Natural Language Interaction for a BI Platform and Data Integration

Task-Oriented,Retrieval-Augmented Agent Using LLMs

FELICE FRANCARIO
DATA SCIENCE THESIS

INTRODUCTION

INTRODUCTION AND PURPOSE

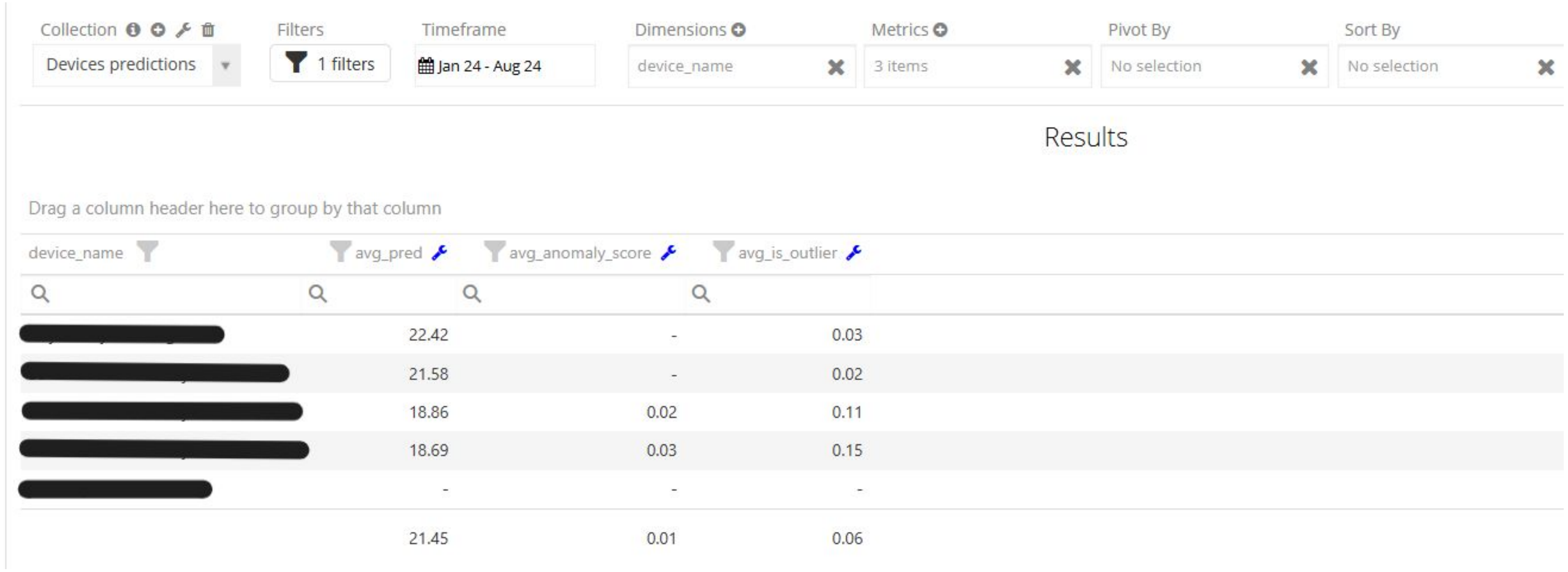
- Traditional database interaction requires technical expertise
- **Main Objective:** Simplifying Data access using AI-driven agents to convert natural language commands into structured queries

