Microsoft

DSToolKit - Text2SQL and Azure AI Search

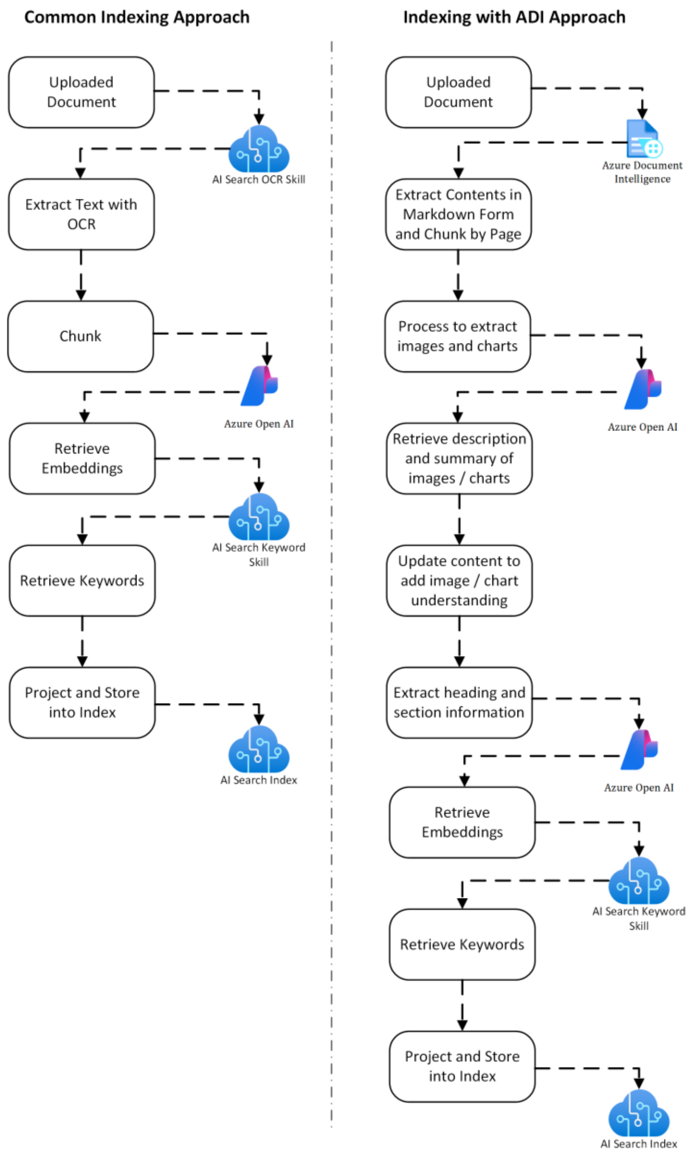
Microsoft

Document Cracking With ADI

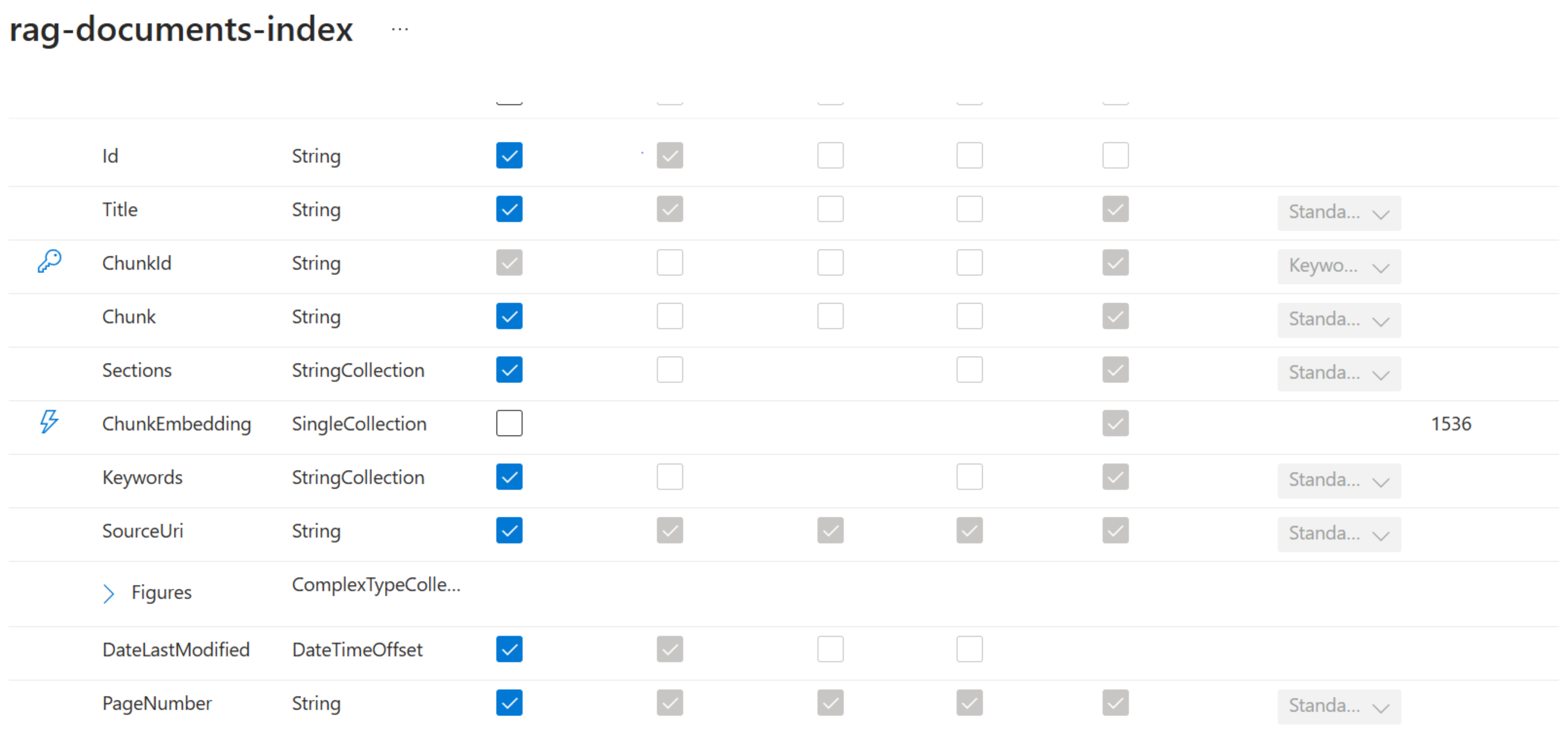
Unstructured Data Search

Industry Solutions | Data & AI

Unstructured Search - Document Cracking Approach


Common Indexing Approach
Uploaded Document
AI Search OCR Skill
Extract Text with OCR
Chunk
Azure Open AI
Retrieve Embeddings
AI Search Keyword Skill
Retrieve Keywords
Project and Store into Index
AI Search Index
Indexing with ADI Approach
Uploaded Document
Azure Document Intelligence
Extract Contents in Markdown Form and Chunk by Page
Process to extract images and charts
Azure Open AI
Retrieve description and summary of images / charts
Update content to add image / chart understanding
Extract heading and section information
Azure Open AI
Retrieve Embeddings
AI Search Keyword Skill
Retrieve Keywords
Project and Store into Index
AI Search Index

Unstructured Search Index Set Up



rag-documents-index ···

	Id	String	✓	✓	☐	☐	☐		
	Title	String	✓	✓	☐	☐	✓	Standa...	
🔑	ChunkId	String	✓	☐	☐	☐	✓	Keywo...	
	Chunk	String	✓	☐	☐	☐	✓	Standa...	
	Sections	StringCollection	✓	☐		☐	✓	Standa...	
⚡	ChunkEmbedding	SingleCollection	☐				✓		1536
	Keywords	StringCollection	✓	☐		☐	✓	Standa...	
	SourceUri	String	✓	✓	✓	✓	✓	Standa...	
	> Figures	ComplexTypeColle...							
	DateLastModified	DateTimeOffset	✓	✓	☐	☐			
	PageNumber	String	✓	✓	✓	✓	✓	Standa...	

Document Cracking with Figure Analysis

Model	Ctx Size	Python	C++	Rust	Java	TypeScript	Average
gpt-4O-2024-05-13	128k	95	80	85	96	97	90.6
gemini-1.5-flash-latest	1000k	93	79	87	94	97	90
Phi-3.5-MoE	128k	89	74	81	88	95	85
Phi-3.5-Mini	128k	86	67	73	77	82	77
Llama-3.1-8B-Instruct	128k	80	65	73	76	63	71
Mixtral-8x7B-Instruct-v0.1	32k	66	65	64	71	74	68
Mixtral-8x22B-Instruct-v0.1	64k	60	67	74	83	55	67.8

Table 1: Comparison results on RepoQA benchmark.

such as Arabic, Chinese, Russian, Ukrainian, and Vietnamese, with average MMLU-multilingual scores of 55.4 and 47.3, respectively. Due to its larger model capacity, **phi-3.5-MoE** achieves a significantly higher average score of 69.9, outperforming **phi-3.5-mini**.

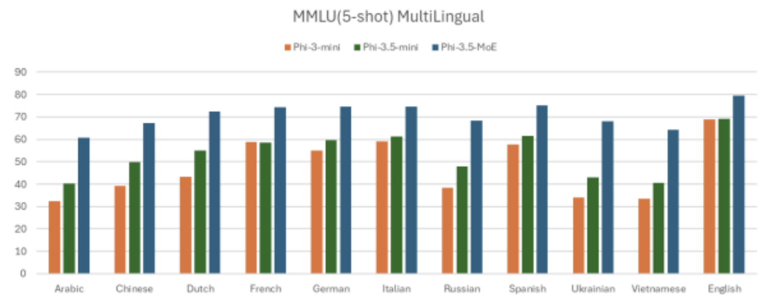
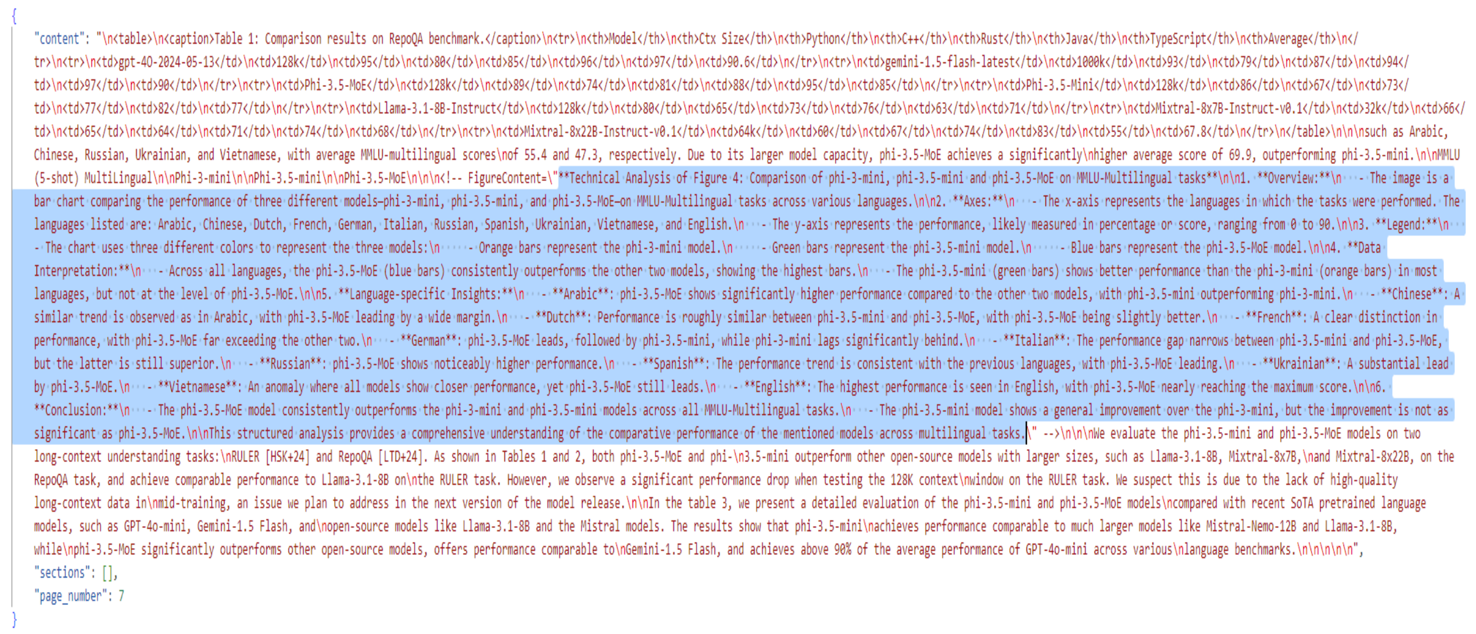


Figure 4: Comparison of **phi-3-mini**, **phi-3.5-mini** and **phi-3.5-MoE** on MMLU-Multilingual tasks

We evaluate the **phi-3.5-mini** and **phi-3.5-MoE** models on two long-context understanding tasks: RULER [HSK+24] and RepoQA [LTD+24]. As shown in Tables 1 and 2, both **phi-3.5-MoE** and **phi-3.5-mini** outperform other open-source models with larger sizes, such as Llama-3.1-8B, Mixtral-8x7B, and Mixtral-8x22B, on the RepoQA task, and achieve comparable performance to Llama-3.1-8B on the RULER task. However, we observe a significant performance drop when testing the 128K context window on the RULER task. We suspect this is due to the lack of high-quality long-context data in mid-training, an issue we plan to address in the next version of the model release.

