Prompting Techniques

## Few-shot Prompting

Few-shot prompting is a technique in natural language processing where a language model is given a few examples of the desired task along with the input it needs to process. This helps the model understand the task and generate more accurate and contextually relevant responses. The examples serve as a guide, enabling the model to generalize from the provided instances to similar, but unseen tasks.

Schema:  
 - Node Labels: Company, SalesPartner, Customer, Quote, SalesOrder, WorkOrder, Part, ValueStream, ShipmentProvider, ShipmentLocation  
 - Relationships: [:PLACED, :GENERATES, :SHIPS\_TO]  
 - Properties:  
 - Company: name, revenue, marketDescription  
 - SalesOrder: id, date, revenue  
   
 Examples:  
 Question: "List all customers who have placed orders this month."  
 Cypher Query: "MATCH (c:Customer)-[:PLACED]->(o:SalesOrder) WHERE o.date >= date('2024-07-01') RETURN c.name"  
   
 Question: "Retrieve the total revenue for each market description for the past year."  
 Cypher Query: "MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue"  
   
 Question: "Find the top 5 sales partners based on the number of orders they have generated."  
 Cypher Query: "MATCH (sp:SalesPartner)-[:GENERATES]->(o:SalesOrder) RETURN sp.name, COUNT(o) AS orderCount ORDER BY orderCount DESC LIMIT 5"  
   
 Question: "Retrieve the names of all shipment providers that have shipped orders to New York."  
 Cypher Query: "MATCH (s:ShipmentProvider)-[:SHIPS\_TO]->(l:ShipmentLocation {city: 'New York'}) RETURN s.name"  
   
 Question: "How many work orders are associated with each value stream?"  
 Cypher Query: "MATCH (vs:ValueStream)-[:ASSOCIATED\_WITH]->(wo:WorkOrder) RETURN vs.name, COUNT(wo) AS workOrderCount"  
   
 Question: "Retrieve all parts associated with a specific work order ID."  
 Cypher Query: "MATCH (wo:WorkOrder {id: 'WO12345'})-[:USES]->(p:Part) RETURN p.name"  
   
 Question: "List all quotes created by a specific sales partner."  
 Cypher Query: "MATCH (sp:SalesPartner {name: 'Partner A'})-[:CREATED]->(q:Quote) RETURN q.id, q.date, q.amount"  
   
 Question: "Retrieve the names of companies with a specific market description."  
 Cypher Query: "MATCH (c:Company {marketDescription: 'Tech'}) RETURN c.name"  
   
 Question: "Find the total revenue generated by each sales partner for the past quarter."  
 Cypher Query: "MATCH (sp:SalesPartner)-[:GENERATES]->(so:SalesOrder) WHERE so.date >= date('2024-04-01') RETURN sp.name, SUM(so.revenue) AS totalRevenue"  
   
 Question: "List all work orders created within the last month."  
 Cypher Query: "MATCH (wo:WorkOrder) WHERE wo.creationDate >= date('2024-06-01') RETURN wo.id, wo.creationDate"  
   
 Question: "Find the total number of parts used in all work orders."  
 Cypher Query: "MATCH (wo:WorkOrder)-[:USES]->(p:Part) RETURN COUNT(p) AS totalPartsUsed"  
   
 Question: "Retrieve the shipment locations for orders placed by a specific customer."  
 Cypher Query: "MATCH (c:Customer {name: 'Customer A'})-[:PLACED]->(o:SalesOrder)-[:SHIPS\_TO]->(l:ShipmentLocation) RETURN l.city, l.state"  
   
 Question: "Find the average revenue generated by each company."  
 Cypher Query: "MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) RETURN c.name, AVG(s.revenue) AS averageRevenue"

New Question: 'Retrieve the total revenue for all companies in the tech market for the past six months.'

