity between the question and the most representative example or between the question template and the most representative example template. In either scenario, we obtain the most representative example of the given question, either based on the structural similarity or both structural and named entity similarity.

Filtering

We then compare this maximum similarity m_{max} to an arbitrary threshold of 0.6. High-quality synthetic reference questions have a similarity greater than or equal to 0.6, while low-quality questions with a similarity less than 0.6 are filtered out. We then perform further filtering, by excluding questions with newline characters since these questions contain long, conversational outputs. We then conducted a manual check of the low-quality samples and found that most low-quality samples have newline characters. This suggests that most filtering is done based on the second criterion, removing conversational output with newline characters. Finally, we record medical concepts, synthetic reference questions, prompt example similarity and quality tags (e.g., low and high) as quadruples for both high-quality and low-quality synthetic reference questions. The quadruples are saved in JSON[7] file format.

[5] https://www.nltk.org/
[6] https://github.com/princeton-nlp/SimCSE?tab=readme-ov-filemodel-list
[7] https://www.json.org/json-en.html

B.5 Paraphrasing

B.5.1 Prompt Template

FIGURE B.4: Paraphrase Prompt Template. The placeholders *{medical_term}* and *{reference_question}* are replaced with the medical term and synthetic reference question at inference.

Context:
You are a doctor's assistant. Your task is to paraphrase a medical question.

Instruction:
For example, the doctor asks you a question, you then paraphrase the question before asking the nurse the question in your own words. Generate only one paraphrase. Do not attempt to answer the question. Do not alter the medical term: *{medical_term}*.

Example:
Here is an example:
Input Question: Could you please tell me about the potential side effects of estradiol?
Paraphrase: What are the side effects associated with estradiol?

Now take a deep breath, and provide only one paraphrase for the question *{reference_question}* but keep the medical term *{medical_term}* as is.

B.6 Value-Synonym Replacement

B.6.1 WordNet and SNOMED CT

To ascertain whether WordNet (Fellbaum, 2010) or SNOMED CT (Donnelly et al., 2006) should be used for value-synonym replacement of diagnoses terms in an NLQ, we investigate the medical synonyms obtained by either synonym source.

First, we compose a set of unique medical concepts from the synthetic reference questions generated in Section 3.6.1, where n = 3760 unique medical concepts. We find that for this subset of medical concepts, the average similarity between a medical concept and its best synonym for: (i) WordNet is 73.04% (n = 322) and (ii) SNOMED CT is 92.24% (n = 2357). When computing the average, the numerator is the sum of all similarities between a medical concept and its best synonym across all medical concepts; and the denominator (n) is the total number of medical concepts that return at least one synonym other than itself for a specific knowledge source. This suggests that SNOMED CT is most reliable in returning medical synonyms. In **Table B.9**, we observe that SNOMED CT has a larger proportion of medical concepts that return synonyms (62.69%) relative to WordNet (8.56%) for n = 3760 medical concepts. In addition, we see that WordNet only returns synonyms for single-word medical concepts. In contrast, SNOMED CT returns medical concepts for both multi-word medical concepts (55.37% for n = 2357 medical concepts that return at least one synonym) and single-word medical concepts (44.63% for n = 2357 medical concepts that return at least one synonym).

TABLE B.9: WordNet and SNOMED CT: Percentage of Medical Concepts that Return at Least One Medical Synonym. The row name *At least 1 synonym* shows the percentage of all medical concepts (n = 3760) that return at least one medical synonym for a given synonym source. The row name *Multi-word medical concepts* shows the percentage of multi-word medical concepts among the medical concepts that return at least one synonym. The row name *Single-word medical concepts* shows the percentage of single-word medical concepts among the medical concepts that return at least one synonym.

	WordNet (%)	**SNOMED CT (%)**
At least 1 synonym	8.56 (n = 3760)	62.69 (n = 3760)
Multi-word medical concepts	0 (n = 322)	55.37 (n = 2357)
Single-word medical concepts	100 (n = 322)	44.63 (n = 2357)

B.6.2 SpiderSyn

FIGURE B.5: A SpiderSyn Example. A SpiderSyn example showing a Spider question and its associated SpiderSyn question. The blue font shows the word to be replaced, whereas the green font shows the synonym that replaces the word.

Database: hospital_1

Spider Question: What are the names of patients who made an appointment:?

SpiderSyn Question: What are the names of patients who made a reservation:?

SQL Query:

```
SELECT name
FROM appointment AS T1
JOIN patient AS T2 ON T1.patient = T2.ssn
```

B.6.3 Obtaining Synonyms

In this section, we discuss how synonyms for a medical concept within an NLQ are obtained from the knowledge sources, namely SNOMED CT (Le et al., 2024) and RxNorm (Nelson et al., 2011).