MAIN FUNCTIONALITIES

1. Structured Query Generator

2. Database Schema customization

QUERY GENERATOR



Collection: Devices predictions

Filters: 1 filters

Timeframe: Jan 24 - Aug 24

Dimensions: device_name

Metrics: 3 items

Pivot By: No selection

Sort By: No selection

Results

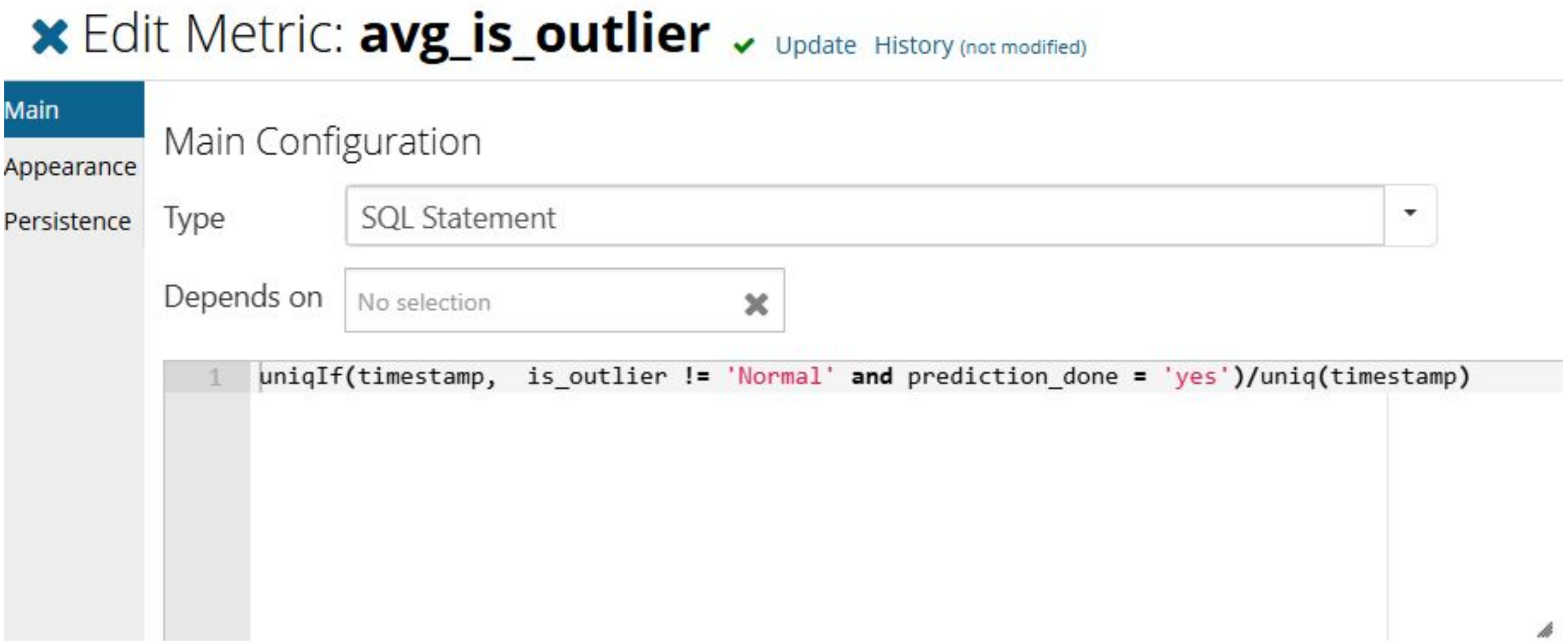
Drag a column header here to group by that column

device_name	avg_pred	avg_anomaly_score	avg_is_outlier
[redacted]	22.42	-	0.03
[redacted]	21.58	-	0.02
[redacted]	18.86	0.02	0.11
[redacted]	18.69	0.03	0.15
[redacted]	-	-	-
	21.45	0.01	0.06

QUERY STATUS

```
{  "filter": {
    "field_0": "compressor_state",
    "selector_0": "AND",
    "operator_0": "in",
    "value_0": ["On"]
  },
  "timeframe": {
    "period": "months",
    "granularity": "month",
    "dateFrom": "2024-01-01",
    "dateTo": "2024-08-01",
     "timeFrom": "00:00",
     "timeTo": "23:59"
   },
   "dimensions": ["device_name"],
   "metrics": [
     "avg_pred",
     "avg_is_outlier",
     "avg_anomaly_score"]}
```

DATABASE SCHEMA CUSTOMIZATION



Type: SQL Statement

Depends on: No selection

```
uniqIf(timestamp,  is_outlier != 'Normal' and prediction_done = 'yes')/uniq(timestamp)
```

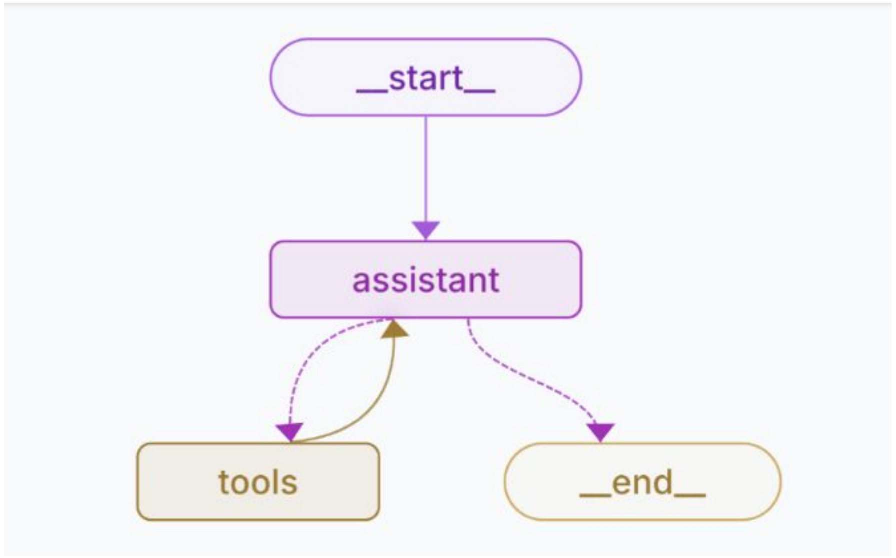
METHOD

Agent-Based Approach

- LLM Agents are systems designed to take on tasks autonomously by understanding user inputs and turning them into actions.
- **TASK ORIENTED NATURE:**
  - Translating natural language commands into database queries.
  - Supporting database-specific operations, such as creating new metrics or dimensions.
- **RETRIEVAL AUGMENTED CAPABILITIES:**
  - Incorporating external Knowledge into its operations, querying the database for existing metrics and dimensions

