Xpert: Empowering Incident Management with Query Recommendations via Large Language Models

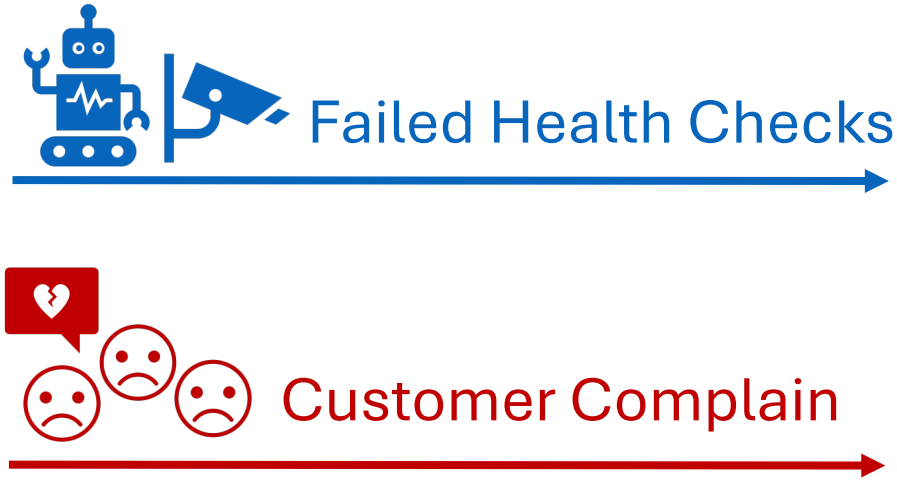
Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang

@_MattJiang_

ICSE'24 @ Lisbon

jyuxuan@umich.edu

Cloud Incidents Produce Incident Tickets



Incident Tickets

Writing Queries is Crucial For Resolution

Incident Tickets

On-call Engineers (OCE)

KQL Query

Database table to query from

```
MonRdmsInstanceAgent
| where LogicalServerName == “fpgsqlserver”
| where msg_metadata contains “OpenNewConnection”
| project TIMESTAMP, msg_metadata, event, severity
| order by TIMESTAMP desc
```

Filtering operation

Telemetry Data (Traces, Logs, Metrics)

Diagnosis & Solution

Writing Correct Queries is Challenging

Task: Finding relevant data in **a haystack of databases.**

OCE

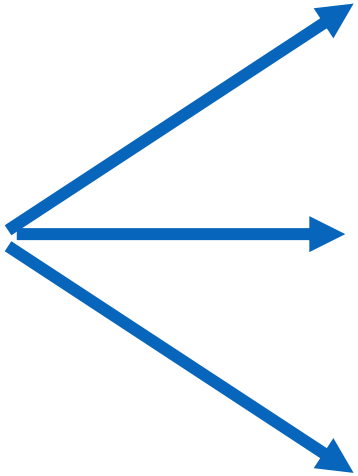
Telemetry Data

```
MonRdmsInstanceAgent
| where LogicalServerName == “fpgsqlserver”
| where msg_metadata contains “OpenNewConnection”
| project TIMESTAMP, msg_metadata, event, severity
| order by TIMESTAMP desc
```

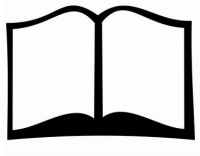
Writing Correct Queries is Challenging

Task: Finding relevant data in **a haystack of databases.**

OCE



Domain Experience



Troubleshooting Guide (TSG)

External Help

```
MonRdmsInstanceAgent
| where LogicalServerName == “fpgsqlserver”
| where msg_metadata contains “OpenNewConnection”
| project TIMESTAMP, msg_metadata, event, severity
| order by TIMESTAMP desc
```

Generating Queries is Immensely Helpful

- ~ 500 incidents daily on Azure for Top-30 services

New Incident Ticket

ID XXX	Title
ACTIVE	Summary
	Discussion
Severity x	...