SNOMED CT: The SNOMED CT (Donnelly et al., 2006) vocabulary is accessed via the PyMedtermino[8] module in Python. This tool enables us to query a diagnosis term and obtain the SNOMED CT normalized medical concept. We can then use the normalized medical concept as a search term to find suitable synonyms in SNOMED CT. If more than one synonym is returned, synonyms with the sub-string *(disorder)* are removed. We find that some synonyms contain the normalized medical term + *(disorder)*, which does not offer much lexical variation to the original medical term. Subsequently, the best synonym is chosen based on the maximum cosine similarity of a synonym and its target medical concept. Similarity is computed between the medical concept and each of its synonyms. For this task, we compute the embeddings of the medical concept and each synonym using the SapBERT (Liu et al., 2020) language model. SapBERT is a pre-trained language model trained on the UMLS 2020AA (English only) metathesaurus. SapBERT uses Microsoft's BiomedBERT (Gu et al., 2020) model as the base model. Cosine similarity is computed between the CLS token of a medical concept and the CLS token of each of its synonyms. The CLS token represents the weighted average of the words in a body of text, thus capturing the meaning of the medical phrase. The best synonym is chosen based on the maximum cosine similarity with its target medical concept.

RxNorm: The RxNorm (Nelson et al., 2011) vocabulary is also accessed in Python via PyMedtermino. We load the RxNorm vocabulary and query a medication name against RxNorm. We obtain the normalized RxNorm medical concept which we can use as a query term to find suitable synonyms (otherwise known as medication tradenames) in RxNorm. We then select a synonym by randomly sampling a synonym from the available set of synonyms. We randomly sample a synonym because our analysis shows that the maximum cosine similarity between a medication name and its synonyms favors those that are lexically similar to the target medication name, as determined by embeddings computed with SapBERT (Liu et al., 2020). See **Table B.10** for the experimental results.

[8] https://owlready2.readthedocs.io/en/latest/pymedtermino2.html

TABLE B.10: Analysis of SapBERT when Computing Cosine Similarity between Medication Names and their Synonyms. Target medical concepts are shown in blue, lexically favoured synonyms are shown in red and other synonyms are shown in green. Here the similarity score refers to the cosine similarity between a medication name (in blue) and its best synonym. The best synonym is shown in red or green in the final paraphrase.

Category	**Details**
Sentence	Nurse, could you please let me know the commercial brand names of the medication Tacrolimus?
Tradenames	Protopic, Tacrolimus Suspension
Similarity Score	0.77
Paraphrase	Nurse, could you please let me know the commercial brand names of the medication tacrolimus suspension?
Sentence	Could you please tell me the commercial brand names of the medication warfarin?
Tradenames	Coumadin
Similarity Score	0.854
Paraphrase	Could you please tell me the commercial brand names of the medication coumadin?
Sentence	Could you please let me know the brand names for the medication furosemide:?
Tradenames	Disal, Furoscix, Salix - substance, Lasix
Similarity Score	0.868
Paraphrase	Could you please let me know the brand names for the medication furoscix?
Sentence Tradenames	Could you share the tradenames of mycophenolate mofetil with me? Cellcept, Mycophenolate Mofetil (*IND*)
Similarity Score	0.97
Paraphrase	Could you share the tradenames of mycophenolate mofetil (*ind*) with me?

B.6.4 Synonym Replacement

In this task, we replace the medical jargon in a given synthetic reference question (generated in Section 3.6.1). The rest of the question remains in place. Unlike the out-of-schema questions which consist of both synthetic reference questions and paraphrases, the unanswerable questions requiring medical knowledge only consist of the value-synonym replacement questions. Since the out-of-schema questions consist of two unanswerable groups and the unanswerable questions requiring medical knowledge only consist of one unanswerable group, we do not have a strict sample size for the value-synonym replacement questions.

B.7 Ground-Truth SQL Templates

FIGURE B.6: Ground-Truth SQL Query Templates for Unanswerable Questions in the Medication Unanswerable Subcategory.

```
a. Side-effects
SELECT SIDE_EFFECTS
FROM DRUG_SIDE_EFFECTS
WHERE DRUG_NAME = {medical_term} COLLATE NOCASE

b. Tradenames
SELECT TRADENAMES
FROM DRUG_TRADENAMES
WHERE DRUG_NAME = {medical_term} COLLATE NOCASE

c. Medication-to-diagnoses treatment mapping
SELECT TREATMENT_OF
FROM MEDICATION_TO_DIAGNOSIS
WHERE DRUG_NAME = {medical_term} COLLATE NOCASE

d. Medication-to-diagnoses prevention mapping
SELECT PREVENTION_OF
FROM MEDICATION_TO_DIAGNOSIS
WHERE DRUG_NAME = {medical_term} COLLATE NOCASE
```

