Lecture 25: Question Answering (QnA) Systems: Basics, Methods, and Rule-based/Pattern Matching

1. QnA System Basics

A **Question Answering (QnA) system** is a form of information retrieval that aims to provide direct, precise answers to natural language questions posed by users, rather than simply returning a list of relevant documents or web pages. While traditional search engines (Information Retrieval - IR) find documents containing keywords from a query, a QnA system attempts to understand the user's intent and extract or synthesize a concise answer from a vast knowledge base.

Key Goal: To bridge the gap between a user's natural language question and the information available in a structured or unstructured data source, providing a definitive answer.

Difference from Information Retrieval (IR):

- **IR:** Returns a list of *documents* that might contain the answer. Focuses on document relevance.
- **QnA:** Returns the *specific answer* itself. Focuses on answer extraction/generation.

2. General QnA Methods / Workflow

Most QnA systems, regardless of their underlying approach, typically follow a generalized multi-stage pipeline:

1. **Question Analysis:**
    - **Parsing:** Breaking down the question into its grammatical components (e.g., noun phrases, verb phrases).
    - **Semantic Role Labeling:** Identifying the roles of words in the sentence (e.g., who did what to whom).
    - **Question Type Classification:** Determining the type of question (e.g., "Who," "When," "Where," "How many," "Definition," "List," "Factoid"). This is crucial for guiding the search for an answer.
    - **Named Entity Recognition (NER):** Identifying entities like persons, organizations, locations, dates, etc., in the question.
    - **Keyword Extraction:** Identifying important terms to search for.
2. **Document/Passage Retrieval:**
    - Using keywords and potentially refined queries from the question analysis, the system searches its knowledge base (e.g., a corpus of documents, a database,

    the web) to find documents or passages that are likely to contain the answer. This stage often uses standard IR techniques (e.g., TF-IDF, BM25).

3. **Answer Extraction / Generation:**
    - This is the core QnA stage. From the retrieved documents/passages, the system attempts to pinpoint the exact answer.
    - **Extraction:** If the answer exists directly in the text (e.g., a short phrase), the system extracts it.
    - **Generation:** For more complex questions, the system might need to synthesize an answer from multiple pieces of information or rephrase information into a coherent response.
4. **Answer Ranking & Presentation:**
    - If multiple potential answers are found, they are ranked based on confidence scores.
    - The highest-ranked answer(s) are presented to the user. This might include supporting evidence (e.g., the sentence from which the answer was extracted).

3. Types of QnA Systems

QnA systems can be broadly categorized based on their underlying methodology and the nature of their knowledge source:

- **Based on Methodology:**
    - **Rule-based / Pattern Matching:** (Discussed in detail below)
    - **Statistical / Machine Learning-based:** Uses statistical models (e.g., n-grams, SVMs, CRFs) trained on large datasets to learn patterns for question analysis, passage retrieval, and answer extraction.
    - **Information Extraction (IE) based:** Relies on structured information extracted from text into templates or knowledge graphs.
    - **Deep Learning / Neural Network-based (Modern Approach):** Employs sophisticated neural architectures (e.g., LSTMs, Transformers like BERT, GPT) for end-to-end question understanding, passage ranking, and answer generation. These excel at semantic understanding.
- **Based on Knowledge Source:**
    - **Closed-Domain QnA:** Operates within a specific, limited domain (e.g., medical QnA, technical support QnA). Can achieve high accuracy due to specialized knowledge.
    - **Open-Domain QnA:** Can answer questions on almost any topic, drawing from a vast and diverse knowledge base (e.g., the entire web, Wikipedia). More challenging to build.

    - **Knowledge Graph (KG) based:** Answers questions by querying a structured graph of entities and their relationships.
    - **Text-based:** Answers questions by searching and extracting information directly from unstructured text documents.

4. Rule-based / Pattern Matching QnA Systems

This was one of the earliest and foundational approaches to QnA. It relies on handcrafted linguistic rules, patterns, and templates to analyze questions and identify answers.

Core Idea: The system contains a predefined set of rules that map specific question patterns to expected answer patterns. When a question is posed, it's matched against these known patterns, and if a match is found, the system looks for an answer that fits the corresponding answer pattern in the text.