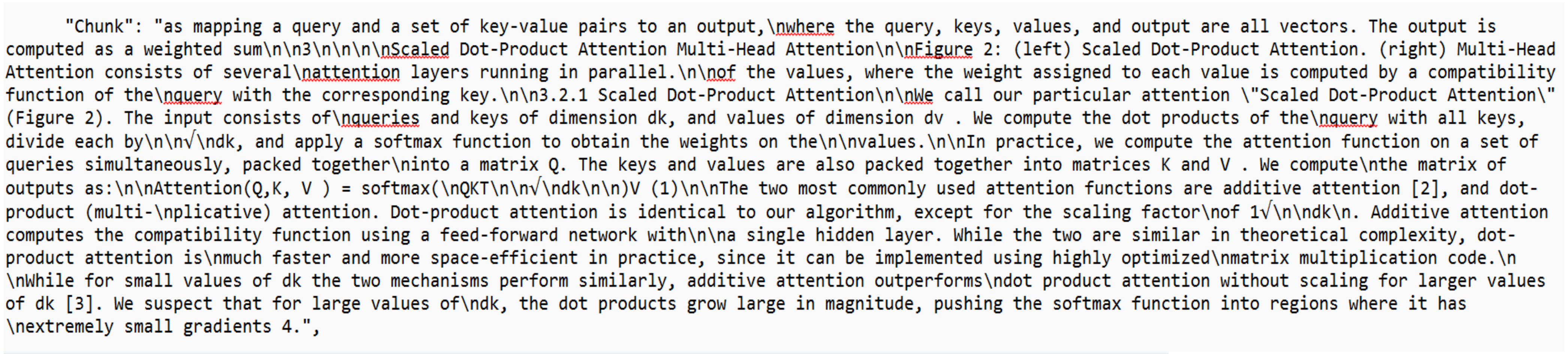
In the table 3, we present a detailed evaluation of the **phi-3.5-mini** and **phi-3.5-MoE** models compared with recent SoTA pretrained language models, such as GPT-4o-mini, Gemini-1.5 Flash, and open-source models like Llama-3.1-8B and the Mistral models. The results show that **phi-3.5-mini** achieves performance comparable to much larger models like Mistral-Nemo-12B and Llama-3.1-8B, while **phi-3.5-MoE** significantly outperforms other open-source models, offers performance comparable to Gemini-1.5 Flash, and achieves above 90% of the average performance of GPT-4o-mini across various language benchmarks.



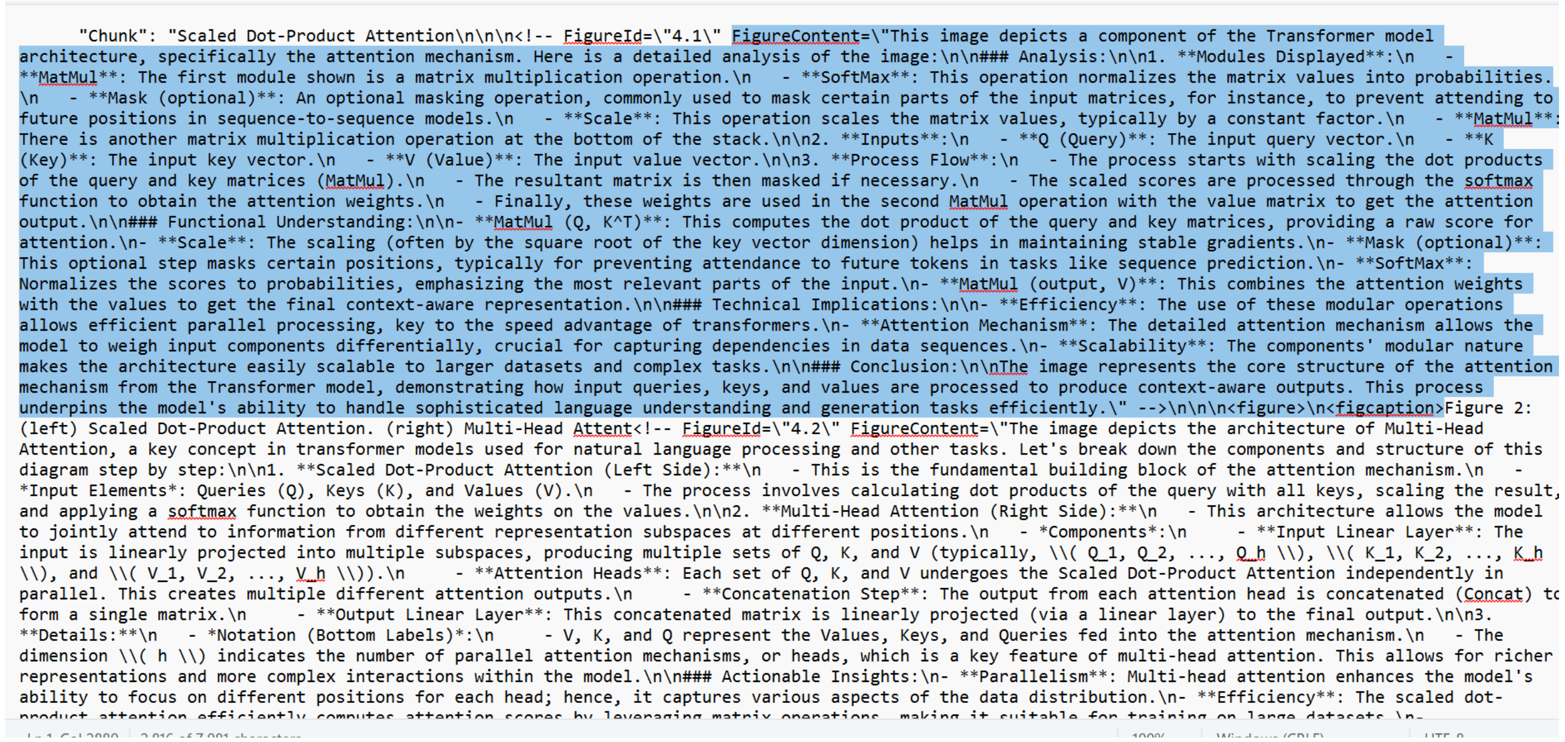
```
{
  "content": "\n<table>\n<caption>Table 1: Comparison results on RepoQA benchmark.</caption>\n<tr>\n<th>Model</th>\n<th>Ctx Size</th>\n<th>Python</th>\n<th>C++</th>\n<th>Rust</th>\n<th>Java</th>\n<th>TypeScript</th>\n<th>Average</th>\n</
  tr>\n<tr>\n<td>gpt-4O-2024-05-13</td>\n<td>128k</td>\n<td>95</td>\n<td>80</td>\n<td>85</td>\n<td>96</td>\n<td>97</td>\n<td>90.6</td>\n</tr>\n<tr>\n<td>gemini-1.5-flash-latest</td>\n<td>1000k</td>\n<td>93</td>\n<td>79</td>\n<td>87</td>\n<td>94</
  td>\n<td>97</td>\n<td>90</td>\n</tr>\n<tr>\n<td>Phi-3.5-MoE</td>\n<td>128k</td>\n<td>89</td>\n<td>74</td>\n<td>81</td>\n<td>88</td>\n<td>95</td>\n<td>85</td>\n</tr>\n<tr>\n<td>Phi-3.5-Mini</td>\n<td>128k</td>\n<td>86</td>\n<td>67</td>\n<td>73</
  td>\n<td>77</td>\n<td>82</td>\n<td>77</td>\n</tr>\n<tr>\n<td>Llama-3.1-8B-Instruct</td>\n<td>128k</td>\n<td>80</td>\n<td>65</td>\n<td>73</td>\n<td>76</td>\n<td>63</td>\n<td>71</td>\n</tr>\n<tr>\n<td>Mixtral-8x7B-Instruct-v0.1</td>\n<td>32k</td>\n<td>66</
  td>\n<td>65</td>\n<td>64</td>\n<td>71</td>\n<td>74</td>\n<td>68</td>\n</tr>\n<tr>\n<td>Mixtral-8x22B-Instruct-v0.1</td>\n<td>64k</td>\n<td>60</td>\n<td>67</td>\n<td>74</td>\n<td>83</td>\n<td>55</td>\n<td>67.8</td>\n</tr>\n</table>\n\n\nsuch as Arabic,
  Chinese, Russian, Ukrainian, and Vietnamese, with average MMLU-multilingual scores\nof 55.4 and 47.3, respectively. Due to its larger model capacity, phi-3.5-MoE achieves a significantly\nhigher average score of 69.9, outperforming phi-3.5-mini.\n\nMMLU
  (5-shot) MultiLingual\n\nPhi-3-mini\n\nPhi-3.5-mini\n\nPhi-3.5-MoE\n\n\n<!-- FigureContent=\"**Technical Analysis of Figure 4: Comparison of phi-3-mini, phi-3.5-mini and phi-3.5-MoE on MMLU-Multilingual tasks**\n\n1. **Overview:**\n   - The image is a
  bar chart comparing the performance of three different models—phi-3-mini, phi-3.5-mini, and phi-3.5-MoE—on MMLU-Multilingual tasks across various languages.\n\n2. **Axes:**\n   - The x-axis represents the languages in which the tasks were performed. The
  languages listed are: Arabic, Chinese, Dutch, French, German, Italian, Russian, Spanish, Ukrainian, Vietnamese, and English.\n   - The y-axis represents the performance, likely measured in percentage or score, ranging from 0 to 90.\n\n3. **Legend:**\n
  - The chart uses three different colors to represent the three models:\n     - Orange bars represent the phi-3-mini model.\n     - Green bars represent the phi-3.5-mini model.\n     - Blue bars represent the phi-3.5-MoE model.\n\n4. **Data
  Interpretation:**\n   - Across all languages, the phi-3.5-MoE (blue bars) consistently outperforms the other two models, showing the highest bars.\n   - The phi-3.5-mini (green bars) shows better performance than the phi-3-mini (orange bars) in most
  languages, but not at the level of phi-3.5-MoE.\n\n5. **Language-specific Insights:**\n   - **Arabic**: phi-3.5-MoE shows significantly higher performance compared to the other two models, with phi-3.5-mini outperforming phi-3-mini.\n   - **Chinese**: A
  similar trend is observed as in Arabic, with phi-3.5-MoE leading by a wide margin.\n   - **Dutch**: Performance is roughly similar between phi-3.5-mini and phi-3.5-MoE, with phi-3.5-MoE being slightly better.\n   - **French**: A clear distinction in
  performance, with phi-3.5-MoE far exceeding the other two.\n   - **German**: phi-3.5-MoE leads, followed by phi-3.5-mini, while phi-3-mini lags significantly behind.\n   - **Italian**: The performance gap narrows between phi-3.5-mini and phi-3.5-MoE,
  but the latter is still superior.\n   - **Russian**: phi-3.5-MoE shows noticeably higher performance.\n   - **Spanish**: The performance trend is consistent with the previous languages, with phi-3.5-MoE leading.\n   - **Ukrainian**: A substantial lead
  by phi-3.5-MoE.\n   - **Vietnamese**: An anomaly where all models show closer performance, yet phi-3.5-MoE still leads.\n   - **English**: The highest performance is seen in English, with phi-3.5-MoE nearly reaching the maximum score.\n\n6.
  **Conclusion:**\n   - The phi-3.5-MoE model consistently outperforms the phi-3-mini and phi-3.5-mini models across all MMLU-Multilingual tasks.\n   - The phi-3.5-mini model shows a general improvement over the phi-3-mini, but the improvement is not as
  significant as phi-3.5-MoE.\n\nThis structured analysis provides a comprehensive understanding of the comparative performance of the mentioned models across multilingual tasks.\" -->\n\n\nWe evaluate the phi-3.5-mini and phi-3.5-MoE models on two
  long-context understanding tasks:\nRULER [HSK+24] and RepoQA [LTD+24]. As shown in Tables 1 and 2, both phi-3.5-MoE and phi-\n3.5-mini outperform other open-source models with larger sizes, such as Llama-3.1-8B, Mixtral-8x7B,\nand Mixtral-8x22B, on the
  RepoQA task, and achieve comparable performance to Llama-3.1-8B on\nthe RULER task. However, we observe a significant performance drop when testing the 128K context\nwindow on the RULER task. We suspect this is due to the lack of high-quality
  long-context data in\nmid-training, an issue we plan to address in the next version of the model release.\n\nIn the table 3, we present a detailed evaluation of the phi-3.5-mini and phi-3.5-MoE models\ncompared with recent SoTA pretrained language
  models, such as GPT-4o-mini, Gemini-1.5 Flash, and\nopen-source models like Llama-3.1-8B and the Mistral models. The results show that phi-3.5-mini\nachieves performance comparable to much larger models like Mistral-Nemo-12B and Llama-3.1-8B,
  while\nphi-3.5-MoE significantly outperforms other open-source models, offers performance comparable to\nGemini-1.5 Flash, and achieves above 90% of the average performance of GPT-4o-mini across various\nlanguage benchmarks.\n\n\n\n\n",
  "sections": [],
  "page_number": 7
}
```