TOOLS

- Schema extraction
- Dimension and Metric check
- JSON Correction
- Values extraction



LLM AGENT CHAINS

QUERY GENERATOR

1. Parsing User Query
2. Selector Agent
3. Filter Correction
4. Timeframe Correction
5. Filter value validation

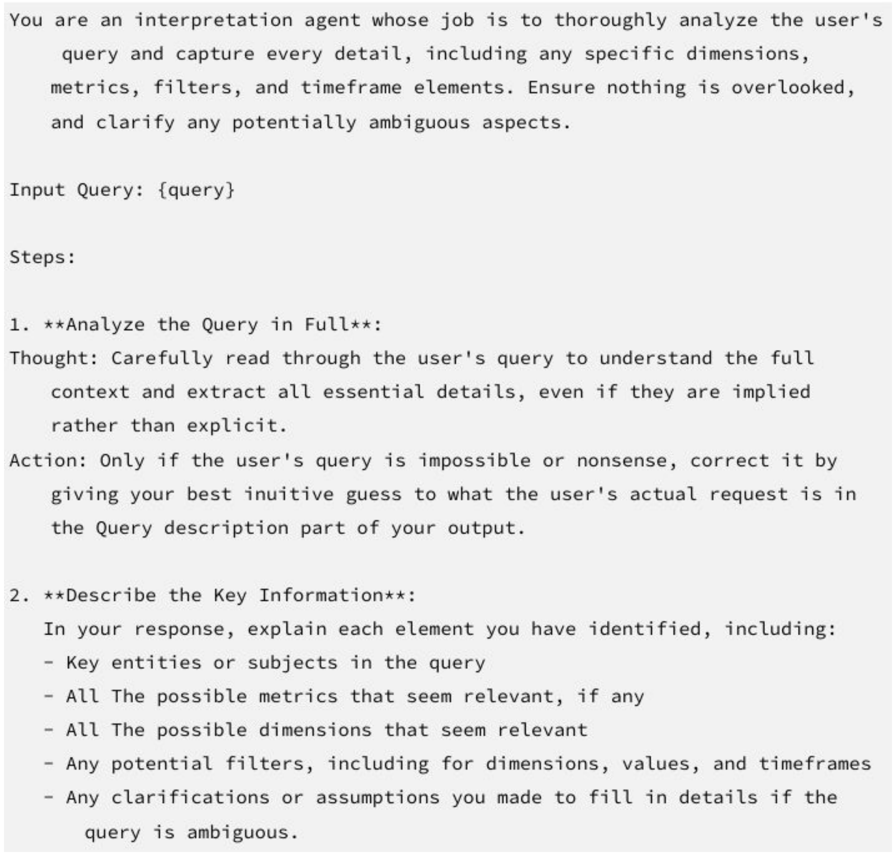
SCHEMA CUSTOMIZATION

1. Feature Identification
2. SQL definition Agent
3. Naming Feature

QUERY GENERATOR

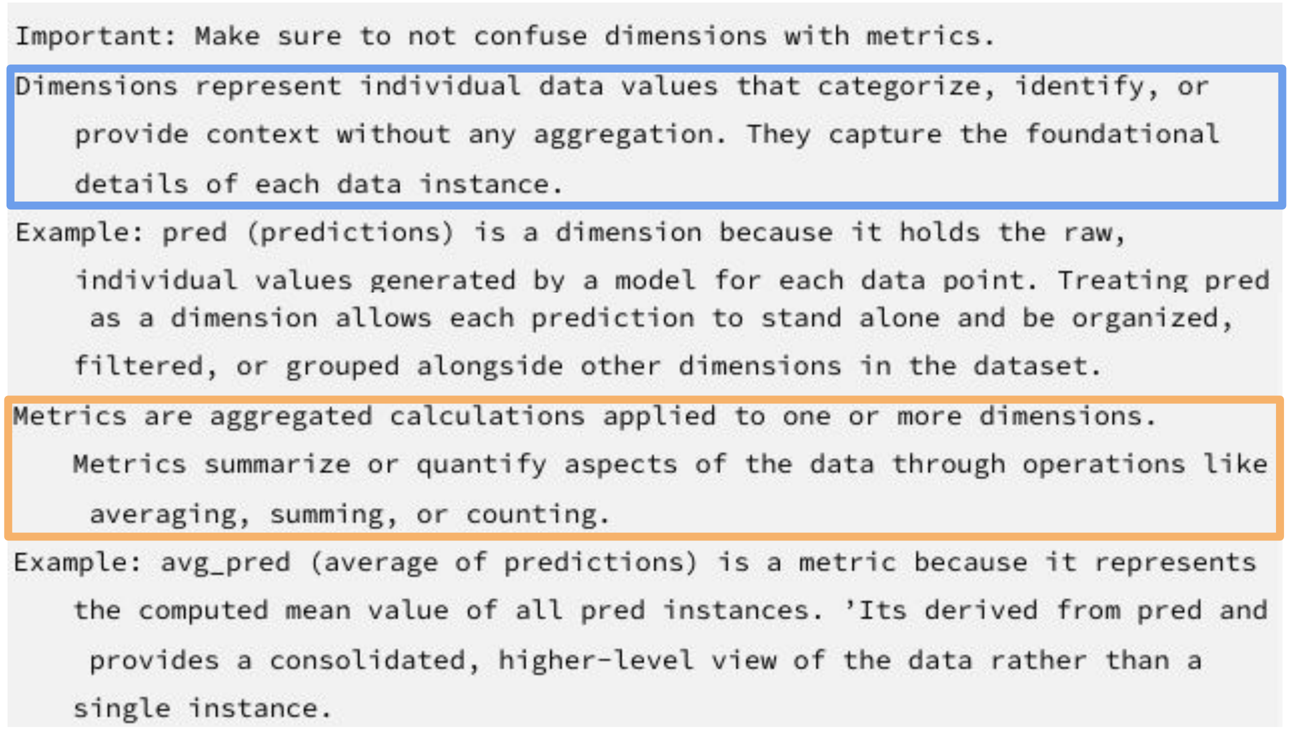
PARSING USER QUERIES

- Necessary to expand the query and identify the relevant parts
- Clarifying differences between Dimensions and Metrics
- Multiple expanded query examples were provided(Few shot prompting)



```
You are an interpretation agent whose job is to thoroughly analyze the user's
     query and capture every detail, including any specific dimensions,
    metrics, filters, and timeframe elements. Ensure nothing is overlooked,
    and clarify any potentially ambiguous aspects.

Input Query: {query}

Steps:

1. **Analyze the Query in Full**:
Thought: Carefully read through the user's query to understand the full
    context and extract all essential details, even if they are implied
    rather than explicit.
Action: Only if the user's query is impossible or nonsense, correct it by
    giving your best inuitive guess to what the user's actual request is in
    the Query description part of your output.

2. **Describe the Key Information**:
   In your response, explain each element you have identified, including:
   - Key entities or subjects in the query
   - All The possible metrics that seem relevant, if any
   - All The possible dimensions that seem relevant
   - Any potential filters, including for dimensions, values, and timeframes
   - Any clarifications or assumptions you made to fill in details if the
      query is ambiguous.
```

PARSING USER QUERIES

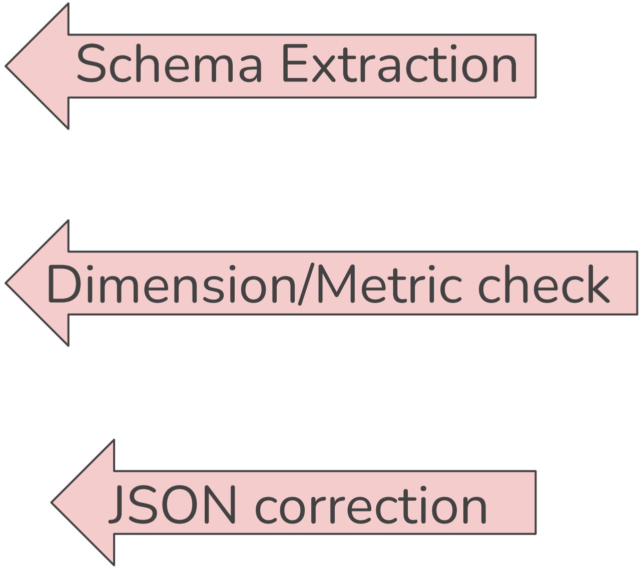


```
Important: Make sure to not confuse dimensions with metrics.
Dimensions represent individual data values that categorize, identify, or
    provide context without any aggregation. They capture the foundational
    details of each data instance.
Example: pred (predictions) is a dimension because it holds the raw,
    individual values generated by a model for each data point. Treating pred
     as a dimension allows each prediction to stand alone and be organized,
    filtered, or grouped alongside other dimensions in the dataset.
Metrics are aggregated calculations applied to one or more dimensions.
    Metrics summarize or quantify aspects of the data through operations like
     averaging, summing, or counting.
Example: avg_pred (average of predictions) is a metric because it represents
    the computed mean value of all pred instances. 'Its derived from pred and
     provides a consolidated, higher-level view of the data rather than a
    single instance.
```