FIGURE B.7: Ground-Truth SQL Query Templates for Unanswerable Questions in the Diagnoses Unanswerable Subcategory.

```
a. ICD-10-CM
SELECT ICD10CM_CODE
FROM ICD10CM_DIAGNOSIS
WHERE DIAGNOSIS_SHORT_TITLE = {medical_term} COLLATE NOCASE
OR
DIAGNOSIS_LONG_TITLE = {medical_term} COLLATE NOCASE

b. Diagnoses-to-medication treatment mapping
SELECT MEDICATION_TREATMENT
FROM DIAGNOSIS_TO_MEDICATION
WHERE DIAGNOSIS = {medical_term} COLLATE NOCASE

c. Diagnoses-to-medication prevention mapping
SELECT MEDICATION_PREVENTION
FROM DIAGNOSIS_TO_MEDICATION
WHERE DIAGNOSIS = {medical_term} COLLATE NOCASE
```

Appendix C

Results

C.1 RQ1: EM and EX Results

TABLE C.1: Text-to-SQL Model Performance on the Synthetic Reference Validation Questions and Validation Paraphrases. The exact matching (EM) and execution (EX) accuracies are reported for T5-Base and T5-Large models with various schema augmentation strategies. The best EM and EX values are highlighted in blue.

Model ID	EM (%)	EX (%)
A. T5-Base		
Experiment 1a: MIMIC-III + eICU		
Synthetic Reference Questions	0	0
Paraphrases: MIMIC-III + eICU	0	0
Experiment 2a: MIMIC-III + eICU + schema serialization at inference		
Synthetic Reference Questions	0	0
Paraphrases	0	0
Experiment 3a: MIMIC-III + eICU + synthetic data		
Synthetic Reference Questions	93.96	95.58
Paraphrases	85.33	91.33
Experiment 4a: MIMIC-III + eICU + synthetic data + schema serialization at inference		
Synthetic Reference Questions	69.45	71.87
Paraphrases	56.17	68.33
Experiment 5a: MIMIC-III + eICU + synthetic data + schema serialization		
Synthetic Reference Questions	91.79	93.96
Paraphrases	79.67	88.83
Experiment 6a: MIMIC-III + eICU + synthetic data + schema pruning		
Synthetic Reference Questions	0	0
Paraphrases	0	0
B. T5-Large		
Experiment 1b: MIMIC-III + eICU		
Synthetic Reference Questions	0	0
Paraphrases	0	0
Experiment 2b: MIMIC-III + eICU + schema serialization at inference		
Synthetic Reference Questions	0	0
Paraphrases	0	0
Experiment 3b: MIMIC-III + eICU + synthetic data		
Synthetic Reference Questions	93.4	95.46
Paraphrases	81.33	88.83
Experiment 4b: MIMIC-III + eICU + synthetic data + schema serialization at inference		
Synthetic Reference Questions	76.54	77.66
Paraphrases	66.33	74
Experiment 5b: MIMIC-III + eICU + synthetic data + schema serialization		
Synthetic Reference Questions	91.23	93.34
Paraphrases	80.67	88.83

Bibliography

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. In: 2023.

Aoki, Y., Kudo, K., Kuribayashi, T., Brassard, A., Yoshikawa, M., Sakaguchi, K., & Inui, K. (2023). Empirical investigation of neural symbolic reasoning strategies.

Arthur, P., Neubig, G., Sakti, S., Toda, T., & Nakamura, S. (2015). Semantic parsing of ambiguous input through paraphrasing and verification. *Transactions of the Association for Computational Linguistics*, *3*, 571–584.

Ba, J. L. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bae, S., Kim, D., Kim, J., & Choi, E. Question answering for complex electronic health records database using unified encoder-decoder architecture. In: In *Machine learning for health*. 2021, 13–25.

Bae, S., Kyung, D., Ryu, J., Cho, E., Lee, G., Kweon, S., Oh, J., Ji, L., Chang, E., Kim, T., et al. (2024). Ehrxqa: A multi-modal question answering dataset for electronic health records with chest x-ray images. *Advances in Neural Information Processing Systems*, *36*.

Balakrishnan, V., & Lloyd-Yemoh, E. (2014). Stemming and lemmatization: A comparison of retrieval performances.

Bardhan, J., Colas, A., Roberts, K., & Wang, D. Z. (2022). DrugEHRQA: A question answering dataset on structured and unstructured electronic health records for medicine related queries.

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Blei, D. M., Ng, A., & Jordan, M. I. Latent dirichlet allocation. In: 2009.

Boag, W., Sergeeva, E., Kulshreshtha, S., Szolovits, P., Rumshisky, A., & Naumann, T. (2018). Cliner 2.0: Accessible and accurate clinical concept extraction. *arXiv preprint arXiv:1803.02245*.

