Building a Trustworthy Question Answering System for Covid-19 Tracking

Yiqing Liu

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Science and Applications

Chandan K. Reddy
Chang-Tien Lu
Clifford A. Shaffer

August 13, 2021
Arlington, Virginia

Keywords: Information Retrieval, Question Answering, Database, Machine Learning,
Natural Language Processing, Healthcare, Covid-19 Dashboard

Building a Trustworthy Question Answering System for Covid-19 Tracking

Yiqing Liu

ABSTRACT

During the unprecedented global pandemic of Covid-19, the general public is suffering from inaccurate Covid-19 related information including outdated information and fake news [1]. The most used media: TV, social media, newspaper, and radio are incompetent in providing certitude and flash updates that people are seeking. In order to cope with this challenge, several public data resources that are dedicated to providing Covid-19 information were born. They rallied with experts from different fields to provide authoritative and up-to-date pandemic updates. However, the general public cannot still make complete use of such resources since the learning curve is too steep, especially for the aged and under-aged users.

To address this problem, in this Thesis, we propose a question answering system that can be interacted with using simple natural language-based sentences. While building this system, we investigate qualified public data resources and from the data content they are providing, and we collect a set of frequently asked questions for Covid-19 tracking. We further build a dedicated dataset named CovidQA for evaluating the performance of the question answering system with different models. Based on the new dataset, we assess multiple machine learning-based models that are built for retrieving relevant information from databases, and then propose two empirical models which utilize the pre-defined templates to generate SQL queries. In our experiments, we demonstrate both quantitative and qualitative results and provide a comprehensive comparison between different types

of methods. The results show that the proposed template-based methods are simple but effective in building question answering systems for specific domain problems.

Building a Trustworthy Question Answering System for Covid-19 Tracking

Yiqing Liu

GENERAL AUDIENCE ABSTRACT

During the unprecedented global pandemic of Covid-19, the general public is suffering from inaccurate Covid-19 related information including outdated information and fake news [1]. The most used media: TV, social media, newspaper, and radio are incompetent in providing certitude and flash updates that people are seeking. In order to cope with this challenge, several public data resources that are dedicated to providing Covid-19 information were born. They rallied with experts from different fields to provide authoritative and up-to-date pandemic updates. However, there is room for improvement in terms of user experience.

To address this problem, in this Thesis, we propose a system that can be interacted with using natural questions. While building this system, we evaluate and choose six qualified public data providers as the data sources. We further build a testing dataset for evaluating the performance of the system. We assess two Artificial Intelligence-powered models for the system, and then propose two rule-based models for the researched problem. In our experiments, we provide a comprehensive comparison between different types of methods. The results show that the proposed rule-based methods are simple but effective in building such systems.

Dedication

I dedicate my thesis work to Dr. Chandan Reddy and Mrs. Ping Wang who helped me prepare for graduate research. I am also grateful to my wife for giving me unconditional support while I was preparing the whole thesis and oral defense.

Acknowledgements

I want to thank Dr. Chandan Reddy for providing me the opportunity to engage in graduate thesis study. I also want to thank him for his understanding during my hardship in the quarantine times. This work cannot be finished without his support and help. I want to thank Dr. Chang-Tien Lu and Dr. Clifford A. Shaffer for the additional guidance and expertise. Finally, I would like to thank Mrs. Ping Wang for providing thorough work reviews and constructive advice regularly.

Contents

List of Figures

List of Tables

1 Introduction

The coronavirus disease (Covid-19) has taken the world by storm and has been declared a global pandemic by the World Health Organization (WHO). As of July 27, 2021, an astounding 198 million cases have been documented worldwide with over 4.22 million deaths [2]. In addition, the pandemic has prompted massive social change with respect to numerous quarantine measures and the shutting down of many businesses globally, inciting widespread paranoia among researchers, public officials, and most importantly, the general public. Because of the current pandemic, dependency on internet sources for reliable information has also seen a dramatic increase. However, even with this heightened reliance on internet resources, much of this information is misrepresented by inaccurate sources and is extremely difficult and time-consuming to retrieve as it is present across a wide range of databases and web authorities.

1.1 Public Data Sources for Covid-19 Tracking

Compared to other typical data science applications, the epidemiologic data is characterized by their distributed nature and are rapidly updated dynamically [3]. Government and local public health surveillance systems around the world operate with completely different methodologies and levels of transparency. As a result, it is not realistic to build a unified database for all the distributed data. Further, given the nature of the epidemic, not only the data itself can be updated rapidly, but also the ways how people present and organize the data can be altered over time: it is a gradual process for humans to learn about the disease.

Government-funded, Institute-funded, and crowd-funded organizations are the main types of groups in addressing such data challenges. Such groups often team up with epidemiology experts, data scientists, and database/web developers, offering access to their work to everyone. They constructed a closed-loop methodology for specific application in epidemiologic data from raw data gathering, data pre-processing, data cleaning, schema defining, data integration, data transformation, to data exploration. In addition, some of them are able to provide the data in an advanced way for specific applications. For example, they could present the data in an interactive interface with integrated spatial information and advanced chart/diagram elements [2]. Furthermore, with the collection of historical data, some of them developed prediction models to provide different perspectives in epidemic forecasting.

1.2 Problem Statement

The general public needs the guidance of accurate and up-to-date tracking information to keep them safe during the pandemic, for example, if there is an outbreak going on in a local community, people should consider staying at home as much as possible for a certain period of time. Current public data resources simplified this issue immensely, but that's not all. Overall, such public data resources are still too complex and specialized for most general people.

Due to the limitation of the team size and their working scope, there is no one ideally centralized data resource that covers all information that most people are interested in. Hence for the general public, even with the help of the existing excellent data collection works, most of the information they collect are still in "distributed" status. They need to turn to multiple data resources or authorities to get the answers they need. They have to learn about what data resource covers what kind of information, which is way too

confusing, not to mention that the covered topics might change over time as the teams could adjust their goals at different stages of the pandemic.

On the other hand, as a typical paradigm, the pandemic tracking-related data are usually saved in a predefined database schema with numbers of tables. For well-educated users, they know how to retrieve the desired information from such tables. Usually, experienced users know what table to choose and know how to search, rank and filter the table. After such operations, they know which field displays the answers they want. They can also handle most of the terminologies or abbreviations from the healthcare domain and data science domain that appear on the tables or interface. But the problem is that not all users are able to finish such actions with ease, especially for the aged and under-aged users [4]. Even for the experienced users, the potential changes and extensions made on the system require them to adapt, or they are not able to fully utilize the powerful resources. Sophisticated user tutorials and interfaces can mitigate such inconvenience. Interactive graphic interfaces are provided by certain data resources, and they are advancing the user experience by a huge leap. Users do not have to tweak the tables directly. Instead, they can interact with the graphic interface like buttons, tabs, maps, and diagrams. However, such extensions increase the requirement of the computational ability to the devices that access the data resources, and the system responding time is much worse than simply using the table interface.

1.2.1 Ways to Access Information

Aside from the professional data resources, multiple alternative media are available for accessing the pandemic information. Generally, there are two types of information media: traditional media and Internet media. Traditional media such as short message services, TV, and newspapers are playing a big role in information dissemination, especially in some extreme cases like natural disasters or unavailability of

Internet services. Internet media, on the other hand, is a more favorable type with the surge of information innovation. It now has better coverage and higher frequency of utilization among the general public.

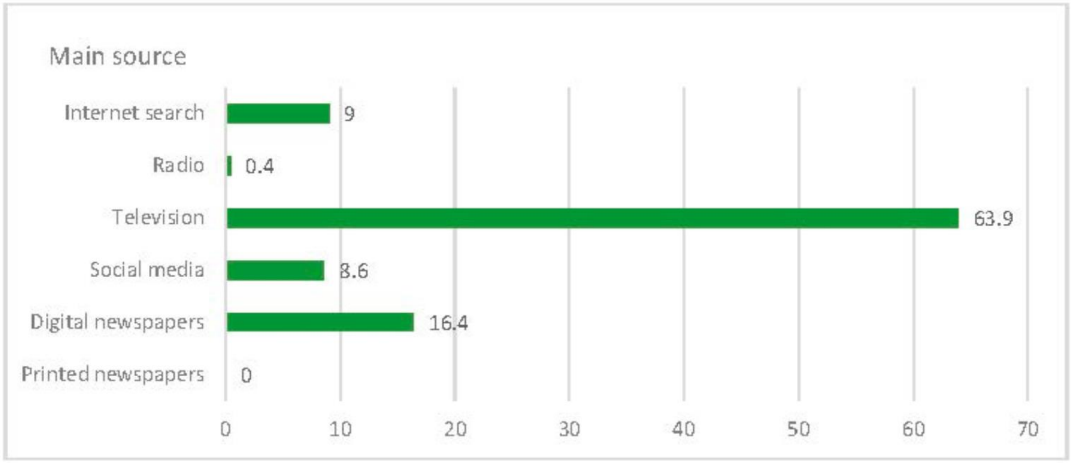


Figure 1: Which media is your main information source about the Covid-19 global crisis in the last seven days (%) [5]?

Figure 1 shows the preference of the survey respondents for the favored media source for the Covid-19 related information. The result that traditional media is having an overwhelmingly picking rate shows a major distrust towards the Internet from the respondents. However, people can only be passively exposed to the information from traditional media, and it is not always easy to get the information you want from it.

1.3 Research Question

In order to effectively combat these issues, a reliable, precise, and efficient method is required to leave the user well-informed about the Covid-19 pandemic. Thus, we propose an implementation of a question answering system for Covid-19 dashboards. To be more specific, the research questions to be solved are

as follows: (1). Carry out a survey on the existing public data resources for Covid-19, placing emphasis on timeliness and authoritativeness; (2). With the trustworthy data resources, build a downstream, user-friendly system that can answer Covid-related, natural questions from users; (3). Build a dedicated database to evaluate the performance of the question answering system.

For the general public, the most comfortable way to get information is to ask what they want through natural language questions. Question answering (QA) systems have recently gained a lot of traction in the research community [6], the overarching target of QA systems delves into both concisely and correctly answering a given natural language question through the use of either a knowledge base or a compilation of natural language sources. For QA tasks, various types of questions are present, but, in this work, we mainly focus on factoid questions where the questions typically possess a specific short answer.

Different from typical information retrieval techniques, the existing question answering systems utilize many progresses made in the field of Natural Language Processing (NLP) including natural language parsing, question type classification, and terminology recognition [7]. Some even make use of complex logical reasoning mechanisms to infer the answer from knowledge bases. From the perspective of the domain, the question answering system can be divided into two types: closed-domain question answering and open-domain question answering. The Covid-19 topic we are focusing on is a closed-domain question answering since it is dedicated to answering questions in a narrowed knowledge field, hence the performance of systems can be improved when structured answer/data collections are introduced.

Model Name	Model Type	Model Output	Support Multiple Tables
GeoQuery [8]	Template-based	SQL Query	Yes
COVIDQA	Template-based	SQL Query	Yes
TAPAS [9]	Machine Learning	Answers	No
TypeSQL [10]	Machine Learning	SQL Query	Yes
TREQS [11]	Machine Learning	SQL Query	Yes

Table 1: Some of the representative modeling approaches for question answering databases.

Database technology is the most commonly used media for storing information. Table 1 shows a list of representative modeling approaches for a QA system to retrieve information from databases. Structured Query Language (SQL) is a dedicated program-designing language for accessing, modifying, and managing information stored in database systems [12]. The goal for most of the question answering models is to predict a corresponding SQL query for natural language questions. Table Parser (TAPAS) developed by the google NLP teams is an exception to this [9]. TAPAS model is able to directly run with structured tables in databases for some specific operations, it retrieves information from the database without SQL and has outstanding accuracy on numbers of benchmarks [9].

Other than models like TAPAS, most of the modeling approaches abstract this problem as a sort of "translation" task. The SQL rules can be taken as one kind of language and the main goal is to translate the natural question to another language with precise, structured grammar. Template based modeling approaches are essentially using pre-defined question templates and SQL templates to match the questions asked. It defines multiple placeholders in the SQL templates and fills them out with identity recognition mechanisms. Such solutions were proven to be effective in some specific closed-domain applications

(GeoQuery) [8]. For some inherent drawbacks of template-based modeling approaches that will be discussed in chapter 6 of this work, machine learning-based approaches are developed. Machine learning is a powerful technology for NLP applications that leverages the potential of neural networks [13]. In this special translation task, typical machine learning solutions are modeling the question-to-SQL problem in a sequence-to-sequence manner, and often use the attention mechanism to capture the internal correlations between the elements that appeared in the natural questions.

In order to construct a QA model that is tailor-made for Covid-19 questions, we need to further establish our database for our system (for the training of machine learning-based models and the system evaluation) by gathering both reliable and comprehensive Covid-19 sources. After solidifying our databases, we defined distinct question templates and correlated SQL templates that contain entities that correspond to the type of data for each database as much as possible. Besides, the diversity of questions and the templates' variations are also important criteria.

1.4 Contributions and Organization

The primary contributions of this work are as follows:

(1) Conduct a comprehensive investigation of the current open-access data resources for Covid-19 and select six databases as the trusty raw data source for our proposed QA system.

(2) Create a large-scale dataset for question answering about Covid-19 databases based on template generation.

(3) Propose two template-based question answering methods which consist of five main elements: 1) Matching an input question to a predefined question template 2) recognizing the specific entities present

in the given question 3) identifying the correct query template that matches with the question template 4) populating the query template with the proper values and 5) retrieving the correct answer by running the populated query.

(4) Evaluate the performance of the template-based methods along with other two state-of-the-art methods for retrieving information from tables and analyze their advantages and disadvantages.