How Rules/Patterns are Created:

1. **Linguistic Analysis:** Human experts (linguists) analyze common question structures (e.g., "Who is X?", "When did Y happen?") and the typical structures of sentences that contain answers to such questions.
2. **Template Definition:** These analyses are then formalized into templates or regular expressions.
    - **Question Templates:** Define patterns for different types of questions.
        - Example: `WH-PERSON is <NP>?` (e.g., "Who is the CEO of Google?")
        - Example: `WHEN did <NP> <VP>?` (e.g., "When did the Berlin Wall fall?")
    - **Answer Templates/Extraction Patterns:** Define patterns for how answers are likely to appear in text based on the question type.
        - Example for `WH-PERSON is <NP>?`: Look for sentences where `NP` is followed by a verb like 'is' or 'was' and then a `PERSON` named entity.
            - Pattern: `<NP> (is|was) <PERSON_ENTITY>`
            - From: "Sundar Pichai is the CEO of Google." rightarrow match "Sundar Pichai"
3. **Entity Recognition:** Rules often incorporate the identification of specific Named Entity Types (e.g., PERSON, ORGANIZATION, LOCATION, DATE, MONEY) to constrain searches.

Example of Rule-based/Pattern Matching Workflow:

Let's use the example: **"Who is the CEO of Google?"**

1. **Question Analysis (using Rules):**

    - Rule: `WH-PERSON is <NP_QUESTION>?`
    - Match: "Who" as WH-PERSON, "NP_QUESTION>" as "the CEO of Google".

    - Inferred Answer Type: `PERSON`
2. **Document/Passage Retrieval:**
    - Keywords: "CEO", "Google"
    - Retrieves documents containing these keywords. Suppose it finds a passage: "Sundar Pichai was appointed CEO of Google in 2015."
3. **Answer Extraction (using Rules):**
    - Based on the inferred answer type (`PERSON`) and the question pattern, the system looks for an answer pattern that matches the question's structure and expected answer type.
    - Rule: Look for a `PERSON_ENTITY` immediately following an NP or a linking verb like "is/was" related to the NP from the question.
    - Matching Pattern: `<NOUN_PHRASE> (is|was|appointed) <PERSON_ENTITY> (of|at) <ORGANIZATION_ENTITY>`
    - Applying to passage: "Sundar Pichai was appointed CEO of Google in 2015."
        - Matches "Sundar Pichai" as `PERSON_ENTITY`.
        - Matches "Google" as `ORGANIZATION_ENTITY`.
    - Extracted Answer Candidate: "Sundar Pichai"
4. **Answer Ranking & Presentation:**
    - If multiple passages yielded candidates, they would be ranked.
    - The system presents: "Sundar Pichai"

Pros of Rule-based/Pattern Matching Systems:

- **High Precision for Covered Cases:** When a question perfectly matches a defined pattern and the answer exists in the expected format, these systems can provide highly accurate and precise answers.
- **Explainable:** The logic is transparent; you can trace why a particular answer was given based on the rules.
- **Good for Narrow Domains:** In very specific domains with predictable question and answer structures, they can be highly effective.

Cons of Rule-based/Pattern Matching Systems:

- **Scalability Issues (Knowledge Acquisition Bottleneck):** Handcrafting rules for all possible question variations and answer patterns is extremely time-consuming, labor-intensive, and requires deep linguistic expertise. It's almost impossible to cover all real-world language variations.
- **Brittleness:** They are very sensitive to variations in wording or sentence structure. A slight deviation from a defined pattern can cause the system to fail completely. They lack robustness to unseen linguistic phenomena.

- **Poor Generalization:** They do not generalize well to new types of questions or unseen text.
- **Limited Semantic Understanding:** They operate primarily on syntactic and lexical patterns, with very little genuine understanding of the underlying meaning or context.
- **Maintenance Overhead:** As language evolves or the knowledge base changes, rules need constant updating and refinement.

Due to these significant limitations, particularly the scalability and brittleness issues, pure rule-based/pattern matching systems have largely been superseded by statistical, machine learning, and deep learning approaches in modern open-domain QnA. However, pattern matching still plays a supporting role in some hybrid systems or for very specific, tightly controlled tasks.