Bodenreider, O. (2004). The unified medical language system (umls): Integrating biomedical terminology. *Nucleic acids research*, *32*, D267–D270.

Bogin, B., Gardner, M., & Berant, J. (2019). Global reasoning over database structures for text-to-sql parsing. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Brown, E. G., Wood, L., & Wood, S. (1999). The medical dictionary for regulatory activities (meddra). *Drug safety*, *20*, 109–117.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Brunner, U., & Stockinger, K. Valuenet: A natural language-to-sql system that learns from database information. In: In *2021 ieee 37th international conference on data engineering (icde)*. IEEE. 2021, 2177–2182.

Cao, R., Chen, L., Chen, Z., Zhao, Y., Zhu, S., & Yu, K. (2021). Lgesql: Line graph enhanced text-to-sql model with mixed local and non-local relations.

Cao, Z., Zheng, Y., Fan, Z., Zhang, X., Chen, W., & Bai, X. (2024). Rsl-sql: Robust schema linking in text-to-sql generation.

Chang, E., & Mostafa, J. (2021). The use of snomed ct, 2013-2020: A literature review. *Journal of the American Medical Informatics Association*, *28*, 2017–2026.

Chang, E., & Sung, S. (2024). Use of snomed ct in large language models: Scoping review. *JMIR Med Inform*, *12*, e62924.

Chang, S., Wang, J., Dong, M., Pan, L., Zhu, H., Li, A. H., Lan, W., Zhang, S., Jiang, J., Lilien, J., et al. (2023). Dr. spider: A diagnostic evaluation benchmark towards text-to-sql robustness. *arXiv preprint arXiv:2301.08881*.

Chen, J., Wang, X., Xu, R., Yuan, S., Zhang, Y., Shi, W., Xie, J., Li, S., Yang, R., Zhu, T., et al. (2024). From persona to personalization: A survey on role-playing language agents. *arXiv preprint arXiv:2404.18231*.

Choi, D., Shin, M. C., Kim, E., & Shin, D. R. (2021). Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics*, *47*, 309–332.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., … Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, *24*, 1–113.

Conneau, A., & Lample, G. (2019). Cross-lingual language model pretraining. *Advances in neural information processing systems*, *32*.

Dahl, D. A., Bates, M., Brown, M., Fisher, W. M., Hunicke-Smith, K., Pallett, D. S., Pao, C., Rudnicky, A. I., & Shriberg, E. Expanding the scope of the atis task: The atis-3 corpus. In: In *Human language technology - the baltic perspectiv*. 1994.

Das, M., Alphonse, P., et al. (2023). A comparative study on tf-idf feature weighting method and its analysis using unstructured dataset. *arXiv preprint arXiv:2308.04037*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Dijkstra, E. W. A note on two problems in connexion with graphs. In: In *Edsger wybe dijkstra: His life, work, and legacy*. 2022, pp. 287–290.

Dong, G., Yuan, H., Lu, K., Li, C., Xue, M., Liu, D., Wang, W., Yuan, Z., Zhou, C., & Zhou, J. How abilities in large language models are affected by supervised fine-tuning data composition. In: In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*. 2024, 177–198.

Dong, L., & Lapata, M. Coarse-to-fine decoding for neural semantic parsing. In: In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*. 2018, 731–742.

Donnelly, K., et al. (2006). Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, *121*, 279.

Dou, L., Gao, Y., Liu, X., Pan, M., Wang, D., Che, W., Zhan, D., Kan, M.-Y., & Lou, J.-G. (2023). Towards knowledge-intensive text-to-sql semantic parsing with formulaic knowledge. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Dou, L., Gao, Y., Pan, M., Wang, D., Che, W., Zhan, D., & Lou, J.-G. (2022). Unisar: A unified structure-aware autoregressive language model for text-to-sql. *International Journal of Machine Learning and Cybernetics*.

Fan, Y., Jiang, F., Li, P., & Li, H. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning. In: In *Ccf international conference on natural language processing and chinese computing*. 2023, 69–80.

Fellbaum, C. Wordnet. In: In *Theory and applications of ontology: Computer applications*. 2010, pp. 231–243.

Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., & Radev, D. (2018). Improving text-to-sql evaluation methodology. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Gan, Y., Chen, X., Huang, Q., Purver, M., Woodward, J. R., Xie, J., & Huang, P. (2021a). Towards robustness of text-to-sql models against synonym substitution. *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)*.

Gan, Y., Chen, X., & Purver, M. (2021b). Exploring underexplored limitations of cross-domain text-to-sql generalization.

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2024). Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, *17*, 1132–1145.