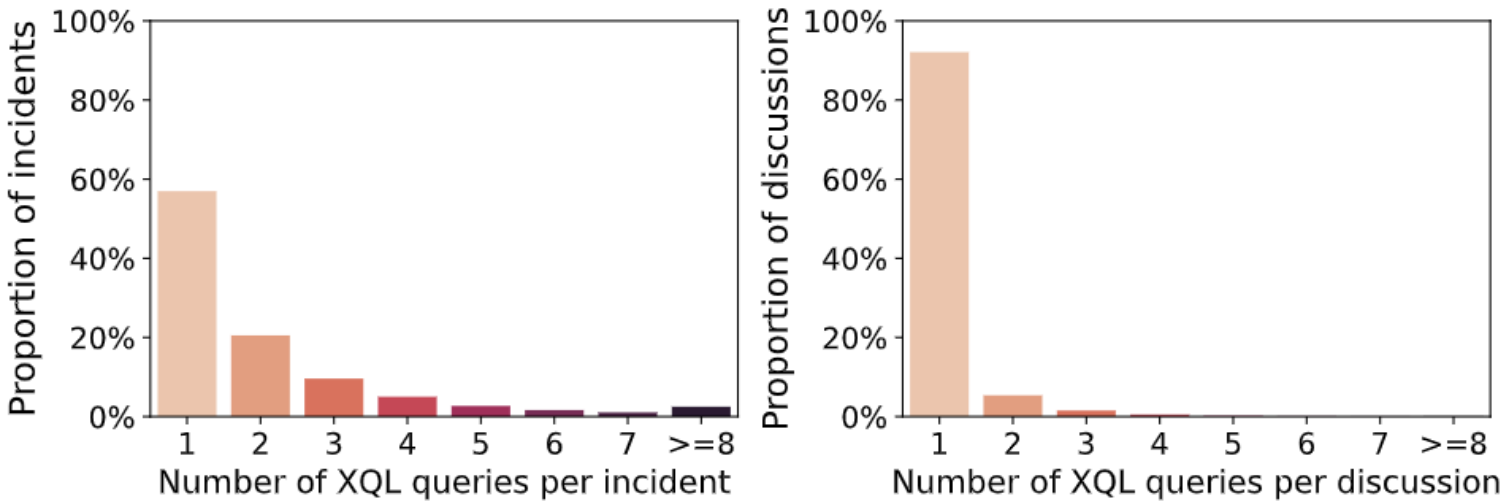
Automation?

```
MonRdmsInstanceAgent
| where LogicalServerName == “fpgsqlserver”
| where msg_metadata contains “OpenNewConnection”
| project TIMESTAMP, msg_metadata, event, severity
| order by TIMESTAMP desc
```

How to Generate Queries?

- What input do we need?
- Number of queries to generate for each incident?
- How to guarantee queries are executable?
- …

RQ1: Frequency of Queries



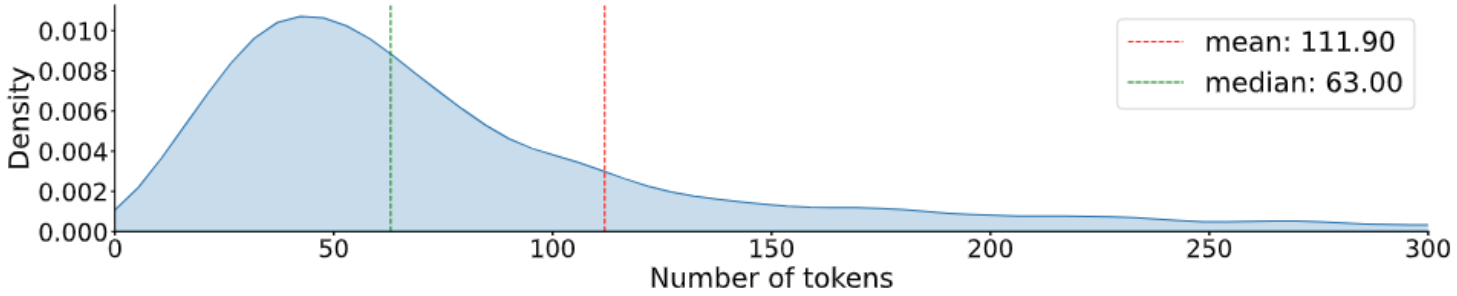
Study Result:

- Over half of the incidents have only one Kusto query.
- Over 90% of the discussions within an incident contain just one query.

Design Implication:

- **Recommending one query suffices.**

RQ2: Complexity of Queries



64 tokens:

```
VMApiQosEvent
| where PreciseTimeStamp >= ago(1d)
| where errorDetails contains "AllocationError"
| where MonitoringApplication contains "Service A"
| project PreciseTimeStamp, operationId, vMSize, errorDetails
| order by PreciseTimeStamp desc
| take 10
```

Study Result:

- Median of tokens per query is 63, average tokens are ~112.
- Majority of incidents are managed using relatively concise Kusto queries.