We organize the rest of this work in the following manner: In chapter 2, we discuss previous related works within the field of question answering systems. Chapter 3 offers an in-depth breakdown of the databases employed and also reports the methodology for the question and data generation steps. Chapter 4 outlines the template-based matching steps and the mechanisms of the tested machine learning models, namely TAPAS model and TREQS model. Chapter 5 analyzes the results of the template-based methods and compares the results with machine learning methods while chapter 6 summarizes and explains the potential future work in this direction.

2 Related Work

With QA systems garnering great research attention [6], numerous advancements in the field have come to fruition, bolstering the performance of such systems. Template-based QA models, in specific, operate by linking a general template to a specific question and provide multiple benefits. Template-based QA models not only enable the system to become more maintainable by offering a straightforward method of question alteration, but they also reduce the vast ambiguity involved with natural language through entity placeholders.

With respect to previous template-based question answering systems, SPARQL templates reflect the structure of given input questions [14]. These templates require a table that maps natural language jargon to the vocabulary that is present in the query template. This methodology has been proved with Resource Description Framework (RDF) data to be an effective approach to addressing the issue of maintaining the semantic structure of a given natural language question within a query template [15]. Deep learning techniques have also been adapted to template-based systems. A recursive neural network-based approach has been proposed to effectively map natural language questions to their corresponding question template, enabling one to bypass the need for feature engineering of input questions. [16]

Template-based approaches have also made headway in question generation for the improvement of QA models. One of the most popular methods of QA has been to calibrate pre-trained QA models on a

specified dataset for a given task. The problem with this approach, however, is that acquiring such data is expensive. Fabbri et. al [17] have offered a question generation method that revolves around utilizing a template of a related sentence rather than the original text. Training unsupervised QA models on this template-based generated dataset has shown to relatively enhance performance.

SQLNet developed by Xu et. al [18] combined template-filling and machine learning approaches in an effective manner. This modeling approach defines a concise "sketch" for SQL command and changes the sequence-to-sequence modeling to sequence-to-set operation, which overturned the serialized problem that was a problem when the training set contains multiple equivalent SQL commands. They also came up with a column attention mechanism to dig the relationship between keywords and specific columns in the databases. In addition, Yu et. al [10] with the improved TypeNet has a more concrete sketch definition and uses certain entities and numbers to fill the sketch as a different type of data.

In regards to the efforts in the area of recent Covid-19 QA models, Oniani et. al [19] have constructed a web-based conversational Covid-19 chatbot. The chatbot integrates natural language processing and artificial intelligence (AI) techniques to instantly specify answers to given Covid-19 questions. The chatbot's performance was thoroughly tested across four different embedding approaches which included tf-idf, BERT and Bio-BERT, to polish the generated answers. CovidASK is another type of Covid-19 question answering system that has been recently developed. This system couples QA procedures with biomedical text mining in order to offer accurate answers in real-time. It incorporates Bio-BERT methods with an entity-based search engine in order to draw out necessary biomedical entities from documents.

Co-search is a ranking-based semantic search engine that takes in a search query (or question) and returns relevant scientific coronavirus articles [20]. To perform this task, Co-search first retrieves documents with an SBERT model and formulates a comprehensive document list, labeling each document with a specific retrieval score. Then, through a QA model and an abstractive summarizer, answers and summaries for each question are generated. Finally, a ranker then returns an ordered document set by adjusting each document's score based on the extent that each document provides the proper answer and summary.

3 Data Preparation

In this chapter, the overview of the selected Covid-19 data resources is introduced first. Then, we depict the workflow for building the CovidQA database, from the template definition to database generation. Finally, we also present a basic data statistic and data example for the generated database.

3.1 Overview of Covid-19 Database Resources

After we posed the problem to be solved, we came up with multiple principles in data resource selection for the QA system, based on personal experience: (1). Select databases from reliable authorities and the data sources should be public and transparent. (2). Selected databases should be well-managed and should be updated rapidly to ensure the answer to the query does not have any outdated information. (3). There should be a reasonable level of diversity among selected databases and minimum overlap from different databases. Overall, the selected databases should be able to reflect the most up-to-date Covid-19 information from different perspectives so that a wide range of questions can be answered by our system.

3.1.1 Data Resources Evaluation

In the process of finding the qualified data sources, we evaluated numbers of data providers with the aforementioned principles. Global COVID-19 Tracker is a non-profit service provided by 1point3acres [21], an online community for foreign students in North America. Comprehensive data like new case number, vaccination number, positive rate and test rate are provided on a daily basis. This team also brings

multiple intuitive and diverse diagrams to reflect the situation from different perspectives. Their main data source is the daily Covid-19 reports from the public health surveillance systems at all levels, but they also accept individual data submission, Figure 2 allows everyone to submit the tracking information, then the volunteers will review the submissions manually. This practice is efficient in terms of increasing the frequency of information updating, but it also violates our principle that data should be provided from reliable authorities.

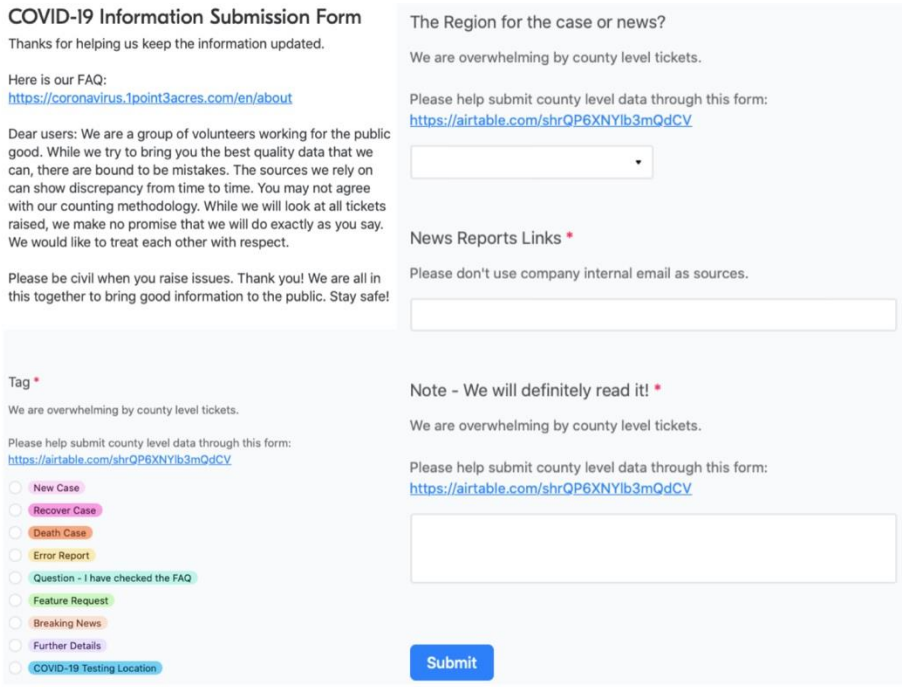


Figure 2: Information submission form of Global COVID-19 Tracker by 1pont3acres [22].

Baidu, known as a tech giant in China, is contributing its strength in fighting Covid-19 with the web service named Epidemic Big Data Live Report [23]. It provides numbers of existing cases, new cases, total cured patients, and death on a national/state scale. One advantage of this service is that it highlights the hot zones which are undergoing local breakout. Nevertheless, as stated on the data description page

from the official site, there will be unavoidable delay for the Covid-19 tracking information, in particular, the data for the regions that are outside China, because they need extra manpower and time to verify the data sources. Since the timeliness cannot be guaranteed from the Epidemic Big Data Live Report, we had to exclude this data provider from our system.

Covid in the U.S. is an excellent data provider with the best user experience among the data resources we evaluated [24]. This project is maintained by The New York Times and updated every day. The published data includes the daily cumulative number of cases and deaths in each county and state in the U.S., and they derive much advanced data and deliver them in multiple well-organized tables and charts. Overall, Covid in the U.S. is of high quality in terms of information delivery. The reason we did not choose it as our data source is that most of their information is overlapped with the databases directly from the CDC. We need to choose data providers that is able to lend a higher information diversity.

3.1.2 Selected Databases

Following part is the description of the selected databases.

Database 1: Covid-19 Data Repository by the Center for Science and Engineering (CSSE) at John Hopkins University [2]**.**

The Covid-19 Data Repository is monitored by the JHU CSSE and is updated daily with information on a global, state, province, and county scale, pulling data from reliable sources such as the CDC and WHO. Specifically, the repository mainly provides numerical information relating to Covid-19 cases, deaths,

tests, and various rates. The two tables that were utilized from this repository include the global database and the United States database. The global database returns data related to total confirmed cases, active cases, deaths, recovered cases, the incidence rate, and the case fatality rate. On the other hand, the United States database covers the same information as the global database but at a State level for the United States.

Database 2: Covid Racial Tracker/Covid Tracking Project [25]**.**

The Covid Tracking Project is a volunteer organization from the Atlantic magazine that collects Covid-19 testing and patient conditions from all over the United States. The Covid Tracking Project offers three primary tables: the United States table, the Individual States Table, and the Racial Data Dashboard. The United States table and the Individual States table are both updated on a daily basis and provide relevant data relating to the total number of hospitalizations, ICU patients, patients on ventilators on a state/national standard scale. These tables also report the daily and total negative, positive, and total tests are done. The Racial Data Dashboard is a joint cooperation between the Covid Tracking Project and the Boston University Center and is updated twice per week. The table summarizes the statistical breakdown of Covid-19 current and fatal cases by race in all US states.

Database 3: Definitive Healthcare: USA Hospital Beds [26]**.**

This database is supervised by Definitive Healthcare, a provider of data and analytics for health organizations. The dataset specifically receives data from various hospitals and health organizations across the United States and is necessary for understanding the impact of Covid-19 on the health system. The

database primarily offers a table with the number of licensed beds, staffed beds, and ICU beds on a county/state/national level.

Database 4: CDC Covid Data Tracker [27]**.**

The CDC Covid Data Tracker is operated by the CDC in order to effectively visualize trends in the spread of Covid-19 strictly in the context of the United States. The tracker provides information ranging from the number of cases to future forecasts about Covid-19 deaths per state and is updated whenever new information is gathered. The three specific data tables that were pulled from the CDC are the demographics database, the forecast table, and the mobility index dataset. The demographics database looks at the national percentage distribution of Covid-19 cases and deaths by sex, age, and race/ethnicity. The forecast table offers predictions from multiple research models regarding the cumulative number of deaths up till the next month on both a state/national scope. Finally, the mobility index dataset outlines the percentage change in social activities, such as parks and workplaces, from the start of the year due to Covid-19 on a county/state/national level.

Database 5: CDC Provisional Covid-19 Death Counts by County and Race [28].

The CDC Provisional Covid-19 database is monitored by the National Center for Health and Statistics. The database receives data from only counties with over 100 cumulative Covid-19 deaths and aims to help further the understanding of how the impact of Covid-19 varies by race. The dataset consists of one table, outlining the percentage racial breakdown of Covid-19 deaths by county and the total deaths reported per county.

Database 6: Our World in Data [29]**.**

Our World in Data is a scientific research organization that concentrates its efforts on tackling large-scale global issues. The organization provides a database that is generally updated twice a week and contains data ranging from total cases to the daily tests done by each country around the globe. Four tables were utilized in order to comprehend the testing infrastructure and the widespread influence of Covid-19 on a global scale: the total test table, the daily test table, the short-term positivity table, and the daily testing rate table. The total test table gives the cumulative tests performed by each country, while the daily test table provides the number of tests each country does per day. The short-term positivity rate table returns the daily positive rate of tests as a 7-day rolling average. Finally, information about the number of tests done per day per 1000 people is answered by the daily testing rate table.

3.2 Question Templates Definition

After reviewing the tables from each database that were going to be utilized, we formulated a list of specific questions referring to each repository's distinct information. For instance, since database 1 provides relevant case data on county, state, and global levels, questions that are asked about this type of information were added to the Database 1 list. There are two main types of questions that each database question list consists of, namely, retrieval questions and reasoning questions. Retrieval questions focus on directly retrieving the data from a given table. On the other hand, reasoning questions concentrate on asking for the minimum and maximum values in the table and require an “Order by” command in SQL to accomplish this.

After each database receives its own question list, the question templates are then defined. However, in order to begin generating these question templates, certain "entities" had to be defined. An "entity" refers to a broader aspect that the data relates to. For instance, cases and deaths could be generalized to a "Case Entity" and hospitalization rate, testing rate, etc. could be characterized by a "Rate Entity." After the entities were designated, the specific questions were then grouped into a comprehensive question template based on whether or not they shared the same type of entities. Various types of data were generalized through these entities primarily due to the fact that human annotation is both expensive and time-consuming. As a result, it is not reasonable to generate question-SQL pairs for all the six databases. Rather, the goal of utilizing entity-based question templates is to map a given natural language question to a question template in order to produce a viable SQL query. Through entity-based question templates, any natural language question can be assigned to a question template. This eliminates the need to manually formulate multiple question-SQL pairs for every single question while also providing an efficient mechanism by which a broader range of questions could be addressed for the Covid-19 dashboard.

Additionally, we further narrowed down the number of entity-based question templates through a Null-Entity Method. This method suggests that each template should try to include as many entities as reasonably possible which could be treated as "Null" when unnecessary. For instance, in the following question-template "What is the number of (Case Entity) in (State Entity) (Time Entity)?" given an input question that does not denote the time entity, the entity within the question template can be marked as "null", rather than having a separate question-template without the Time Entity. More information about the defined entity type lists can be found in Appendix 1.