Figure Understanding with gpt4o to Update Markdown

Chunk Comparison (Old vs New Indexing Approach)



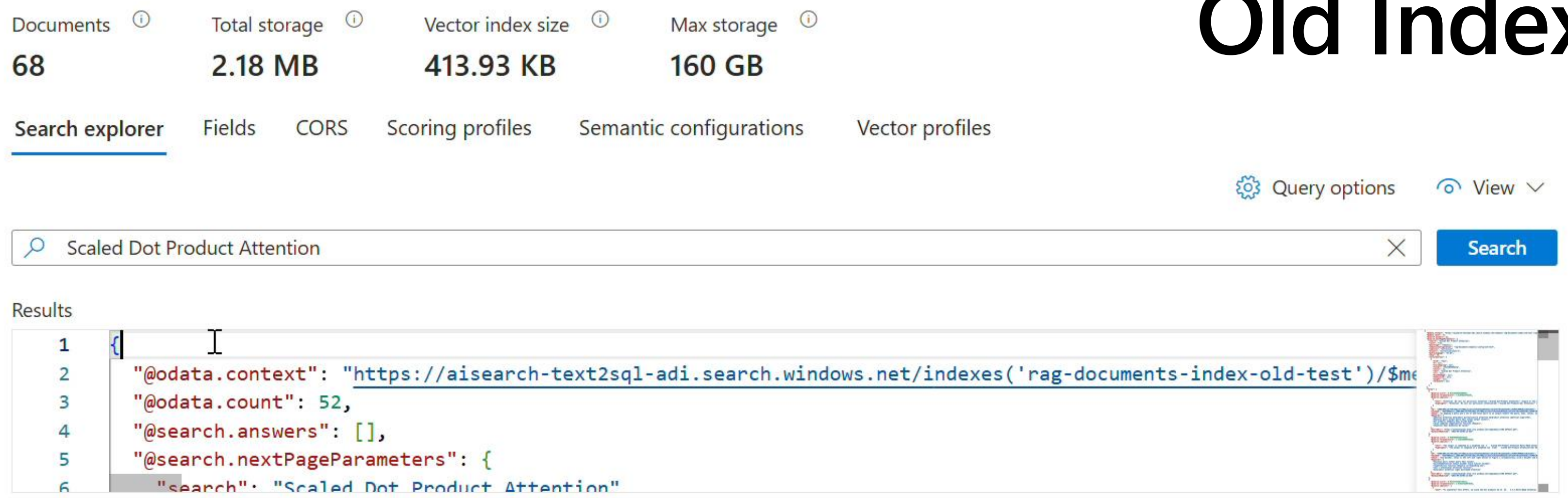
```
    "Chunk": "as mapping a query and a set of key-value pairs to an output,\nwhere the query, keys, values, and output are all vectors. The output is computed as a weighted sum\n\n3\n\n\n\nScaled Dot-Product Attention Multi-Head Attention\n\nFigure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several\nattention layers running in parallel.\n\nof the values, where the weight assigned to each value is computed by a compatibility function of the\nquery with the corresponding key.\n\n3.2.1 Scaled Dot-Product Attention\n\nWe call our particular attention \"Scaled Dot-Product Attention\" (Figure 2). The input consists of\nqueries and keys of dimension dk, and values of dimension dv . We compute the dot products of the\nquery with all keys, divide each by\n\n√\ndk, and apply a softmax function to obtain the weights on the\n\nvalues.\n\nIn practice, we compute the attention function on a set of queries simultaneously, packed together\ninto a matrix Q. The keys and values are also packed together into matrices K and V . We compute\nthe matrix of outputs as:\n\nAttention(Q,K, V ) = softmax(\nQKT\n\n√\ndk\n\n)V (1)\n\nThe two most commonly used attention functions are additive attention [2], and dot-product (multi-\nplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor\nof 1√\n\ndk\n. Additive attention computes the compatibility function using a feed-forward network with\n\na single hidden layer. While the two are similar in theoretical complexity, dot-product attention is\nmuch faster and more space-efficient in practice, since it can be implemented using highly optimized\nmatrix multiplication code.\n\nWhile for small values of dk the two mechanisms perform similarly, additive attention outperforms\ndot product attention without scaling for larger values of dk [3]. We suspect that for large values of\ndk, the dot products grow large in magnitude, pushing the softmax function into regions where it has\nextremely small gradients 4.",
```



```
    "Chunk": "Scaled Dot-Product Attention\n\n\n<!-- FigureId=\"4.1\" FigureContent=\"This image depicts a component of the Transformer model architecture, specifically the attention mechanism. Here is a detailed analysis of the image:\n\n### Analysis:\n\n1. **Modules Displayed**:\n   - **MatMul**: The first module shown is a matrix multiplication operation.\n   - **SoftMax**: This operation normalizes the matrix values into probabilities.\n   - **Mask (optional)**: An optional masking operation, commonly used to mask certain parts of the input matrices, for instance, to prevent attending to future positions in sequence-to-sequence models.\n   - **Scale**: This operation scales the matrix values, typically by a constant factor.\n   - **MatMul**: There is another matrix multiplication operation at the bottom of the stack.\n\n2. **Inputs**:\n   - **Q (Query)**: The input query vector.\n   - **K (Key)**: The input key vector.\n   - **V (Value)**: The input value vector.\n\n3. **Process Flow**:\n   - The process starts with scaling the dot products of the query and key matrices (MatMul).\n   - The resultant matrix is then masked if necessary.\n   - The scaled scores are processed through the softmax function to obtain the attention weights.\n   - Finally, these weights are used in the second MatMul operation with the value matrix to get the attention output.\n\n### Functional Understanding:\n\n- **MatMul (Q, K^T)**: This computes the dot product of the query and key matrices, providing a raw score for attention.\n- **Scale**: The scaling (often by the square root of the key vector dimension) helps in maintaining stable gradients.\n- **Mask (optional)**: This optional step masks certain positions, typically for preventing attendance to future tokens in tasks like sequence prediction.\n- **SoftMax**: Normalizes the scores to probabilities, emphasizing the most relevant parts of the input.\n- **MatMul (output, V)**: This combines the attention weights with the values to get the final context-aware representation.\n\n### Technical Implications:\n\n- **Efficiency**: The use of these modular operations allows efficient parallel processing, key to the speed advantage of transformers.\n- **Attention Mechanism**: The detailed attention mechanism allows the model to weigh input components differentially, crucial for capturing dependencies in data sequences.\n- **Scalability**: The components' modular nature makes the architecture easily scalable to larger datasets and complex tasks.\n\n### Conclusion:\n\nThe image represents the core structure of the attention mechanism from the Transformer model, demonstrating how input queries, keys, and values are processed to produce context-aware outputs. This process underpins the model's ability to handle sophisticated language understanding and generation tasks efficiently.\" -->\n\n\n<figure>\n<figcaption>Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attent<!-- FigureId=\"4.2\" FigureContent=\"The image depicts the architecture of Multi-Head Attention, a key concept in transformer models used for natural language processing and other tasks. Let's break down the components and structure of this diagram step by step:\n\n1. **Scaled Dot-Product Attention (Left Side):**\n   - This is the fundamental building block of the attention mechanism.\n   - *Input Elements*: Queries (Q), Keys (K), and Values (V).\n   - The process involves calculating dot products of the query with all keys, scaling the result, and applying a softmax function to obtain the weights on the values.\n\n2. **Multi-Head Attention (Right Side):**\n   - This architecture allows the model to jointly attend to information from different representation subspaces at different positions.\n   - *Components*:\n     - **Input Linear Layer**: The input is linearly projected into multiple subspaces, producing multiple sets of Q, K, and V (typically, \\( Q_1, Q_2, ..., Q_h \\), \\( K_1, K_2, ..., K_h \\), and \\( V_1, V_2, ..., V_h \\)).\n     - **Attention Heads**: Each set of Q, K, and V undergoes the Scaled Dot-Product Attention independently in parallel. This creates multiple different attention outputs.\n     - **Concatenation Step**: The output from each attention head is concatenated (Concat) to form a single matrix.\n     - **Output Linear Layer**: This concatenated matrix is linearly projected (via a linear layer) to the final output.\n\n3. **Details:**\n   - *Notation (Bottom Labels)*:\n     - V, K, and Q represent the Values, Keys, and Queries fed into the attention mechanism.\n   - The dimension \\( h \\) indicates the number of parallel attention mechanisms, or heads, which is a key feature of multi-head attention. This allows for richer representations and more complex interactions within the model.\n\n### Actionable Insights:\n- **Parallelism**: Multi-head attention enhances the model's ability to focus on different positions for each head; hence, it captures various aspects of the data distribution.\n- **Efficiency**: The scaled dot-product attention efficiently computes attention scores by leveraging matrix operations, making it suitable for training on large datasets.\n-
```

search query : scaled dot product attention

Old Index



```
{
  "@odata.context": "https://aisearch-text2sql-adi.search.windows.net/indexes('rag-documents-index-old-test')/$me
  "@odata.count": 52,
  "@search.answers": [],
  "@search.nextPageParameters": {
    "search": "Scaled Dot Product Attention"
```

Search Results

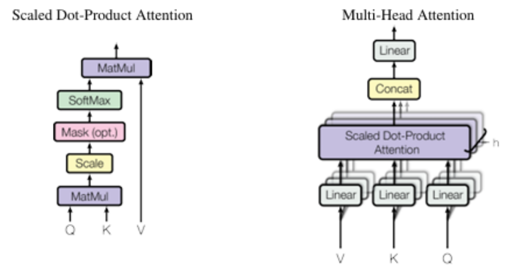


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

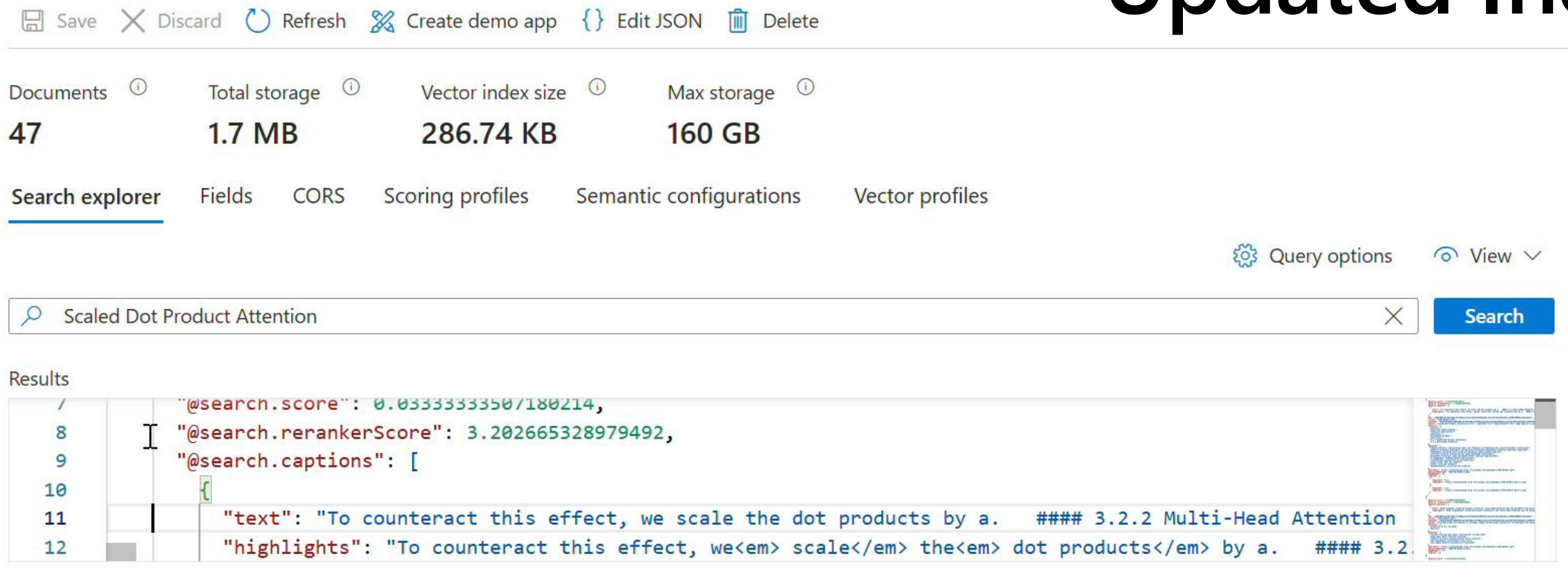
In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