Gaur, B., Saluja, G. S., Sivakumar, H. B., & Singh, S. (2021). Semi-supervised deep learning based named entity recognition model to parse education section of resumes. *Neural Computing and Applications*, *33*, 5705–5718.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. Convolutional sequence to sequence learning. In: In *International conference on machine learning*. PMLR. 2017, 1243–1252.

Gomes, L., da Silva Torres, R., & Côrtes, M. L. (2023). Bert-and tf-idf-based feature extraction for long-lived bug prediction in floss: A comparative study. *Information and Software Technology*, *160*, 107217.

Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., Naumann, T., Gao, J., & Poon, H. (2020). Domain-specific language model pretraining for biomedical natural language processing.

Gu, Z., Fan, J., Tang, N., Zhang, S., Zhang, Y., Chen, Z., Cao, L., Li, G., Madden, S., & Du, X. (2023). Interleaving pre-trained language models and large language models for zero-shot nl2sql generation. *arXiv preprint arXiv:2306.08891*.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., & Zhang, D. Towards complex text-to-SQL in cross-domain database with intermediate representation. In: In *Proceedings of the 57th annual meeting of the association for computational linguistics*. 2019, 4524–4535.

Gupta, S., Ranjan, R., & Singh, S. N. (2024). A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions.

Hanif, U. (2023). *Research paper summarization using text-to-text transfer transformer (t5) model* [Doctoral dissertation, Dublin, National College of Ireland].

Hazoom, M., Malik, V., & Bogin, B. (2021). Text-to-SQL in the wild: A naturally-occurring dataset based on stack exchange data, 77–87.

He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. In: In *Proceedings of the ieee conference on computer vision and pattern recognition*. 2016, 770–778.

Hemphill, C. T., Godfrey, J. J., & Doddington, G. R. The atis spoken language systems pilot corpus. In: In *Speech and natural language: Proceedings of a workshop held at hidden valley, pennsylvania, june 24-27, 1990*. 1990.

Hirsch, J., Nicola, G, McGinty, G, Liu, R., Barr, R., Chittle, M., & Manchikanti, L. (2016). Icd-10: History and context. *American Journal of Neuroradiology*, *37*, 596–599.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–1780.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification.

Huang, J., Wang, Y., Wang, Y., Dong, Y., & Xiao, Y. (2021). Relation aware semi-autoregressive semantic parsing for nl2sql. *arXiv preprint arXiv:2108.00804*.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. (2019). Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, *32*.

Huang, Y., Yuan, Z., Zhou, Y., Guo, K., Wang, X., Zhuang, H., Sun, W., Sun, L., Wang, J., Ye, Y., et al. (2024). Social science meets llms: How reliable are large language models in social simulations? *arXiv preprint arXiv:2410.23426*.

Hui, B., Shi, X., Geng, R., Li, B., Li, Y., Sun, J., & Zhu, X. (2021). Improving text-to-sql with schema dependency learning. *arXiv preprint arXiv:2103.04399*.

Hwang, W., Yim, J., Park, S., & Seo, M. (2019). A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Jain, S. M. Hugging face. In: In *Introduction to transformers for nlp: With the hugging face library and models to solve problems*. 2022, pp. 51–67.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., & Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, *3*, 1–9.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., & Levy, O. (2020). Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, *8*, 64–77.

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., et al. A comparative study on transformer vs rnn in speech applications. In: In *2019 ieee automatic speech recognition and understanding workshop (asru)*. 2019, 449–456.

Katsogiannis-Meimarakis, G., & Koutrika, G. (2023). A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 1–32.

Khyani, D., Siddhartha, B., Niveditha, N., & Divya, B. (2021). An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology*, *22*, 350–357.

Kim, S., Shen, S., Thorsley, D., Gholami, A., Kwon, W., Hassoun, J., & Keutzer, K. Learned token pruning for transformers. In: In *Proceedings of the 28th acm sigkdd conference on knowledge discovery and data mining*. 2022, 784–794.

Kimura, E., Kawakami, Y., Inoue, S., & Okajima, A. (2024). Mapping drug terms via integration of a retrieval-augmented generation algorithm with a large language model. *Healthcare Informatics Research*, *30*, 355–363.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. *Advances in neural information processing systems*, *4*.

Kuhn, M., Letunic, I., Jensen, L. J., & Bork, P. (2016). The sider database of drugs and side effects. *Nucleic acids research*, *44*, D1075–D1079.

Le, H., Chen, R., Harris, S., Fang, H., Lyn-Cook, B., Hong, H., Ge, W., Rogers, P., Tong, W., & Zou, W. (2024). Rxnorm for drug name normalization: A case study of prescription opioids in the fda adverse events reporting system. *Frontiers in Bioinformatics*, *3*, 1328613.

Lee, C.-H., Polozov, O., & Richardson, M. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In: In *Annual meeting of the association for computational linguistics*. 2021.