After the entity-based question templates were created, we finalize a lookup table that contains the possible specific forms that the entities can take up in order to efficiently retrieve the necessary values from a given input question. For example, with the Rate Entity, all the relevant rates, such as hospitalization rate, testing rate, etc. were loaded into a lookup table. However, certain entities could be phrased in multiple variations. For example, the State Entity in a given question could be expressed as "California" or "in the state of California," making it infeasible to simply list all combinations of specific values and the phrasing variations within one lookup table. Thus, to solve this issue, for entities that can be presented in different ways, we label a broader template within the first lookup table and input the specific values inside the second lookup table. For instance, with the previous example of the State Entity, we would designate the two phrases as "(state)" or "in the state of (state)" and retrieve "California" from the second lookup table under the "State" column, containing the names of all possible states. More information about defined entity type lists can be found in Appendix 2.

3.3 SQL Query Templates Definition

In this part, we discussed the steps involved in generating the corresponding SQL query templates.

We first begin by constructing a SQL template for each of the previously defined entity-based question templates. Each SQL template will have consistent placeholder values that reflect the entities present in the question templates. For instance, looking at the question-query table, it can be seen how the first question template in Database 2's list contains "State Entity" and how this is demonstrated in the SQL template. The reasoning behind this methodology lies in the question answering task. When given a natural language question, we aim to isolate the entity names from the questions and then populate these values

into the query template. Additionally, we administer a “Column” notation within the query templates in situations where the column name in the database is different from the specific entity values. This “Column” notation indicates that when we find a specific value within an input question, we search for this within an individual lookup that maps the specific value to its complementary column name in the database. We then construct the actual query by replacing the entity and the “Column” notation within the query template with the retrieved item. Furthermore, since certain SQL queries are dependent on the specific candidate values of the entities, one single SQL template for each question template cannot guarantee the retrieval of the necessary results. For example, the “Time Entity” can refer to either a single date or a range of dates. In order to address this complicated scenario, we generated multiple SQL templates for some question templates (see Appendix B for more details).

3.4 CovidQA Data Generation

With a comprehensive list mapping the question templates to query templates, we began to populate the question and query templates with specific entity values. More specifically, we generated a real question and its corresponding SQL query. In order to do this, the lookup tables that contain the candidate values for each entity were loaded. Next, the question template and query template that will be utilized for the data generation was specified. We then identified the entities present in the question template and randomly acquired specific candidate values from the lookup tables for each entity within the question template. After selecting the candidate values, we populated the question and SQL query, generating a real question and running the generated query to retrieve the specific data from each database. Furthermore, since we do not want to generate all possible questions for each question template, we placed a limit on

the number of questions that will be generated. This range of real question numbers are depending on the question templates. We finalized the data generation step by gathering the populated questions and queries into a .json file with other related information, such as the question and query template used, the entities within the question template, and the chosen candidate values for each entity. Figure 3 shows a fragment of the generated dataset.


```
[
  {
    "question_template_id": "2q4",
    "entities_type": [
      "Testing Entity",
      "State Entity",
      "Month"
    ],
    "entities": [
      "total tests",
      "Ohio",
      "February"
    ],
    "question": "What is the number of total tests done in Ohio in 8th, February?",
    "sql": "Select totalTestResults from db2State where date = '20200208' and state = \"OH\" ",
    "real_question": "What is the number of (Testing Entity) done in (State Entity) in (Month)?",
    "database": "database 2"
  }
]
```

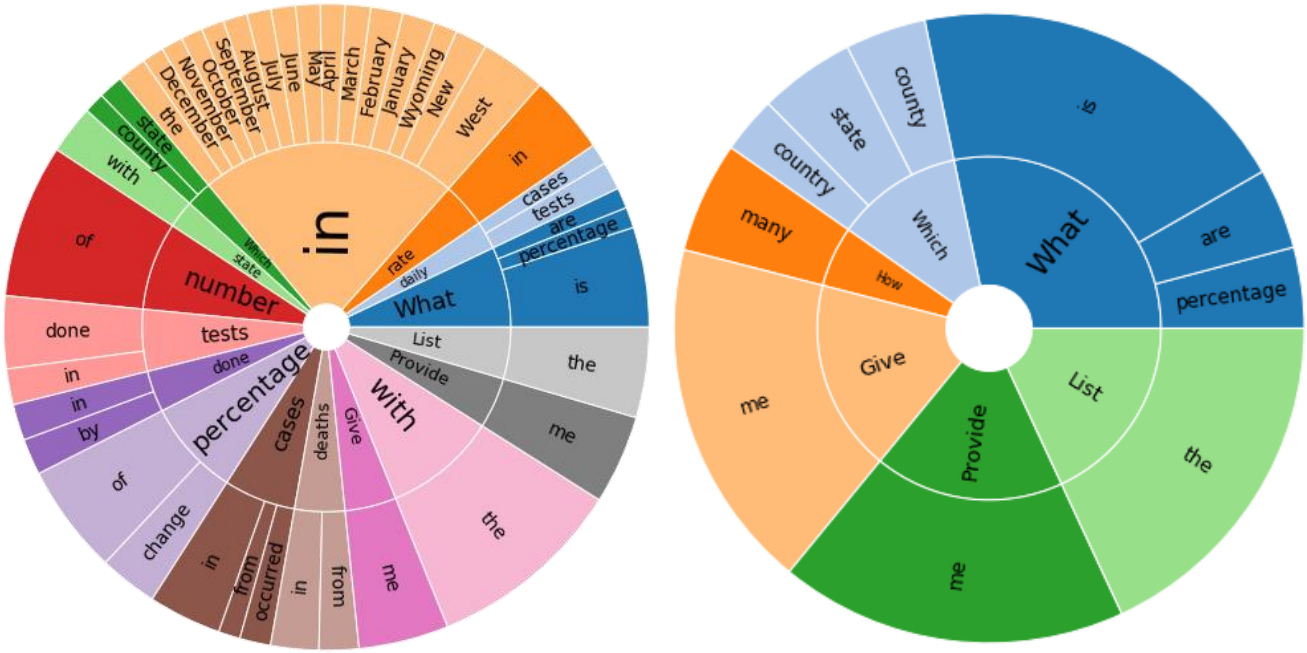

Figure 3: A fragment of the CovidQA dataset.

3.5 Data Statistics

The primary objective of our template-based Covid-19 question answering system is to efficiently obtain data from multiple Covid-19 dashboards in a comprehensive manner that is adaptable to the vast intricacies of natural language. As such, all the questions generated in our CovidQA dataset focus on acquiring accurate information on Covid-19 from a multitude of timeframes and locations. Figure 4a highlights the distribution of the question types present in the CovidQA dataset solely based on the first two words in the question. Table 2 provides a further quantitative percent examination of the question types while also illustrating specific examples.

A dominant portion of the CovidQA dataset consists of “What”, “How”, and “Give/Provide/List" questions (as shown in Figure 4a). These questions typically form retrieval questions that aim to inquire for numerical and statistical details, such as the number of cases and the recovery rate in a specific location. On the other hand, questions starting with “Which” and a fraction of the “Give/Provide/List” questions serve as the reasoning questions that only ask for specific locations where certain numerical aspects of Covid-19 are at a specific maximum or minimum.

Bigram is defined as a sequence of two adjacent words from a sentence. In Figure 4b, we also depict the breakdown of the most common bigrams in the CovidQA dataset. This breakdown offers a detailed synopsis of certain aspects of Covid-19 that the generated questions look to draw out from the six databases.



(a) **(b)**

Figure 4: Breakdown of the real generated CovidQA questions. Plot (a) depicts the breakdown based on the first two words. Plot (b) showcases the breakdown based on the bigrams.

Question Type	Examples	Percentage
What	What is the incidence rate in New Mexico? What is the racial breakdown of recovered cases in Indiana? What is the total forecasted number of deaths in Oklahoma in August?	24.65%
How	How many people are currently in ICU in Arizona? How many Asian deaths occurred in West Virginia in May? How many confirmed cases occurred in Stavropol Krai, Russia in July?	5.80%
Which	Which state has the highest percentage of deaths from American Indians? Which country has the least total tests in January? Which state has the lowest number of people cumulatively on ventilators?	12.08%
List/Give/Provide	Give me the daily testing rate in Cameroon in December. List the country with the least negative tests in December. Provide me with the county in Wyoming with the lowest percentage of deaths from Native Hawaiians.	57.47%

Table 2: Question types in CovidQA along with examples.

4 Question Answering Methods on Databases

In this chapter, we begin with the formulation of the natural language questions-to-SQL queries generation problem. Then, we introduce two template-based matching methods in detail. In addition, we present two machine learning models in resolving the question-to-SQL problem.

4.1 Template-based Methods

In this work, our goal is to convert Covid-19 tracking-related questions to database queries and retrieve the right answer from appropriate data sources. To handle this problem in an empirical manner, we proposed two template-based SQL query generation methods. In chapter 3, we defined a question template collection and corresponding SQL query template collection. Both of our proposed methods will utilize these resources: first, map the natural language question with a question template. Next, find what entities and entity types are used in the natural language question. Then combine the information retrieved and use them to generate the output SQL query with the SQL query template. The final step is to use the generated SQL query to retrieve the answer to the question from the corresponding database. We will introduce our methods in flow charts with more detail.

4.1.1 Type Group Recognition Matching

We found that different natural language question templates hold different combinations of entity types, and this forms the core idea in the type group recognition matching solution. Figure 5 provides the workflow of this method. When the natural language question is given, we first go over the entities among the entity lists defined in chapter 3, then find all matches and record the entities and entity types. Once the entity type that appears in the natural language question is determined, we can find a unique natural language question template mapped to the question. Since we defined fixed SQL query templates for every natural language question template, it is convenient to generate SQL queries with those templates and entities recorded by replacing the placeholders conditionally. The type group recognition matching strategy is highly scalable for it does not require any modifications when the entity lists and question templates change. However, it suffers from a few drawbacks. If we keep expanding the volume of the entity lists in chapter 3, the matching efficiency will decrease accordingly. Also, if we decide to expand the pool of natural language questions and SQL query templates, it may introduce some extreme cases like two different natural language question templates share the same entity type group combination.

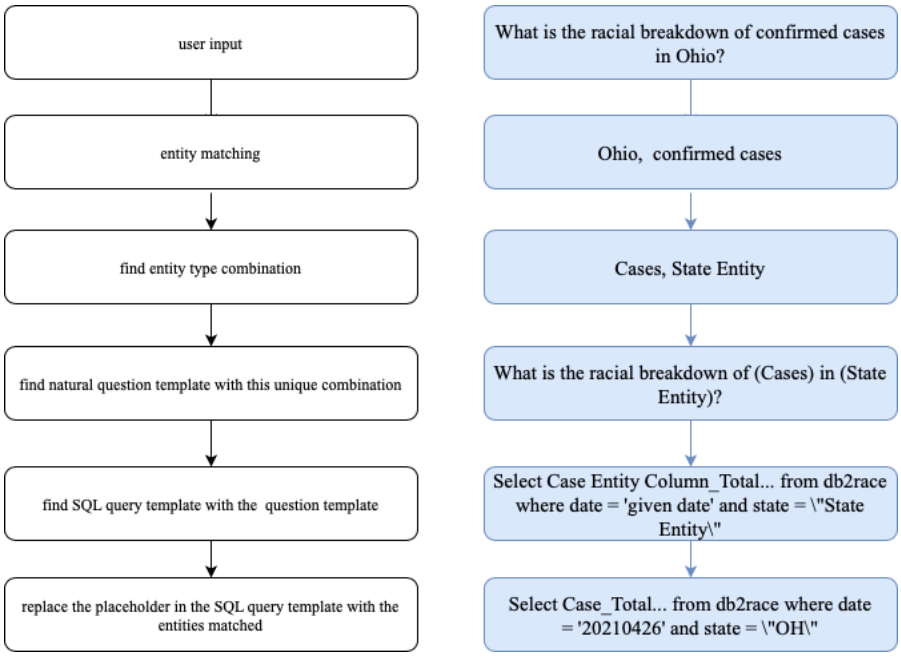


Figure 5: Flowchart of the type group recognition matching.

4.1.2 Two-step Matching

To eliminate certain uncertainties in the type group recognition matching strategy mentioned previously, we further propose a second matching strategy in the natural question-to-SQL conversion: two-step matching.

The workflow of this method is shown in Figure 6. Using the natural language question templates pool, we can calculate the similarities between the user input with all the templates in the pool, then take the templates with the highest similarity as the mapped templates for the user input for the following process [30]. To achieve the similarity calculation, we chose a light-weighted python package named sumeval [31]. Sumeval provides a convenient API that can return different similarity metrics given any two English strings. ROUGE-L is the metric we chose from sumeval. ROUGE is an acronym for Recall-Oriented

Understudy for Gisting Evaluation and L stands for longest common subsequence [32]. If we denote the user input as C and the template to be calculated as S, ROUGE-L is calculated by

$$R_{LCS} = \frac{LCS(C,S)}{len(S)}$$

$$P_{LCS} = \frac{LCS(C,S)}{len(C)}$$

$$F_{LCS} = \frac{(1+\beta^2)R_{LCS}P_{LCS}}{R_{LCS}+\beta^2 P_{LCS}}$$

where LCS(C, S) is the biggest common subsequence of the two input strings and β is the coefficient defined by P and R.

The next step requires a closer look into the question templates. The question templates in the pool are padded with a number of placeholders where every one of the placeholders stand for an entity type. Once the mapping from the user input with the question template is determined, it also suggests the mapping from the user input to the entity types is determined. Different from the implementation in type group recognition matching, we no longer need to go over all possible entities in all entities list, only those in the matched entity type lists will be used, which could considerably improve the matching efficiency. After we retrieve the entity that appeared in our lookup table, the last thing that needs to be done is replacing it with the placeholders in the SQL query templates. It will generate an available SQL query command that can be used for specific databases mentioned in chapter 3.