What is a Knowledge Graph (KG)?

Imagine a vast network where:

- **Nodes (or Entities) represent real-world "things" or concepts. These could be people (e.g., "Albert Einstein"), places (e.g., "Paris"), organizations (e.g., "Google"), events (e.g., "World War II"), or abstract concepts (e.g., "Physics").**
- **Edges (or Relationships/Predicates) connect these nodes, describing the relationships between them. For example:**
  - **"Albert Einstein" --(was born in)--> "Ulm"**
  - **"Google" --(has CEO)--> "Sundar Pichai"**
  - **"Paris" --(is capital of)--> "France"**
  - **"World War II" --(started in)--> "1939"**

So, a Knowledge Graph is a structured, interconnected representation of facts about the world. It organizes information as a graph of entities and the relationships between them, making it easy for machines to understand and query complex, interconnected data.

Think of it like a highly structured, machine-readable encyclopedia or a vast semantic network. Google's famous "Knowledge Graph" that powers the info boxes you see in search results is a prime example.

How do Knowledge Graphs work in QnA Systems?

Traditional QnA systems often rely on searching through unstructured text (like articles, books, web pages). When you ask a question, they try to find sentences or paragraphs that contain the answer.

KG-based QnA systems take a different approach. Instead of just looking for keywords in text, they try to convert your natural language question into a query that can be executed against the structured Knowledge Graph.

Here's the general workflow:

1. **Question Understanding / Semantic Parsing:**
    - **The user asks a question in natural language (e.g., "Who directed 'Inception'?").**
    - **The QnA system's first step is to "understand" this question by parsing it into its core components and mapping them to entities and relationships in the Knowledge Graph. This is often called semantic parsing.**
    - **It identifies:**
        - **Entities: "'Inception'" (a movie)**
        - **Relationship (Implicit): "directed by" (looking for the director)**
        - **Answer Type: `PERSON` (a director is a person)**
2. **Query Generation:**
    - **Once the entities and relationships are identified, the system translates them into a formal query language that the Knowledge Graph database can understand. Common query languages for KGs include SPARQL or Cypher.**
    - **For "Who directed 'Inception'?", the system might generate a query like:**
        - **`MATCH (movie:Movie {name: 'Inception'})-[r:DIRECTED_BY]->(person:Person) RETURN person.name`**
        - ***(This translates to: "Find a node (movie) named 'Inception', follow the relationship 'DIRECTED_BY' from it to another node (person), and return that person's name.")***
3. **Graph Traversal / Query Execution:**
    - **The generated query is executed directly on the Knowledge Graph.**
    - **The KG database efficiently traverses its network of nodes and edges to find the matching pattern.**
    - **It might find: "Inception" --(DIRECTED_BY)--> "Christopher Nolan".**
4. **Answer Formulation:**
    - **The result of the graph query (e.g., "Christopher Nolan") is directly the answer.**
    - **The system then presents this answer to the user.**

Example Scenario:

- **User Question: "What is the capital of France?"**
- **KG Analysis:**
    - **Identify entity: "France" (a Country)**
    - **Identify relationship: "capital of" (a relationship linking Country to City)**
    - **Expected answer type: `CITY`**

- **KG Query (Conceptual): Find the `CITY` that is the `capital of France`.**
- **KG Data:**
    - **(France) --(capital_of)--> (Paris)**
- **Answer: "Paris"**

Advantages of Knowledge Graph (KG) based QnA:

- **Precision and Accuracy: KGs contain factual, structured data, leading to highly precise and accurate answers for factual questions. There's less ambiguity compared to extracting answers from unstructured text.**
- **Reasoning and Multi-Hop Questions: KGs excel at answering complex questions that require "multi-hop" reasoning (following multiple relationships).**
    - **Example: "Which actors starred in movies directed by Christopher Nolan?"**
        - **KG can first find all movies directed by "Christopher Nolan".**
        - **Then, for each of those movies, find all the actors who starred in them.**
- **Explainability: Because the answer is derived from explicit relationships in the graph, it's often easier to explain *how* the system arrived at the answer.**
- **Handling Ambiguity: KGs can differentiate between entities with the same name (e.g., "Paris, France" vs. "Paris, Texas") because each entity has a unique identifier and specific relationships.**

