PYTHON.ID SHARING SESSION

BUILDING RAG ON LARGE DOCUMENT USING OPEN SOURCE LLM

Alibaba Cloud

indigo

INTRODUCTION

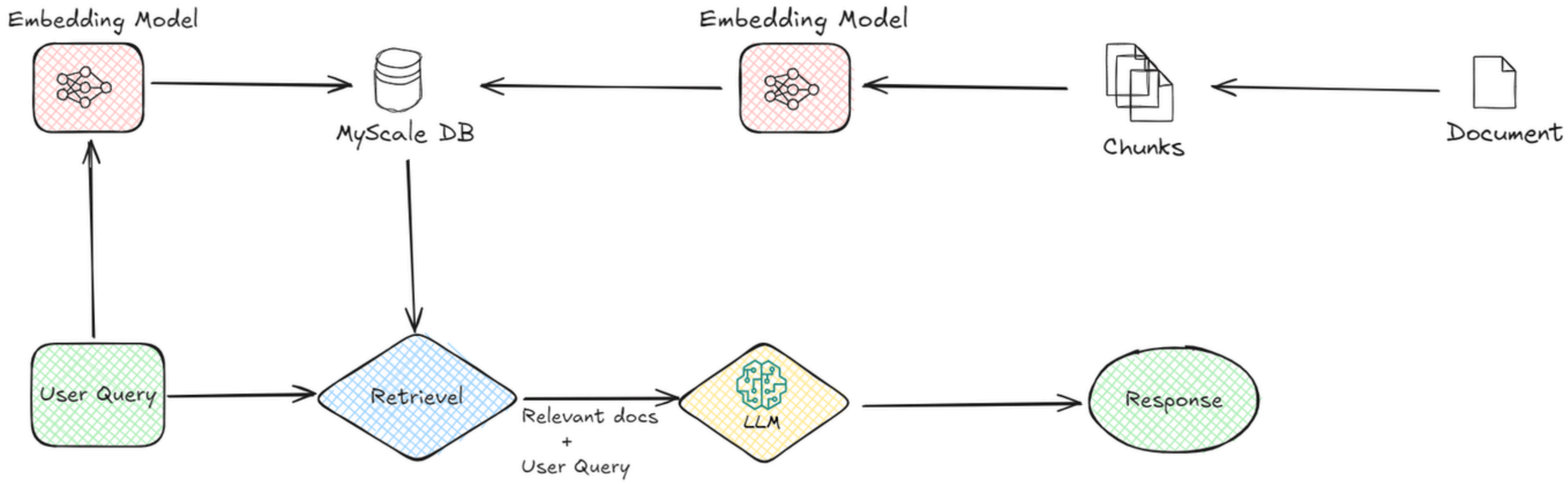
M. ALIF RAMADHAN

AI Engineer | **Pecinta AI**

LATEST ACTIVITY

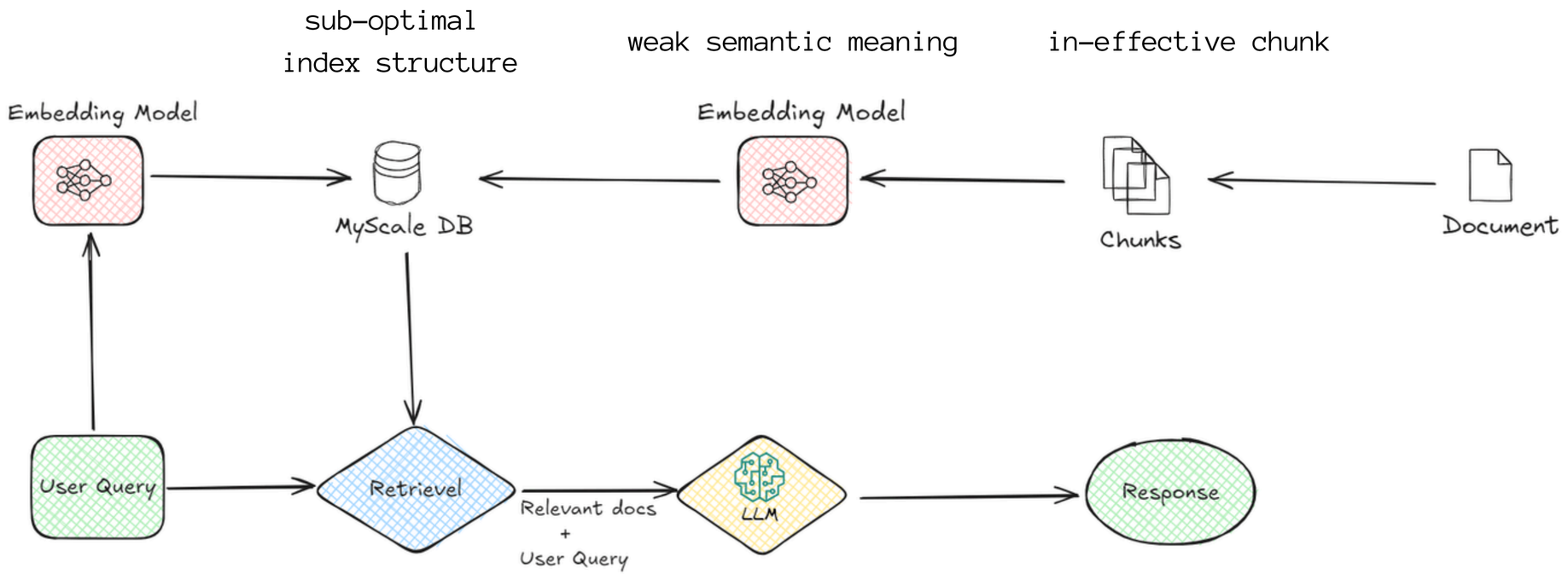
- Mahasiswa Teknik Elektro
- Built community on facebook: IMPHNEN
- Make other people’s tasks easier with Gen. AI