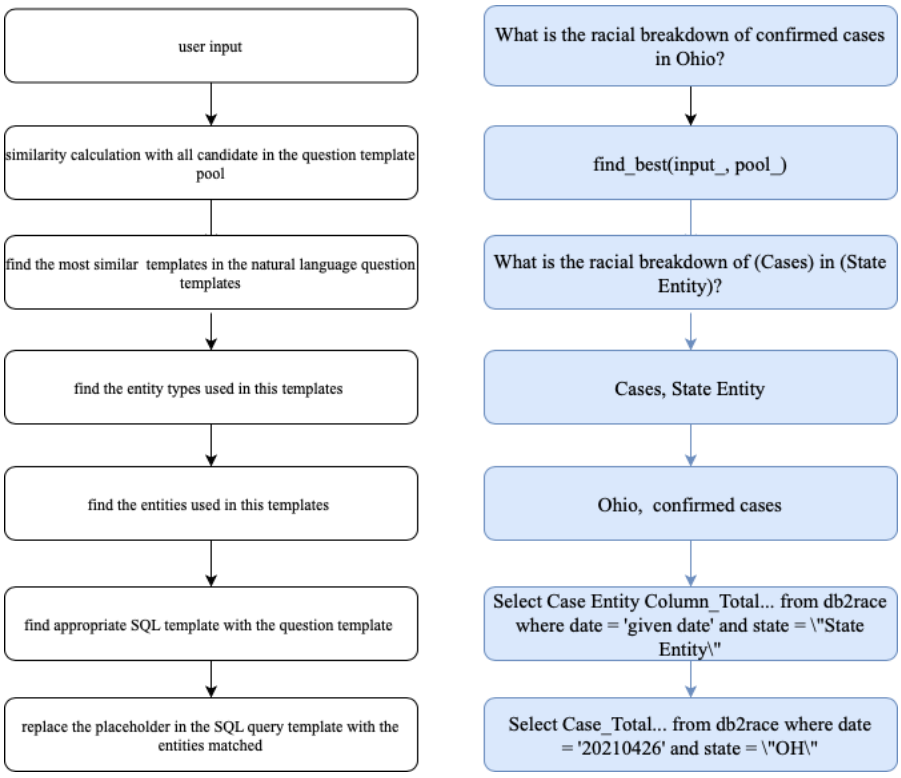


Figure 6: Flowchart of the two-step matching procedure.

4.2 Machine Learning-based Methods

In addition to the template matching methods, we further investigated two information retrieval solutions built with neural networks and the availability to apply them in our QA system.

4.2.1 TAPAS Model

Recently, many models utilize semantic parsing methods as a baseline in the field of question answering and NLP. However, these semantic parsing methods that work to convert a question to a corresponding query are relatively ineffective when scaled up to the size and complexity of real-world data. Additionally,

logical forms present additional costs to question answering methods. A recent question answering model that has been successful is Google Research's TAPAS [9]. TAPAS is a unique system that approaches question answering without the use of logical queries. Instead of incorporating logical forms, TAPAS skips this traditional step and trains itself with weak supervision to make a valid prediction on the given natural language statement. It offers its prediction by either selecting a cell in the input table or performing an appropriate aggregation operator. TAPAS applies a variation of the BERT architecture by taking in both the natural language questions and the numerical data present in the table as a single sequence [9, 33]. Figure 7 shows the model overview of TAPAS.

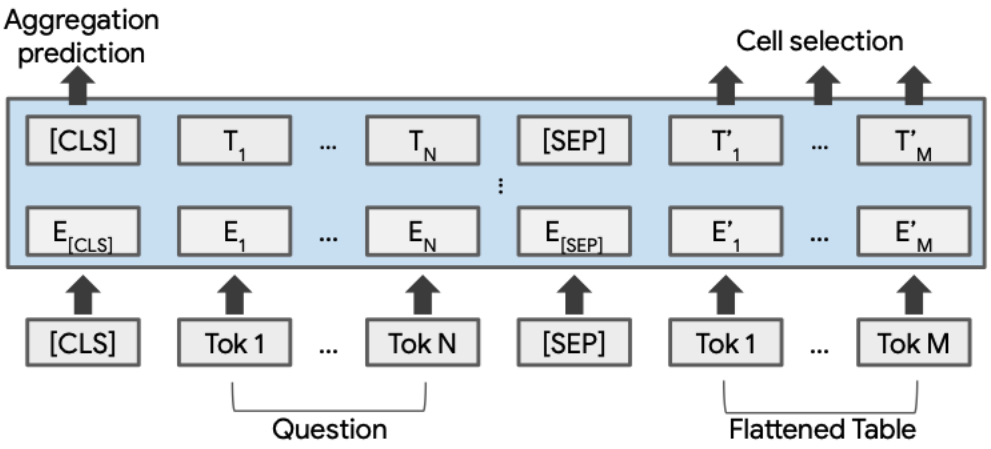


Figure 7: An overview of the TAPAS model [9].

In Figure 7, the input is at the bottom, combining the natural language embeddings and the flattened, serialized table content embeddings with [SEP] separator in the middle. The input also includes the location embeddings like the location of word embeddings in the natural language and the location of table content embeddings (the column number and the row number of the table content). Figure 8 shows a typical encoding method.

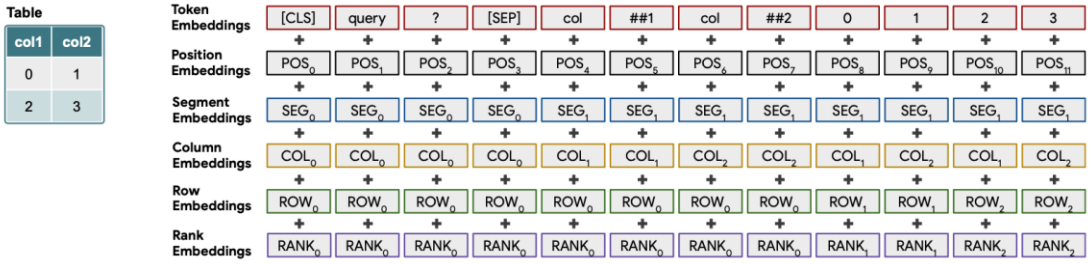


Figure 8: An illustration of the embeddings built in the TAPAS model [9].

All token cells in Figure 8 have a special row embedding, column embedding, and a ranking embedding to calculate the model's outputs: one type is a probability score for the table content embedding that indicates the likelihood of being the wanted answer; the other type is an aggregate operation result that indicates if an aggregate operator exists, and if the answer is yes, what the kind of operator will be used.

High-quality data is vital to building an excellent model. Pretrained on 6.2 million Wikipedia data tables, TAPAS proved its significance by outperforming or matching previous semantic parsing approaches across three benchmark datasets. Furthermore, the TAPAS team opened access to their pre-trained model from Wikipedia, which makes it possible for us to test this model on our CovidQA dataset.

4.2.2 TREQS Model

Different from the TAPAS model, TRanslate-Edit Model for Question-to-SQL (TREQS) model is able to handle broader operator types, and TREQS model is also a SQL-generating model [11]. Therefore, it is possible to have more parallel comparisons with the two empirical methods in this chapter.

The overall model structure of TREQS is given in Figure 9. The encoder-decoder framework for sequences, attention mechanism both in the question and SQL end, a controlled generation and pointer network, placeholder replacing mechanism, and term matching mechanism are the essential building blocks of the model. For the encoder-decoder framework, a bidirectional LSTM is used as an encoder in the natural question end and a unidirectional LSTM is used as a decoder in the SQL end [34, 35]. The vectorized features in the questions are denoted by h_e, and for the chosen decoder, the processed hidden state is denoted by h_t^d where the *t* stands for the time step. The initializing function for the decoder's hidden state and cell state is given by:

$$h_0^d = \tanh\left(W_{\text{e2dh}}\left(\overrightarrow{h}_J^e \oplus \overleftarrow{h}_1^e\right) + b_{\text{e2dh}}\right)$$
$$c_0^d = \tanh\left(W_{\text{e2dc}}\left(\overrightarrow{c}_J^e \oplus \overleftarrow{c}_1^e\right) + b_{\text{e2dc}}\right)$$

where the W_{e2dh} and W_{e2dc} are the weighting matrices and the b_{e2dh} and b_{e2dc} are parameters that can be updated during the training process. The TREQS model also introduced the attention mechanism both in the question and SQL end to capture the correlation between the elements in the sequences [36]. However, modifications are required for this specific task because, in the natural question end, the vanilla attention mechanism is not able to prevent the redundancy of the decoder. A balancing module is shown in Figure 10 to reduce the attention score from tokens that have already drawn high attention. In the SQL end, TREQS uses a tailored dynamic attention mechanism to align the score from previous tokens:

$$\alpha_{t\tau}^d = \frac{\exp(s_{t\tau}^d)}{\sum_{k=1}^{t-1}\exp(s_{tk}^d)}$$

$$z_t^d = \sum_{\tau=1}^{t-1}\alpha_{t\tau}^d h_\tau^d$$

Where the s_t^d stands for the alignment scores, a_t^d stands for the attention weight of all tokens, and the z_t^d stands for the context vectors. Also, given the characteristic that the question-to-SQL problem has a structured grammar in SQL and there are special patterns in the entities of the SQL, TREQS proposed a generation and pointer network to adopt such features. To be more specific, it uses a pointer network to generate placeholders in the primary generated SQL for keywords that are already in the vocabulary and the keywords that are not in the vocabulary will be taken as potential condition values for the operators. Then the generation network will be used to determine the structure entities like table or column names. Combining both, the generation and pointer network is able to calculate the probability for a given token. Once the primary SQL with placeholders is generated, the next step is to replace the placeholders with the tokens. This combination rule is given by:

$$P_{\text{rps}}(y_{t'}) = \begin{cases} \sum_{j:x_j=y_{t'}} \alpha^e_{t'j} & y_{t'} \in \mathcal{X} - \mathcal{V} \\ 0 & \text{otherwise} \end{cases}$$

Where y_t is the token to be determined, X is the entire set of the tokens and V is the given vocabulary of the dataset. Finally, for some outlier questions without any table content in the sequence, the TREQS model utilizes the ROUGE-L score to retrieve the most similar entities in the vocabulary for the placeholder replacement.

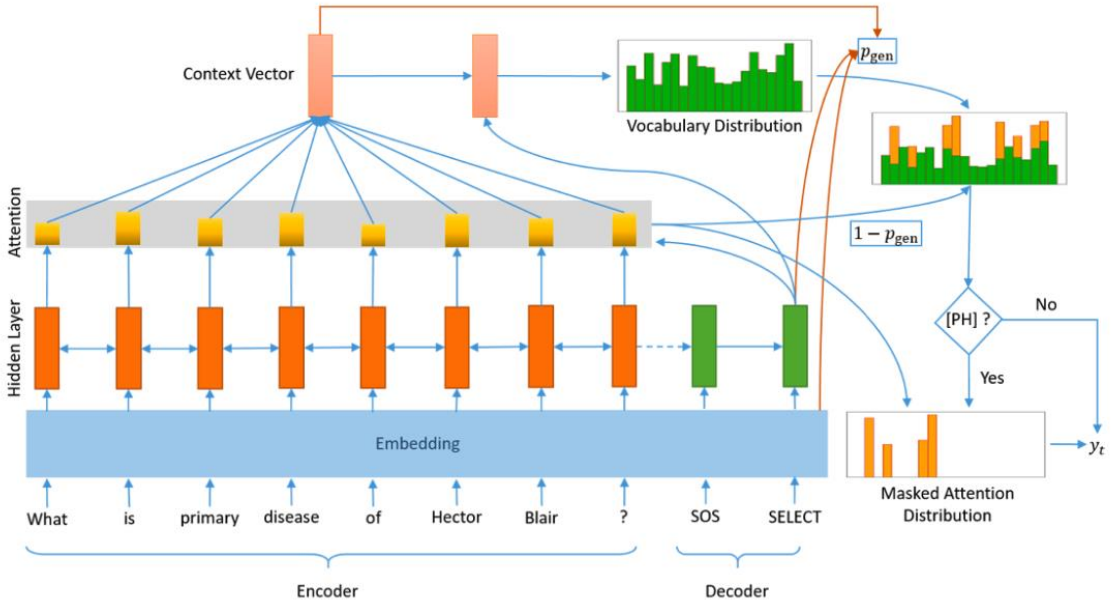


Figure 9: An overview of the TREQS model [11].

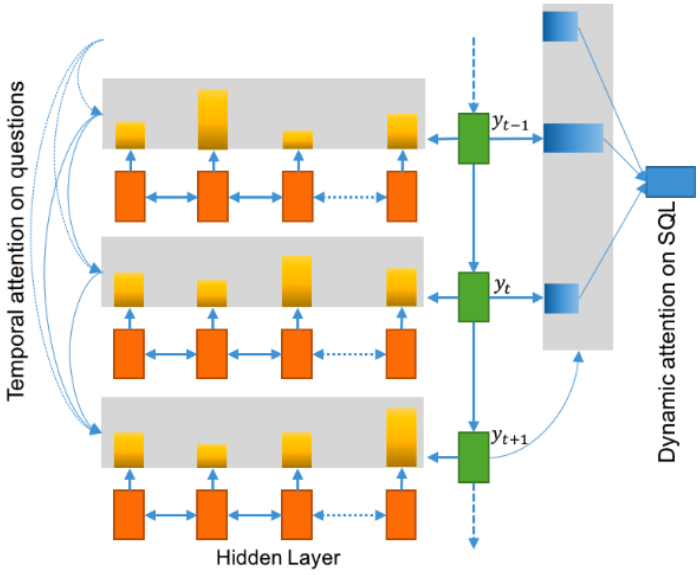


Figure 10: Dynamic and temporal attention used in the TREQS model [11].

5 Experiments

In this chapter, we primarily introduce the implementation details, and evaluation metrics are introduced primarily. Then we present different types of qualitative and quantitative analysis detailing the performance of our proposed methods.

