# Survey on Construction of Knowledge Graphs for Clinical Practice Guidelines and Question Answering on Clinical Practice Guidelines

**Vasudhan Varma Kandula**
CSE, IITB / Bombay
vasudhanvarma@gmail.com

**Pushpak Bhattacharyya**
CSE, IITB / Bombay
pushpakbh@gmail.com

## Abstract

In the medical domain, several disease treatment procedures have been documented properly as a set of instructions known as Clinical Practice Guidelines (CPGs). CPGs have been developed over the years on the basis of past treatments, and are updated frequently. A doctor treating a particular patient can use these CPGs to know how past patients with similar conditions were treated successfully and can find the recommended treatment procedure. In this paper, we present a Decision Knowledge Graph (DKG) representation to store CPGs and to perform question-answering on CPGs. CPGs are very complex and no existing representation is suitable to perform question-answering and searching tasks on CPGs. As a result, doctors and practitioners have to manually wade through the guidelines, which is inefficient. Representation of CPGs is challenging mainly due to frequent updates on CPGs and decision-based structure. Our proposed DKG has a decision dimension added to a Knowledge Graph (KG) structure, purported to take care of decision based behavior of CPGs. Using this DKG has shown 40% increase in accuracy compared to fine-tuned BioBert model in performing question-answering on CPGs. To the best of our knowledge, ours is the first attempt at creating DKGs and using them for representing CPGs.

## 1 Introduction

### 1.1 Problem Statement

Clinical Practice Guidelines (CPGs) are a set of systematically developed statements intended to assist a doctor or a practitioner to make decisions about appropriate health care to be given to a patient under a specific clinical circumstance. CPGs serve as critical resources for healthcare practitioners, providing evidence-based recommendations to guide decision-making and improve patient care. However, the current utilization of CPGs faces significant challenges due to the lack of a suitable representation. These guidelines, developed over years and regularly updated, consist of extensive documentation detailing disease treatments and steps. As a result, healthcare professionals are required to invest considerable time and effort in manually searching and comprehending the guidelines, leading to inefficiencies and potential delays in treatment decisions.

The absence of a structured and accessible format hampers the efficient use of CPGs, hindering the translation of evidence-based knowledge into clinical practice. To overcome these limitations, there is a pressing need to develop an improved representation for CPGs that enables easier searching, navigation, and question-answering. By addressing this problem, healthcare professionals can enhance their ability to provide optimal care, improve patient outcomes, and efficiently allocate healthcare resources.

### 1.2 Motivation

The main motivations are:

- Failure to follow Guidelines
- Digitizing Clinical Practice Guidelines
- Increasing load on doctors
- Lack of Data

#### 1.2.1 Failure to follow Clinical Practice Guidelines (CPGs)

According to American Hospital Association (aha), in 2022, there were more than 33 million admissions of patients in hospitals in the US, which is an average of 91,000 admissions per day. As the number of patients is increasing, there is a heavy workload on doctors, and they may have limited time to review and implement complex guidelines. Also, doctors may be unfamiliar with CPGs due to lack of training, and frequent changes in guidelines over time. Lack of familiarity with CPGs

can be a barrier to their use in clinical practice, as doctors may not be aware of the most up-to-date recommendations or may not know how to apply the guidelines to their patients. So, to promote the usage of CPGs, the above barriers need to be overcome. One way to achieve this is by digitizing the guidelines and providing assistance when referring to the guidelines using technology.

#### 1.2.2 Digitizing Clinical Practice Guidelines (CPGs)

Nowadays, when everything is being digitized, CPGs are still finding difficulty because of their complexity and lack of proper representation in digital format. Adding to these, there is one more challenge that CPGs face. CPGs are updated regularly based on discussions. So, even if digitized, the model should allow those changes in digital representations. If CPGs are digitized properly, searching and navigating becomes easier and saves a lot of time for doctors, and motivates doctors to use CPGs. The main motivation for digitizing a CPG is to assist doctors and practitioners when treating cancer patients. An application using a suitable CPG representation should be able to answer basic questions like what are the next treatment steps given the current patient's condition. It should be easy for users to navigate the CPG. The existing Knowledge Graph representation on which searching and question-answering can be performed is not suitable for storing CPGs as CPGs contain a decision-based structure along with factual data and these decisions in CPGs are updated frequently. Given the following guideline:

> *"Patient can be treated with chemotherapy if age less than 65"*

The existing KG extraction model gave: Subject: *Patient*; Predicate: *can be treated with*; *Object: chemotherapy*. So, the extracted triple is *(patient, can be treated with, chemotherapy)*. The model ignored the condition of age less than 65, which is important for guiding the doctor. So, a good CPG knowledge graph should represent not only concepts but also decisions (attributes). If the above guideline is updated to:

> *"Patient can be treated with chemotherapy if age less than 65 and greater than 35. He should not have any substantial comorbidities."*

The existing KG model will require many changes in its structure (i.e, number of nodes and relations). A good CPG knowledge graph representation should have an efficient updating capability with few changes.

#### 1.2.3 Increasing Load on Doctors

In a survey done by NCI, it has been found that 1 in 10 men and 1 in 8 women in India can expect to develop cancer of any form, in their life span after the age of 35 years. The rate of new cases of cancer is 442.4 per 100,000 men and women per year from 2013-2017 across the world. In India, around 2.7 million people are suffering from cancer and every year, 13.9 lakh new cases are registered. A cancer patient's treatment generates a lot of clinical reports and notes. Doctors find it difficult to go through all the reports and information present in them. To remember a patient's data there is much cognitive load on the doctor. With the increase in the number of patients, the doctor's load is also getting increased for going through the reports. The increased time in reading reports makes a doctor spend less time treating a patient. Consequently, he won't be able to treat many patients. With the increasing number of patients every year, this will be a problem in the future to treat cancer patients.