Design Implication:

- **Query generation can be feasible.**

RQ3: Diversity of Queries

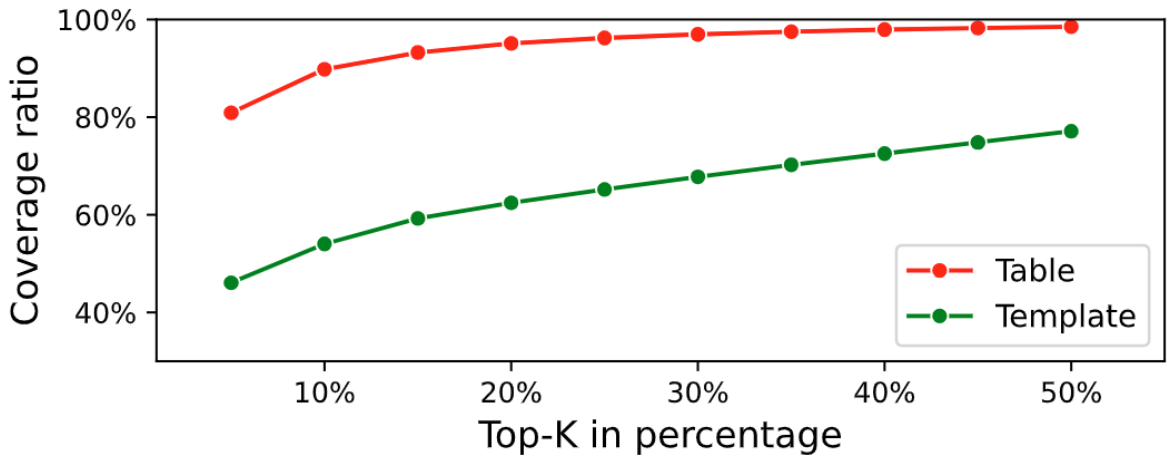
Table

XQL Query

```
VMNodeMeta
| where date > ago(1d) and date < now()
| where region == "west us"
| join kind = leftouter
LatencyEvolution on NodeId
| summarize AvgDelay = avg(delay) by NodeId
| where AvgDelay > 1
| order by AvgDelay
| take 10
```

XQL Query Template

```
VMNodeMeta
| where date > ago({timespan}) and date < {time}
| where region == {string}
| join kind = leftouter
LatencyEvolution on NodeId
| summarize AvgDelay = avg(delay) by NodeId
| where AvgDelay > {int}
| order by AvgDelay
| take {int}
```



Study Result:

- KQL usage exhibit low-diversity.
  - 80.9% queries use 5% of all tables.
  - 46.1% queries consist of 5% unique templates.
- KQL usage differs across services.

Design Implication:

- **Use similar incidents' queries to guide query generation.**

RQ3: Diversity of Queries

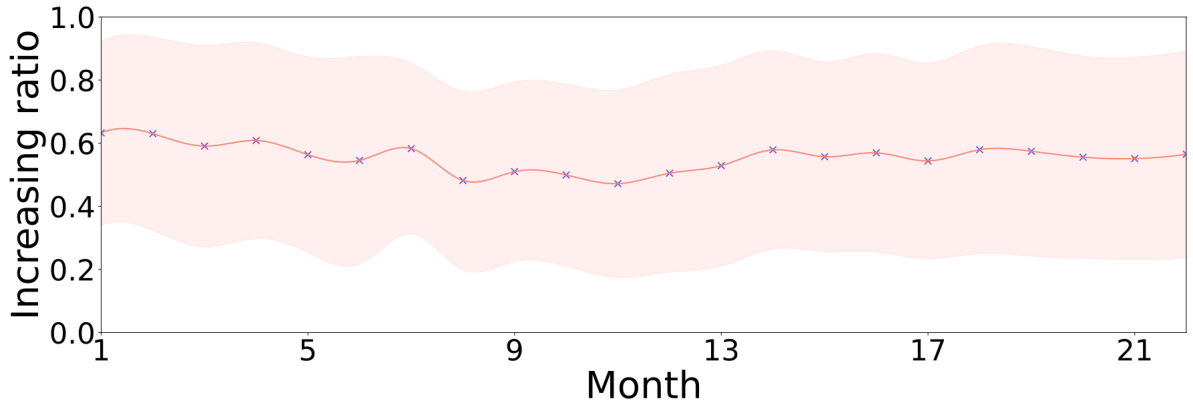


Figure 5: The mean±std. of the monthly ratio of KQL queries covered by novel templates across all services.