5.1 Experimental Settings

5.1.1 Template-based Methods

We used the CovidQA dataset introduced in Chapter 3 as our testing data and used all the six public databases described in Chapter 3.1 for querying the answer from the generated SQL queries. As the input, CovidQA database will be fed into our proposed methods in Chapter 4, and generate corresponding output in the same format as the dataset shown in Figure 3.

5.1.2 TAPAS Model

We organized the TAPAS experiments by evaluating the TAPAS-SQA pre-trained model. SQA is one of the three large datasets in which TAPAS was trained on, which consisted of questions about given tables in a highly conversational manner. One of the key elements of TAPAS-SQA pre-trained model is that it can only select cells from the table, meaning that the questions inputted into the model should reflect answers that can be acquired through simple cell selection. It is also important to note that the TAPAS-

SQA pre-trained model does not perform any aggregation operators. For the purpose of these experiments, only the TAPAS-SQA pre-trained model was examined. This is primarily due to the fact that the other two datasets used for TAPAS's fine-tuning, WTQ and WIKISQL, take in questions that comprise aggregation. However, since the aggregation operators used in the TAPAS pre-trained models for these two datasets do not apply to our question templates, we focused on only the TAPAS-SQA pre-trained model. Hence, we inspected each of our question templates to discover those that fit the criteria of being a cell-selection question. After comprising a comprehensive list of applicable question templates, we inputted our generated questions from the CovidQA dataset that corresponds to these selected question templates to TAPAS. Since TAPAS cannot execute predictions on large tables, we reduce the tables to a suitable size that the BERT-based TAPAS model can encode by eliminating extraneous rows and columns that are irrelevant to the question. We then generated TAPAS predictions and analyze the validity of these predictions with the ground truth results and obtain the final accuracy. Since only a selected set of question templates in CovidQA database could be tested, we took the result from TAPAS model as a partial reference and partial baselines.

5.1.3 TREQS Model

TREQS defines a dedicated data format for its training and testing process as shown in Figure 11. To run the TREQS model with our CovidQA dataset, first we resolve the data compatibility issue. The data formatting conversion work is finished with: (1) combined all the tables defined by the six selected data resources; (2) all tables were assigned with a number; (3) added operation index to the dataset ('=': 0, '>': 1, '<': 2, '>=': 3, '<=': 4); (4) added the numerical index of aggregation operation ('': 0, 'count': 1, 'max': 2, 'min': 3, 'avg': 4); (5) tokenized all the natural questions and SQL queries; (6) generated a vocabulary for

the TREQS model to calculate the best entities for the placeholder. After the data conversion is done, we then randomly split the entire dataset into 3 subsets with the 8:1:1 ratio, and mark them as the training set, development set, and test set, respectively. The model is implemented with Pytorch with the hyperparameter shown in Table 3 [37].

Figure 11: Proposed data formatting of TREQS [11].

Dimension of Word Embeddings	Hidden State Size	Learning Rate	Maximum Gradient Norm	Batch Size
128	256	0.0005	2	16

Table 3: Selected hyperparameters for TREQS.

5.2 Evaluation Metrics

In this work, we adopt the following evaluation metrics:

(1). **SQL logic form accuracy** is defined as the Acc_OA = N_oa/N, where N denotes the number of Question-SQL pairs in CovidQA, and N_oa represents the number of correctly generated SQL queries. This metric only applies to models that generate SQL. In other words, the TAPAS model cannot be evaluated with the SQL logic form accuracy, because the TAPAS model aims to directly retrieve/predict answers instead of predicting the SQL queries. We used the execution accuracy to generally compare the accuracy of the returned answers for the input questions.

(2). **Breakdown accuracy** of matching is defined as Acc_BD =N_bd/N, which is used to provide a closer perspective generated SQL query and the ground truth query. N_bd denotes the number of correct parts that match exactly with the ground truth data. This metric only applies to models that generate SQL.

(3). **Execution accuracy** is defined as Acc_EX = N_ex/N, where N denotes the number of question-SQL pairs in CovidQA, and N_ex denotes the number that matches exactly the execution result from the ground truth SQL query. This metric applies to all the models we tested.

5.3 Experimental Results

5.3.1 SQL Logic Form Accuracy

Table 4 shows the SQL logic form accuracy of the two proposed methods and the TREQS model. By comparing the overall performance of the two empirical methods, it can be concluded that both methods are showing good accuracy in generating the correct SQL queries with respect to all six databases. After checking the mis-generated SQL queries for the two-step matching, it can be observed that using ROUGE-L as the similarity metric will lead to mismatched results when the entities appearing in the natural language questions are close to some of the expressions in the natural language questions templates. Also, by checking some specific template ID with SQL logic form accuracy, it can be inferred that some entities

that share the same name in different entity type lists are the reasons for the mismatching. As an example, the word "*Idaho*" is both a state entity and county entity. And regarding the CovidQA dataset we generated, the entity combination matching approach is generally more accurate than the other one. However, for some particular natural language question templates, the two-step matching shows a better generalizability since it can overcome the drawbacks of homonymy in different entity types. Moreover, the machine learning-based TREQS model shows a reasonable overall accuracy, but the performance is surpassed by the two empirical approaches in this domain-specific task. Uneven performance over different databases can also be observed in the TREQS model, but the reason is different: There is not enough data from database 3 to fit the TREQS model.

Methods	DB1	DB2	DB3	DB4	DB5	DB6	Overall
TREQS	0.549	0.555	0.385	0.580	0.556	0.547	0.554
Type Group Recognition	0.625	0.990	1.000	1.000	1.000	1.000	0.948
Two-step	0.600	0.836	1.000	0.952	1.000	0.959	0.879

Table 4: SQL logic form accuracy results for various methods.

5.3.2 Breakdown and Execution Accuracy

To further evaluate the accuracy over each component in our CovidQA dataset, in Table 5, we present the entity matching accuracy of the three SQL generating models. To provide an approximate baseline in Table 6, we present the execution accuracy of the TAPAS model on question templates that can be tested, along with other models' results on the same question types. In this table, 2q1 stands for question template 1 of database 2. It is noteworthy that for 2q8, the two-step matching is not able to give back correct answers because the matching strategy is mapping a similar but wrong first layer templates to all the test samples in 2q8. And the corresponding execution accuracy for all the other three models is shown in Table 7. By comparing the two tables, it can be concluded that the TAPAS model is not the ideal solution for our

proposed QA system, for it cannot handle all the SQL query types we have, and it also performed a relatively low accuracy.

Methods	DB1	DB2	DB3	DB4	DB5	DB6	Overall
TREQS	0.573	0.570	0.464	0.585	0.571	0.568	0.572
Type Group Recognition	0.685	1.000	1.000	1.000	1.000	1.000	0.957
Two-step	0.613	0.838	1.000	0.954	1.000	0.965	0.883

Table 5: Breakdown accuracy of template-based methods and TREQS model.

Method	2q1	2q2	2q3	2q6	2q7	2q8	2q9	4q2	6q1
TAPAS	0.267	0.208	0.267	0.091	0.228	0.325	0.250	0.510	0.428
TREQS	0.616	0.073	0.621	0.581	0.566	0.572	0.291	0. 652	0.589
Type Group Recognition	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Two-step	1.000	1.000	1.000	1.000	0.925	0	1.000	0.758	1.000

Table 6: Execution accuracy of TAPAS model and other models on selected question templates.

Methods	DB1	DB2	DB3	DB4	DB5	DB6	Overall
TREQS	0.613	0.575	0.488	0.623	0.587	0.591	0.592
Type Group Recognition	0.695	0.992	1.000	1.000	1.000	1.000	0.955
Two-step	0.641	0.845	1.000	0.978	1.000	0.975	0.895

Table 7: Overall execution accuracy of template-based methods and TREQS model on the complete dataset.

Besides the quantitative results shown above, we additionally provide qualitative examples in Table 8 and Table 9 to further illustrate the performance of all the SQL-generating models. In the two cases demonstrated in Table 8, it can be observed that the two empirical methods are able to generate the corresponding SQL queries with correct SQL templates, entity types, and entities themselves. On the other hand, the machine learning-based TERQS model has also shown a reasonable ability to "translate" the natural questions to the SQL, except for some time entities in the questions. In the example of "*What are the number of total tests done by Cameroon in 1st, July?*", the generated SQL query from TREQS is

partially correct except the time entity is "*Jun 01, 2020*" instead of the correct "*Jul 01, 2020*". In fact, many of the mis-generated SQL queries are showing the same error pattern: date is the only incorrect component in the SQL queries. The underlying reason for it is that the TREQS model lacks the ability to recognize the correlation to "translate" fragments like "*1st, January*" to "*Jan 01, 2020*" in the SQL queries. For the SQL queries that are being generated correctly by TREQS model, there must be a number of training items with exactly the same time entity, so that the neural networks can recognize the matching pairs for the SQL-generating task.

Method	Example 1	Example 2
Question	Give me the age with the lowest percentage of daily cases in the United States.	What are the number of total tests done by Cameroon in 1st, July?
Ground truth	Select Age_Group, Percentage from db4caseage Order by Percentage asc limit 0, 1	Select Total from db6 where Entity = "Cameroon" and Date = 'Jul 01, 2020'
TREQS	Select Age_Group, Percentage from db4caseage Order by Percentage asc limit 0, 1	Select Total from db6 where Entity = "Cameroon" and Date = 'Jun 01, 2020'
Type Group Recognition	Select Age_Group, Percentage from db4caseage Order by Percentage asc limit 0, 1	Select Total from db6 where Entity = "Cameroon" and Date = 'Jul 01, 2020'
Two-step	Select Age_Group, Percentage from db4caseage Order by Percentage asc limit 0, 1	Select Total from db6 where Entity = "Cameroon" and Date = 'Jul 01, 2020'

Table 8: Queries from different models on the example questions. Text in Red color indicates the incorrect part in the query.

Template-based question-to-SQL models provide incorrect SQL queries in some corner cases that the pre-defined rules fail to cover. In Table 9, in the example of "*What is the number of ICU beds in Iowa?*", the type group recognition method takes "*Iowa*" as a county entity instead of a state name. Thus, the corresponding SQL query is mistakenly generated. Although this model is showing promising performance in the quantitative analysis, it fails in simple cases like multiple entities sharing the same name across different entity categories. It will also fail if question templates with the same entity type group combination are being introduced to the empirical rules. In the example, "*How many African-American confirmed cases occurred in Ohio in 3rd, January?*", all the models are able to give the correct query except the two-step matching method: it matches the asked natural question with the wrong template

"*How many (Cases) occurred in (State Entity) in (day) (Month)?*" from the templates pool. The reason for this is that this template returns the highest ROUGE similarity score with the given question, which leads to the following inaccurate SQL generation. For the two-step matching, such corner cases cannot be eliminated and can only be minimized by a well-defined set of step-one templates.

Method	Example 1	Example 2
Question	What is the number of ICU beds in Iowa?	How many African-American confirmed cases occurred in Ohio in 3rd, January?
Ground truth	Select SUM(NUM_ICU_BEDS) from db3 where STATE_Name = "Iowa"	Select Cases_Black from db2race where date = '20210103' and state = "OH"
TREQS	Select SUM(NUM_ICU_BEDS) from db3 where STATE_Name = "Iowa"	Select Cases_Black from db2race where date = '20210103' and state = "OH"
Type Group Recognition	Select SUM(NUM_ICU_BEDS) from db3 where STATE_Name = "Iowa" and COUNTY_Name = "Iowa"	Select Cases_Black from db2race where date = '20210103' and state = "OH"
Two-step	Select SUM(NUM_ICU_BEDS) from db3 where STATE_Name = "Iowa"	Select SUM(Confirmed) from db1state where date = '01-03-2021' and Province_State = "Ohio"

Table 9: Queries from different models on the example questions. Text in Red color indicates the incorrect part in the query.

6 Conclusion

In summary, we conducted many real-world experiments for building a trustworthy Covid-19 tracking QA system, from both the data and modeling perspectives. While we believe that the insights we gathered in this study can be extremely important for the future covid-QA systems, there is certainly further room for improvement.

6.1 Lessons Learned

6.1.1 Insights for Information Retrieval

While exploring the related work, we learned that it is fundamental to have a suitable and precise depiction of the information retrieving task. An inappropriate definition of the problem will make it difficult and costly in the model building phase. For example, once the question of interest is determined, the following questions should be answered: is it a single-domain problem or a cross-domain problem? How many databases are likely to be involved in the retrieval process? What are the relative sizes of the structured data? For the natural questions asked, are they relatively arbitrary or wil there be any underlying patterns that can be utilized? How complex (in terms of average length, diversity, operator types involved) the generated SQL queries will be? Getting the answers to all the questions above will greatly reduce the efforts in finding the best modeling approach for the researched information retrieving task.

When it comes to machine learning-based modeling, it is true that neural networks are incredibly capable of handling non-linear patterns in complex tasks. But to leverage this power, a large amount of high-quality data is indispensable. In most application scenarios, collecting such data is expensive, and sometimes impossible to find high quality datasets. In our work, the machine learning-based TREQS model was not showing promising performance when the training set was not sufficiently large, and this issue was finally resolved after we managed to expand the scale of our CovidQA dataset. Also, the trained model has poor generalizability, which makes it difficult to evaluate its applicability to other similar applications.

Another important lesson from this work is that machine learning-based models are not silver bullets for all the information retrieval tasks. In recent years, machine learning is in the spotlight of almost every computer science-related area. For comprehensive and complex tasks in laboratories, machine learning is typically able to improve the system performance in terms of the accuracy or response time, sometimes by a huge margin compared to the traditional solution. However, for real-world applications, it matters less for the novelty and complexity of the used models. Thus, it is always worth trying the empirical, mature solutions when it comes to emerging real-world problems.

6.1.2 Strengths and Weaknesses of Modeling Approaches

Extending the analysis from chapter 5, we can further conclude the corresponding strengths and weaknesses of each type of model we tested from parallel comparison.