Lee, D., Cornet, R., Lau, F., & De Keizer, N. (2013). A survey of snomed ct implementations. *Journal of biomedical informatics*, *46*, 87–96.

Lee, D., de Keizer, N., Lau, F., & Cornet, R. (2014). Literature review of snomed ct use. *Journal of the American Medical Informatics Association*, *21*, e11–e19.

Lee, G., Chay, W., Cho, S., & Choi, E. (2024). Trustsql: A reliability benchmark for text-to-sql models with diverse unanswerable questions. *arXiv preprint arXiv:2403.15879*.

Lee, G., Hwang, H., Bae, S., Kwon, Y., Shin, W., Yang, S., Seo, M., Kim, J.-Y., & Choi, E. Ehrsql: A practical text-to-sql benchmark for electronic health records. In: In *Advances in neural information processing systems*. *35*. 2022, 15589–15601.

Lee, K., Lee, K., Lee, H., & Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, *31*.

Lei, W., Wang, W., Ma, Z., Gan, T., Lu, W., Kan, M.-Y., & Chua, T.-S. Re-examining the role of schema linking in text-to-sql. In: In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*. 2020, 6943–6954.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Li, H., Zhang, J., Li, C., & Chen, H. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In: In *Proceedings of the aaai conference on artificial intelligence*. *37*. 2023, 13067–13075.

Li, H., Zhang, J., Liu, H., Fan, J., Zhang, X., Zhu, J., Wei, R., Pan, H., Li, C., & Chen, H. (2024). Codes: Towards building open-source language models for text-to-sql. *Proc. ACM Manag. Data*, *2*.

Li, J., Hui, B., Qu, G., Li, B., Yang, J., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., Huo, N., Ma, C., Chang, K. C., Huang, F., Cheng, R., & Li, Y. (2023b). Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Li, Y., Chen, B., Liu, Q., Gao, Y., Lou, J.-G., Zhang, Y., & Zhang, D. (2020). What do you mean by that?" a parser-independent interactive approach for enhancing text-to-sql. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lipscomb, C. E. (2000). Medical subject headings (mesh). *Bulletin of the Medical Library Association*, *88*, 265.

Liu, F., Shareghi, E., Meng, Z., Basaldella, M., & Collier, N. (2020). Self-alignment pretraining for biomedical entity representations. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Maamari, K., Abubaker, F., Jaroslawicz, D., & Mhedhbi, A. (2024). The death of schema linking? text-to-sql in the age of well-reasoned language models. *ArXiv*, *abs/2408.07702*.

MacQueen, J. Some methods for classification and analysis of multivariate observations. In: 2018.

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, *30*.

McKeown, K. (1983). Paraphrasing questions using given and new information. *American Journal of Computational Linguistics*, *9*, 1–10.

Micheletti, N., Belkadi, S., Han, L., & Nenadic, G. (2024). Exploration of masked and causal language modelling for text generation. *arXiv preprint arXiv:2405.12630*.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work?

Moody, G. B. Physionet. In: In *Encyclopedia of computational neuroscience*. 2022, pp. 2806–2808.

Muftie, F., & Haris, M. Indobert based data augmentation for indonesian text classification. In: In *2023 international conference on information technology research and innovation (icitri)*. IEEE. 2023, 128–132.

Murphy, A. H. (1996). The finley affair: A signal event in the history of forecast verification. *Weather and forecasting*, *11*, 3–20.

Nelson, S. J., Zeng, K., Kilbourne, J., Powell, T., & Moore, R. (2011). Normalized names for clinical drugs: Rxnorm at 6 years. *Journal of the American Medical Informatics Association*, *18*, 441–448.

Niculescu, M. A., Ruseti, S., & Dascalu, M. (2022). Rosummary: Control tokens for romanian news summarization. *Algorithms*, *15*, 472.

Nothman, J., Qin, H., & Yurchak, R. Stop word lists in free open-source software packages. In: In *Proceedings of workshop for NLP open source software (NLP-OSS)*. 2018, 7–12.

Ohri, K., & Kumar, M. (2024). Supervised fine-tuned approach for automated detection of diabetic retinopathy. *Multimedia Tools and Applications*, *83*, 14259–14280.

Pampari, A., Raghavan, P., Liang, J. J., & Peng, J. (2018). Emrqa: A large corpus for question answering on electronic medical records.

Paulsen, D., Govind, Y., & Doan, A. (2023). Sparkly: A simple yet surprisingly strong tf/idf blocker for entity matching. *Proceedings of the VLDB Endowment*, *16*, 1507–1519.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Louppe, G., Prettenhofer, P., Weiss, R., Weiss, R. J., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *ArXiv*, *abs/1201.0490*.