#### 1.2.4 Lack of Data

Now-a-days pretrained models are performing well in question-answering tasks. The limitation in the case of CPGs is that sufficient data for training the model is not available. Even if we create huge data and train the model, since the treatment guidelines are changed frequently, the dataset should also be updated with the guidelines which is another limitation. Considering this, storing the CPGs as DKG is a better approach compared to using a pre-trained transformer model.

### 1.3 Contributions

The main contributions of our work are:

1. Creation and releasing of a knowledge graph (KG) with an additional decision dimension added to some nodes in existing KG structure for storing clinical practice guidelines, *i.e., Decision Knowledge Graph (DKG)*.

2. Creation of dataset of triples containing 8300 questions from acute lymphoblastic leukemia, kidney, and bone cancer. Each *triple* consists

of *question*, *answer*, and *cypher query* (used to query decision knowledge graph).

3. Question-answering model on Clinical Practice Guidelines with the help of Decision Knowledge Graphs. The proposed model gives 40% better results compared to fine-tuned transformer question-answering model.

To the best of our knowledge, ours is the first attempt at (i) creating a knowledge graph for CPGs and (ii) adding a decision dimension to a node in KG.

### 1.4 Challenges

Some of the challenges that are faced are:

- Clinical Practice Guidelines are regularly updated based on meetings conducted by member organizations. The digital representation of guidelines should be able to update quickly for the changes made in the guidelines.
- In guidelines, there are superscripts and subscripts which add some more specific details of the entities used in the flowchart, making it hard to parse.
- Capturing both the flow of treatment and interrelation between nodes in Clinical Practice Guidelines.

## 2 Background

In this chapter, we establish the groundwork for multiple concepts related to the project. To delve further into the project, it is essential to have an understanding of concepts such as knowledge graphs, differentiating between static and dynamic data, clinical practice guidelines (CPGs) with NCCN Guidelines, the neo4j database management system, cypher query language, constraint extractor and question-answering systems.

### 2.1 Knowledge Graphs

Knowledge Graphs are structured representations of data, often referred to as semantic networks, that store information in a graph-like format. While the concept of knowledge graphs has been referenced as early as 1972, their popularity surged when Google started utilizing them in 2012. These graphs serve as a network of real-world entities interconnected by relationships. They are typically stored in graph databases and visualized as graphs, hence the term "graph." Each node in the graph represents an entity, while the links between nodes capture the relationships between those entities. In a knowledge graph, information is organized in the form of tuples, consisting of a subject, relation, and object. Figure 1, derived from a survey paper on Knowledge graphs by Dai et al. (2020), presents a simplified knowledge graph illustrating a sequence of events and their corresponding venues.

Some of the prominent features of a knowledge graph are

- Variety of relations and,
- Information is organized in a complex network.

These knowledge graphs act as a knowledge base for various downstream tasks like question-answering, recommending systems, searching and navigating etc.

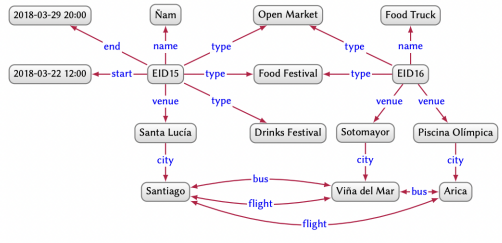


Figure 1: Knowledge Graph (KG) fragment representing a series of events and venues, derived from a survey paper on Knowledge graphs by Dai et al. (2020)

### 2.2 Clinical Practice Guidelines

Doctors may need assistance when they treat patients. There are many diseases and each disease is treated differently based on knowledge of how it has been treated before. Many discussions have been made on what is the recommended way to treat a patient based on his condition, and all the details of those discussions have been documented properly which are called Clinical Practice Guidelines (CPG). These CPG are updated regularly based on discussions held. A clear definition of CPGs is specified by the Institute of Medicine in 1990 **?**:

> ... systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances.

CPGs are exhaustively documented, but not enforced rigorously. They recommend the best treatment that can be chosen under given circumstances. CPGs are complex to understand for doctors, specifically beginner practitioners. As CPGs are currently not digitalized into a precise structured format, a practitioner searches manually through PDF files, which consumes a lot of time.

An example CPG fragment is shown in Figure 2. There are flowcharts that guide on where to start based on the disease state and the treatment goes to the caring stage (post-treatment phase). At the end of each document, an elaborate verbal description of what the flowchart means and other details about treatment steps and conditions are available. There are superscripts and subscripts which add some more specific details of the entities used in the flowchart, making it hard to parse.

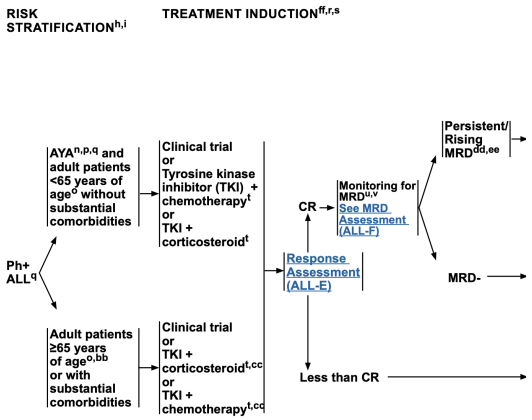


Figure 2: Fragment of Clinical Practice Guidelines by National Comprehensive Cancer Network from page 12 of Acute Lymphoblastic Leukemia (ALL) cancer Version 1.2022 which shows how a ph+ (Philadelphia chromosome) ALL patient should be treated in the induction phase of ALL cancer

| Acronym | Abbreviation |
|---|---|
| ALL | Acute Lymphoblastic Leukemia |
| Ph+ | Philadelphia chromosome-positive |
| AYA | Adolescents and Young Adults |
| TKI | Tyrosine Kinase Inhibitor |
| CR | Complete Recovery |
| MRD | Measurable Residual Disease |