RAG: **SOLUTION** FOR AI **HALLUCINATION?**
OR **ANOTHER ISSUE?**



source - https://myscale.com/blog/naive-rag-vs-advanced-rag

HIGHLIGHT THE ISSUE



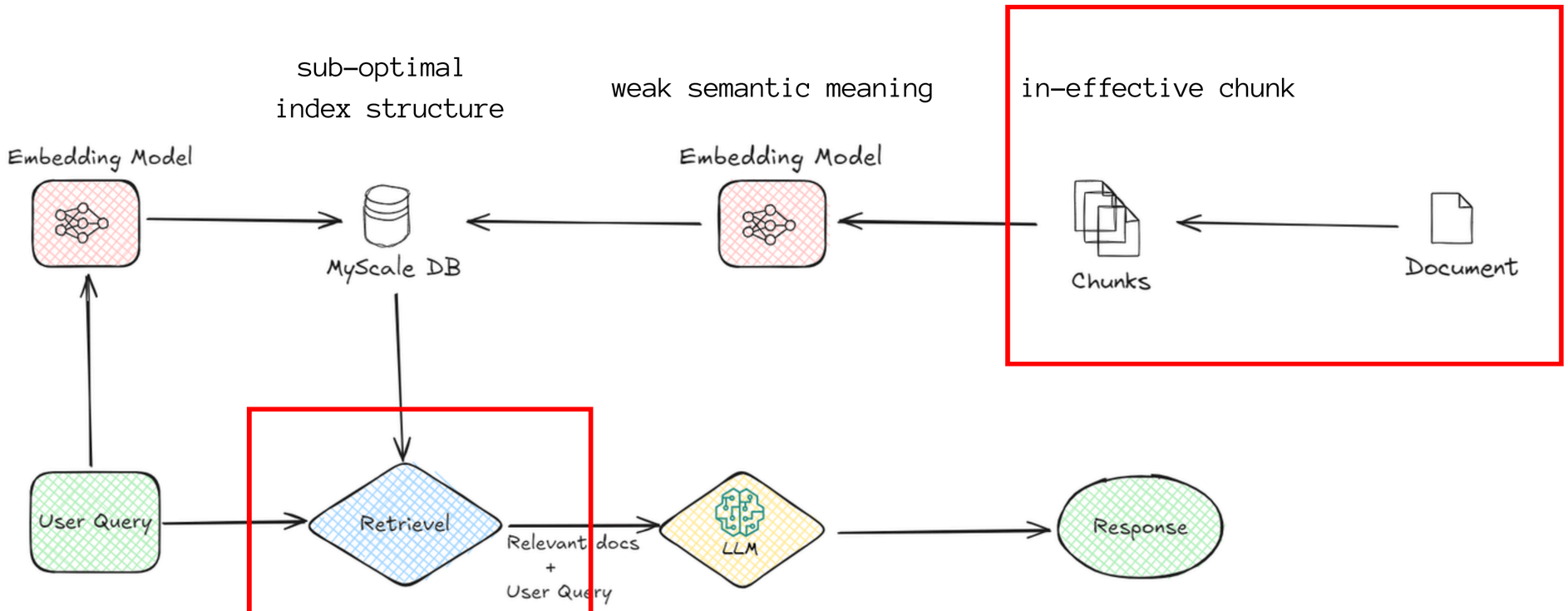
query objective not clear

- irrelevent context
- redundancy
- etc.

- resource intensity
- model optimization
- smaller context length
- etc.

FOCUS TOPIC

INGEST



query objective not clear

- irrelevent context
- redundancy
- etc.

- resource intensity
- model optimization
- smaller context length
- etc.

RETRIEVAL

INGEST STRATEGY

WHAT ARE THE MOST COMMON VULNERABILITIES? TOP 5 VULNERABILITY CATEGORIES

What's different: Where we saw a notable difference is the rise of the Broken Access Control category, and we're not alone in our observations. OWASP noticed the same uptick. In the 2017 version of their list, Broken Access Control was at 5th place, but it shot up to the top position in their 2021 update with this stat attached: "94% of applications were tested for some form of Broken Access Control."