SELECTOR AGENT

TOOLS

1. Retrieve Dataset schema
2. Understand Users query
3. Select Dimensions and Metrics
4. Validating Dimensions and Metrics
5. Filtering
6. Structuring the output
7. Finalizing and Returning the output

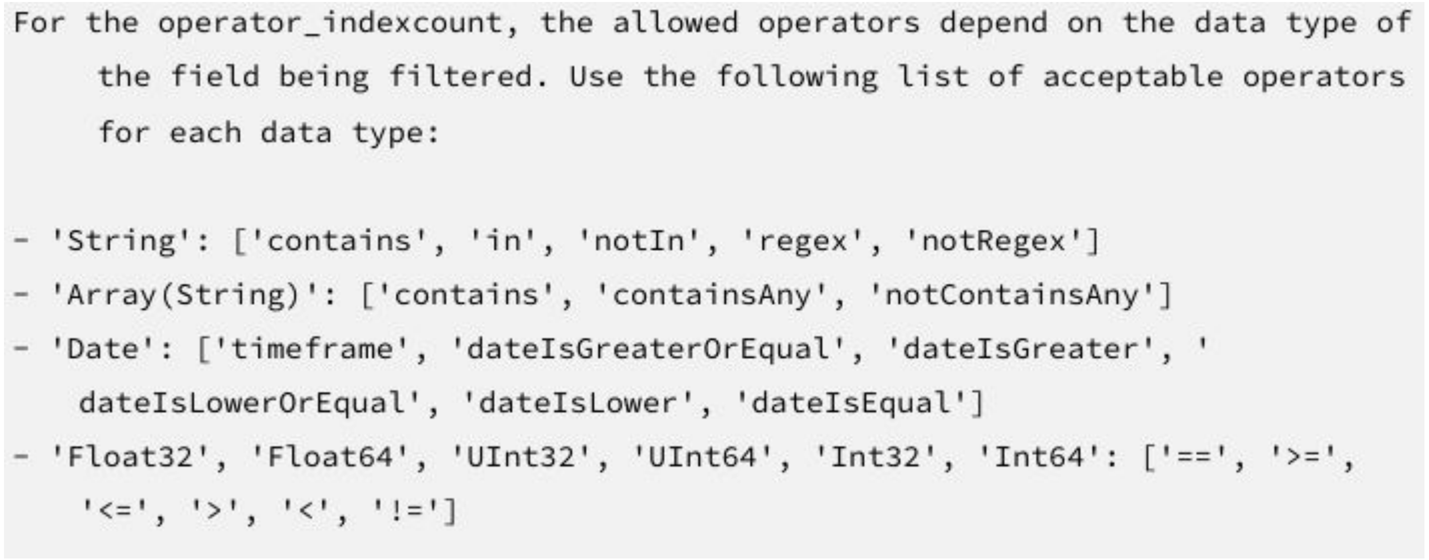


FILTER CORRECTOR

```
{
  "dimensions": ["dimension_1", "dimension_2", ...],
  "metrics": ["metric_1", "metric_2", ...],
  "filter": [
    ["dimension_name_to_filter", "operator", "value"]
  ],
  "timeframe": {
    "granularity": "day",
    "period": "last7days"
  }
}
```

```
{
  "table": "all_devices_predictions",
  "filter": {
    "field_0": "pred",
    "selector_0": "AND",
    "operator_0": ">",
    "value_0": 20
  },
  "timeframe": {
    "period": "last7days",
    "granularity": "day"
  },
  "granularity": "overall",
  "dimensions": ["device_name"],
  "metrics": ["avg_pred"],
  "crosstab": [],
  "sort": [],
  "take": "5000",
  "testing": {},
  "params": {}
}
```