Disadvantages:

- **Cost of Construction: Building and maintaining a comprehensive Knowledge Graph is a massive undertaking. It requires significant effort in data extraction, entity linking, and schema design.**
- **Coverage Limitations: A KG can only answer questions about knowledge that is explicitly represented within it. If a fact isn't in the graph, it can't be answered.**
- **Handling Unstructured Information: They are less effective at handling questions that require understanding nuanced language, opinions, or information solely present in unstructured text that hasn't been extracted into the graph. (This is where hybrid systems, combining KGs with LLMs, come into play today).**

In essence, KG-based QnA systems are powerful for answering factual, relationship-driven questions by leveraging highly organized and interconnected data structures. They provide a precise and explainable way to retrieve information, especially for complex queries.

What is a Knowledge Graph (KG)?

Imagine a vast network where:

- **Nodes (or Entities)** represent real-world "things" or concepts. These could be people (e.g., "Albert Einstein"), places (e.g., "Paris"), organizations (e.g., "Google"), events (e.g., "World War II"), or abstract concepts (e.g., "Physics").
- **Edges (or Relationships/Predicates)** connect these nodes, describing the relationships between them. For example:
  - "Albert Einstein" --(was born in)--> "Ulm"
  - "Google" --(has CEO)--> "Sundar Pichai"
  - "Paris" --(is capital of)--> "France"
  - "World War II" --(started in)--> "1939"

So, a Knowledge Graph is a structured, interconnected representation of facts about the world. It organizes information as a graph of entities and the relationships between them, making it easy for machines to understand and query complex, interconnected data.

Think of it like a highly structured, machine-readable encyclopedia or a vast semantic network. Google's famous "Knowledge Graph" that powers the info boxes you see in search results is a prime example.

How do Knowledge Graphs work in QnA Systems?

Traditional QnA systems often rely on searching through unstructured text (like articles, books, web pages). When you ask a question, they try to find sentences or paragraphs that contain the answer.

KG-based QnA systems take a different approach. Instead of just looking for keywords in text, they try to **convert your natural language question into a query that can be executed against the structured Knowledge Graph.**