Study Result:

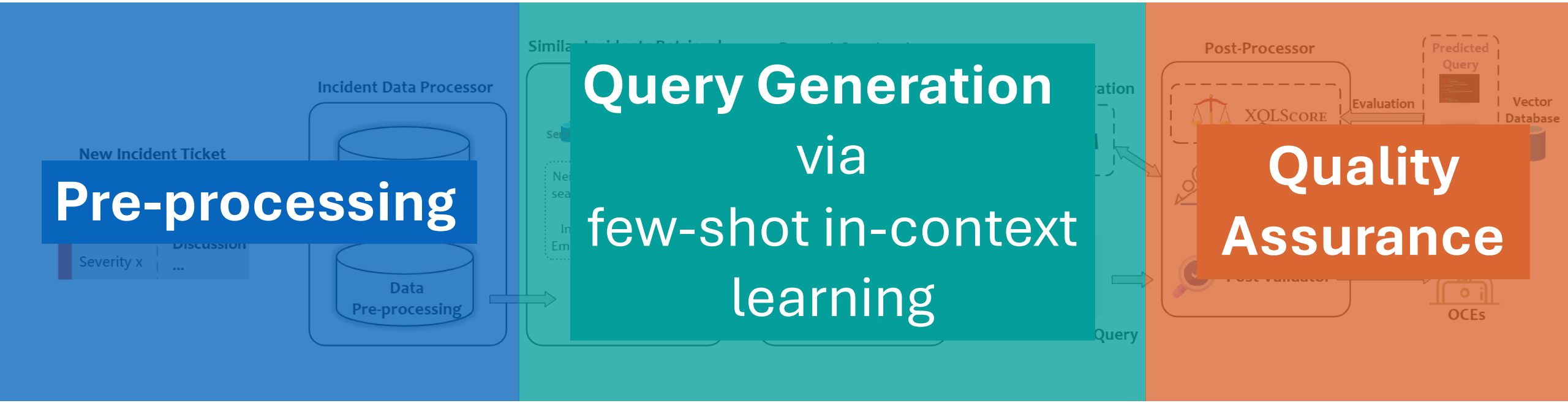
- KQL usage change significantly over time (>60% per month).

Design Implication:

- **Model training/finetuning might not be feasible.**

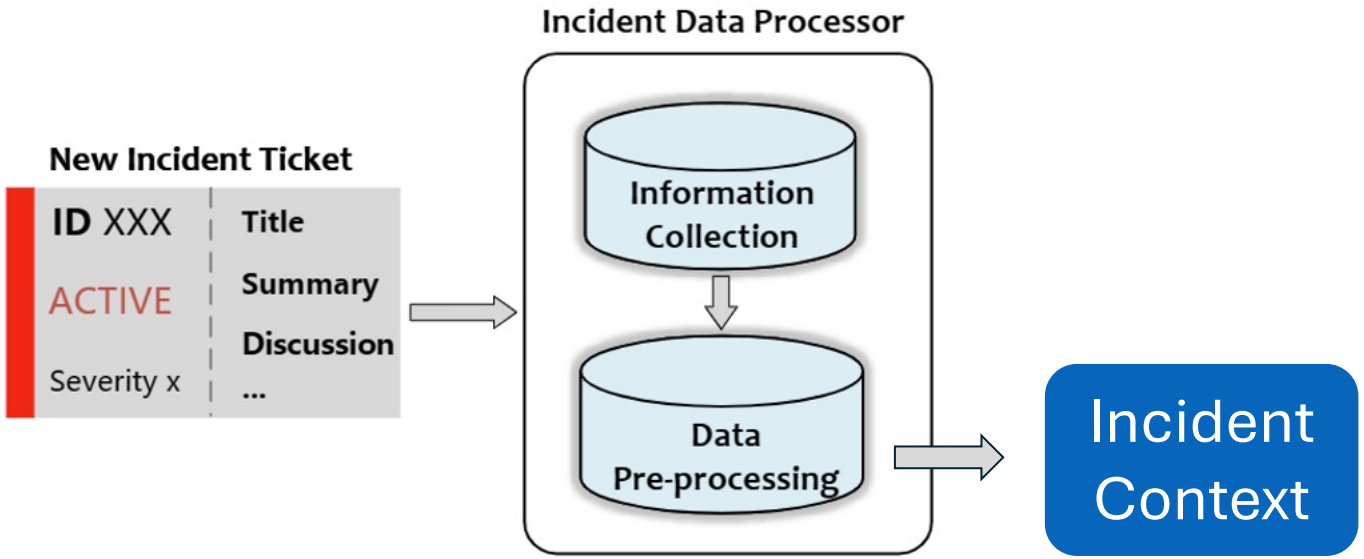
Solution

Xpert: An End-to-End Query Generation Framework



- **End-to-End Automated Query Generation**
- **Pattern Extraction from Abundant Historical Incidents**
- **Customized Recommendations for New Incidents**