The two most commonly used attention functions are additive attention [2], and dot-product (multi-plicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

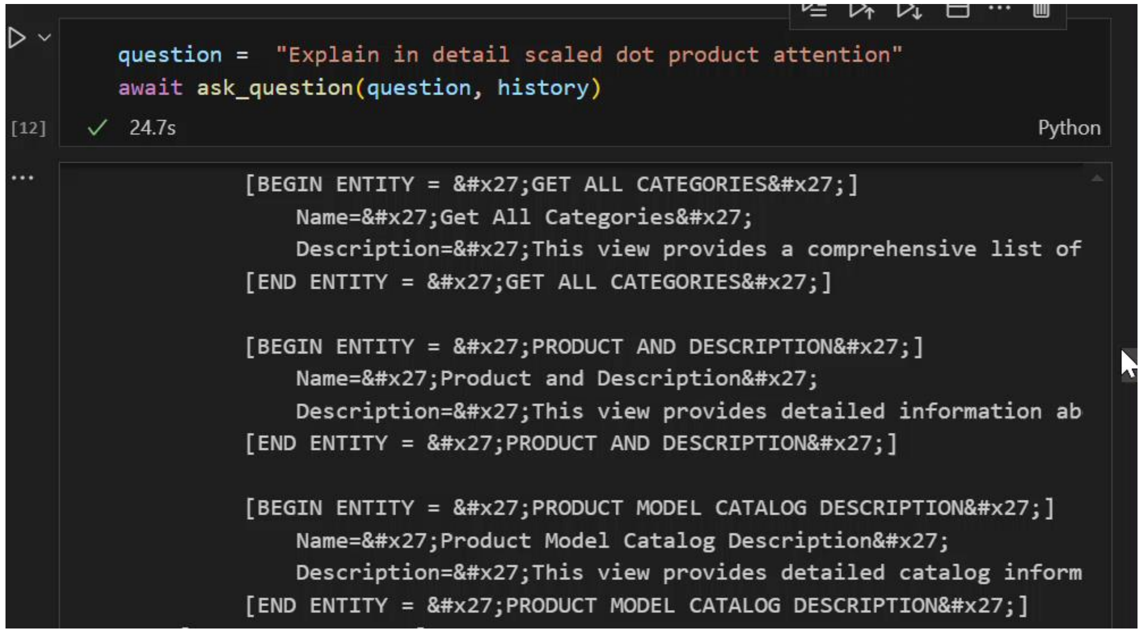
While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients[4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

Updated Index



```
    "@search.score": 0.03333333507180214,
    "@search.rerankerScore": 3.202665328979492,
    "@search.captions": [
      {
        "text": "To counteract this effect, we scale the dot products by a.   #### 3.2.2 Multi-Head Attention
        "highlights": "To counteract this effect, we<em> scale</em> the<em> dot products</em> by a.   #### 3.2.
```

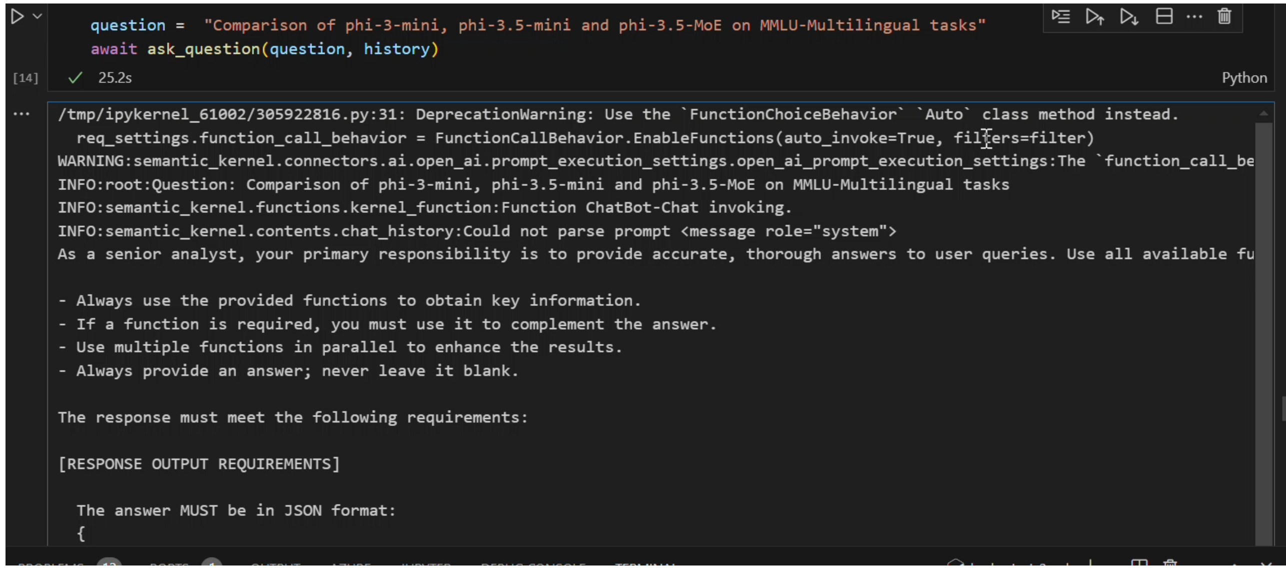
AutoFunction Calling



```python
question =  "Explain in detail scaled dot product attention"
await ask_question(question, history)
```

[12] ✓ 24.7s Python

```
            [BEGIN ENTITY = &#x27;GET ALL CATEGORIES&#x27;]
                Name=&#x27;Get All Categories&#x27;
                Description=&#x27;This view provides a comprehensive list of
            [END ENTITY = &#x27;GET ALL CATEGORIES&#x27;]

            [BEGIN ENTITY = &#x27;PRODUCT AND DESCRIPTION&#x27;]
                Name=&#x27;Product and Description&#x27;
                Description=&#x27;This view provides detailed information ab
            [END ENTITY = &#x27;PRODUCT AND DESCRIPTION&#x27;]

            [BEGIN ENTITY = &#x27;PRODUCT MODEL CATALOG DESCRIPTION&#x27;]
                Name=&#x27;Product Model Catalog Description&#x27;
                Description=&#x27;This view provides detailed catalog inform
            [END ENTITY = &#x27;PRODUCT MODEL CATALOG DESCRIPTION&#x27;]
```



```python
question =  "Comparison of phi-3-mini, phi-3.5-mini and phi-3.5-MoE on MMLU-Multilingual tasks"
await ask_question(question, history)
```

[14] ✓ 25.2s Python

```
/tmp/ipykernel_61002/305922816.py:31: DeprecationWarning: Use the `FunctionChoiceBehavior` `Auto` class method instead.
  req_settings.function_call_behavior = FunctionCallBehavior.EnableFunctions(auto_invoke=True, filters=filter)
WARNING:semantic_kernel.connectors.ai.open_ai.prompt_execution_settings.open_ai_prompt_execution_settings:The `function_call_be
INFO:root:Question: Comparison of phi-3-mini, phi-3.5-mini and phi-3.5-MoE on MMLU-Multilingual tasks
INFO:semantic_kernel.functions.kernel_function:Function ChatBot-Chat invoking.
INFO:semantic_kernel.contents.chat_history:Could not parse prompt <message role="system">
As a senior analyst, your primary responsibility is to provide accurate, thorough answers to user queries. Use all available fu

- Always use the provided functions to obtain key information.
- If a function is required, you must use it to complement the answer.
- Use multiple functions in parallel to enhance the results.
- Always provide an answer; never leave it blank.

The response must meet the following requirements:

[RESPONSE OUTPUT REQUIREMENTS]

  The answer MUST be in JSON format:
  {
```

Challenges & Learnings

- Azure Document Intelligence unable to parse PPTX ad XLSX files
- Moving from inbuilt skills for enrichment to custom skills
- Monitoring and debugging of each session for failures
- Moving to MI implementation for Search
- Metadata enrichment for index
- Page wise chunking retrieved better search results given the structure of majority documents , however semantic chunking can also be tried out

Microsoft

Text2SQL

Structured Data Search

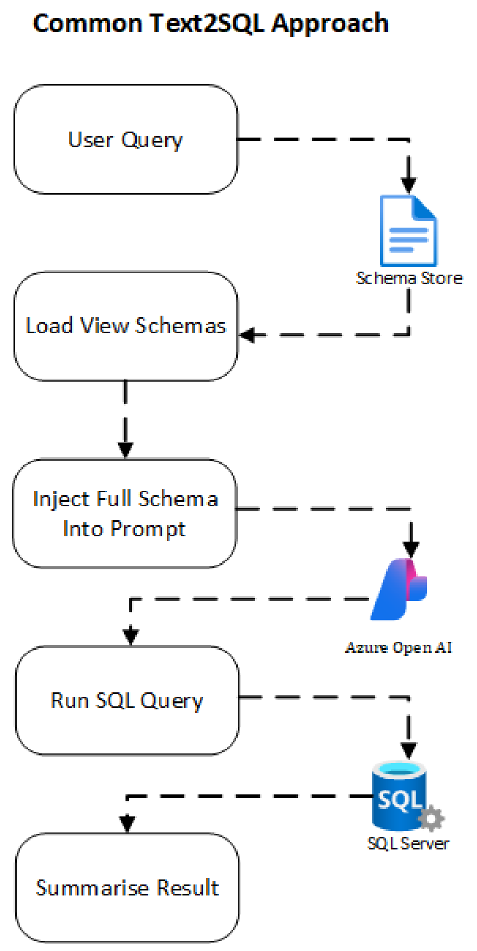
Industry Solutions | Data & AI

Why Text2SQL?

- Allows aggregations and calculations on data quickly
  - Without SQL integration, questions may need specifically pre-computed report
  - e.g. What is our top performing sales per by quantity of units sold this month? is answerable with a few simple SQL query if the correct views are exposed.
- Without Text2SQL, automatic report generation needs to be built.
- Pushing numerical calculations onto the source SQL engine ensures accuracy in the maths.
- Data can be updated real-time in the source database and be immediately accessible to the LLM.

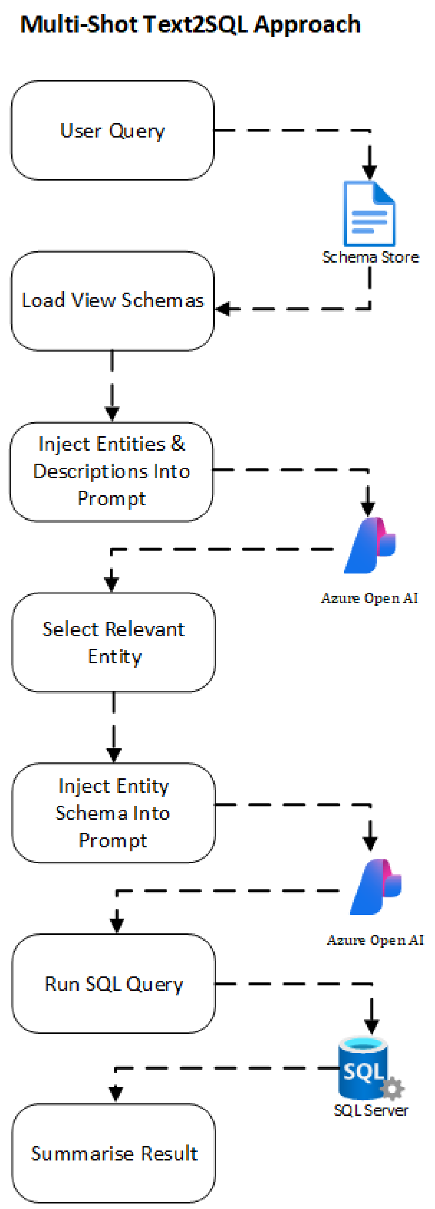
Common Text2SQL Approach – Single Shot

- Limitations of single shot:
  - More tables / views significantly increases the number of tokens used within the prompt and the cost of inference.
    - Increasingly expensive operation
  - Excessive amounts of schema information can cause confusion with the LLM.
    - In our original use case, when exceeding 5 complex tables / views, we found that the LLM could get confused between which columns belonged to which entity and as such, would generate invalid SQL queries.