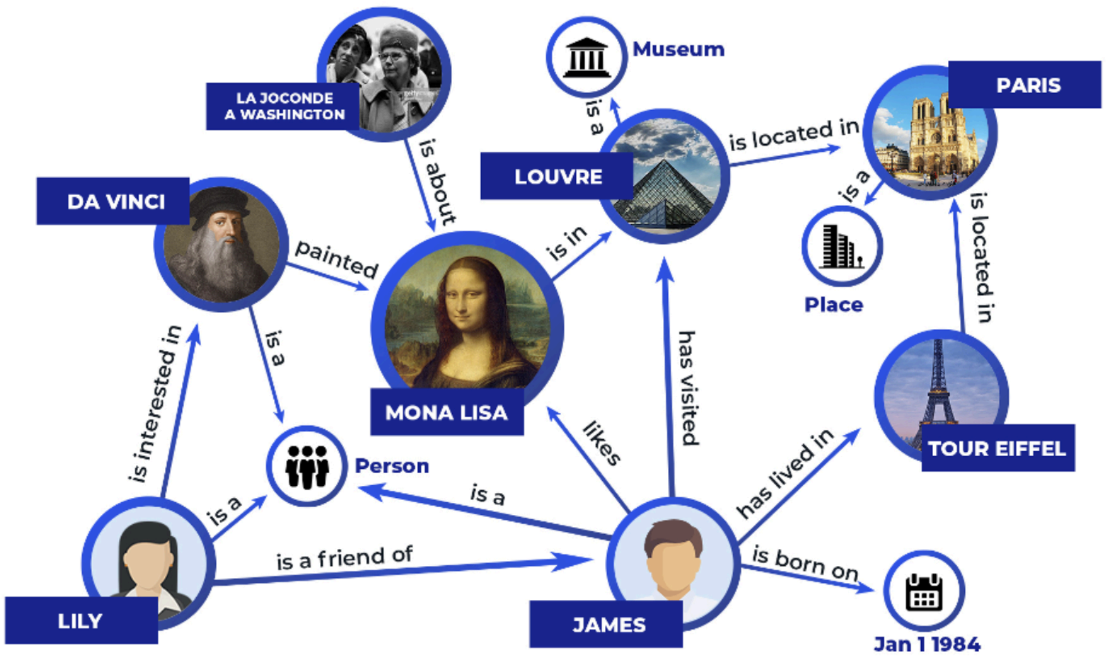

Figure 1 Knowledge graphs illustration.png

Here's the general workflow:

1. **Question Understanding / Semantic Parsing:**
    - The user asks a question in natural language (e.g., "Who directed 'Inception'?").
    - The QnA system's first step is to "understand" this question by parsing it into its core components and mapping them to entities and relationships in the Knowledge Graph. This is often called **semantic parsing**.
    - It identifies:
        - **Entities:** "'Inception'" (a movie)
        - **Relationship (Implicit):** "directed by" (looking for the director)
        - **Answer Type:** `PERSON` (a director is a person)
2. **Query Generation:**
    - Once the entities and relationships are identified, the system translates them into a formal query language that the Knowledge Graph database can understand. Common query languages for KGs include **SPARQL** or **Cypher**.
    - For "Who directed 'Inception'?", the system might generate a query like:
        - `MATCH (movie:Movie {name: 'Inception'})-[r:DIRECTED_BY]->(person:Person) RETURN person.name`

        - *(This translates to: "Find a node (movie) named 'Inception', follow the relationship 'DIRECTED_BY' from it to another node (person), and return that person's name.")*
3. **Graph Traversal / Query Execution:**
    - The generated query is executed directly on the Knowledge Graph.
    - The KG database efficiently traverses its network of nodes and edges to find the matching pattern.
    - It might find: "Inception" --(DIRECTED_BY)--> "Christopher Nolan".
4. **Answer Formulation:**
    - The result of the graph query (e.g., "Christopher Nolan") is directly the answer.
    - The system then presents this answer to the user.

Example Scenario:

- **User Question:** "What is the capital of France?"
- **KG Analysis:**
    - Identify entity: "France" (a Country)
    - Identify relationship: "capital of" (a relationship linking Country to City)
    - Expected answer type: `CITY`
- **KG Query (Conceptual):** Find the `CITY` that is the `capital of France`.
- **KG Data:**
    - (France) --(capital_of)--> (Paris)
- **Answer:** "Paris"

Advantages of Knowledge Graph (KG) based QnA:

- **Precision and Accuracy:** KGs contain factual, structured data, leading to highly precise and accurate answers for factual questions. There's less ambiguity compared to extracting answers from unstructured text.
- **Reasoning and Multi-Hop Questions:** KGs excel at answering complex questions that require "multi-hop" reasoning (following multiple relationships).
    - Example: "Which actors starred in movies directed by Christopher Nolan?"
        - KG can first find all movies directed by "Christopher Nolan".
        - Then, for each of those movies, find all the actors who starred in them.
- **Explainability:** Because the answer is derived from explicit relationships in the graph, it's often easier to explain *how* the system arrived at the answer.
- **Handling Ambiguity:** KGs can differentiate between entities with the same name (e.g., "Paris, France" vs. "Paris, Texas") because each entity has a unique identifier and specific relationships.

Disadvantages:

- **Cost of Construction:** Building and maintaining a comprehensive Knowledge Graph is a massive undertaking. It requires significant effort in data extraction, entity linking, and schema design.
- **Coverage Limitations:** A KG can only answer questions about knowledge that is explicitly represented within it. If a fact isn't in the graph, it can't be answered.
- **Handling Unstructured Information:** They are less effective at handling questions that require understanding nuanced language, opinions, or information solely present in unstructured text that hasn't been extracted into the graph. (This is where hybrid systems, combining KGs with LLMs, come into play today).

In essence, KG-based QnA systems are powerful for answering factual, relationship-driven questions by leveraging highly organized and interconnected data structures. They provide a precise and explainable way to retrieve information, especially for complex queries.