From our experiment results, it can be inferred that template-based models are suitable for specific sub-field QA tasks, especially when the frequently asked questions can be exhaustively listed. Different from applications in typical translation tasks, in QA tasks, the target language for the template-based modeling methods is a strict and highly structured command language, which is more suitable for applying such methods in practice. With respect to system modification like the add/drop question templates and entity types or the change of the schema of the databases, the empirical methods hold better scalability compared to the models that require retraining. However, as the number of question templates and entity types increases, the matching accuracy will inevitably drop.

TAPAS, or the cell-selecting machine learning model, could be the best solution for general question answering tasks. It overcomes two major challenges in the information retrieval process: the randomness in the natural questions and SQL queries; and the complexity of the large-scale data collections. While assessing the trade-offs of these challenges, TAPAS is not able to handle queries with certain operators or scenarios when multiple tables are involved. On the other hand, the SQL-generating machine learning model offers a solution that can potentially balance out these trade-offs. TREQS model is able to cope with both complex question templates and complex queries. It abstracts the question-to-SQL task as a special translation task. With multiple tailored modifications on the general translation model, the overall accuracy is satisfactory, although it is still outperformed by the empirical methods. Overall, the machine learning-based modeling approaches are computationally expensive compared to the template-based methods, but they have better applicability for a wider range of QA applications.

6.2 Summary

There are large amounts of Covid-19 related data that are collected and updated in a wide range of public databases. It is a challenging work to develop a reliable natural language information retrieval system in the healthcare field and a user-friendly QA mechanism can improve the data accessibility to general users across the world. In our work, we focus our QA system on a relatively narrower problem, namely, the Covid-19 tracking task. We investigate and select six trustworthy public data resources and define a number of template pools that are commonly used, then tested two template-based matching strategies and two machine learning-based modeling approaches on the CovidQA dataset that we generated with a predefined set of question templates. From Chapter 5, it can be concluded that our proposed methods are both feasible and effective for such problems, and with a favorable potential of migration to other similar problems.

6.3 Future Work

In the future, our work can be expanded in multiple ways. These ways include: (1). More entity types and additional natural language question templates can be introduced to cover corner cases in frequently asked questions for our system. Also, as the public data resources are being updated regularly, we could add new questions that people are concerned about such as the virus variant-related questions. (2). We could generate some questions with designated defectiveness like typos or missing pieces to test the robustness of our system. Following this practice, a matching module could be added to the two empirical modeling approaches. Similar to the condition recovering mechanism in TREQS, such a module is able to map the

keywords with the most similar entities, which empowers the system to cope with more randomly asked natural questions. (3). We could apply some semantic analysis tools to the template-based methods in response to the potential expansion of the entity types and natural language question templates. (4). Future work could incorporate more modeling approaches like SQLNet and TypeSQL [10, 18]. Adding such experiments will provide a more comprehensive perspective for the state-of-the-art information retrieval technologies on our researched question. (5). Increasing the accessibility of the QA system is crucial for disabled people. The system can integrate with some user-friendly interfaces like voice control. There are several readily available voice recognition APIs for web development, and it is feasible to embed our QA system in a voice control-supported webpage. (6). Based on the web application, we could further build a feedback pipeline for the users. Such a pipeline could be used both for experience improvements and system glitch reporting. As we discussed, for the mismatch of the Type Group Recognition in example 1 from Table 9, the reason is that the model lacks the ability to distinguish different entities with the same name. In this case, we could use the feedback pipeline to ask the users whether they are referring to a county name or a state name, then use the additional information to generate the correct queries and deliver the correct answers. On the other hand, when the users are provided with answers which are conspicuously wrong, such a pipeline makes it easy for the users to report such errors to the developers for them to further improve the reliability of the system.

References

[1] Apuke, Oberiri Destiny, and Bahiyah Omar, "Alone: Fake news and COVID-19: modelling the predictors of fake news sharing among social media users," *Telematics and Informatics,* p. 56, 2020.

[2] "COVID-19 Map - Johns Hopkins Coronavirus Resource Center," [Online]. Available: https://coronavirus.jhu.edu/map.html.

[3] S. Selvin, "Statistical Analysis of Epidemiological Data," *New York: Oxford University Press,* 1996.

[4] C. Yan, A. Cheung, J. Yang and S. Lu, "View-driven optimization of database-backed web applications," in *10th BiAnnualennial Conference on Innovative Data Systems Research,* 2020.

[5] Ferreira GB, Borges S, "Media and Misinformation in Times of COVID-19: How People Informed Themselves in the Days Following the Portuguese Declaration of the State of Emergency," p. 108–121, 1 2020.

[6] A. M. N. Allam and M. H. Haggag, "The question answering systems: A survey," *Intemational Journal of Research and Reviews in Information Sciences,* vol. 2, 2012.

[7] Sanglap Sarkar, Venkateshwar Rao, S. M. Baala Mithra and Subrahmanya VRK Rao, "NLP Algorithm Based Question and Answering System," *Seventh International Conference on Computational Intelligence Modeling and Simulation,* 2015.

[8] Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ra- manathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev, "Improving text-to-sql evaluation methodology," *the 56th Annual Meeting of the Association for Computational Linguistics,* p. 351–360, 2018.

[9] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos, "TaPas: Weakly supervised table parsing via pre-training," *the 58th Annual Meeting of the Association for Computational Linguistics,* p. 4320– 4333, 2020.

[10] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev, "TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation," *NAACL-HLT 2018,* p. 588–594, 2018.

[11] Ping Wang, Tian Shi, and Chandan K. Reddy, "Text-to-SQL Generation for Question Answering on Electronic Medical Records," *The Web Conference,* 2020.

[12] Jamison DC, "Structured Query Language (SQL) fundamentals," *Curr Protoc Bioinformatics,* 2003.

[13] Mitchell, T., Machine Learning, McGraw Hill, New York, NY, USA, 1997.

[14] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," 2008. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/.

[15] C.Unger,L.Bu ̈hmann,J.Lehmann,A.-C.NgongaNgomo,D.Gerber,andP.Cimi- ano, "Template-based question answering over rdf data," *The 21st international conference on World Wide Web,* 2012.

[16] R.G.Athreya,S.Bansal,A.N.Ngomo,andR.Usbeck, "Template-basedquestionanswering using recursive neural networks," *CoRR,* p. abs/2004.13843.

[17] Alexander R Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang, "Template-based question generation from retrieved sentences for improved unsupervised question answering," *arXiv preprint arXiv:2004.11892*, 2020.

[18] Xiaojun Xu, Chang Liu, and Dawn Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.

[19] David Oniani and Yanshan Wang, "A quali- tative evaluation of language models on automatic question-answering for covid-19," *the 11th ACM International Conference on Bioin- formatics, Computational Biology and Health Infor- matics*, p. 1–9, 2020.

[20] Andre Esteva, Anuprit Kale, Romain Paulus, Kazuma Hashimoto, Wenpeng Yin, Dragomir Radev, and Richard Socher, "CO-Search: COVID-19 information retrieval with semantic search, ques- tion answering, and abstractive summarization," *arXiv:2006.09595*, 2020.

[21] "Global COVID-19 Tracker," [Online]. Available: https://coronavirus.1point3acres.com/en.

[22] "COVID-19 Information Submission Form," [Online]. Available: https://airtable.com/shrxPB5CCBlxLV8wJ.

[23] "Epidemic Big Data Live Report," [Online]. Available: https://voice.baidu.com/act/newpneumonia/newpneumonia?.

[24] "Covid in the U.S.," [Online]. Available: https://www.nytimes.com/interactive/2021/us/covid-cases.html.

[25] "The COVID Racial Data Tracker," [Online]. Available: https://covidtracking.com/race.

[26] "Definitive Healtcare USA. Definitive healthcare bed loca- tions ISO-19139 metadata," [Online]. Available: https://services7. arcgis.com/LXCny1HyhQCUSueu/arcgis/rest/services/ Definitive_Healthcare_USA_Hospital_Beds/FeatureServer/ 0/metadata?format=default&f=html.

[27] "COVID Data Tracker," [Online]. Available: https://covid.cdc.gov/covid-data-tracker/.

[28] "Provisional COVID-19 Deaths by County, and Race and Hispanic Origin," [Online]. Available: https://data.cdc.gov/NCHS/Provisional-COVID-19-Deaths-by-County-and-Race-and/k8wy-p9cg.

[29] "Statistics and Research Coronavirus Pandemic (COVID-19)," [Online]. Available: https://ourworldindata.org/coronavirus.

[30] Alberga, C. N., "String similarity and misspellings," *Communications of the ACM,* vol. 10, pp. 302-313, May 1967.

[31] "Well tested & Multi-language evaluation framework for Text Summarization," [Online]. Available: https://github.com/chakki-works/sumeval.

[32] Lin, Chin-Yew, "Rouge: A package for automatic evaluation of summaries," *Text Summarization Branches Out*, pp. 74-81, 2004.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[34] Hochreiter, S. and Schmidhuber, "Long short-term memory," *Neural computation*, p. 1735–1780, 1997.

[35] Zhiheng Huang, Wei Xu, and Kai Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR, abs/1508.01991*, 2015.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems,* p. 6000–6010, 2017.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al, "An imperative style, high-performance deep learning library," *NIPS,* 2019.

Appendix A Entity Type Lists

Gender Entity: "Male", "Female", "Of the male gender", "of the female gender", "men", "women"
Testing Entity: "total tests", "daily tests", "positive tests", "negative tests"
Bed Entity: "staffed beds", "licensed beds", "ICU beds"
Rate Entity: "daily testing rate", "percent positive rate", "percent negative rate", "daily percent positive rate", "hospitalization rate", "testing rate", "recovery rate", "Case-Fatality rate", "Incidence Rate"
Mobility Entity: "retail and recreation", "grocery and pharmacy", "parks", "transit stations", "workplaces", "residential areas"
Hospitalization Entity: "Currently in ICU", "Cumulatively in ICU", "Currently on ventilators", "Cumulatively on ventilators", "Currently hospitalized", "Cumulatively hospitalized"
State Entity: "Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado", "Connecticut", "Delaware", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa", "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts", "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana", "Nebraska", "Nevada", "New Hampshire", "New Jersey", "New Mexico", "New York", "North Carolina", "North Dakota", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota", "Tennessee", "Texas", "Utah", "Vermont", "Virginia", "Washington", "West Virginia", "Wisconsin", "Wyoming", "Guam", "Virgin Islands", "Guam", "District of Columbia"
Cases: "confirmed cases", "deaths", "new cases", "daily cases", "cases"
Month: "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"
Value Entity: "highest", "lowest", "most", "least"
Amount Entity: "number of","percentage of"
Number: "0", "1","2","3","4","5","6","7","8","9","10"
Day: "1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th", "9th", "10th", "11th", "12th", "13th", "14th", "15th", "16th", "17th", "18th", "19th", "20th", "21st", "22nd", "23rd", "24th", "25th", "26th", "27th", "28th", "29th", "30th", "31st"
Weekday: "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"
Race: "African-American", "Hispanic", "Latino", "Black", "Asian", "American Indian", "White", "Caucasian", "Native Hawaiian", "Alaska Native", "American Indian or Alaska Native", "Pacific Islander", "Pacific Islander and Native Hawaiian", "multiracial", "mixed"
Demographic Entity: "sex", "race", "age", "age group", "race and ethnicity", "ethnicity", "gender"
State Abbreviation: "AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY", "PR", "GU", "VI"

Appendix B Question-SQL Templates Table

	Question Template	SQL-Template
Database 1:	**How many (Case Entity) occurred in (State Entity) (Time Entity)?** **Give me the number of (Case Entity) that occurred in (State Entity) (Time Entity)?** **Provide me with the number of (Case Entity) that occurred in (State Entity) (Time Entity)?** **List the number of (Case Entity) that occurred in (State Entity) (Time Entity)?**	**For Queries Involving Single Dates:** **(Note: The time entity in this query determines the table_name)** **Select SUM(Case Entity Column) from Time Entity where Province_State = 'State Entity'** **For Queries Involving a Range of Dates:** **(Note: The time entity in this query determines the table_name)** **Select(Select SUM(Case Entity Column) from Time Entity where Province_State = 'State Entity') - (Select SUM(Case Entity Column) from Time Entity where Province_State = 'State Entity')**
	How many (Case Entity) occurred in (County Entity) (State Entity) (Time Entity)? **Give me the number of (Case Entity) that occurred in (County Entity) (State Entity) (Time Entity)?** **Provide me with the number of (Case Entity) that occurred in (County Entity) (State Entity) (Time Entity)?** **List the number of (Case Entity) that occurred in (County Entity) (State Entity) (Time Entity)?**	**For Queries Involving Single Dates:** **(Note: The time entity in this query determines the table_name)** **Select SUM(Case Entity Column) from Time Entity where Admin2 = 'County Entity' and Province_State = 'State Entity'** **For Queries Involving a Range of Dates:** **(Note: The time entity in this query determines the table_name)** **Select(Select SUM(Case Entity Column) from Time Entity where Admin2 = 'County Entity' and Province_State = 'State Entity') - (Select SUM(Case Entity Column) from Time Entity where Admin2 = 'County Entity' and Province_State = 'State Entity')**
	How many (Case Entity) occurred in (Province Entity) (Country Entity) (Time Entity)? **Give me the number of (Case Entity) that occurred in (Province Entity) (Country Entity) (Time Entity)?** **Provide me with the number of (Case Entity) that occurred in (Province Entity) (Country Entity) (Time Entity)?** **List the number of (Case Entity) that occurred in (Province Entity) (Country Entity) (Time Entity)?**	**For Queries Involving Single Dates:** **(Note: The time entity in this query determines the table_name)** **Select SUM(Case Entity Column) from Time Entity where Province_State = 'Province Entity' and Country_Region = 'State Entity'** **For Queries Involving a Range of Dates:** **(Note: The time entity in this query determines the table_name)** **Select(Select SUM(Case Entity Column) from Time Entity where Province_State = 'Province Entity' and Country_Region = 'Country Entity') - (Select SUM(Case Entity Column) from Time Entity where Province_State = 'Province Entity' and Country_Region = 'Country Entity')**
	How many (Case Entity) occurred in (Country Entity) (Time Entity)? **Give me the number of (Case Entity) that occurred in (Country Entity) (Time Entity)?** **Provide me with the number of (Case Entity) that occurred in (Country Entity) (Time Entity)?** **List the number of (Case Entity) that occurred in (Province Entity) (Country Entity) (Time Entity)?**	**For Queries Involving Single Dates:** **(Note: The time entity in this query determines the table_name)** **Select SUM(Case Entity Column) from Time Entity where Country_Region = 'State Entity'** **For Queries Involving a Range of Dates:** **(Note: The time entity in this query determines the table_name)** **Select(Select SUM(Case Entity Column) from Time Entity where Country_Region = 'Country Entity') - (Select SUM(Case Entity Column) from Time Entity where Country_Region = 'Country Entity')**
	What is the (Rate Entity) in (State Entity) (Time Entity)? **Give me the (Rate Entity) in (State Entity) (Time Entity).**	**(Note: only single dates are taken. The time entity in this query determines the table_name)** **Select Rate Entity Column from Time Entity where Province_State = 'State Entity'**