Peters, M. E., Neumann, M., Zettlemoyer, L., & Yih, W.-t. (2018). Dissecting contextual word embeddings: Architecture and representation.

Pi, X., Wang, B., Gao, Y., Guo, J., Li, Z., & Lou, J.-G. (2022). Towards robustness of text-to-sql models against natural and realistic adversarial table perturbation.

Pollard, T. J., Johnson, A. E., Raffa, J. D., Celi, L. A., Mark, R. G., & Badawi, O. (2018). The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, *5*, 1–13.

Pourreza, M., Li, H., Sun, R., Chung, Y., Talaei, S., Kakkar, G. T., Gan, Y., Saberi, A., Ozcan, F., & Arik, S. O. (2024). Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *arXiv preprint arXiv:2410.01943*.

Pourreza, M., & Rafiei, D. (2024). Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, *36*.

Price, P. J. Evaluation of spoken language systems: The atis domain. In: In *Human language technology - the baltic perspectiv*. 1990.

Qin, B., Hui, B., Wang, L., Yang, M., Li, J., Li, B., Geng, R., Cao, R., Sun, J., Si, L., et al. (2022). A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*, 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*, 1–67.

Ragazzi, L., Italiani, P., Moro, G., & Panni, M. What are you token about? differentiable perturbed top-k token selection for scientific document summarization. In: In *Findings of the association for computational linguistics acl 2024*. 2024, 9427–9440.

Raghavan, P., Liang, J. J., Mahajan, D., Chandra, R., & Szolovits, P. Emrkbqa: A clinical knowledge-base question answering dataset. In: In *Workshop on biomedical natural language processing*. 2021.

Rangan, K. K., & Yin, Y. (2024). A fine-tuning enhanced rag system with quantized influence measure as ai judge. *Scientific Reports*, *14*.

Ren, J., Fort, S., Liu, J., Roy, A. G., Padhy, S., & Lakshminarayanan, B. (2021). A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*.

Ren, J., Luo, J., Zhao, Y., Krishna, K., Saleh, M., Lakshminarayanan, B., & Liu, P. J. Out-of-distribution detection and selective generation for conditional language models. In: In *The eleventh international conference on learning representations*. 2022.

Robertson, S., Zaragoza, H., et al. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, *3*, 333–389.

Rumelhart, D., Smolensky, P., McClelland, J., & Hinton, G. Sequential thought processes in pdp models. In: In *Parallel distributed processing: Explorations in the microstructures of cognition*. Vol. 2. 1986, pp. 3–57.

Saxena, A., Kochsiek, A., & Gemulla, R. (2022). Sequence-to-sequence knowledge graph completion and question answering. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*, 2673–2681.

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Shi, T., Tatwawadi, K., Chakrabarti, K., Mao, Y., Polozov, O., & Chen, W. (2018). Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Shin, H. J., Eom, D.-H., & Kim, S.-S. (2005). One-class support vector machines—an application in machine fault detection and classification. *Computers & Industrial Engineering*, *48*, 395–408.

Song, Y., Wang, G., Li, S., & Lin, B. Y. (2024). The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism. *arXiv preprint arXiv:2407.10457*.

Sorokin, D., & Gurevych, I. (2018). Mixing context granularities for improved entity linking on question answering data across entity categories. *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*.

Steinhaus, H. (1957). Sur la division des corps matériels en parties: Bulletin de l'académie polonaise des sciences.

Stengel-Eskin, E., & Van Durme, B. (2023). Calibrated interpretation: Confidence estimation in semantic parsing. *Transactions of the Association for Computational Linguistics*, *11*, 1213–1231.

Suhr, A., Chang, M.-W., Shaw, P., & Lee, K. Exploring unexplored generalization challenges for cross-database semantic parsing. In: In *Proceedings of the 58th annual meeting of the association for computational linguistics*. 2020, 8372–8388.

Sun, H., Cohen, W. W., & Salakhutdinov, R. (2021). Conditionalqa: A complex reading comprehension dataset with conditional answers. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Sun, S., Zhuang, S., Wang, S., & Zuccon, G. (2024). An investigation of prompt variations for zero-shot llm-based rankers. *arXiv preprint arXiv:2406.14117*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Tan, K. L., Lee, C. P., Anbananthen, K. S. M., & Lim, K. M. (2022). Roberta-lstm: A hybrid model for sentiment analysis with transformer and recurrent neural network. *IEEE Access*, *10*, 21517–21525.

Thakur, S., Gupta, B., Mathur, U., & Bansal, D. Electronic health record systems for enhanced medical care: A survey. In: In *2023 international conference on intelligent systems for communication, iot and security (iciscois)*. 2023, 257–262.

Toporkov, O., & Agerri, R. (2024). On the role of morphological information for contextual lemmatization. *Computational Linguistics*, *50*, 157–191.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *ArXiv*.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. Attention is all you need. In: In *Neural information processing systems*. 2017.