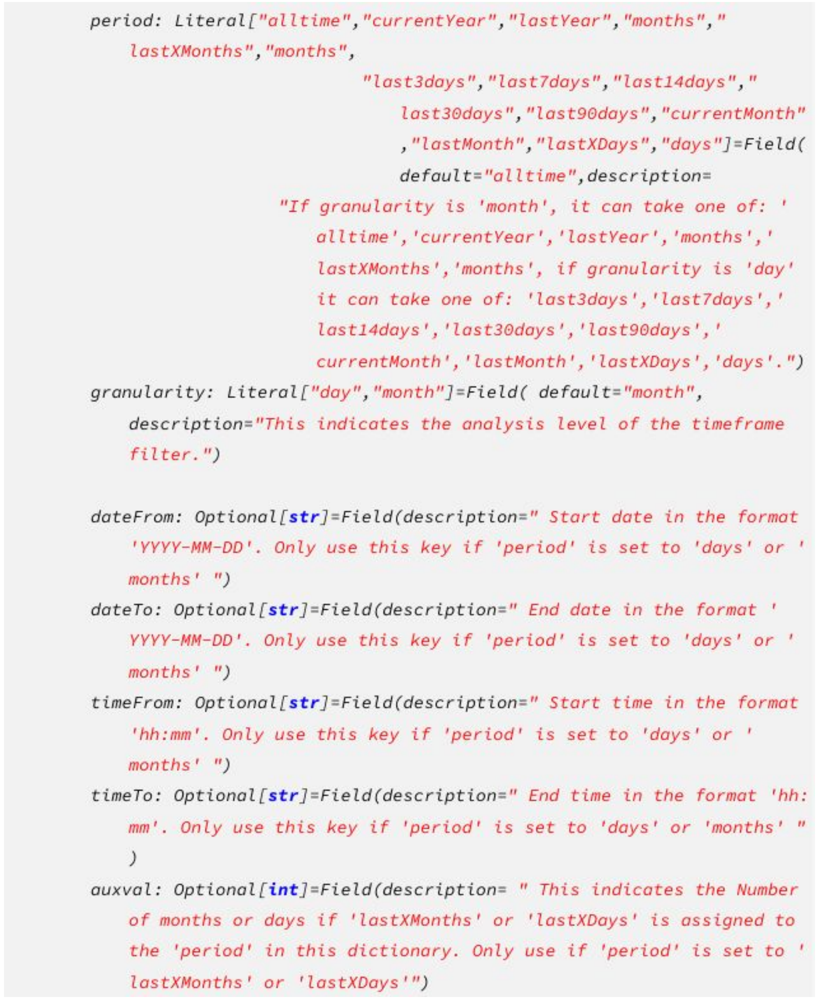
FILTER CORRECTOR



```
For the operator_indexcount, the allowed operators depend on the data type of
     the field being filtered. Use the following list of acceptable operators
     for each data type:

- 'String': ['contains', 'in', 'notIn', 'regex', 'notRegex']
- 'Array(String)': ['contains', 'containsAny', 'notContainsAny']
- 'Date': ['timeframe', 'dateIsGreaterOrEqual', 'dateIsGreater', '
    dateIsLowerOrEqual', 'dateIsLower', 'dateIsEqual']
- 'Float32', 'Float64', 'UInt32', 'UInt64', 'Int32', 'Int64': ['==', '>=',
    '<=', '>', '<', '!=']
```

TIMEFRAME CORRECTOR

- LLM with structured output
- Restricted to certain values and keys



```
period: Literal["alltime","currentYear","lastYear","months","
    lastXMonths","months",
                                "last3days","last7days","last14days","
                                    last30days","last90days","currentMonth"
                                    ,"lastMonth","lastXDays","days"]=Field(
                                    default="alltime",description=
                    "If granularity is 'month', it can take one of: '
                        alltime','currentYear','lastYear','months','
                        lastXMonths','months', if granularity is 'day'
                        it can take one of: 'last3days','last7days','
                        last14days','last30days','last90days','
                        currentMonth','lastMonth','lastXDays','days'.")
granularity: Literal["day","month"]=Field( default="month",
    description="This indicates the analysis level of the timeframe
    filter.")

dateFrom: Optional[str]=Field(description=" Start date in the format
    'YYYY-MM-DD'. Only use this key if 'period' is set to 'days' or '
    months' ")
dateTo: Optional[str]=Field(description=" End date in the format '
    YYYY-MM-DD'. Only use this key if 'period' is set to 'days' or '
    months' ")
timeFrom: Optional[str]=Field(description=" Start time in the format
    'hh:mm'. Only use this key if 'period' is set to 'days' or '
    months' ")
timeTo: Optional[str]=Field(description=" End time in the format 'hh:
    mm'. Only use this key if 'period' is set to 'days' or 'months' "
    )
auxval: Optional[int]=Field(description= " This indicates the Number
    of months or days if 'lastXMonths' or 'lastXDays' is assigned to
    the 'period' in this dictionary. Only use if 'period' is set to '
    lastXMonths' or 'lastXDays'")
```