	Provide me with the (Rate Entity) in (State Entity) (Time Entity). **List the (Rate Entity) in (State Entity) (Time Entity).**	
	What is the (Rate Entity) in (County Entity) (State Entity) (Time Entity)? **Give me the (Rate Entity) in (County Entity) (State Entity) (Time Entity).** **Provide me with the (Rate Entity) in (County Entity) (State Entity) (Time Entity).** **List the (Rate Entity) in (County Entity) (State Entity) (Time Entity).**	**(Note: only single dates are taken. The time entity in this query determines the table_name)** **Select Rate Entity Column from Time Entity where Admin2 = 'County Entity' and Province_State = 'State Entity'**
	What is the (Rate Entity) in (Province Entity) (Country Entity) (Time Entity)? **Give me the (Rate Entity) in (Province Entity) (Country Entity) (Time Entity).** **Provide me with the (Rate Entity) in (Province Entity) (Country Entity) (Time Entity).** **List the (Rate Entity) in (Province Entity) (Country Entity) (Time Entity).**	**(Note: only single dates are taken. The time entity in this query determines the table_name)** **Select Rate Entity Column from Time Entity where Province_State = 'Province Entity' and Country_Region = 'Country Entity'**
	What is the (Rate Entity) in (Country Entity) (Time Entity)? **Give me the (Rate Entity) in (Country Entity) (Time Entity).** **Provide me with the (Rate Entity) in (Country Entity) (Time Entity).** **List the (Rate Entity) in (Country Entity) (Time Entity).**	**(Note: only single dates are taken. The time entity in this query determines the table_name)** **Select Rate Entity Column from Time Entity where Country_Region = 'Country Entity'**
	Which (Location Entity) has the (Value Entity) number of (Case Entity) (Time Entity) **Give me the (Location Entity) that has the (Value Entity) number of (Case Entity) (Time Entity).** **Provide me with the (Location Entity) that has the (Value Entity) number of (Case Entity) (Time Entity).** **List the (Location Entity) that has the (Value Entity) number of (Case Entity) (Time Entity).** **Note: Location Entity can be [state, country, county, province]**	**For Queries Involving Single Dates:** **If Location Entity = 'county':** **Select Admin2, Province_State from Time Entity and (Null) Group by Admin2, Province_State order by SUM(Case Entity Column) Value Entity** **If Location Entity = 'state':** **Select Province_State from Time Entity and (Null) Group by Province_State order by SUM(Case Entity Column) Value Entity** **If Location Entity = 'province':** **Select Province_State, Country_Region from Time Entity and (Null) Group by Province_State, Country_Region order by SUM(Case Entity Column) Value Entity** **If Location Entity = 'country':** **Select Country_Region from Time Entity and (Null) Group by Country_Region order by SUM(Case Entity Column) Value Entity**
		For Queries Involving a Range of Dates: **If Location Entity = 'county':** **Select t1.Admin2, t1.Province_State from (Select Admin2, Province_State, SUM(Case Entity Column) as Value (Null) from Time Entity Group by Admin2, Province_State) as t1 Inner Join(Select Admin2, Province_State, SUM(Case Entity Column) as**

		Value2 (Null) from Time Entity Group by Admin2, Province_State) as t2 on t1.Admin2 = t2.Admin2 and t1.Province_State = t2.Province_State order by t1.Value - t2.Value Value Entity **If Location Entity = 'state':** **Select t1.Province_State from (Select Province_State, SUM(Case Entity Column) as Value (Null) from Time Entity Group by Province_State) as t1 Inner Join(Select Province_State, SUM(Case Entity Column) as Value2 (Null) from Time Entity Group by Province_State) as t2 on t1.Province_State = t2.Province_State order by t1.Value - t2.Value** **Value Entity** **If Location Entity = 'province':** **Select t1.Province_State, t1.Country_Region from (Select Province_State, Country_Region, SUM(Case Entity Column) as Value (Null) from Time Entity Group by Province_State, Country_Region) as t1 Inner Join(Select Province_State, Country_Region, SUM(Case Entity Column) as Value2 (Null) from Time Entity Group by Province_State, Country_Region) as t2 on t1.Province_State = t2.Province_State and t1.Country_Region = t2.Country_Region order by t1.Value - t2.Value** **Value Entity** **If Location Entity = 'country':** **Select t1.Country_Region from (Select Country_Region, SUM(Case Entity Column) as Value (Null) from Time Entity Group by Country_Region) as t1 Inner Join(Select Country_Region, SUM(Case Entity Column) as Value2 (Null) from Time Entity Group by Country_Region) as t2 on t1.Country_Region = t2.Country_Region order by t1.Value - t2.Value** **Value Entity**
	Which (Location Entity) has the (Value Entity) (Rate Entity) (Time Entity) **Give me the (Location Entity) that has the (Value Entity) (Rate Entity)(Time Entity).** **Provide me with the (Location Entity) that has the (Value Entity) (Rate Entity) (Time Entity).** **List the (Location Entity) that has the (Value Entity) (Rate Entity) (Time Entity).** **Note: Location Entity can be [state, country, county, province]**	**For Queries Involving Single Dates:** **If Location Entity = 'county':** **Select Admin2, Province_State from Time Entity and (Null) Group by Admin2, Province_State order by Rate Entity Column Value Entity** **If Location Entity = 'state':** **Select Province_State from Time Entity and (Null) Group by Province_State order by Rate Entity Column Value Entity** **If Location Entity = 'province':** **Select Province_State, Country_Region from Time Entity and (Null) Group by Province_State, Country_Region order by Rate Entity Column Value Entity** **If Location Entity = 'country':** **Select Country_Region from Time Entity and (Null) Group by Country_Region order by Rate Entity Column Value Entity** **For Queries Involving a Range of Dates:** **If Location Entity = 'county':** **Select t1.Admin2, t1.Province_State from (Select Admin2, Province_State, Rate Entity Column as Value (Null) from Time Entity Group by Admin2, Province_State) as t1 Inner Join(Select Admin2, Province_State, Rate Entity Column as Value2 (Null) from Time Entity Group by Admin2, Province_State) as t2 on t1.Admin2 = t2.Admin2 and t1.Province_State = t2.Province_State order by t1.Value - t2.Value Value Entity** **If Location Entity = 'state':** **Select t1.Province_State from (Select Province_State, Rate Entity Column as Value (Null) from Time Entity Group by Province_State) as t1 Inner Join(Select Province_State, Rate Entity Column as Value2 (Null) from Time Entity Group by**

		Province_State) as t2 on t1.Province_State = t2.Province_State order by t1.Value - t2.Value **Value Entity** **If Location Entity = 'province':** **Select t1.Province_State, t1.Country_Region from (Select Province_State, Country_Region, Rate Entity Column as Value (Null) from Time Entity Group by Province_State, Country_Region) as t1 Inner Join(Select Province_State, Country_Region, Rate Entity Column as Value2 (Null) from Time Entity Group by Province_State, Country_Region) as t2 on t1.Province_State = t2.Province_State and t1.Country_Region = t2.Country_Region order by t1.Value - t2.Value** **Value Entity** **If Location Entity = 'country':** **Select t1.Country_Region from (Select Country_Region, Rate Entity Column as Value (Null) from Time Entity Group by Country_Region) as t1 Inner Join(Select Country_Region, Rate Entity Column as Value2 (Null) from Time Entity Group by Country_Region) as t2 on t1.Country_Region = t2.Country_Region order by t1.Value - t2.Value** **Value Entity**
Database 2:	**What is the (Rate Entity) in (State Entity)?** **Give me the (Rate Entity) in (State Entity).** **Provide me with the (Rate Entity) in (State Entity).** **List the (Rate Entity) in (State Entity),**	**Select Rate Entity Column from table_name where date = 'current date' and state = "State Entity"**
	What is the (Rate Entity) in the United States? **Give me the (Rate Entity) in the United States.** **Provide me with the (Rate Entity) in the United States,** **List the (Rate Entity) in the United States.**	**Select Rate Entity Column from table_name where date = 'current date'**
	Which state has the (Value Entity) (Hospitalization Entity)? **Give me the state with the (Value Entity) (Hospitalization Entity).** **Provide me with the state with the (Value Entity) (Hospitalization Entity).** **List the state with the (Value Entity) (Hospitalization Entity).**	**Select state from table_name where date = 'given date' and (Null) order by Hospitalization Entity Column Value Entity**
	How many people are (Hospitalization Entity) in (State Entity)? **Give me the number of people who are (Hospitalization Entity) in (State Entity).** **Provide me with the number of people who are (Hospitalization Entity) in (State Entity).** **List the number of people who are (Hospitalization Entity) in (State Entity).**	**Select Hospitalization Entity Column from table_name where date = 'current date' and state = "State Entity"**
	How many people are (Hospitalization Entity) in the United States? **Give me the number of people who are (Hospitalization Entity) in the United States.** **Provide me with the number of people who are (Hospitalization Entity) in the United States.**	**Select Hospitalization Entity Column from table_name where date = 'current date'**

	List the number of people who are (Hospitalization Entity) in the United States.	
	What is the number of (Testing Entity) done in (State Entity) (Time Entity)? **Give me the number of (Testing Entity) done in (State Entity) (Time Entity).** **Provide me with the number of (Testing Entity) done in (State Entity) (Time Entity).** **List the number of (Testing Entity) done in (State Entity) (Time Entity).**	**For Queries Involving a Single Date:** **Select Testing Entity Column from table_name where date = 'Time Entity' and state = "State Entity"** **For Queries Involving a Range of Dates:** **Select (Select Testing Entity Column from table_name where date = 'Time Entity' and state = "State Entity") - (Select Testing Entity Column from table_name where date = 'Time Entity' and state = "State Entity")**
	What is the number of (Testing Entity) done in the United States (Time Entity)? **Give me the number of (Testing Entity) done in the United States (Time Entity).** **Provide me with the number of (Testing Entity) done in the United States (Time Entity),** **List the number of (Testing Entity) done in the United States (Time Entity).**	**For Queries Involving a Single Date:** **Select Testing Entity Column from table_name where date = 'Time Entity'** **For Queries Involving a Range of Dates:** **Select (Select Testing Entity Column from table_name where date = 'Time Entity') - (Select Testing Entity Column from table_name where date = 'Time Entity')**
	Which state has the (Value Entity) (Testing Entity) (Time Entity)? **Give me the state with the (Value Entity) (Testing Entity) (Time Entity).** **Provide me with the state with the (Value Entity) (Testing Entity) (Time Entity).** **List the state with the (Value Entity) (Testing Entity) (Time Entity).**	**For Queries Involving a Single Date:** **Select state from table_name where date = 'Time Entity' and (Null) order by Testing Entity Column Value Entity** **For Queries Involving a Range of Dates:** **Select t1.state from (Select state, Testing Entity Column from table_name where date = 'Time Entity' and (Null)) as t1 Inner Join (Select state, Testing Entity Column from table_name where date = 'Time Entity' and (Null)) as t2 on t1.state = t2.state order by t1.Testing Entity Column-t2.Testing Entity Column Value Entity**
	What percentage of (Case Entity) in (State Entity) are (Race Entity)? **Give me the percentage of (Case Entity) in (State Entity) that are (Race Entity).** **Provide me with the percentage of (Case Entity) in (State Entity) that are (Race Entity).** **List the percentage of (Case Entity) in (State Entity) that are (Race Entity).**	**Select Case Entity Column_Race Entity Column from table_name where date = 'given date' and state = "State Entity"**
	Which state has the (Value Entity) percentage of (Race Entity) (Case Entity)? **Give me the state with the (Value Entity) percentage of (Race Entity) (Case Entity).** **Provide me with the state with the (Value Entity) percentage of (Race Entity) (Case Entity).** **List the state with the (Value Entity) percentage of (Race Entity) (Case Entity).**	**Select state from table_name where date = 'given date' and Case Entity Column_Race Entity Column is not null order by Case Entity Column_Race Entity Column Value Entity**
	How many (Race Entity) (Case Entity) occurred in (State Entity)(Time Entity)?	**For Queries Involving a Single Date:** **Select Case Entity Column_Race Entity Column from table_name where date = 'Time Entity' and state = "State Entity"**