Table 1: Shows the Abbreviations of Acronyms used in Figure 2

Figure 2 shows how a Ph+ ALL patient should be treated. This Ph+ ALL is determined by the doctors after doing initial tests on the patient. Here AYA$^{n,p,q}$ tells that the panel believes patients in the age range of 15–39 years. If the above patients do not have any substantial comorbidities. Substantial Comorbidities means having more than one illness at once. Then the patient is advised with the following treatments Clinical trial or TKI+Chemotherapy$^{t}$ which means systematic chemotherpy or TKI+corticosteroid$^{t}$ which means systematic corticosteroid. After these treatments, Response Assessment is done which classifies the patient as either CR or less than CR. If he is observed as CR, then MRD is monitored which can be persistent rising MRD or MRD- (MRD Negative). If the patient is of greater than 65 years of age then recommended treatments are Clinical trial or TKI+corticosteroid$^{t,cc}$ means we have to consider modifying dose according to patient's age or TKI+chemotherapy$^{t,cc}$ i.e., dose modifications required. Then Response Assessment is done on the patient.

The structure of CPGs contains flowcharts with decision nodes and detailed descriptions of these flowcharts are given in the discussion section in the same guidelines. This CPG structure resembles something like a flowchart combined with a knowledge graph. A flowchart captures on what is the next treatment step whereas a knowledge graph captures the node and the relationship of this node with other nodes. So, the challenge here is to capture both, the flow of treatment and the interrelation between nodes.

### 2.3 NCCN Guidelines

Clinical Practice Guidelines (CPGs) from National Comprehensive Cancer Network (NCCN) are used for building Decision Knowledge Graph (DKG). These are also referred to as Cancer Guidelines, NCCN Guidelines, or Oncology Guidelines. NCCN is a non-profit alliance dedicated to facilitating effective, quality, and accessible cancer care. The organization is home to around 60 types of cancer research and guidelines including breast cancer, lung cancer, kidney cancer, etc. For the past 25 years, these guidelines are updated regularly based on discussions among world-renowned experts from NCCN member institutions. A snapshot of the NCCN Guidelines, taken from page 12 of Acute Lymphoblastic Leukemia (ALL) Cancer Version 1.2022, is shown in Figure 2.

The NCCN guidelines include:

1. List of members and institutions that participated in the specified discussions.
2. Flowcharts for better understanding of decision making.
3. Discussions to provide support for flowcharts.
4. Evidence for recommendations and disclosure of potential conflicts of interest by panel members.

The flowchart section of guidelines consists of text boxes and arrows connecting these boxes as shown in Figure 2. Some of the words in the text have superscripts and subscripts. Superscripts and subscripts contain a detailed description in the footnote of the paper. There are hyper-texts in some text that refer to other pages in the same document.

In this work, Acute Lymphoblastic Leukemia (ALL), Bone, and Kidney cancer types are used from NCCN guidelines to build DKG. ALL cancer guidelines is a 135-page document consisting of more than 35 pages of flowcharts and algorithms for decision-making, 59 pages of discussion, and the remaining pages for references to evidence. Bone cancer guidelines is a 102-page document consisting of 34 pages of flowcharts and algorithms for decision-making, 32 pages of discussion, and the remaining pages for references to evidence. Kidney cancer guidelines is an 81-page document consisting of 23 pages of flowcharts and algorithms for decision-making, 34 pages of discussion, and the remaining pages for references to evidence.

### 2.4 Terminologies

In this section we define some new terminologies used in the report, patient constraints, static data, and dynamic data.

#### 2.4.1 Patient Constraints

Data related to patients' parameters and conditions of patient is called as *Patient's Constraints* which are often referred to as *Constraints* in rest of the report. Some of the examples of patients' constraints are *Age, tumor size, disease stage, past medical history, etc.*

#### 2.4.2 Static Data and Dynamic Data

Static data is the data in CPGs that changes less frequently or doesn't change at all. Example: The treatment steps of chemotherapy, etc. Dynamic data is the data in CPGs that changes frequently. Example: Patient Constraints. Here, dynamic data doesn't refer to data from a query like the name of the patient, etc. It refers to the data that should be present in the KG to make a decision. *For example, treatment procedure like chemotherapy is static data and patients' constraints like age>60, MRD rising, etc., is dynamic data.*

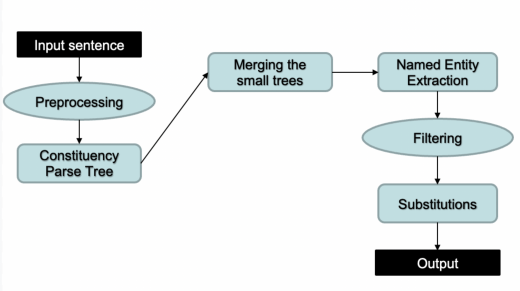


Figure 3: Block Diagram of Constraint Extraction

### 2.5 Constraint Extractor

Constraint Extractor is an extraction tool, which takes input a line from the oncology guidelines and outputs the constraints related to patient present in the guidelines. This tool has 2 major parts. They are:

- Constituency Parser
- Rules for extracting constraints

### 2.6 Constituency Parser

- Input: Natural Language sentence
- Output: Dependency tree

#### 2.6.1 Rules for extracting constraints

- Input: Dependency tree
- Output: List of constraints

A detailed information of how constraint extraction is done is defined in this section.

Step 1: Preprocessing Initially, data that is given as input to constraint extraction is tokenized using NLTK. From the tokenized output, all the symbols are removed. Then, it is converted into a string with no leading and trailing spaces and given to the constituency parser.