Expected Output:  
Cypher Query: 'MATCH (c:Company {marketDescription: 'Tech'})-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2024-01-31') RETURN SUM(s.revenue) AS totalRevenue'

## Prompt Chaining

In Prompt chaining multiple prompts are linked together in a sequence to solve more complex tasks. Each prompt in the chain builds on the output of the previous one, effectively breaking down a complicated problem into smaller, manageable steps. This approach allows the model to progressively refine its responses and tackle tasks that require a multi-step process or a combination of different capabilities. Prompt chaining can enhance the model's performance by leveraging the strengths of each prompt in the chain.

Use a sequence of prompts to build the query incrementally.

Prompt 1: "Identify the nodes and relationships in the schema for the question: 'Retrieve the total revenue for each market description for the past year.'"  
 Response: "Nodes: Company, SalesOrder. Relationship: Company generates SalesOrder."  
   
 Prompt 2: "What properties are relevant for this question?"  
 Response: "Properties: Company: marketDescription, SalesOrder: date, revenue."  
   
 Prompt 3: "Construct the Cypher query using the identified nodes, relationships, and properties."  
 Response: "MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue"

## Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a prompting technique that supplies domain-relevant data as context to produce responses based on that data and the prompt. This technique is similar to fine-tuning. However, rather than having to fine-tune an FM with a small set of labelled examples, RAG retrieves a small set of relevant documents from a large corpus and uses that to provide context to answer the question. RAG will not change the weights of the foundation model whereas fine-tuning will change model weights.

This approach can be more cost-efficient than regular fine-tuning because the RAG approach doesn't incur the cost of fine-tuning a model. RAG also addresses the challenge of frequent data changes because it retrieves updated and relevant information instead of relying on potentially outdated sets of data.

**Example:**

**Prompt:** "Retrieve the total revenue for each market description for the past year using the latest sales data from the database."

**Output:**

1. Retrieve the latest sales data from the database.
   * External Data Source: Sales database
2. Provide the relevant sales data as context to the model.
   * Context: Sales data for the past year with market descriptions and revenues
3. Generate the Cypher query based on the retrieved data and context.
   * Cypher Query: MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue

## Active-Prompt

Engage interactively to refine and correct the output.

Initial Question: "Retrieve the total revenue for each market description for the past year."  
   
 Initial Cypher Query: "MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue"  
   
 Feedback: "Ensure the revenue property is correctly identified as 'revenue' or 'Rev' and handle any dollar signs."  
   
 Refined Cypher Query: "MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') WITH c, toFloat(replace(s.revenue, '$', '')) AS revenue RETURN c.marketDescription, SUM(revenue) AS totalRevenue"

## Self-Consistency

This technique involves generating multiple outputs for the same input and selecting the most consistent one. self-consistency prompts the model to sample a variety of reasoning paths. Then, the model aggregates the final answer based on multiple data points from the various paths, self-consistency improves CoT reasoning prompting when used in a range of common arithmetic and common-sense reasoning benchmarks.

Example:

- Question: "Retrieve the total revenue for each market description for the past year."  
- Generated Outputs:  
 - `MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue`  
 - `MATCH (company:Company)-[:GENERATES]->(order:SalesOrder) WHERE order.date >= date('2023-07-31') RETURN company.marketDescription, SUM(order.revenue) AS totalRevenue`  
- Selected Output: The output that consistently appears across multiple generations.

## Tree of Thoughts

Tree of Thoughts (ToT) is another technique that builds on the Chain-of-Thought (CoT) prompting technique. CoT prompting samples thoughts sequentially, but ToT prompting follows a tree-branching technique. With the ToT technique, the LLM can learn in a nuanced way, considering multiple paths instead of one sequential path.

ToT prompting is an especially effective method for tasks that involve important initial decisions, strategies for the future, and exploration of multiple solutions. Most LLMs make decisions by following a standard left-to-right token-level inference, but with ToT, LLMs can self-evaluate choices.