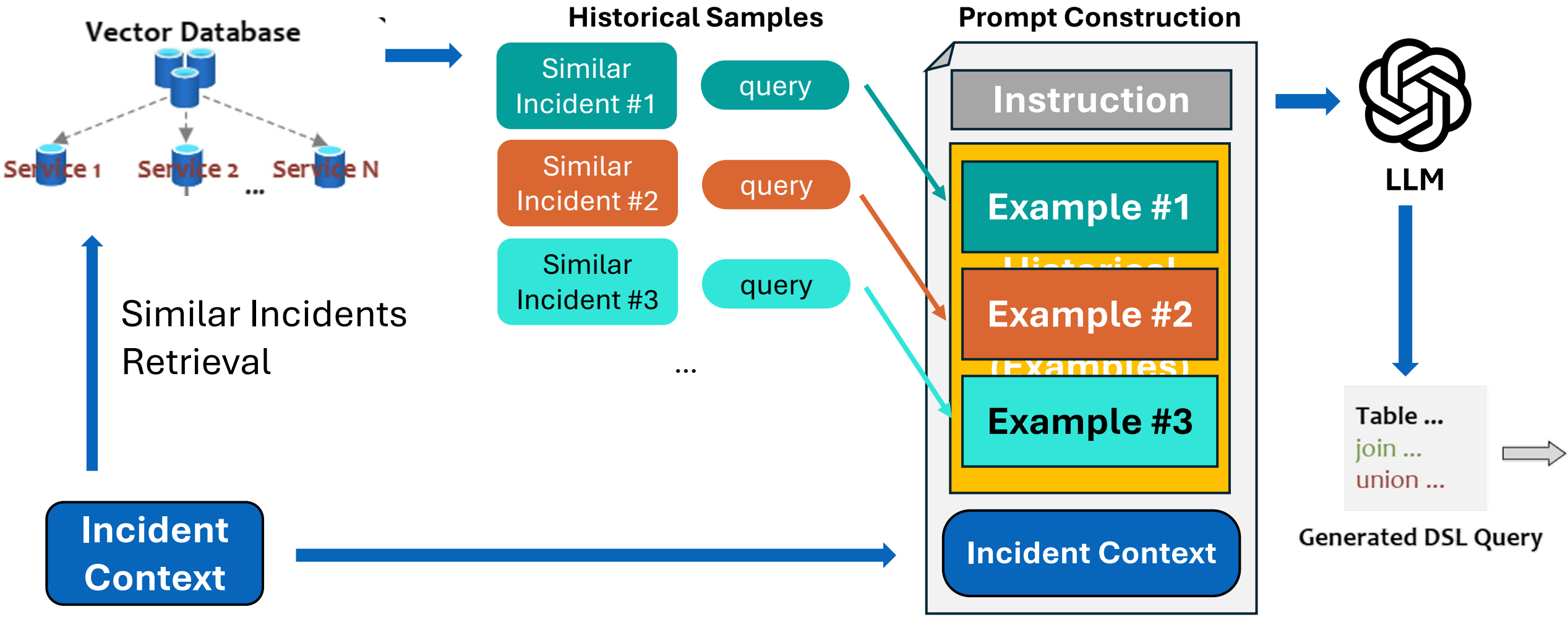
Data Pre-Processor



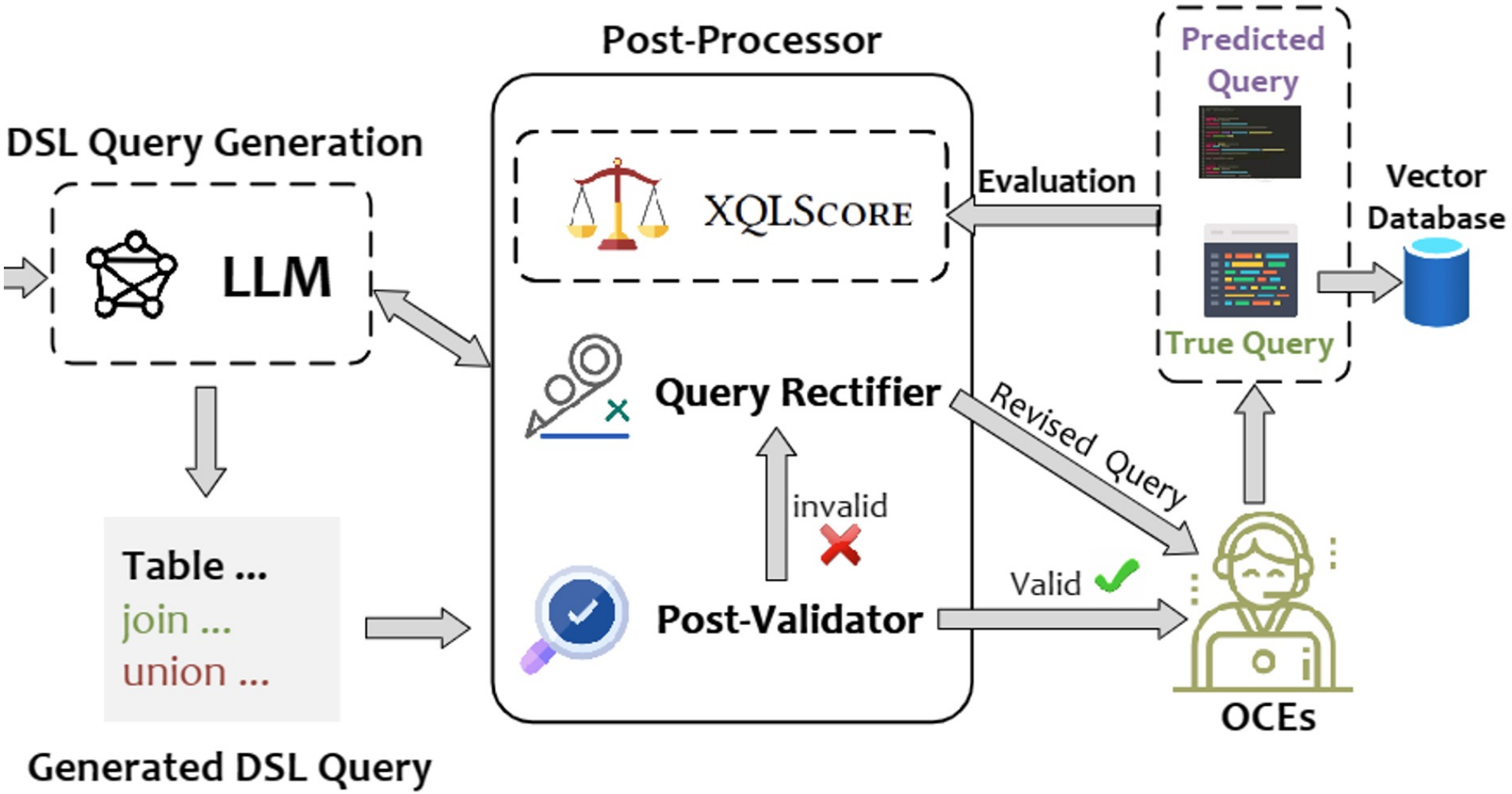
- **Metadata**
  - Timestamp
  - Error Message
- **Title**
- **Summary**
  - Impact of outage
- **Discussion**
  - Attempted diagnosis
  - Additional outage description

Query Generation: Few-shot in-context learning

Design Implication: Use similar incidents' queries as reference.



Post Processor



Evaluating Generated Queries Without Execution

	Ground Truth	Generated Sample
1	Errors	JobErrors
2	| where TIMESTAMP >= datetime(2022-03-27 22:04:06Z)	| where TIMESTAMP < datetime(2022-03-27 22:04:06Z)
3	| where SourceNamespace contains “us-east”	| where SourceNamespace contains “us-east”
4	| where Deployment == “svc-mesh-3d21”	| where Deployment = “svc-mesh-3d21”
5	| summarize count() by subscriptionId	| summarize count(subscriptionId)

Tables do not match

`<` is opposite to `>=`

Invalid op `=`

Effectively two different statements

- NLP metrics overlook syntax & semantic properties in code.
  - BLEU: 75.67, METEOR: 87.02 → **The generated query seems to be very good!**

Syntax & semantics should be considered in evaluation.

Evaluation with Xcore

Xcore: 3.54 for the previous example

Syntax and Semantic Check

Ensure query is executable

Examples:

```
| project colB, colC
| sort by colA asc
```

All columns except for **colB**, **colC** are excluded from the data flow.