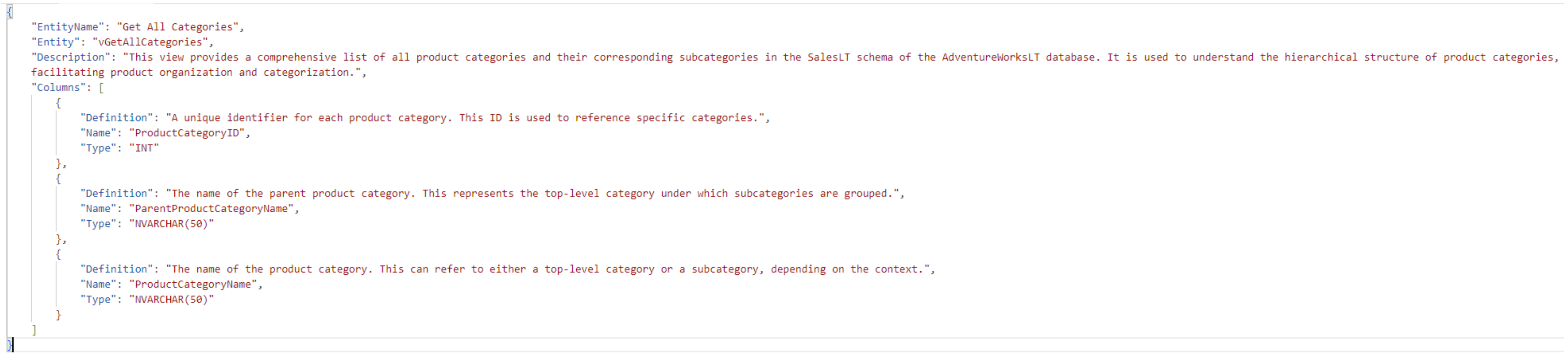
Our Initial Approach – Multi-Shot

- Break down the query generation into 2 steps:
  - Selection of the correct view(s)
  - Retrieval of the full schemas for the selected views
- Advantages:
  - Reduces token usage
  - Reduces confusion between schema attributes



Data Dictionary & Metadata Gathering

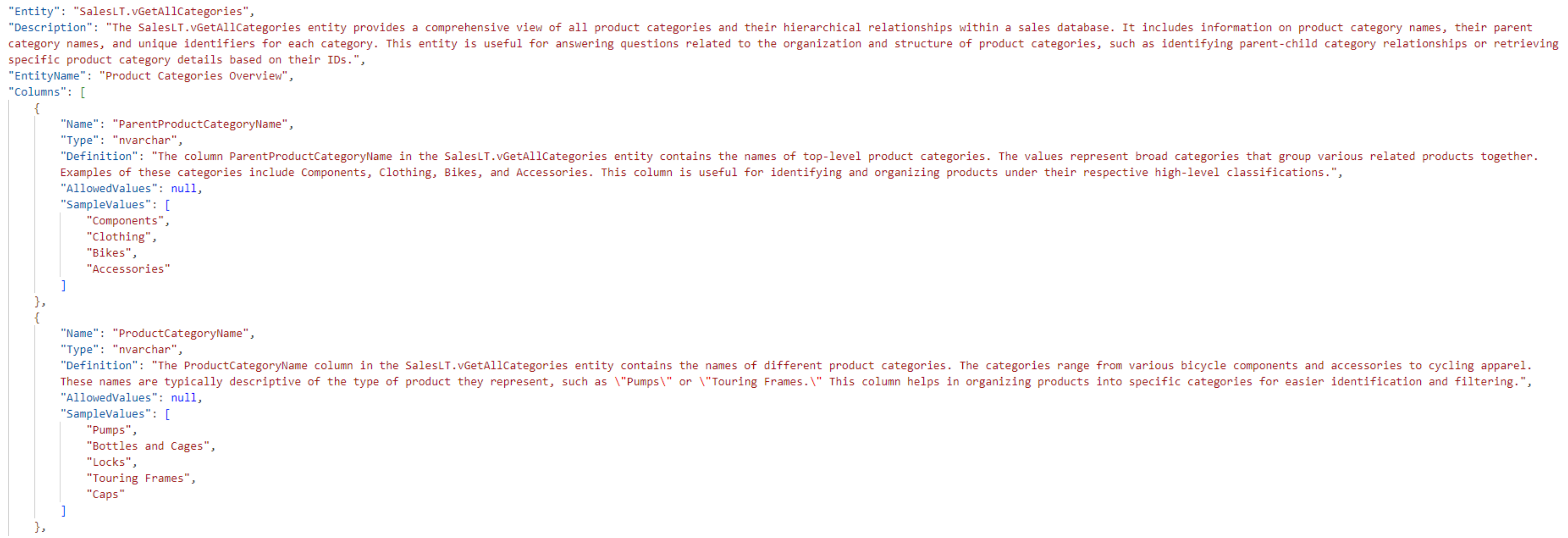
- Maps out all the entities available in the SQL Source
  - Includes detailed descriptions for all columns available and the data types. Effectively “prompt engineering” per column
  - Avoids adding latency in fetching the schemas at runtime
- Quality of metadata directly determines the quality of entity and column selection.



```
{
    "EntityName": "Get All Categories",
    "Entity": "vGetAllCategories",
    "Description": "This view provides a comprehensive list of all product categories and their corresponding subcategories in the SalesLT schema of the AdventureWorksLT database. It is used to understand the hierarchical structure of product categories,
    facilitating product organization and categorization.",
    "Columns": [
        {
            "Definition": "A unique identifier for each product category. This ID is used to reference specific categories.",
            "Name": "ProductCategoryID",
            "Type": "INT"
        },
        {
            "Definition": "The name of the parent product category. This represents the top-level category under which subcategories are grouped.",
            "Name": "ParentProductCategoryName",
            "Type": "NVARCHAR(50)"
        },
        {
            "Definition": "The name of the product category. This can refer to either a top-level category or a subcategory, depending on the context.",
            "Name": "ProductCategoryName",
            "Type": "NVARCHAR(50)"
        }
    ]
}
```

Automated Metadata Gathering

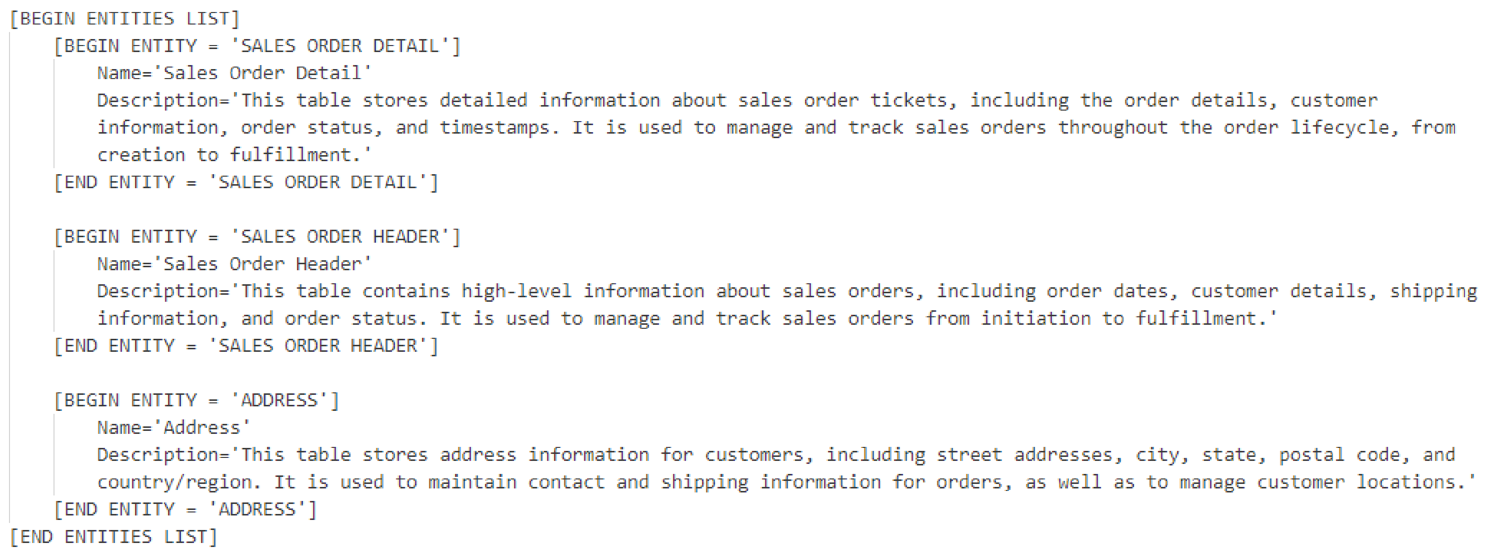
- Metadata collection is a time-consuming exercise.
- Automatic metadata collection is used with a combination of SQL queries and prompting with an LLM.



```
"Entity": "SalesLT.vGetAllCategories",
"Description": "The SalesLT.vGetAllCategories entity provides a comprehensive view of all product categories and their hierarchical relationships within a sales database. It includes information on product category names, their parent
category names, and unique identifiers for each category. This entity is useful for answering questions related to the organization and structure of product categories, such as identifying parent-child category relationships or retrieving
specific product category details based on their IDs.",
"EntityName": "Product Categories Overview",
"Columns": [
    {
        "Name": "ParentProductCategoryName",
        "Type": "nvarchar",
        "Definition": "The column ParentProductCategoryName in the SalesLT.vGetAllCategories entity contains the names of top-level product categories. The values represent broad categories that group various related products together.
        Examples of these categories include Components, Clothing, Bikes, and Accessories. This column is useful for identifying and organizing products under their respective high-level classifications.",
        "AllowedValues": null,
        "SampleValues": [
            "Components",
            "Clothing",
            "Bikes",
            "Accessories"
        ]
    },
    {
        "Name": "ProductCategoryName",
        "Type": "nvarchar",
        "Definition": "The ProductCategoryName column in the SalesLT.vGetAllCategories entity contains the names of different product categories. The categories range from various bicycle components and accessories to cycling apparel.
        These names are typically descriptive of the type of product they represent, such as \"Pumps\" or \"Touring Frames.\" This column helps in organizing products into specific categories for easier identification and filtering.",
        "AllowedValues": null,
        "SampleValues": [
            "Pumps",
            "Bottles and Cages",
            "Locks",
            "Touring Frames",
            "Caps"
        ]
    },
```

Prompt Injection

- Data dictionary transformed at runtime into formatted prompt.