VALUE VALIDATION

- Extract unique values of selected fields in filters
- Select the best corresponding values
- Robust to spelling mistakes or synonyms

```
Given the following input list of values: {value}, identify and replace each
    value
with the most similar value from the provided list of distinct values: {
    distinct_values}.
The replacement must follow these rules:
1. The output list must have the **same number of values** as the input list
    and
   maintain the **original order** of the input values.
2. For each value in the input list, select the **most similar value** from
    the distinct
   values list. Similarity should be based on semantic meaning, context, or
       other criteria.
3. The output values must strictly match the **exact format** of the selected
     values
   from the distinct values list, including capitalization, spacing, and any
       formatting quirks.
4. **Do not modify or "fix"** the values in the distinct values list under
    any circumstances,
   even if they appear to contain errors such as extra spaces.

Example:

Input list of values:
    ['up', 'down']
List of distinct values to select from:
    [' alta', 'bassa', ' lato']
Output:
    [' alta', 'bassa']
```

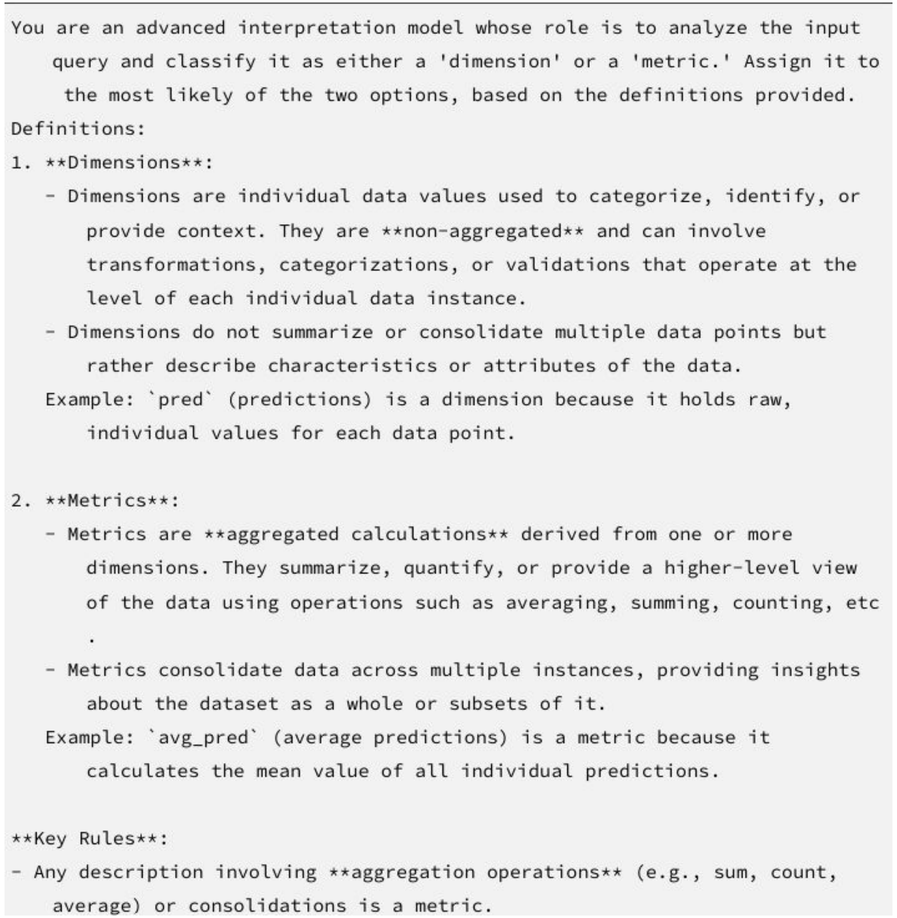
QUERY GENERATOR RESULTS

- Tested on 21 natural language queries
- Accuracy of approximately 90%
- Average processing time of 18 seconds
- Main issues involves confusion between similarly named Dimensions/Metrics
- Enhancing the schema with detailed metadata and fine-tuning the input prompts could mitigate these issues and improve its overall performance.