Step 2: Constituency Parse Tree The preprocessed string is given into the constituency parse tree. Stanza, a Stanford Core NLP package for Python, has been used. Pipelines that are used in

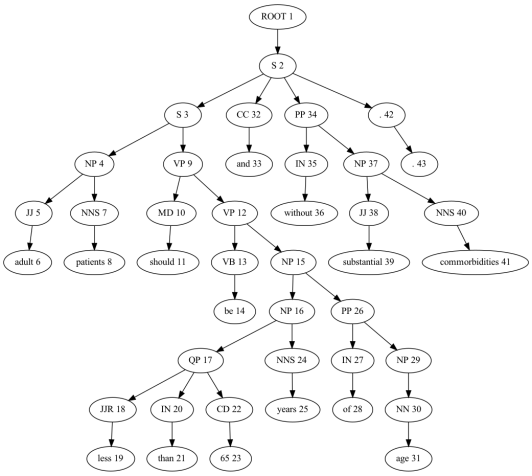


Figure 4: Constituency Parse Tree Output

the stanza are MWT (multiword tokens) followed by POS (parts of speech tagging), and then finally constituency parser. The output of the constituency parse tree is a tree-based structure that is represented as a string in Python. Sample output is: *(ROOT (S (S (NP (JJ adult) (NNS patients)) (VP (MD should) (VP (VB be) (NP (NP (QP (JJR less) (IN than) (CD 65)) (NNS years)) (PP (IN of) (NP (NN age))))))) (CC and) (PP (IN without) (NP (JJ substantial) (NNS comorbidities))) (. .)))*. Visually, it is easy to parse the above structure but since, there are no available parsers for the above output, a parser that parses the above structure and generates a tree has been created. The structure generated by the parser created is then given to networkx package in Python which generates an output, shown in Figure 4. Node numbers have been added at the end to differentiate different nodes with the same content.

Step 3: Merging the Trees The tree structure which is generated using the constituency parse tree is merged recursively. The base case is a list of all 'and' nodes that are taken and sorted according to their depth. The node with maximum depth is picked up and merged across the left side and right side of it and the node which is above the current node is added to the list. This recursively applies till all the nodes have been visited. This step decreases the ambiguity of sentences. Suppose we were given "without comorbidities of diabetes and tuberculosis". Here these merging steps make it into "without comorbidities of diabetes and without comorbidities of tuberculosis". It does the linking of entities that are close to each other.

Step 4: Named Entity Extraction The merged tree's output sentences are taken and the named entity is extracted using the Stanford Core NLP package. These named entities are marked as non-removable which makes further steps not to remove those entities picked up.

Step 5: Filtering The output of merged trees is taken and all the words which are not marked as "non-removable" are removed. Here, non-removable words are the words that are extracted from named entity extraction and words that contribute to making sentences into logical form. These are identified using a set of keywords taken from LexNLP.

Step 6: Substitutions The words which are having logical form are replaced by their mathematical representation like 'less than' is replaced with the symbol '$<$'.

### 2.6.2 Result of Constraint Extractor

The following are the input examples and outputs for those input examples from the Constraint Extractor module:

1. Input to the model is "adult patients should be less than 65 years of age and without substantial comorbidities". In the above sentence, we had 2 constraints which are adult patient's age should be less than 65 years and the other is he should not have any substantial comorbidities. We can see the output in Figure 5 is ['no substantial comorbidities', 'adult patients $<$ 65 years of age'].

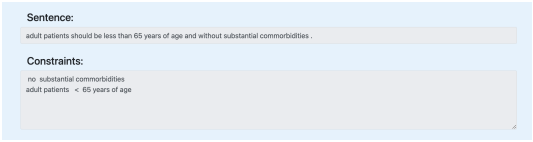


Figure 5: Result 1

2. Input to the model is that "AYA and adult patients should be less than 65 years of age and without substantial comorbidities". In the above sentence, we have 3 constraints which are AYA, age should be less than 65 years he should not have any substantial comorbidities. We can see the output in Figure 6 is ['no substantial comorbidities', 'adult patients $<$ 65 years of age', 'AYA $<$ 65 years of age'].

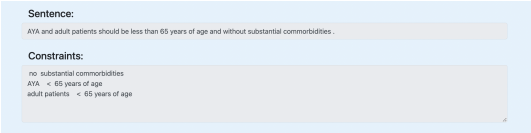


Figure 6: Result 2

3. Input to the model is “tumor size should be greater than 0.5cm and pn0". In the above sentence, we have 1 constraint which is the size of the tumor should be greater than 0.5cm. We can see the output in Figure 7 is [‘tumour size $>$ 0.5 cm’].

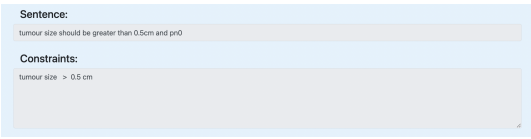


Figure 7: Result 3

4. Input to the model is“without comorbidities of diabetes and hypertension". In the above sentence, we had 2 constraints which are he should not have diabetes and he should not have hypertension. We can see the output in Figure 8 is [‘hypertension comorbidities no’, ‘diabetes comorbidities no].

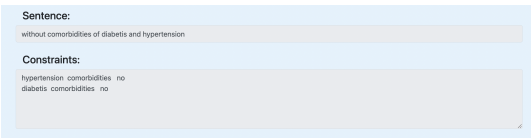


Figure 8: Result 4

5. Input to the model is “tumor size should be less than 5cm and margins greater than 1mm". In the above sentence, we have 2 constraints which are the size of the tumor should be less than 5cm and margins should be less than 1mm. We can see the output in Figure 7 is [‘margins $>$ 1mm’,‘tumour size $<$ 5 cm’].

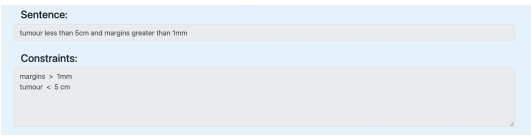


Figure 9: Result 5

### 2.7 Decision Knowledge Graphs

A Knowledge Graph (KG) is used to store factual data. KG consists of many relations and nodes arranged in the form of a triple i.e., *(head node, relation, and tail node)*. Data like Clinical Practice Guidelines have conditional data along with factual data which makes KG not usable for storing this type of data. So, in this work, we have added a new dimension in KG known as the decision dimension used to store conditional data. This new KG structure has additional nodes carrying conditional data added to the existing KG structure.