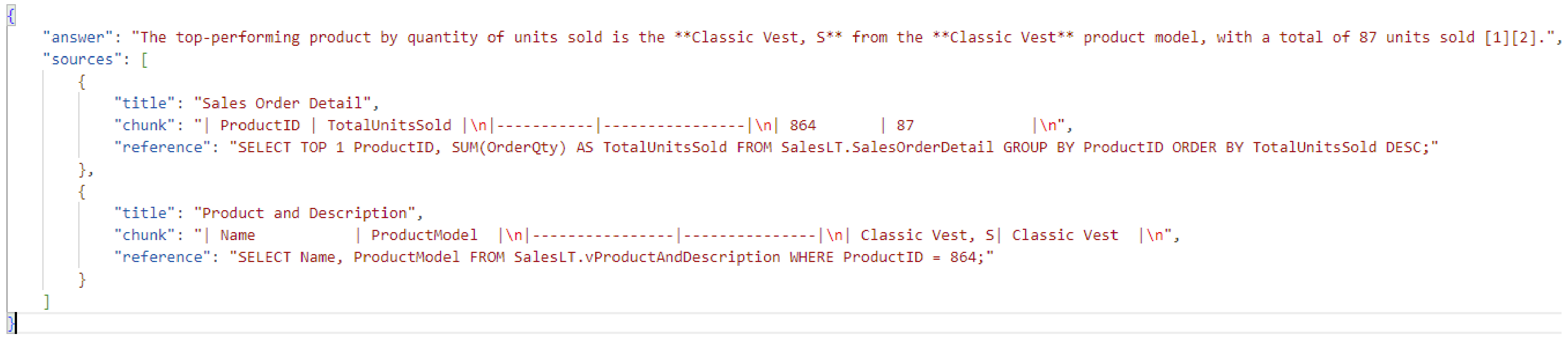


```
[BEGIN ENTITIES LIST]
    [BEGIN ENTITY = 'SALES ORDER DETAIL']
        Name='Sales Order Detail'
        Description='This table stores detailed information about sales order tickets, including the order details, customer
        information, order status, and timestamps. It is used to manage and track sales orders throughout the order lifecycle, from
        creation to fulfillment.'
    [END ENTITY = 'SALES ORDER DETAIL']

    [BEGIN ENTITY = 'SALES ORDER HEADER']
        Name='Sales Order Header'
        Description='This table contains high-level information about sales orders, including order dates, customer details, shipping
        information, and order status. It is used to manage and track sales orders from initiation to fulfillment.'
    [END ENTITY = 'SALES ORDER HEADER']

    [BEGIN ENTITY = 'ADDRESS']
        Name='Address'
        Description='This table stores address information for customers, including street addresses, city, state, postal code, and
        country/region. It is used to maintain contact and shipping information for orders, as well as to manage customer locations.'
    [END ENTITY = 'ADDRESS']
[END ENTITIES LIST]
```

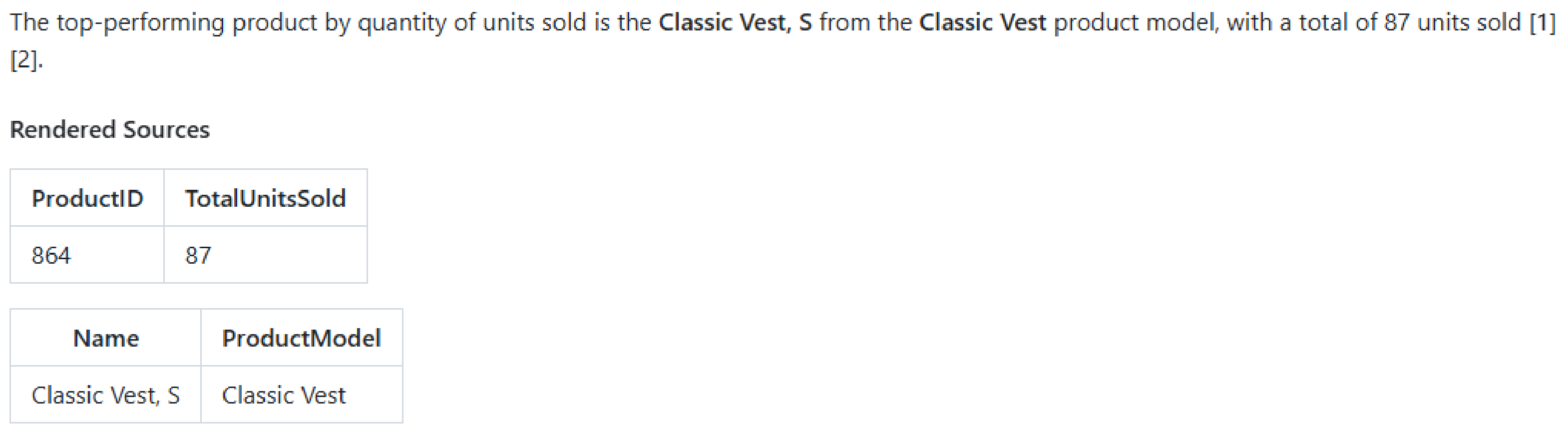
- GetEntitySchema() method is exposed to the LLM.
  - LLM selects and calls GetEntitySchema() via Auto Function calling to retrieve the full schema for the entities it believes will answer the question.
  - Reduces the passing of all schemas to the LLM, to only those that match the user's question.

Text2SQL Examples

What is the top performing quantity in terms of sales numbers?



```
{
    "answer": "The top-performing product by quantity of units sold is the **Classic Vest, S** from the **Classic Vest** product model, with a total of 87 units sold [1][2].",
    "sources": [
        {
            "title": "Sales Order Detail",
            "chunk": "| ProductID | TotalUnitsSold |\n|-----------|----------------|\n| 864       | 87             |\n",
            "reference": "SELECT TOP 1 ProductID, SUM(OrderQty) AS TotalUnitsSold FROM SalesLT.SalesOrderDetail GROUP BY ProductID ORDER BY TotalUnitsSold DESC;"
        },
        {
            "title": "Product and Description",
            "chunk": "| Name           | ProductModel  |\n|----------------|---------------|\n| Classic Vest, S| Classic Vest  |\n",
            "reference": "SELECT Name, ProductModel FROM SalesLT.vProductAndDescription WHERE ProductID = 864;"
        }
    ]
}
```



The top-performing product by quantity of units sold is the **Classic Vest, S** from the **Classic Vest** product model, with a total of 87 units sold [1] [2].

Rendered Sources

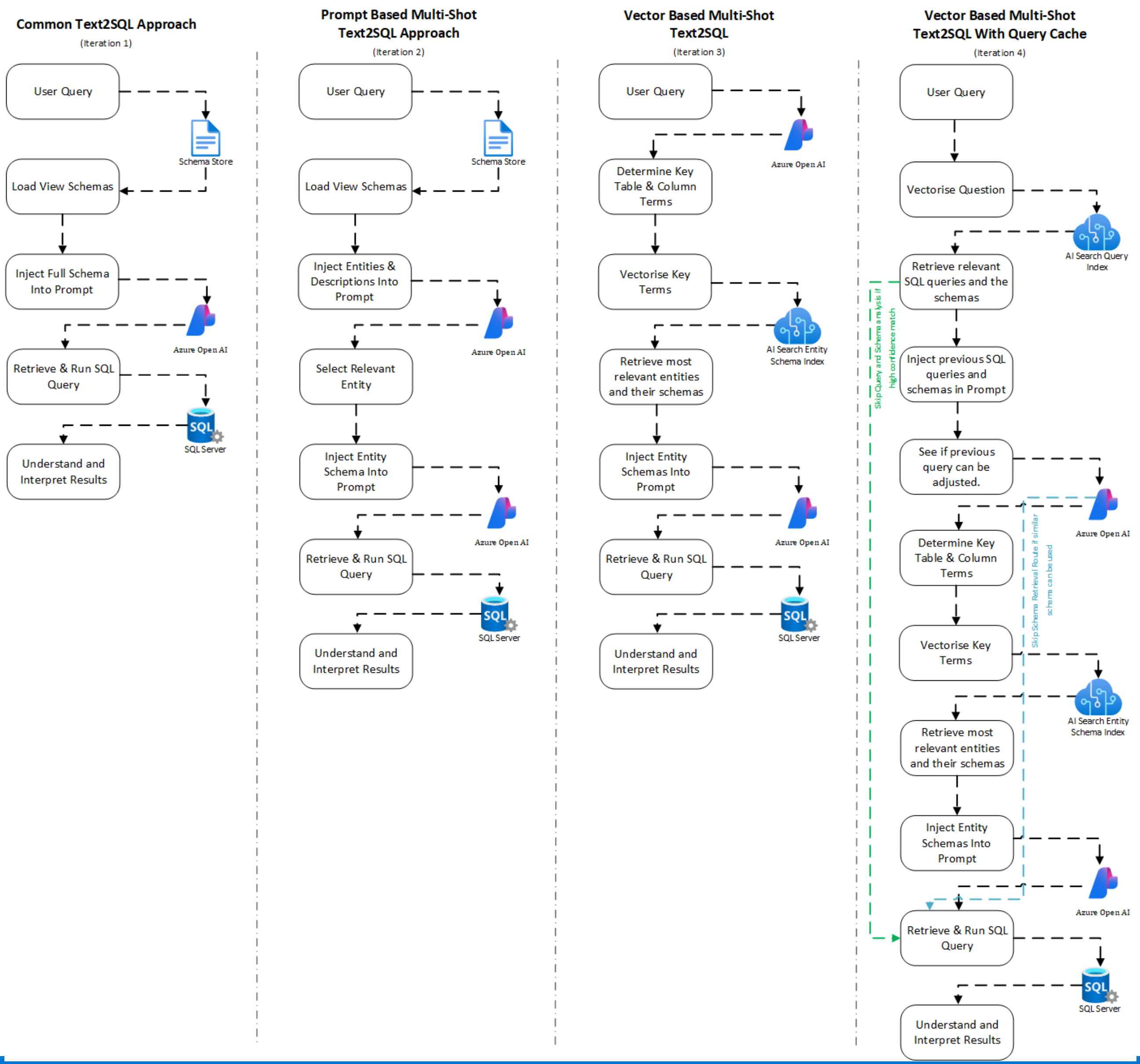
ProductID	TotalUnitsSold
864	87

Name	ProductModel
Classic Vest, S	Classic Vest

Further Improvements

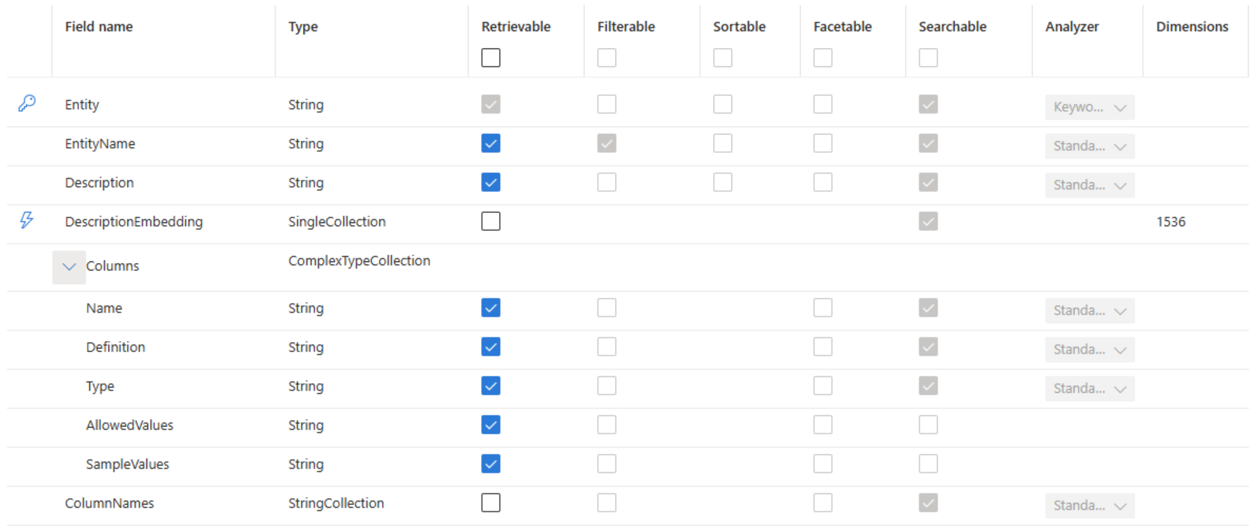
- Offload selection of entities from the LLM.
  - Reduce token usage
  - Speed up selection
  - Improve reliability by using deterministic approach
- Cache the question to query resolution.
  - Speed up resolution for commonly asked questions.
- Pre-fetch database results for commonly asked questions
  - Can we skip query generation completely?

Text2SQL Further Iterations


Common Text2SQL Approach
(Iteration 1)
User Query
Schema Store
Load View Schemas
Inject Full Schema Into Prompt
Azure Open AI
Retrieve & Run SQL Query
SQL
SQL Server
Understand and Interpret Results
Prompt Based Multi-Shot Text2SQL Approach
(Iteration 2)
User Query
Schema Store
Load View Schemas
Inject Entities & Descriptions Into Prompt
Azure Open AI
Select Relevant Entity
Inject Entity Schema Into Prompt
Azure Open AI
Retrieve & Run SQL Query
SQL
SQL Server
Understand and Interpret Results
Vector Based Multi-Shot Text2SQL
(Iteration 3)
User Query
Azure Open AI
Determine Key Table & Column Terms
Vectorise Key Terms
AI Search Entity Schema Index
Retrieve most relevant entities and their schemas
Inject Entity Schemas Into Prompt
Azure Open AI
Retrieve & Run SQL Query
SQL
SQL Server
Understand and Interpret Results
Vector Based Multi-Shot Text2SQL With Query Cache
(Iteration 4)
User Query
Vectorise Question
AI Search Query Index
Retrieve relevant SQL queries and the schemas
Skip Query and Schema analysis if high confidence match
Inject previous SQL queries and schemas in Prompt
See if previous query can be adjusted.
Azure Open AI
Skip Schema Retrieval Route if similar schema can be used
Determine Key Table & Column Terms
Vectorise Key Terms
AI Search Entity Schema Index
Retrieve most relevant entities and their schemas
Inject Entity Schemas Into Prompt
Azure Open AI
Retrieve & Run SQL Query
SQL
SQL Server
Understand and Interpret Results

Schema Index Setup

- Transforms the data dictionary into a index.
- Vectorises the description property.
- Setups column names as “keywords” in semantic config.