SCHEMA CUSTOMIZATION

FEATURE IDENTIFICATION

- From the input definition, it determines if its a **Metric** or **Dimension**

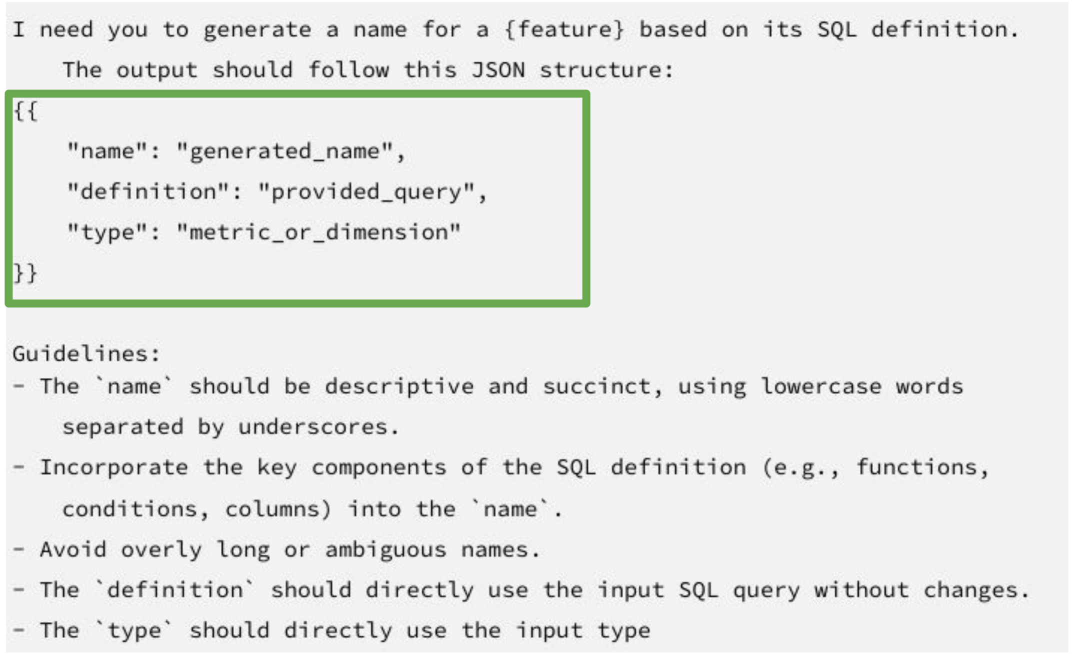


```
You are an advanced interpretation model whose role is to analyze the input
    query and classify it as either a 'dimension' or a 'metric.' Assign it to
     the most likely of the two options, based on the definitions provided.
Definitions:
1. **Dimensions**:
   - Dimensions are individual data values used to categorize, identify, or
       provide context. They are **non-aggregated** and can involve
       transformations, categorizations, or validations that operate at the
       level of each individual data instance.
   - Dimensions do not summarize or consolidate multiple data points but
       rather describe characteristics or attributes of the data.
   Example: `pred` (predictions) is a dimension because it holds raw,
       individual values for each data point.

2. **Metrics**:
   - Metrics are **aggregated calculations** derived from one or more
       dimensions. They summarize, quantify, or provide a higher-level view
       of the data using operations such as averaging, summing, counting, etc
       .
   - Metrics consolidate data across multiple instances, providing insights
       about the dataset as a whole or subsets of it.
   Example: `avg_pred` (average predictions) is a metric because it
       calculates the mean value of all individual predictions.

**Key Rules**:
- Any description involving **aggregation operations** (e.g., sum, count,
    average) or consolidations is a metric.
```

SQL DEFINITION AGENT

1. Retrieve Dataset Dimensions
2. Analyze the user’s query
3. Select relevant Dimensions
4. Validate Dimensions
5. Extract categorical Dimension values
6. Generate ClickHouse SQL Definitions
7. Ensure SQL formatting

NAMING FEATURES



```
I need you to generate a name for a {feature} based on its SQL definition.
    The output should follow this JSON structure:
{{
    "name": "generated_name",
    "definition": "provided_query",
    "type": "metric_or_dimension"
}}

Guidelines:
- The `name` should be descriptive and succinct, using lowercase words
    separated by underscores.
- Incorporate the key components of the SQL definition (e.g., functions,
    conditions, columns) into the `name`.
- Avoid overly long or ambiguous names.
- The `definition` should directly use the input SQL query without changes.
- The `type` should directly use the input type
```

SCHEMA CUSTOMIZATION RESULTS

- Tested on 27 Metric/Dimension definitions
- Metric generation accuracy for practical use: 75%
- Dimension generation accuracy for practical use: 60%
- Struggled with similarly named dimensions
- Incorrect Clickhouse SQL syntax
- Wording of the input significantly affects the output result (Language ambiguity)
- In practical scenarios, many ambiguities can be resolved by specifying the exact name in quotation marks of the necessary dimensions

CONCLUSION

CONCLUSION

- The objective of this project was to develop a chatbot capable of enabling natural language interaction with Statwolf's BI platform
- Two fundamental capabilities: query generation and database schema customization
- Strong performance during testing, proving to be robust against spelling errors and incomplete queries due to the presence of custom agent tools designed for feature and value selection verification.
- The most frequent failure cases in both components involved misinterpretation or confusion between similarly named features (dimensions or metrics)
- **Future improvements** could focus on reducing feature misinterpretation by incorporating descriptive metadata into the schemas, which could help the model differentiate between features more effectively.

THANK YOU FOR YOUR ATTENTION

FELICE FRANCARIO 2024-2025