### 2.8 Neo4j Graph Database Manager

Neo4j is a powerful and widely adopted graph database management system that has revolutionized the way data is stored, managed, and queried. Its unique graph-based approach enables the representation and exploration of complex relationships and interconnected data. Neo4j operates on the principles of graph theory, where data is organized as nodes, relationships, and properties. Nodes represent entities, relationships depict connections between entities, and properties store additional information about nodes and relationships. This intuitive representation allows for efficient modeling of real-world scenarios, enabling a natural exploration of complex relationships. Some of the key features of neo4j are:

- Relationship-centric: Neo4j focuses on relationships as first-class citizens, providing a native and performant way to traverse and query connections within large datasets.
- High Performance: Neo4j’s graph-based data model allows for efficient and fast traversal of relationships, making it well-suited for applications that require real-time insights and rapid query execution.
- Scalability: Neo4j’s architecture is designed to handle large-scale datasets and high-throughput workloads. It supports horizontal scaling, allowing organizations to scale their graph databases as their data grows.
- Flexibility: Neo4j offers flexible schema models, allowing dynamic changes to the data model without requiring a rigid predefined schema. This flexibility makes it suitable for evolving and dynamic data domains.

### 2.9 Cypher Query Language (CQL)

Cypher is a powerful and expressive query language specifically designed for graph databases.

Developed by Neo4j, Cypher provides a simple and intuitive syntax to interact with graph data, enabling developers and analysts to harness the full potential of graph database capabilities. At the core of Cypher lies its graph-pattern matching capability, allowing users to describe patterns and relationships within the graph data. Cypher employs ASCII art-style syntax, using parentheses, arrows, and other symbols to define nodes, relationships, and properties. This intuitive approach simplifies the querying process and facilitates the exploration of complex connections. Some of the key features of CQL are:

- Pattern Matching: Cypher enables the traversal of nodes and relationships in a graph database by specifying patterns to match. It supports various patterns, including node labels, relationship types, properties, and filtering conditions, allowing users to extract precise subsets of data.
- Path Expressions: Cypher offers path expressions, which enable the traversal of multiple nodes and relationships in a single query. This feature facilitates the discovery of connected paths and enables deep analysis of relationships within the graph.
- Aggregation and Analysis: Cypher provides built-in functions and operators for aggregation, filtering, sorting, and statistical analysis. These capabilities allow users to perform calculations, derive insights, and generate reports directly within the query language.
- Read and Write Operations: Cypher supports both read and write operations, allowing users to not only query the graph but also update, insert, and delete data. This versatility makes it a comprehensive language for managing and manipulating graph database

Some of the basic examples of CQL used are:

1. To retrieve all nodes of a specific label:

```
MATCH (n:Label)
RETURN n;
```

This query retrieves all nodes with the label "Label" from the graph database.

2. To find nodes based on property values:

```
MATCH (n:Label)
WHERE n.property = 'value'
RETURN n;
```

This query retrieves nodes with the label "Label" that have a specific property value.

3. To traverse relationships between nodes:

```
MATCH (a)-[:RELATIONSHIP]->(b)
RETURN a, b;
```

This query traverses relationships of type "RELATIONSHIP" from node "a" to node "b" and returns both nodes.

4. To filter and order results:

```
MATCH (n:Label)
WHERE n.property > 10
RETURN n
ORDER BY n.property
DESC;
```

This query retrieves nodes with the label "Label" that have a property value greater than 10, and orders the results in descending order based on that property.

5. To perform aggregations:

```
MATCH (n:Label)
RETURN count(n)
AS count;
```

This query calculates the count of nodes with the label "Label" and returns it as "count".

6. To create new nodes and relationships:

```
CREATE (a:Label {property: 'value
    '})-[:RELATIONSHIP]->(b:Label)
    ;
```

This query creates a new node with the label "Label" and a specific property, and creates a relationship of type "RELATIONSHIP" between the newly created node and another node with the label "Label".

### 2.10 Question Answering Systems

Question-answering (QA) systems are designed to provide human-like responses to user queries by automatically extracting relevant information from various data sources. These systems utilize natural language processing (NLP) techniques to understand and interpret user questions, search for relevant information, and generate accurate and concise answers.

There are two main types of QA systems:

1. Extractive QA: These systems identify and extract the most relevant answer from a given text source, such as documents or web pages.

2. Generative QA: These systems generate answers by understanding the question and generating a response based on the available knowledge.

The development of QA systems began with early AI research, focusing on rule-based approaches and symbolic reasoning. Weizenbaum (1966), developed ELIZA in the 1960s, was one of the earliest chatbot-like systems that could engage in text-based conversations. Later QA systems in the 1990s primarily relied on information retrieval techniques. Systems like START referred in Shinde et al., developed at MIT, used pre-defined patterns and templates to extract answers from textual sources. TREC (Text REtrieval Conference) QA track was initiated to evaluate and benchmark QA systems' performance. The focus shifted to passage retrieval, where systems aimed to find relevant answer passages from large collections of documents. Systems like IBM Watson referred in High (2012) and the development of the Jeopardy!-playing AI showcased advancements in question answering. Deep learning and neural network models revolutionized QA systems. The introduction of large-scale pre-trained language models, such as BERT and GPT, significantly improved QA performance. Transformer-based models, like GPT by OpenAI (2023), achieved state-of-the-art results in both extractive and generative QA tasks. Now with the rise of knowledge graphs and structured data, QA systems have incorporated graph-based approaches to enhance question answering.