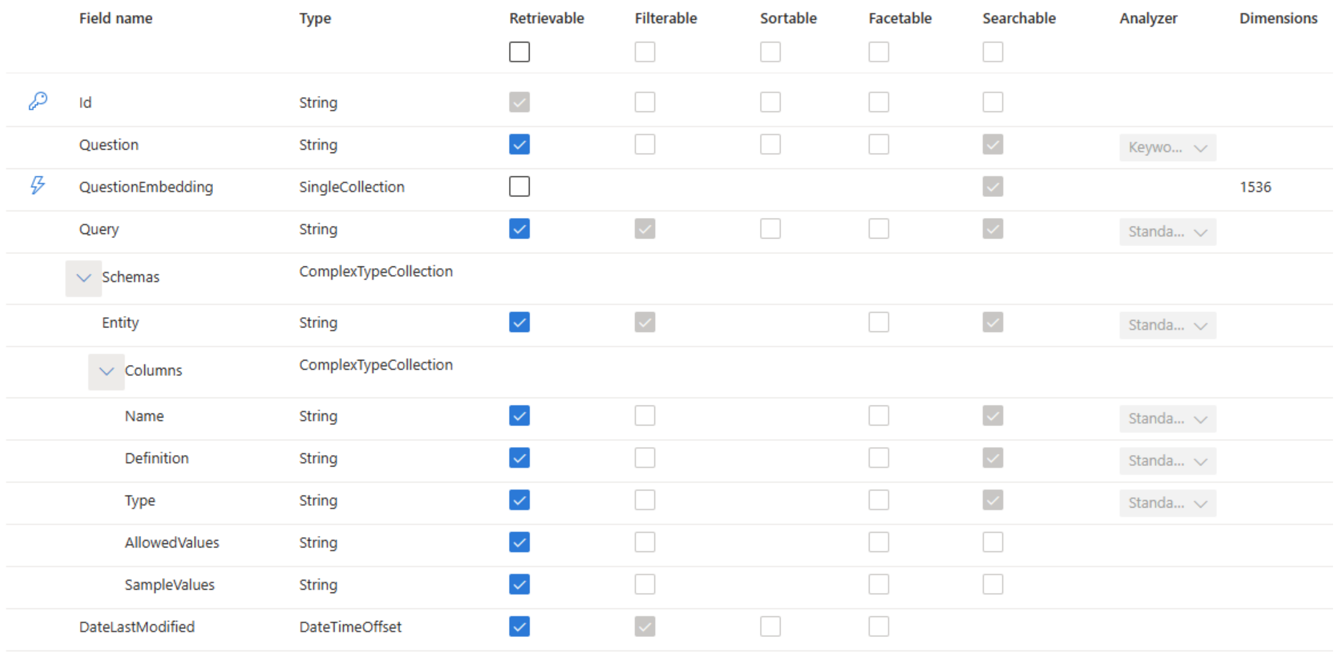


Field name	Type	Retrievable	Filterable	Sortable	Facetable	Searchable	Analyzer	Dimensions
		☐	☐	☐	☐	☐		
Entity	String	✓	☐	☐	☐	✓	Keywo...	
EntityName	String	✓	✓	☐	☐	✓	Standa...	
Description	String	✓	☐	☐	☐	✓	Standa...	
DescriptionEmbedding	SingleCollection	☐				✓		1536
Columns	ComplexTypeCollection							
Name	String	✓	☐		☐	✓	Standa...	
Definition	String	✓	☐		☐	✓	Standa...	
Type	String	✓	☐		☐	✓	Standa...	
AllowedValues	String	✓	☐		☐	☐		
SampleValues	String	✓	☐		☐	☐		
ColumnNames	StringCollection	☐	☐		☐	✓	Standa...	

- Search term of the original question, automatically resolves to the most relevant entities.

Query Cache Index Setup

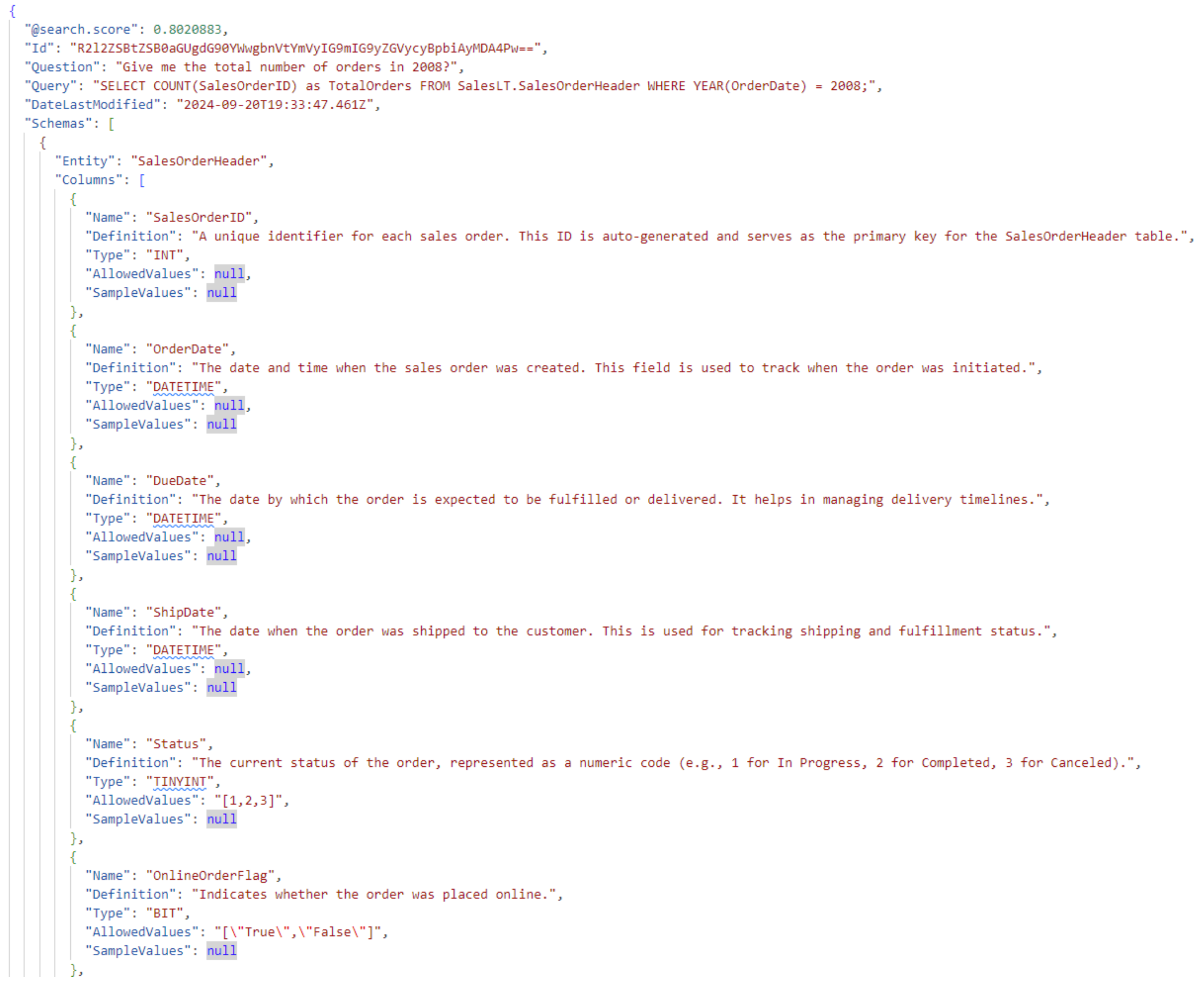
- Stores the user’s question and the queries / schemas it resolves to.
- Vectorises the question property.



Field name	Type	Retrievable	Filterable	Sortable	Facetable	Searchable	Analyzer	Dimensions
		☐	☐	☐	☐	☐		
Id	String	✓	☐	☐	☐	☐		
Question	String	✓	☐	☐	☐	✓	Keywo...	
QuestionEmbedding	SingleCollection	☐				✓		1536
Query	String	✓	✓	☐	☐	✓	Standa...	
Schemas	ComplexTypeCollection							
Entity	String	✓	✓		☐	✓	Standa...	
Columns	ComplexTypeCollection							
Name	String	✓	☐		☐	✓	Standa...	
Definition	String	✓	☐		☐	✓	Standa...	
Type	String	✓	☐		☐	✓	Standa...	
AllowedValues	String	✓	☐		☐	☐		
SampleValues	String	✓	☐		☐	☐		
DateLastModified	DateTimeOffset	✓	✓	☐	☐			

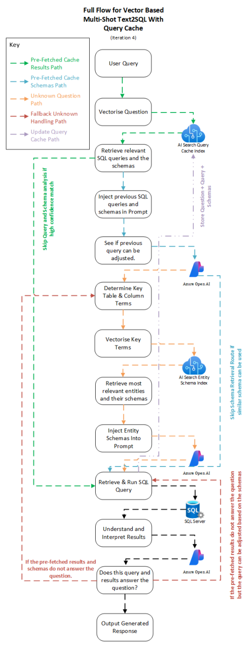
- Ensures consistency that the same question, will always resolve to the same schemas.