**Example:**

**Prompt:** "Generate multiple potential Cypher queries to find the total revenue for each market description for the past year, and evaluate which query yields the best results. (Consider different possible paths)"

**Output:**

1. Generate multiple potential Cypher queries.
   * Query 1: MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue
   * Query 2: MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(toFloat(s.revenue)) AS totalRevenue
   * Query 3: MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') AND c.marketDescription IS NOT NULL RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue
2. Evaluate the generated queries based on execution results and accuracy.
   * Execution and evaluation can involve checking the performance, correctness, and completeness of each query.

## Automatic Prompt Engineer

This technique involves automatically generating the most effective prompts for the task.

Example:

- Question: "Retrieve the total revenue for each market description for the past year."  
- Engineered Prompt: "Given the schema with companies generating sales orders, construct a query to calculate total revenue for each market description over the past year."  
- Cypher Query: `MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue`

## Chain-of-Thought (CoT) Prompting

Chain-of-Thought (CoT) prompting is a technique in natural language processing where a model is encouraged to generate intermediate reasoning steps before arriving at the final answer. This approach involves explicitly instructing the model to articulate its thought process, which can improve its ability to handle complex tasks that require multi-step reasoning and can lead to more accurate and transparent outputs. By breaking down the problem into smaller parts, the model can better navigate through the logical sequence needed to reach the correct conclusion.

Break down the reasoning process step-by-step.

Schema:  
 - Node Labels: Company, SalesOrder  
 - Properties: Company: name, revenue, marketDescription, SalesOrder: date, revenue  
   
 Question: "Retrieve the total revenue for each market description for the past year."  
   
 Step 1: Identify relevant nodes and properties.  
 - Relevant nodes: Company, SalesOrder  
 - Relevant properties: marketDescription, date, revenue  
   
 Step 2: Identify the relationship.  
 - Relationship: Company generates SalesOrder  
   
 Step 3: Formulate the Cypher query.  
 Cypher Query: "MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue"

Chain-of-Thought (CoT) prompting breaks down complex reasoning tasks through intermediary reasoning steps. You can use both zero-shot and few-shot prompting techniques with CoT prompts.  
  
Chain-of-Thought prompts are specific to a problem type. You can use the phrase "Think step by step" to invoke CoT reasoning from your machine learning model.  
  
Tip: Use CoT prompting when the task involves several steps or requires a series of reasoning.

### CoT Zero-shot Prompt

Prompt:  
"Retrieve the total revenue for each market description for the past year. (Think step by step)"

### Output:

1. Identify the relevant nodes and properties.  
 - Nodes: `Company`, `SalesOrder`  
 - Properties: `marketDescription`, `date`, `revenue`  
2. Identify the relationship between the nodes.  
 - Relationship: `Company` generates `SalesOrder`  
3. Construct the Cypher query using the identified nodes, relationships, and properties.  
 - Cypher Query: `MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue`

Note: This prompt did not provide any examples to the model. However, the model was still effective in deciphering the task.

### CoT Few-shot Prompt

Prompt:  
"In the last year, the revenue for different market descriptions is as follows:  
- Market Description: Tech, Revenue: $1,000,000  
- Market Description: Health, Revenue: $800,000  
- Market Description: Retail, Revenue: $600,000  
  
Question: What is the total revenue for all market descriptions in the past year?  
Answer: The total revenue is $1,000,000 (Tech) + $800,000 (Health) + $600,000 (Retail) = $2,400,000.  
  
Question: Retrieve the total revenue for each market description for the past year. (Think step by step)  
Answer:"

### Output:

1. Identify the relevant nodes and properties.  
 - Nodes: `Company`, `SalesOrder`  
 - Properties: `marketDescription`, `date`, `revenue`  
2. Identify the relationship between the nodes.  
 - Relationship: `Company` generates `SalesOrder`  
3. Construct the Cypher query using the identified nodes, relationships, and properties.  
 - Cypher Query: `MATCH (c:Company)-[:GENERATES]->(s:SalesOrder) WHERE s.date >= date('2023-07-31') RETURN c.marketDescription, SUM(s.revenue) AS totalRevenue`

This prompt provided both few-shot context in the form of a question-and-answer example and CoT prompting by asking the model to "Think step by step."