### 2.11 Transformer Model

This work uses transformer model for 2 applications. The Transformer model has revolutionized various natural language processing (NLP) tasks by introducing a novel architecture that relies solely on attention mechanisms, eliminating the need for recurrent or convolutional structures. This summary provides an overview of the Transformer model, highlighting its key components and contributions. The Transformer model, introduced by Vaswani et al. (2017) in 2017, was primarily designed for machine translation tasks. It has since become the backbone of many state-of-the-art models in NLP. The model's success lies in its ability to capture long-range dependencies and effectively handle sequential data without sequential computation.

The key components of the Transformer model are self-attention and feed-forward neural networks. Self-attention allows the model to weigh the importance of different words in a sequence, enabling it to consider both local and global contexts simultaneously. This mechanism replaces recurrent connections, making it more parallelizable and efficient for training on modern hardware. The feed-forward neural networks process the information learned from the self-attention mechanism, providing non-linear transformations to the input. The Transformer model introduces the concept of attention heads, which allow the model to attend to different positions or aspects of the input simultaneously. Multiple attention heads capture diverse information and facilitate better representation learning. Moreover, the model incorporates positional encodings to retain the order of the input sequence, as self-attention is permutation invariant.

To train the Transformer model, Vaswani et al. (2017) introduced the concept of "scaled dot-product attention," which enables efficient computation of self-attention. The model is trained using a variant of the attention mechanism called "multi-head attention," which allows for parallelization and enhances performance. The Transformer model has achieved remarkable results across various NLP tasks, including machine translation, text summarization, sentiment analysis, and question answering. Its ability to capture context dependencies and process input in parallel has contributed to its success. The Transformer model has also inspired subsequent advancements in NLP, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), which have further pushed the boundaries of language understanding and generation.

### 2.12 Evaluation Metrics

We briefly describe the metrics used in the evaluation.

#### 2.12.1 ROUGE Score

The quality of text summarization or machine translation output is assessed using a set of measures called ROUGE (Recall-Oriented Understudy for Gisting Evaluation). Comparing the generated text to the reference text forms the basis for the measurements. Precision, recall, and F1-score are used to construct ROUGE scores. The following is the ROUGE formula:

ROUGE-N:

$$Precision = \frac{overlapping\ ngrams}{total\ ngrams}$$

$$Recall = \frac{number\ of\ overlapping\ ngrams}{number\ of\ ngrams\ in\ reference\ summary}$$

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

The metrics reported in the paper are ROUGE-1 score. The score is calculated using the package rouge_score.

#### 2.12.2 BLEU Score

BLEU (Bilingual Evaluation Understudy) is used to assess the effectiveness by comparison of the generated text and the reference text forms the basis of it.

The nltk.translate.bleu_score module in the NLTK package offers tools for computing BLEU scores. To compare a single generated sentence to a reference sentence and determine the BLEU score, use the sentence_bleu() function. The sentence_bleu() function allows you to specify the n-gram order (default is 4) and a set of weights to assign to each n-gram order. The weights are used to compute the final BLEU score, and they can be specified using the weights parameter. The weights parameter should be a tuple of floats that sum up to 1, where each float corresponds to the weight assigned to the n-gram order.

In this paper we have used sentence_bleu with equal weigthage to all ngrams.

#### 2.12.3 Jaccard Similarity Score

A measure of similarity between two sets of data is the Jaccard similarity score, commonly referred to as the Jaccard index or Jaccard coefficient. It is calculated by dividing the size of the intersection by the sum of the two sets. The following is the Jaccard similarity score formula:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

A and B are two sets, and the symbols for their intersection and union are and, respectively. The symbols |A| and |B| stand for the size or cardinality of the sets A and B, respectively.

The Jaccard similarity score is frequently used in text analysis to assess how similar two texts or text strings are to one another. The sets A and B can be defined as the set of words or tokens in the two documents, and the Jaccard similarity score can be used to measure the overlap between the sets of words.

#### 2.12.4 Accuracy

Accuracy is used to check the correctness of the generated model. We calculated accuracy with the formulae: $Accuracy = \frac{total\ correct\ predictions}{total\ predictions}$

For baseline model we have used if BLEU Score greater than 0.7 as correct statement. For with DKG model we have used 1 as correct statement.

## 3 Literature Survey

### 3.1 CPGs and their ignorance

CPGs are written based on evidence, aiming to improve the quality and efficiency of medical treatment and care. They are useful to a doctor in providing proper insights when he/she is treating a patient. Many physicians don't use CPGs. Cabana et al. (1999) claims that the main reasons for not using CPGs are their complexity, unfamiliarity, and distrust. Trust can be improved once CPGs start gaining positive attention and lead to successful treatment of patients. Complexity and familiarity need to be addressed for the usage of CPGs. CPGs were introduced in the early 90s yet their familiarity is still a problem in the medical domain.

The existing representations of CPGs are complex and unfamiliar as mentioned in Cabana et al. (1999). Manually searching data in CPGs takes time. During emergencies, time is valuable and lack of time can cost lives. A representation for CPGs on which question-answering and searching can be performed will help a lot in emergencies. This representation can also motivate practitioners and doctors to use guidelines. So far, no attempt has been made for representing CPGs to perform question-answering and searching tasks.

Given the structured nature, and factual data present in the CPGs, it is reasonable to organize this information as a Knowledge Graph. Rossetto et al. (2020) describes Knowledge Graph (KG) as static graph triples. If the data is static, KG, once constructed, needs no modifications and can be used to perform question-answering and searching tasks. Once the KG is constructed, modifying the KG is costly and takes time as modification involves updating, changing, or deleting multiple nodes and relations which can propagate. Therefore, at times, KG needs to be reconstructed because of some modifications.