Sub-component Matching

Canonicalize query for lexical comparison

```
where A and B
```

and

```
where B and A
```

should be considered equivalent

Output Schema Matching

Approximate execution result

```
Errors
| where TIMESTAMP >= ago(1d)
| where Name == “svc-3d21”
| project Name, FailureReason
```

would have output schema

Column	**Name**	**FailureReason**
DType	**String**	**Unknown**

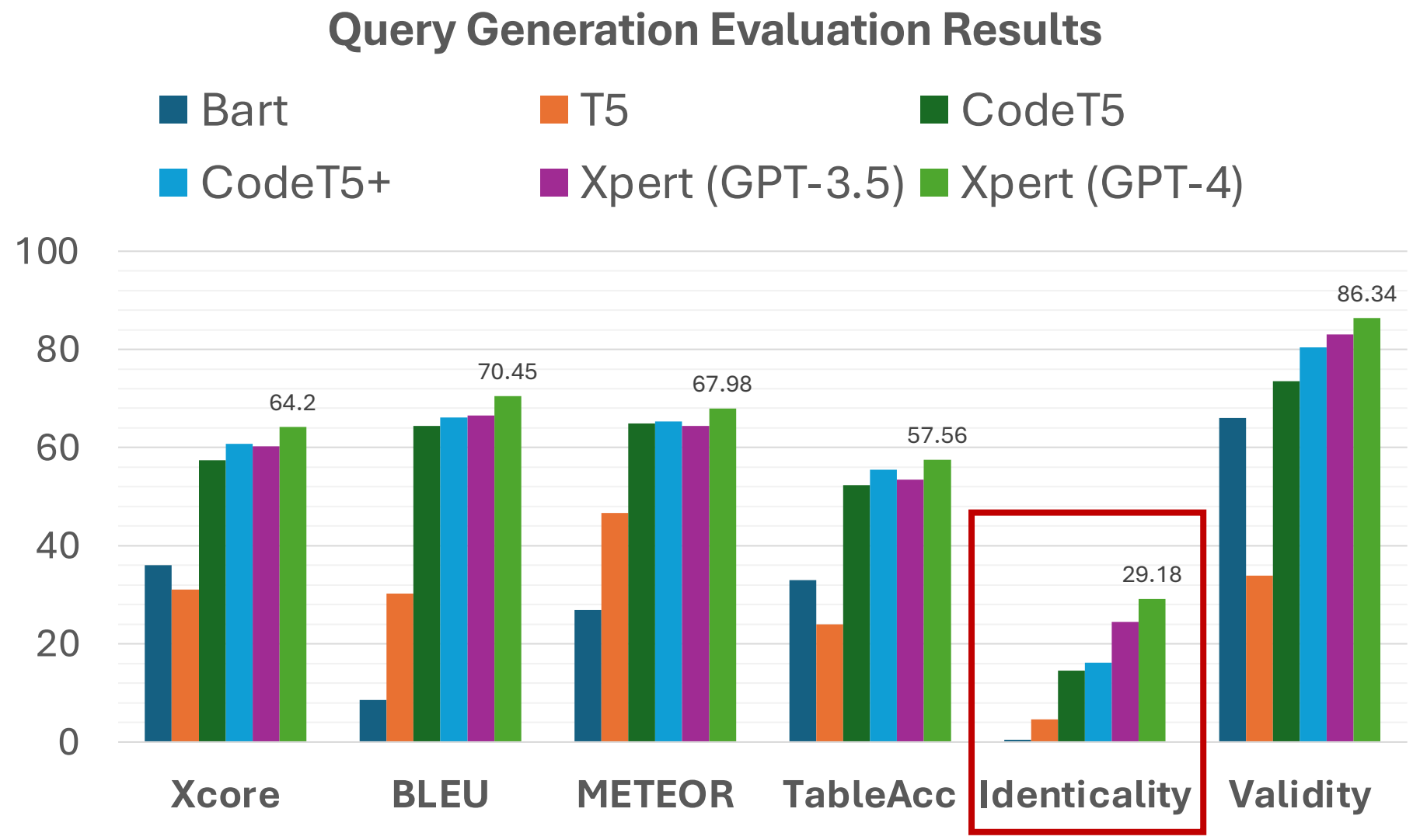
Evaluation: Xpert

- **Baseline Models**: Finetuned transformer models (T5, Bart, CodeT5, CodeT5+)
- **Dataset**: ~200,000 samples for retrieval; 3,000 samples tested across top-10 tenants

- **Task:**
  Generating queries for the 3000 samples.
  - Xpert will use **historical incidents for reference**, while
  - baselines are **finetuned** and perform **zero-shot generation**.