	Give me the number of (Race Entity) (Case Entity) that occurred in (State Entity) (Time Entity). **Provide me with the number of (Race Entity) (Case Entity) that occurred in (State Entity) (Time Entity).** **List the number of (Race Entity) (Case Entity) that occurred in (State Entity) (Time Entity).**	**For Queries Involving a Range of Dates:** **Select(Select Case Entity Column_Race Entity Column from table_name where date = 'Time Entity' and state = "State Entity") - (Select Case Entity Column_Race Entity Column from table_name where date = 'Time Entity' and state = "State Entity")**
	Which state has the (Value Entity) (Rate Entity)? **Give me the state with the (Value Entity) (Rate Entity).** **Provide me with the state with the (Value Entity) (Rate Entity)** **List the state with the (Value Entity) (Rate Entity)**	**Select state from table_name where date = 'current date' and (Null) order by Rate Entity Column Value Entity**
	What is the racial breakdown of (Case Entity) in (State Entity)? **Give me the racial breakdown of (Case Entity) in (State Entity).** **Provide me with the racial breakdown of (Case Entity) in (State Entity).** **List the racial breakdown of (Case Entity) in (State Entity).**	**Select Case Entity Column_Total,Case Entity Column_White, Case Entity Column_Black,Case Entity Column_LatinX, Case Entity Column_Asian,Case Entity Column_NHPI, Case Entity Column_Multiracial, Case Entity Column_Other, Case Entity Column_Unknown from table_name where date = 'given date' and state = "State Entity"**
Database 3:	**What is the number of (Bed Entity) in (State Entity)?** **Give me the number of (Bed Entity) in (State Entity).** **Provide me with the number of (Bed Entity) in (State Entity),** **List the number of (Bed Entity) in (State Entity).**	**For State Entity:** **Select Sum(Bed Entity Column) From table_name Where STATE_Name = "State Entity"**
	What is the number of (Bed Entity) in (County Entity) (State Entity)? **Give me the number of (Bed Entity) in (County Entity) (State Entity),** **Provide me with the number of (Bed Entity) in (County Entity) (State Entity),** **List the number of (Bed Entity) in (County Entity) (State Entity).**	**For County Entity and State Entity:** **Select SUM(Bed Entity Column) from table_name where STATE_Name = "State Entity" and COUNTY_Name = "County Entity"**
	Which (Location Entity) has the (Value Entity) number of (Bed Entity)? **Give me the (Location Entity) that has the (Value Entity) number of (Bed Entity).** **Provide me with the (Location Entity) that has the (Value Entity) number of (Bed Entity).** **List the (Location Entity) that has the (Value Entity) number of (Bed Entity).**	**If Location Entity = 'state'** **Select * from (Select STATE_NAME, SUM(Bed Entity Column) as sb from table_name Group by STATE_NAME Order by SUM(Bed Entity Column)) where sb > 0 Value Entity**
		If Location Entity = 'county' **For County Entity and State Entity:**

		Select * from (Select COUNTY_NAME, STATE_NAME, SUM(Bed Entity Column) as sb from table_name Group by COUNTY_NAME,STATE_NAME Order by SUM(Bed Entity Column)) where sb > 0 Value Entity
Database 4:	**What is the breakdown of (Case Entity) by (Demographic Entity) in the United States?** **Give me the breakdown of (Case Entity) by (Demographic Entity) in the United States?** **Provide me with the breakdown of (Case Entity) by (Demographic Entity) in the United States.** **List the breakdown of (Case Entity) by (Demographic Entity) in the United States.**	**Select * from table_name**
	Which (Demographic Entity) has the (Value Entity) (Amount Entity) (Case Entity) in the United States? **Give me the (Demographic Entity) that has the (Value Entity) (Amount Entity) (Case Entity) in the United States.** **Provide me with the (Demographic Entity) that has the (Value Entity) (Amount Entity) (Case Entity) in the United States.** **List the (Demographic Entity) that has the (Value Entity) (Amount Entity) (Case Entity) in the United States.**	**Select Demographic Entity, Amount Entity from table_name Order by Amount Entity Value Entity**
	What is the total forecasted number of deaths in (State Entity) (Time Entity)? **Give me the total forecasted number of deaths in (State Entity) (Time Entity).** **Provide me with the total forecasted number of deaths in (State Entity) (Time Entity).** **List the total forecasted number of deaths in (State Entity) (Time Entity).**	**Select Max(point) from table_name where target_week_end_date = 'Time Entity' and location_name = "State Entity"**
	What is the total forecasted number of deaths in the United States (Time Entity)? **Give me the total forecasted number of deaths in the United States (Time Entity).** **Provide me with the total forecasted number of deaths in the United States (Time Entity).** **List the total forecasted number of deaths in the United States (Time Entity).**	**Select Max(point) from table_name where target_week_end_date = 'Time Entity' and location_name = 'National'**
	Which state will have the (Value Entity) total forecasted number of deaths (Time Entity)? **Give me the state with the (Value Entity) total forecasted number of deaths (Time Entity).** **Provide me with the state with the (Value Entity) total forecasted number of deaths (Time Entity).** **List the state with the (Value Entity) total forecasted number of deaths (Time Entity).**	**Select location_name, Max(point) from table_name WHERE target_week_end_date = 'Time Entity' and location_name != 'National' group by location_name order by Max(point) Value Entity**
	What is the percentage change in (Mobility Entity) in (State Entity) (Time Entity)?	**For Queries Involving Single Dates:** **For State Entity:**

	Give me the percentage change in (Mobility Entity) in (State Entity) (Time Entity). **Provide me with the percentage change in (Mobility Entity) in (State Entity) (Time Entity).** **List the percentage change in (Mobility Entity) in (State Entity) (Time Entity).** **What is the percentage change in (Mobility Entity) in (County Entity) (State Entity) (Time Entity)?** **Give me the percentage change in (Mobility Entity) in (County Entity) (Time Entity).** **Provide me with the percentage change in (Mobility Entity) in (County Entity) (Time Entity).** **List the percentage change in (Mobility Entity) in (County Entity) (Time Entity).**	**Select Mobility Entity Column FROM table_name WHERE date = "Time Entity" AND country_region = "United States" AND sub_region_1 = "State Entity" and iso_3166_2_code LIKE "US-%"** **For County Entity and State Entity:** **Select Mobility Entity Column FROM table_name WHERE date = "Time Entity" AND country_region = "United States" AND sub_region_1 = "State Entity" AND sub_region_2 = "County Entity"** **For Queries Involving Range of Dates:** **For State Entity:** **Select (Select Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" and iso_3166_2_code LIKE "US-%") - (Select Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" and iso_3166_2_code LIKE "US-%")** **For County Entity and State Entity:** **Select (Select Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" AND sub_region_2 = "County Entity") - (Select Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" AND sub_region_2 = "County Entity")**
	Which county had the (Value Entity) percentage change in (Mobility Entity) (Time Entity)? **Give me the county with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **Provide me with the county with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **List the county with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **Which county in (State Entity) had the (Value Entity) percentage change in (Mobility Entity) (Time Entity)?** **Give me the county in (State Entity) with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **Provide me with the county in (State Entity) with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **List the county in (State Entity) with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **Which state had the (Value Entity) percentage change in (Mobility Entity) (Time Entity)** **Give me the state with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).** **Provide me with the state with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).**	**For Queries Involving Single Dates:** **For state:** **Select sub_region_1, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and iso_3166_2_code like "US-%" and Mobility Entity Column is not null order by Mobility Entity Column Value Entity** **For county in (State Entity):** **Select sub_region_2, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" and sub_region_2 is not null and Mobility Entity Column is not null order by Mobility Entity Column Value Entity** **For county:** **Select sub_region_2, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_2 is not null and Mobility Entity Column is not null order by Mobility Entity Column Value Entity** **For Queries Involving Range of Dates:** **For state:** **Select t1.sub_region_1, t1.Mobility Entity Column-t2.Mobility Entity Column from(Select sub_region_1, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and iso_3166_2_code like "US-%" and Mobility Entity Column is not null) as t1 Inner Join (Select sub_region_1, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and iso_3166_2_code like "US-%" and Mobility Entity Column is not null) as t2 on t1.sub_region_1=t2.sub_region_1 order by t1.Mobility Entity Column-t2.Mobility Entity Column Value Entity** **For county in (State Entity):** **Select t1.sub_region_2, t1.Mobility Entity Column-t2.Mobility Entity Column from(Select sub_region_2, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" and sub_region_2 is not null and Mobility Entity Column is not null) as t1 Inner Join (Select sub_region_2, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_1 = "State Entity" and sub_region_2 is not null and Mobility Entity Column is not null) as t2 on**

	List the state with the (Value Entity) percentage change in (Mobility Entity) (Time Entity).	**t1.sub_region_2=t2.sub_region_2 order by t1.Mobility Entity Column-t2.Mobility Entity Column Value Entity** **For county:** **Select t1.sub_region_2, t1.Mobility Entity Column-t2.Mobility Entity Column from(Select sub_region_2, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_2 is not null and Mobility Entity Column is not null) as t1 Inner Join (Select sub_region_2, Mobility Entity Column from table_name where date = 'Time Entity' and country_region = 'United States' and sub_region_2 is not null and Mobility Entity Column is not null)** **as t2 on t1.sub_region_2=t2.sub_region_2 order by t1.Mobility Entity Column-t2.Mobility Entity Column Value Entity**
Database 5:	**What percentage of deaths from (County Entity) (State Entity) are (Race Entity)?** **Give me the percentage of deaths from (County Entity) (State Entity) that are (Race Entity).** **Provide me with the percentage of deaths from (County Entity) (State Entity) that are (Race Entity).** **List the percentage of deaths from (County Entity) (State Entity) that are (Race Entity).**	**Select Race Entity Column from table_name where Indicator = 'Distribution of COVID-19 deaths (%)' and County_Name = "County Entity" and State = "State Entity"**
	Which county in (State Entity) has the (Value Entity) percentage of (Race Entity) deaths **Give me the county in (State Entity) with the (Value Entity) percentage of (Race Entity) deaths.** **Provide me with the county in (State Entity) with the (Value Entity) percentage of (Race Entity) deaths.** **List the county in (State Entity) with the (Value Entity) percentage of (Race Entity) deaths.**	**Select County_Name from table_name where Indicator = 'Distribution of COVID-19 deaths (%)' and State = 'State Entity' order by Race Entity Column Value Entity**
	How many (Race Entity) deaths occurred in (County Entity) (State Entity)? **Give me the number of (Race Entity) deaths that occurred in (County Entity) (State Entity).** **Provide me with the number of (Race Entity) deaths that occurred in (County Entity) (State Entity).** **List the number of (Race Entity) deaths that occurred in (County Entity) (State Entity).**	**Select Round((Select Race Entity Column from table_name where Indicator = 'Distribution of COVID-19 deaths (%)' and State = "State Entity" and County_Name = "County Entity") * (Select Deaths from table_name where Indicator = 'Distribution of COVID-19 deaths (%)' and State = "State Entity" and County_Name = "County Entity"))**
	What is the racial breakdown of COVID-19 deaths in (County Entity) (State Entity)? **Give me the racial breakdown of COVID-19 deaths in (County Entity) (State Entity).** **List the racial breakdown of COVID-19 deaths in (County Entity) (State Entity).** **Provide me with the racial breakdown of COVID-19 deaths in (County Entity) (State Entity).**	**Select Deaths, Non_Hispanic_White, Non_Hispanic_Black, Non_Hispanic_AIAN, Non_Hispanic_Asian, Other, Hispanic from table_name where Indicator = "Distribution of COVID-19 deaths (%)" and State = "State Entity" and County_Name = "County Entity"**

Database 6:	**What are the number of (Testing Entity) done by (Country Entity) (Time Entity)?** **Give me the number of (Testing Entity) done by (Country Entity) (Time Entity).** **Provide me with the number of (Testing Entity) done by (Country Entity) (Time Entity).** **List the number of (Testing Entity) done by (Country Entity) (Time Entity).**	**For Queries Involving Single Dates:** **Select Testing Entity Column from table_name where Entity = "Country Name" and Date = "Time Entity"** **For Queries Involving Range of Dates:** **Select (Select Testing Entity Column from table_name where date = 'Time Entity' and Entity = "Country Entity") - (Select Testing Entity Column from table_name where date = 'Time Entity' and Entity = "Country Entity")**
	What is the (Rate Entity) in (Country Entity) (Time Entity)? **Give me the (Rate Entity) in (Country Entity) (Time Entity).** **Provide me with the (Rate Entity) in (Country Entity) (Time Entity).** **List the (Rate Entity) in (Country Entity) (Time Entity).**	**Select Rate Entity Column from table_name where date = 'Time Entity' and Entity = "Country Entity"**
	Which country has the (Value Entity) (Testing Entity) (Time Entity)? **Give me the country with the (Value Entity) (Testing Entity) (Time Entity).** **Provide me with the country with the (Value Entity) (Testing Entity) (Time Entity).** **List the country with the (Value Entity) (Testing Entity) (Time Entity).**	**For Queries Involving Single Dates:** **Select Entity from table_name where date = 'Time Entity' order by Testing Entity Column Value Entity** **For Queries Involving Range of Dates:** **Select e1 from (Select Entity as E1, Testing Entity Column as tot1 from table_name where date = 'Time Entity') as t1 Inner Join (Select Entity as e2, Testing Entity Column as tot2 from table_name where date = 'Time Entity') as t2 on t1.e1=t2.e2 order by t1.tot1-t2.tot2 Value Entity**
	Which country has the (Value Entity) (Rate Entity) (Time Entity)? **Give me the country with the (Value Entity) (Rate Entity) (TIme Entity).** **Provide me with the country with the (Value Entity) (Rate Entity) (Time Entity).** **List the country with the (Value Entity) (Rate Entity) (Time Entity).**	**Select Entity from table_name where date = 'Time Entity' order by Rate Entity Column Value Entity**