### 3.2 Existing KG models in Medical Domain

Construction of a KG involves many steps like co-reference resolution, information extraction, etc. Rossanez et al. (2020) provides a detailed pipeline of KG construction for biomedical scientific literature. Many existing approaches to constructing KG

ignore the conditional statements that are present in the sentences. Jiang et al. (2019) explains how existing ScienceIE models capture factual data and will not consider conditional statements. i.e., An existing system would return the tuple (alkaline pH, increases, activity of TRPV5/V6 channels in Jurkat T cells) if the statement "alkaline pH increases the activity of TRPV5/V6 channels in Jurkat T cells" was given. However, in this case the condition tuple (TRPV5/V6 channels, in, Jurkat T cells) was not identified.

Jiang et al. (2020) emphasizes the importance of conditional statements in biomedical data. They also propose a KG representation with conditional statements. The conditional statements are added to the existing KG structure but this structure is not suitable for clinical practice guidelines because updating is not efficient in the current KG structure.

From the survey conducted by Liang et al. (2022), many KG question-answering models were relying on rules, keywords, neural networks, etc. After the introduction of SPARQL by Hu et al. (2021), which is a query language to search and modify a KG, retrieving data from KG became easy. Therefore, many question-answering models were proposed using KG.

## 4 Dataset and Annotation

### 4.1 MIMIC-III Dataset

The MIMIC-III (Medical Information Mart for Intensive Care III) dataset is a widely used and publicly available healthcare dataset that provides a comprehensive collection of de-identified clinical data from intensive care units (ICUs). It includes data from over 60,000 patients who were admitted to the Beth Israel Deaconess Medical Center between 2001 and 2012. The dataset contains a wealth of information, including demographics, vital signs, laboratory measurements, medications, procedures, diagnoses, and clinical notes.

Here are a few examples of the types of data available in the MIMIC-III dataset:

1. Demographics: - Example: Age, gender, ethnicity, and admission and discharge dates of patients. - Usage: Demographic information can be used to study population characteristics and analyze patient outcomes based on demographic factors.

2. Vital Signs and Measurements: - Example: Heart rate, blood pressure, temperature, respiratory rate, and oxygen saturation. - Usage: Vital signs and measurements help monitor a patient's physiological condition and can be analyzed to identify patterns, detect abnormalities, and predict patient outcomes.

3. Laboratory Measurements: - Example: Blood tests, such as complete blood count (CBC), electrolyte levels, glucose, and arterial blood gas (ABG) measurements. - Usage: Laboratory measurements provide valuable insights into a patient's biochemical profile, organ function, and disease progression.

4. Medications: - Example: Administration records of medications, including dosage, route, and frequency. - Usage: Medication data can be utilized to study drug usage patterns, analyze medication interactions, and assess the impact of medications on patient outcomes.

5. Procedures: - Example: Surgical procedures, diagnostic tests, and interventions performed during a patient's ICU stay. - Usage: Procedure data enables the analysis of treatment approaches, outcomes, and the effectiveness of different interventions.

6. Diagnoses: - Example: International Classification of Diseases (ICD) codes representing patient diagnoses. - Usage: Diagnoses data helps in understanding disease prevalence, comorbidities, and the impact of specific conditions on patient outcomes.

7. Clinical Notes: - Example: Free-text notes written by healthcare professionals, including progress notes, discharge summaries, and radiology reports. - Usage: Clinical notes provide valuable narrative information about a patient's medical history, treatment plans, and healthcare provider observations.

The MIMIC-III dataset has been extensively utilized in research and healthcare analytics to explore various topics, including predictive modeling, clinical decision support systems, disease surveillance, treatment effectiveness, and patient outcome prediction. Its availability has contributed significantly to advancing the understanding of critical care and facilitating data-driven research in the healthcare domain.

### 4.2 CPG-QA Dataset

The main objective of a Decision Knowledge Graph (DKG) is to perform question-answering thus reducing the manual effort of a doctor to search through the guidelines. There are no available question-answering datasets on Clinical Prac-

tice Guidelines.

We have created a CPG-QA dataset with 4000 question-answer pairs. This dataset consists of three main types of questions.

Types of questions:

1. **What is next treatment advice given a patient's constraints (refer to Section 2 for more details on constraints).**
   *Example: A patient is ALL positive. After his initial diagnosis he is classified as ph- patient. His age is 65. He is not treated with other cancer treatments. What treatment is recommended in this condition?*

2. **What are the patient's medical constraints that needs to be satisfied given a treatment stage.**
   *Ex: A patient is ALL positive. After his initial diagnosis he is classified as ph+ patient. What are patient constraints for doing chemotherapy?*

3. **Given a patient's medical constraints and treatment stage, whether a particular treatment is advisable or not?**
   *Ex: A patient is ALL positive. After his initial diagnosis he is classified as ph- patient. His age is 65. He is not diagnosed with any other cancer treatment. Can we perform TKI + Chemotherapy on him?*

Annotation The dataset also consists of cypher queries for question-answering pairs which are used to query the DKG. These cypher queries are manually constructed given a question. We have verified the correctness of the queries by running them on DKG and matching the outputs of DKG with the expected answer. The format of the dataset is:

```
[
    {
      "QUESTION": String,
      "ANSWER": String,
      "QUERY": String,
      "Expected_Node": Integer,
      "DKG_response": Integer,
    },...
]
```

Dataset Examples Here are few examples from the dataset

```
1.
{
      "QUESTION": "Upon risk
         stratification, a
         patient is identified to
          have ph- ALL at the age
          of 37. What treatment
         measures are advised?",
      "ANSWER": "clinical trial
         or Pediatric-inspired
         regimes or Multiagent
         chemotherapy(systematic
         therapy)",
      "REMARK": "pediatric-
         inspired regimes is
         preferred more",
      "QUERY": "MATCH (n:
         risk_stratification)
         WHERE n.stratified = 'ph
         -' and n.age_cat='AYA' -
         [:next_step]->k RETURN k
         .treatment",
      "Expected_Node": 14,
      "DKG_response": 14
}

2.
{
      "QUESTION": "A ph- ALL
         patient's response
         assessment is CR. His
         age is 37. He was
         monitored for MRD and
         found negative. What are
          the recommended
         procedures?",
      "ANSWER": "Allogenic HCT (
         especially if high-risk
         features or consider
         continuing multiagent
         chemotherapy or
         Blinatumomab",
      "QUERY": "MATCH (m:
         decision_node{
         stratified='ph-',
         age_cat='AYA', MRD:'
         absent'})-[:next_step]->
          n RETURN n.treatments",
      "Expected_Node": 17,
      "DKG_response": 17
}
```

Dataset can be downloaded from github[1].