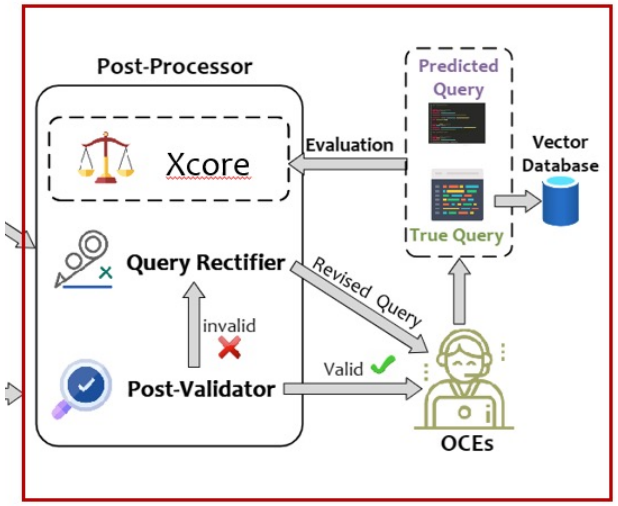
Solutions are assessed by comparing the generated query with the ground truth.

Evaluation: Overall Performance

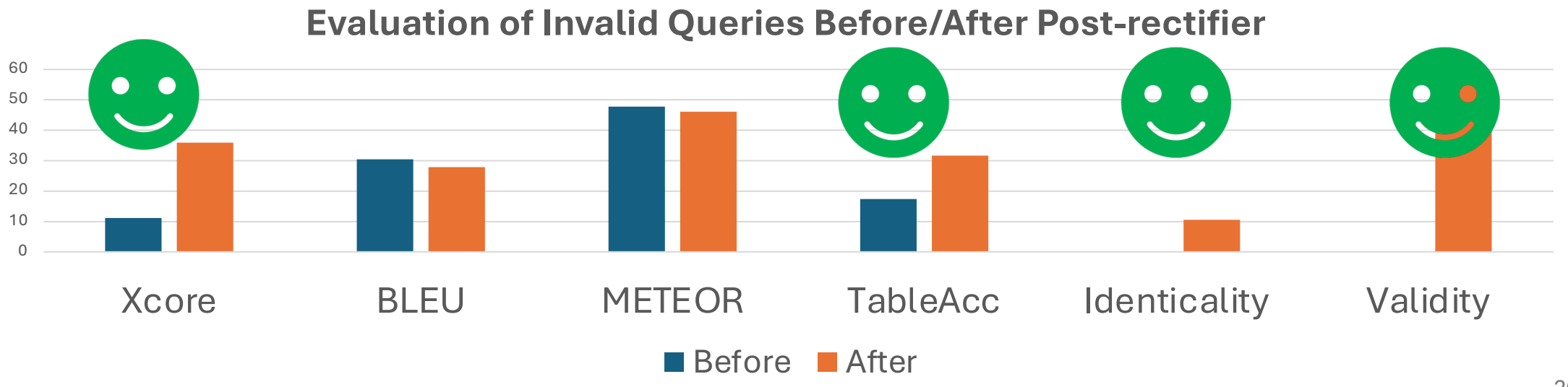


- **Xpert (GPT-4) is the best across all metrics**
- **Xpert (GPT-3.5) are comparable with CodeT5+**
- **Greater advantage on identicality:**
  - Xpert: 29.19%
  - Finetuned Models: 16.18%

Evaluation: Post-processor



- Post-rectifier improves the prediction quality for most of performance dimensions
- Refining over **50%** queries to valid



Summarizing Xpert

- **Pioneering Empirical Study**: Investigating the application of query languages in incident management systems.
- **Innovative Framework**: Introducing *Xpert* for automated generation of incident management queries.
- **Novel Metric**: *Xcore*, assessing query quality independently of the execution environment.
- **Proven Deployment**: Validating *Xpert's* effectiveness through practical implementation in a real-world setting.

Related Works

- **Text2SQL Adaptation:**
  - Moves beyond direct natural language translation.
  - Tackles unstructured, noisy input interpretation.
  - Enhances Text2SQL to aid in root cause analysis from symptomatic data.
  - Showcases the utility of in-context learning for complex queries.

- **Advancements in AIOps:**
  - Aligns with AI-driven IT operational enhancements.
  - Demonstrates AI's role in streamlining incident management.