Wahle, J. P., Ruas, T., Kirstein, F., & Gipp, B. (2022). How large language models are transforming machine-paraphrased plagiarism. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2019). Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *Annual Meeting of the Association for Computational Linguistics*.

Wang, B., Gao, Y., Li, Z., & Lou, J.-G. Know what i don't know: Handling ambiguous and unknown questions for text-to-sql. In: In *Findings of the association for computational linguistics: Acl 2023*. 2023, 5701–5714.

Wang, C., Tatwawadi, K., Brockschmidt, M., Huang, P.-S., Mao, Y., Polozov, O., & Singh, R. (2018). Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Wang, P., Shi, T., & Reddy, C. K. Text-to-sql generation for question answering on electronic medical records. In: 2020, 350–361.

Wang, Q., Liu, L., Jing, C., Chen, H., Liang, G., Wang, P., & Shen, C. Learning conditional attributes for compositional zero-shot learning. In: In *2023 ieee/cvf conference on computer vision and pattern recognition (cvpr)*. 2023, 11197–11206.

Witteveen, S., & Andrews, M. Paraphrasing with large language models. In: In *Proceedings of the 3rd workshop on neural generation and translation*. 2019, 215–220.

Wu, Z., Kao, B., Wu, T.-H., Yin, P., & Liu, Q. Perq: Predicting, explaining, and rectifying failed questions in kb-qa systems. In: In *Proceedings of the 13th international conference on web search and data mining*. 2020, 663–671.

Xia, Y., Huang, Y., Qiu, Q., Zhang, X., Miao, L., & Chen, Y. (2024). A question and answering service of typhoon disasters based on the t5 large language model. *ISPRS Int. J. Geo Inf.*, *13*, 165.

Xie, Y., Aggarwal, K., & Ahmad, A. (n.d.). Efficient continual pre-training for building domain specific large language models.

Xu, X., Liu, C., & Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Yang, S., Su, Q., Li, Z., Li, Z., Mao, H., Liu, C., & Zhao, R. (2024). Sql-to-schema enhances schema linking in text-to-sql.

Yang, Z. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yavuz, S., Gür, I., Su, Y., & Yan, X. What it takes to achieve 100% condition accuracy on wikisql. In: In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2018, 1702–1711.

Yin, P., & Neubig, G. (2017). A syntactic neural model for general-purpose code generation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yin, P., & Neubig, G. (2018). Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Yu, T., Li, Z., Zhang, Z., Zhang, R., & Radev, D. (2018a). Typesql: Knowledge-based type-aware neural text-to-sql generation. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., & Radev, D. (2018b). Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2018.

Yu, Z., Johnson, T. R., & Kavuluru, R. Phrase based topic modeling for semantic information processing in biomedicine. In: In *2013 12th international conference on machine learning and applications*. *1*. 2013, 440–445.

Zelle, J. M., & Mooney, R. J. Learning to parse database queries using inductive logic programming. In: In *Aaai/iaai, vol. 2*. 1996.

Zeng, J., Lin, X. V., Xiong, C., Socher, R., Lyu, M. R., King, I., & Hoi, S. C. (2020). Photon: A robust cross-domain text-to-sql system. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In: In *Proceedings of the 37th international conference on machine learning*. *119*. 2020, 11328–11339.

Zhang, R., Yu, T., Er, H. Y., Shim, S., Xue, E., Lin, X. V., Shi, T., Xiong, C., Socher, R., & Radev, D. (2019). Editing-based sql query generation for cross-domain context-dependent questions. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Zhang, X., Zhang, S., Liang, Y., Chen, Y., Liu, J., Han, W., & Xu, J. (2023). A quality-based syntactic template retriever for syntactically-controlled paraphrase generation. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Zhang, Y., Dong, X., Chang, S., Yu, T., Shi, P., & Zhang, R. (2020b). Did you ask a good question? a cross-domain question intention classification benchmark for text-to-sql. *arXiv preprint arXiv:2010.12634*.

Zhao, C., Su, Y., Pauls, A., & Platanios, E. A. Bridging the generalization gap in text-to-SQL parsing with schema expansion. In: In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)*. 2022, 5568–5578.

Zheng, M., Pei, J., & Jurgens, D. (2023). Is" a helpful assistant" the best role for large language models? a systematic evaluation of social roles in system prompts. *arXiv preprint arXiv:2311.10054*, *8*.

Zheng, Y., Chen, G., & Huang, M. (2020). Out-of-domain detection for natural language understanding in dialog systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 1198–1209.

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *ArXiv*.

Zhu, X., Li, Q., Cui, L., & Liu, Y. (2024). Large language model enhanced text-to-sql generation: A survey.