## 5 Decision Knowledge Graphs

In the Knowledge Graph (KG), data is stored as triples consisting of a head entity, a relation, and a tail entity i.e., *(head, relation, tail)*. If there is some change in the KG (i.e., updating triple, deleting triple, or adding new triple), these changes, in the worst case, can propagate to all nodes. Consider the example of *triple (Barack Obama, president of , US), if we want to update Obama to Trump, then the update should be done in multiple nodes which talk about US presidency or about the individuals*. Therefore sometimes, updating a KG will become equivalent to rebuilding the KG. The update operation, therefore, is time-consuming. Clinical Practice Guidelines (CPGs) are updated frequently. Hence, KG structure won't be of much help for CPGs as it would require the costly update operation frequently.

From the previous few versions of guidelines, we have observed that not all content in the guidelines is changed. The modifications that are made to guidelines, based on discussions, are mainly done on patients' constraints (refer to Section 2 for definition). The treatment steps of chemotherapy are not changed but when to perform chemotherapy based on the patient's condition is changed. Therefore, using this observation, we divide the data into static and dynamic data.

Static data is the data in CPGs that changes less frequently or doesn't change at all. Dynamic data is the data in CPGs which changes frequently. Here, dynamic data doesn't refer to data from a query like the name of the patient, etc. It refers to the data that should be present in the KG to make a decision. For example, treatment procedure like chemotherapy is static data and patients' constraints like *age>60, MRD rising,* etc., is dynamic data.

DKG is a knowledge graph over which we have introduced a decision layer. This decision dimension will consist of dynamic data. Static data is stored as KG triples extracted as proposed by (Rossanez et al., 2020). For example, if there is a node, "chemotherapy", we have relations like "procedure", "drugs used", "duration" etc., which comes under static data. When updating a KG, only dynamic data needs to be changed without changing the structure of the KG and static data. Therefore, performing updates on DKG will be a more cost-effective task than updating a KG. Here the static data is stored as a KG. For example, if there is a node, "chemotherapy", we have relations like "duration", "drugs used" etc. Therefore, factual data is stored as we do in a KG, but conditional data is stored in decision nodes.

[1]https://github.com/Vasu8081/CPG-QA

## 6 Conclusion and Future work

In conclusion, representing clinical practice guidelines (CPGs) digitally is challenging. The proposed novel structure, Decision Knowledge Graph (DKG) can effectively store CPGs. DKG enables the encoding of decision-based structures, which are often changed in CPGs, in addition to factual data. Our work makes a significant addition to the field of representing medical knowledge and can help practitioners and doctors to make well-informed judgments about patient's treatment. Our work also contributes to the NLP community by providing a representation for storage of knowledge which has decision-based structure. The model is intended to be used by professional practitioners and doctors only and for recommendation purpose, not to solely depend on the models recommended treatment.

There are many areas that can be explored in this project, major future goals are listed below: The DKG architecture can be expanded to clinical practice guidelines other than NCCN by building a constraint extractor for the particular guidelines. It can also be expanded to other domains like construction guidelines in Civil engineering, etc.

## References


Fast facts on u.s. hospitals, 2022. http://https://www.aha.org/statistics/fast-facts-us-hospitals. Accessed: 2023-02-15.

Michael D. Cabana, Cynthia S. Rand, Neil R. Powe, Albert W. Wu, Modena H. Wilson, Paul-André C. Abboud, and Haya R. Rubin. 1999. Why Don't Physicians Follow Clinical Practice Guidelines?A Framework for Improvement. *JAMA*, 282(15):1458–1465.

Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. 2020. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5).

Rob High. 2012. The era of cognitive systems: An inside look at ibm watson and how it works. *IBM Corporation, Redbooks*, 1:16.

Xin Hu, Jiangli Duan, and Depeng Dang. 2021. Natural language question answering over knowledge graph: the marriage of sparql query and keyword search. *Knowledge and Information Systems*, 63(4):819–844.

Tianwen Jiang, Qingkai Zeng, Tong Zhao, Bing Qin, Ting Liu, Nitesh V Chawla, and Meng Jiang. 2020. Biomedical knowledge graphs construction from conditional statements. *IEEE/ACM transactions on computational biology and bioinformatics*, 18(3):823–835.

Tianwen Jiang, Tong Zhao, Bing Qin, Ting Liu, Nitesh Chawla, and Meng Jiang. 2019. Multi-input multi-output sequence labeling for joint extraction of fact and condition tuples from scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, and Fuchun Sun. 2022. Reasoning over different types of knowledge graphs: Static, temporal and multi-modal. *arXiv preprint arXiv:2212.05767*.

OpenAI. 2023. Gpt-4 technical report.

Anderson Rossanez, Julio Cesar Dos Reis, Ricardo da Silva Torres, and Hélène de Ribaupierre. 2020. Kgen: a knowledge graph generator from biomedical scientific literature. *BMC medical informatics and decision making*, 20(4):1–24.

Luca Rossetto, Matthias Baumgartner, Narges Ashena, Florian Ruosch, Romana Pernischová, and Abraham Bernstein. 2020. Lifegraph: a knowledge graph for lifelogs. In *Proceedings of the Third Annual Workshop on Lifelog Search Challenge*, pages 13–17.

Kalyani Shinde, Anjika Singh, Reshma Shitole, Pallavi Singh, and Sumit Harale. A survey on question answer system.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.