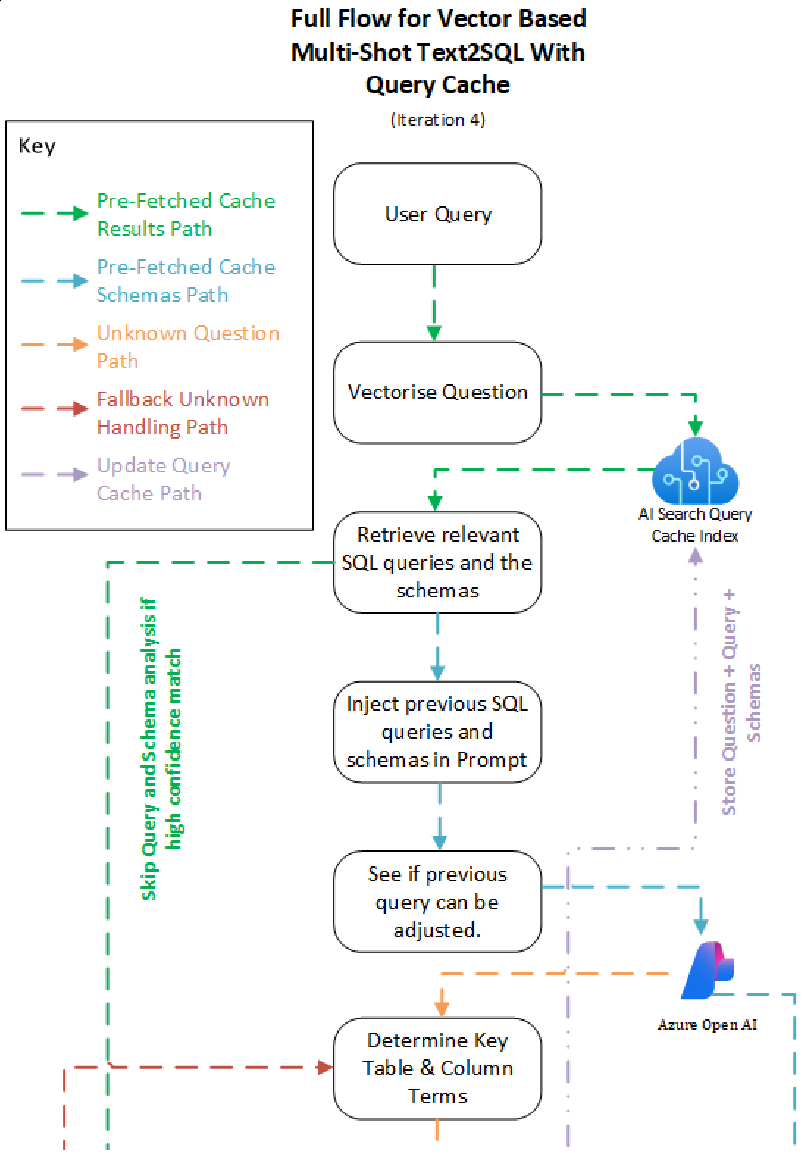
Query Cache Index Setup

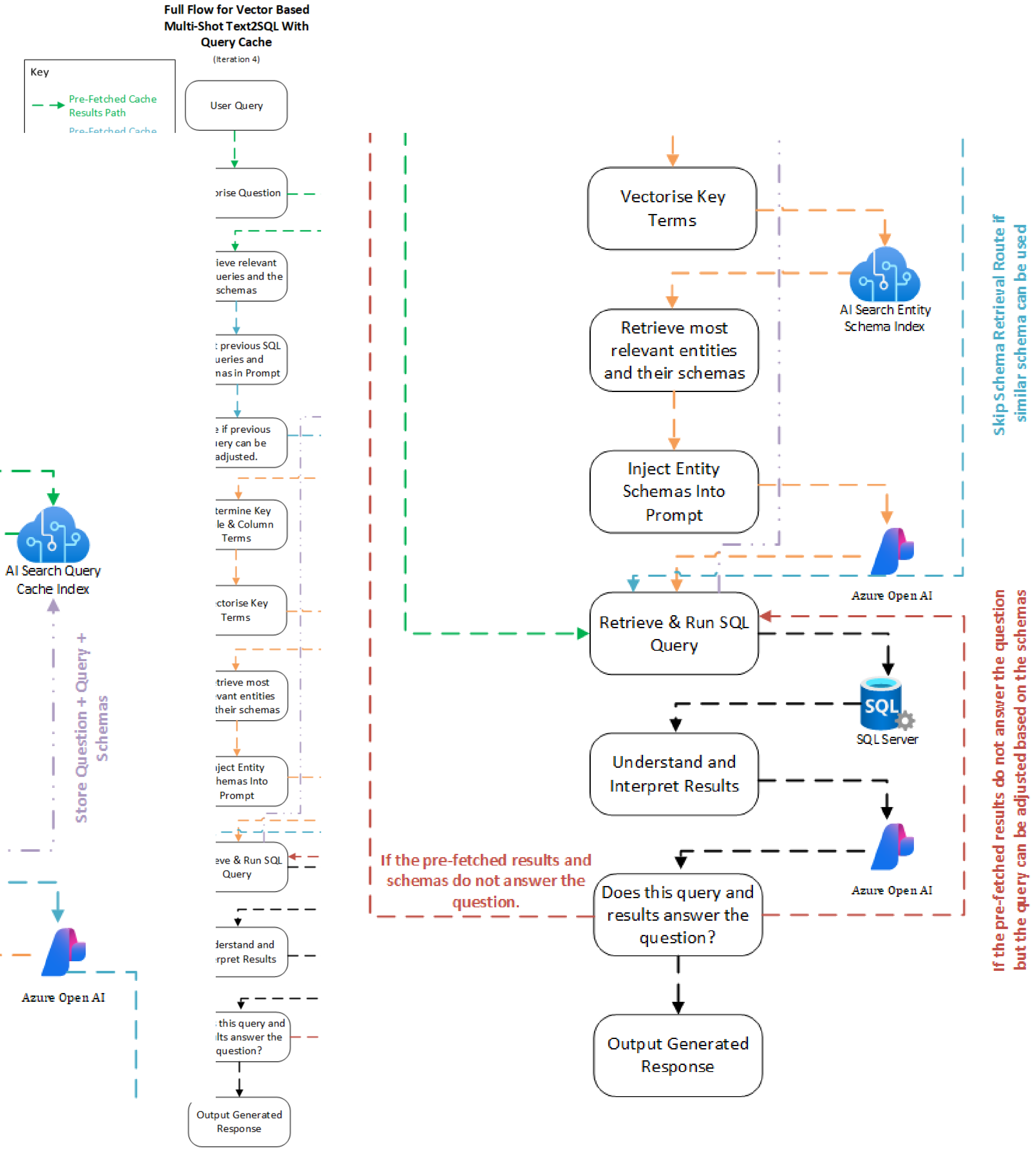


```
{
  "@search.score": 0.8020883,
  "Id": "R2l2ZSBtZSB0aGUgdG90YWwgbnVtYmVyIG9mIG9yZGVycyBpbiAyMDA4Pw==",
  "Question": "Give me the total number of orders in 2008?",
  "Query": "SELECT COUNT(SalesOrderID) as TotalOrders FROM SalesLT.SalesOrderHeader WHERE YEAR(OrderDate) = 2008;",
  "DateLastModified": "2024-09-20T19:33:47.461Z",
  "Schemas": [
    {
      "Entity": "SalesOrderHeader",
      "Columns": [
        {
          "Name": "SalesOrderID",
          "Definition": "A unique identifier for each sales order. This ID is auto-generated and serves as the primary key for the SalesOrderHeader table.",
          "Type": "INT",
          "AllowedValues": null,
          "SampleValues": null
        },
        {
          "Name": "OrderDate",
          "Definition": "The date and time when the sales order was created. This field is used to track when the order was initiated.",
          "Type": "DATETIME",
          "AllowedValues": null,
          "SampleValues": null
        },
        {
          "Name": "DueDate",
          "Definition": "The date by which the order is expected to be fulfilled or delivered. It helps in managing delivery timelines.",
          "Type": "DATETIME",
          "AllowedValues": null,
          "SampleValues": null
        },
        {
          "Name": "ShipDate",
          "Definition": "The date when the order was shipped to the customer. This is used for tracking shipping and fulfillment status.",
          "Type": "DATETIME",
          "AllowedValues": null,
          "SampleValues": null
        },
        {
          "Name": "Status",
          "Definition": "The current status of the order, represented as a numeric code (e.g., 1 for In Progress, 2 for Completed, 3 for Canceled).",
          "Type": "TINYINT",
          "AllowedValues": "[1,2,3]",
          "SampleValues": null
        },
        {
          "Name": "OnlineOrderFlag",
          "Definition": "Indicates whether the order was placed online.",
          "Type": "BIT",
          "AllowedValues": "[\"True\",\"False\"]",
          "SampleValues": null
        },
```

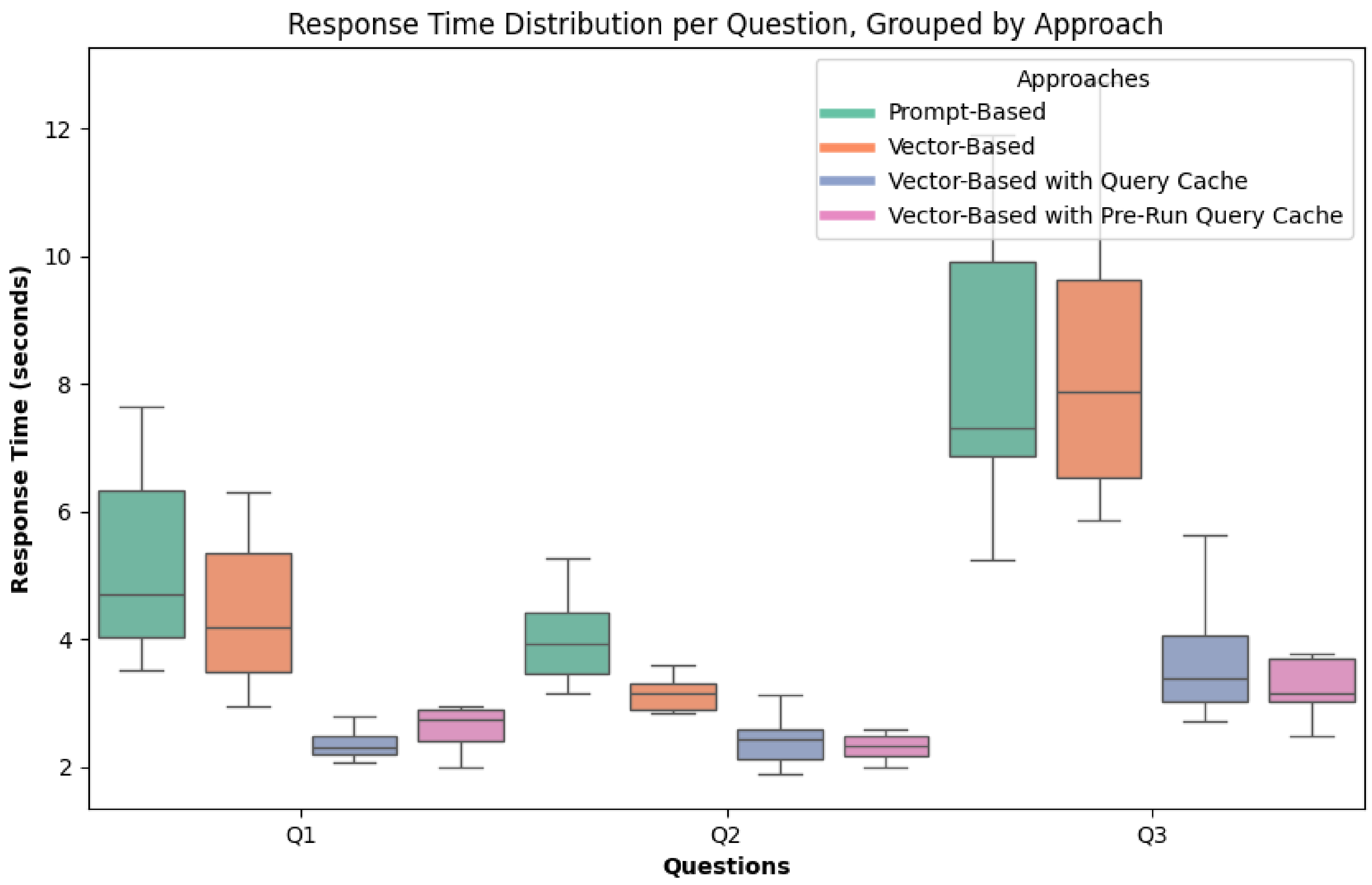
Query Caching




Full Flow for Vector Based Multi-Shot Text2SQL With Query Cache
(Iteration 4)
Key
Pre-Fetched Cache Results Path
Pre-Fetched Cache Schemas Path
Unknown Question Path
Fallback Unknown Handling Path
Update Query Cache Path
User Query
Vectorise Question
AI Search Query Cache Index
Retrieve relevant SQL queries and the schemas
Inject previous SQL queries and schemas in Prompt
See if previous query can be adjusted.
Azure Open AI
Determine Key Table & Column Terms
Skip Query and Schema analysis if high confidence match
Store Question + Query + Schemas


Full Flow for Vector Based Multi-Shot Text2SQL With Query Cache
(Iteration 4)
Key
Pre-Fetched Cache Results Path
User Query
Vectorise Key Terms
AI Search Entity Schema Index
Retrieve most relevant entities and their schemas
Inject Entity Schemas Into Prompt
Azure Open AI
Skip Schema Retrieval Route if similar schema can be used
Retrieve & Run SQL Query
SQL Server
Understand and Interpret Results
Azure Open AI
If the pre-fetched results and schemas do not answer the question.
Does this query and results answer the question?
If the pre-fetched results do not answer the question but the query can be adjusted based on the schemas
Output Generated Response

Timing Comparison



Text2SQL Challenges

- Selecting the best entity / column in cases where multiple columns may be a good match.
  - Similarly named columns may seem relevant for a given question.
- Text representation
  - Different columns are formatted in different ways, achieving accurate SQL statements requires knowledge of the source system representation.
- Performance
  - Initial approach was slow and struggled to scale with multiple tables.

Text2SQL Learnings & Performance Tips

- Pre-assemble views to avoid the LLM having to make complex joins between multiple tables
- Spend time providing detailed names and descriptions in the metadata for all entities and columns e.g.
  - If a column contains a value in each currency, give the currency information in the metadata.
  - Clearly state in the description what sorts of questions a given view / table can provide answers for.
- Use common codes for columns that need filtering e.g.
  - A country can have multiple text representations e.g. United Kingdom or UK.
  - Use ISO codes for countries, instead of text descriptions to increase the likelihood of correct and valid SQL queries.

IP Available

Full Python Code available as part of the Data Science Toolkit:

- Function App for Document Extraction With Azure Document Intelligence
- Text2SQL Plugin for both Prompt & Vector Based Approaches (including cache tweaks)
- Deployment scripts for index / indexer setup

aka.ms/text2sql-adi

Microsoft

Thank you

Industry Solutions | Data & AI