One possible reason for this is growth in operations. We like to think of it like this: If you had a house with one door, all you have to worry about is keeping that one door locked. But if you expand that house and add a garage door and 5 windows—or in our case, adopt more technology, expand systems, widen networks, and manage more users—you're more likely to forget to lock everything. Creating or updating your user/access matrix can help.

Top 5 Findings

For a detailed breakdown of **how to fix and prevent each of the findings** listed here, check the Appendix in The State of Pentesting 2021.

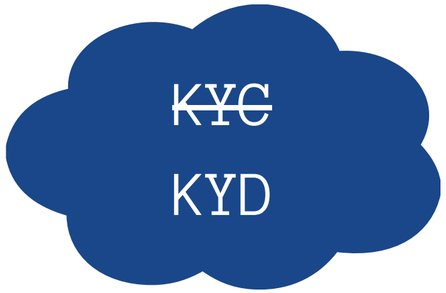
Diving deeper into our data, we uncovered a similar pattern when it came to the top 5 findings: most flaws from 2020 stayed at the top, but a couple of new issues made their way onto the list.

	2020	2021
#01	Broken Access Control: Insecure Direct Object References (IDOR)	Cross-Site Scripting (XSS): Stored
#02	Cross-Site Scripting (XSS): Stored	Broken Access Control: Insecure Direct Object References (IDOR)
#03	Components with Known Vulnerabilities: Outdated Software Versions	Components with Known Vulnerabilities: Outdated Software Versions
#04	Broken Access Control: Username/Email Enumeration	Server Security Misconfiguration: Insecure SSL or TLS protocols
#05	Cross-Site Scripting (XSS): Reflected	Server Security Misconfiguration: Lack of Security Headers

Figure 2: The top 5 most common findings in our database for 2020 and 2021

For a detailed breakdown of the **top vulnerabilities according to asset type**, check APPENDIX A.

Cobalt THE STATE OF PENTESTING 2022 | 08



- Do Some Qualitative Survey
- Gather User Expectation
- **Define Scope per Milestone**

- ingest strategy
- retrieval strategy
- benchmark evaluation

source - https://github.com/securestep9/CyberSecurity-Reports

INGESTION - 101

FIRST ITER.

- Text Parser + Recursive Chunking + RegEx Pattern

SECOND ITER.

- Layout Parser + Text Parser + Parent-Child + Recursive Chunking

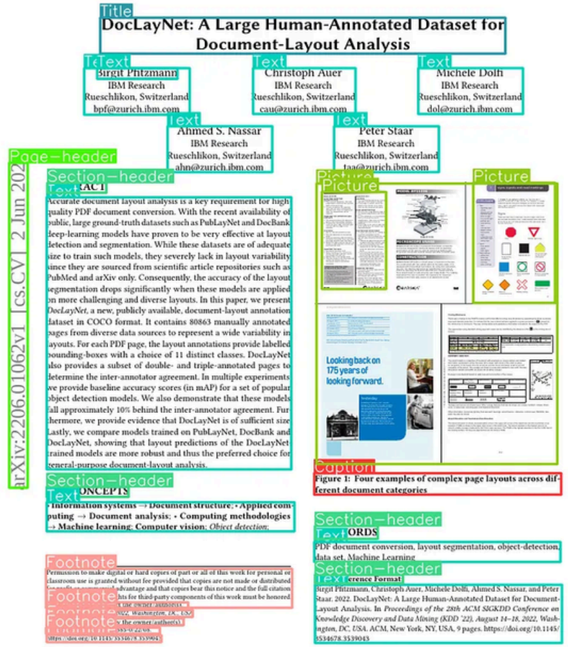
THIRD ITER.

- Multimodal Parser + Parent-Child + Semantic Chunking + Metadata Extraction

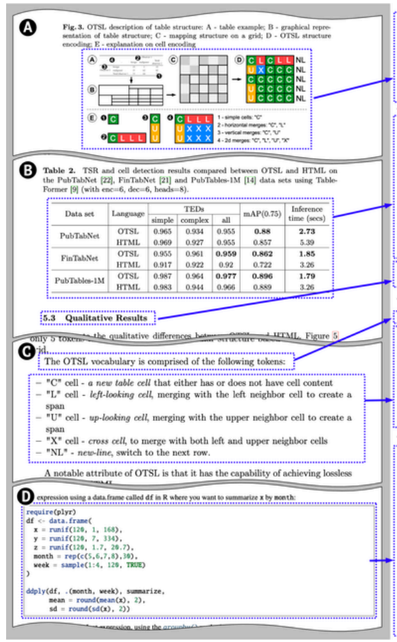
N ITER.

- DYOR

Layout Parser



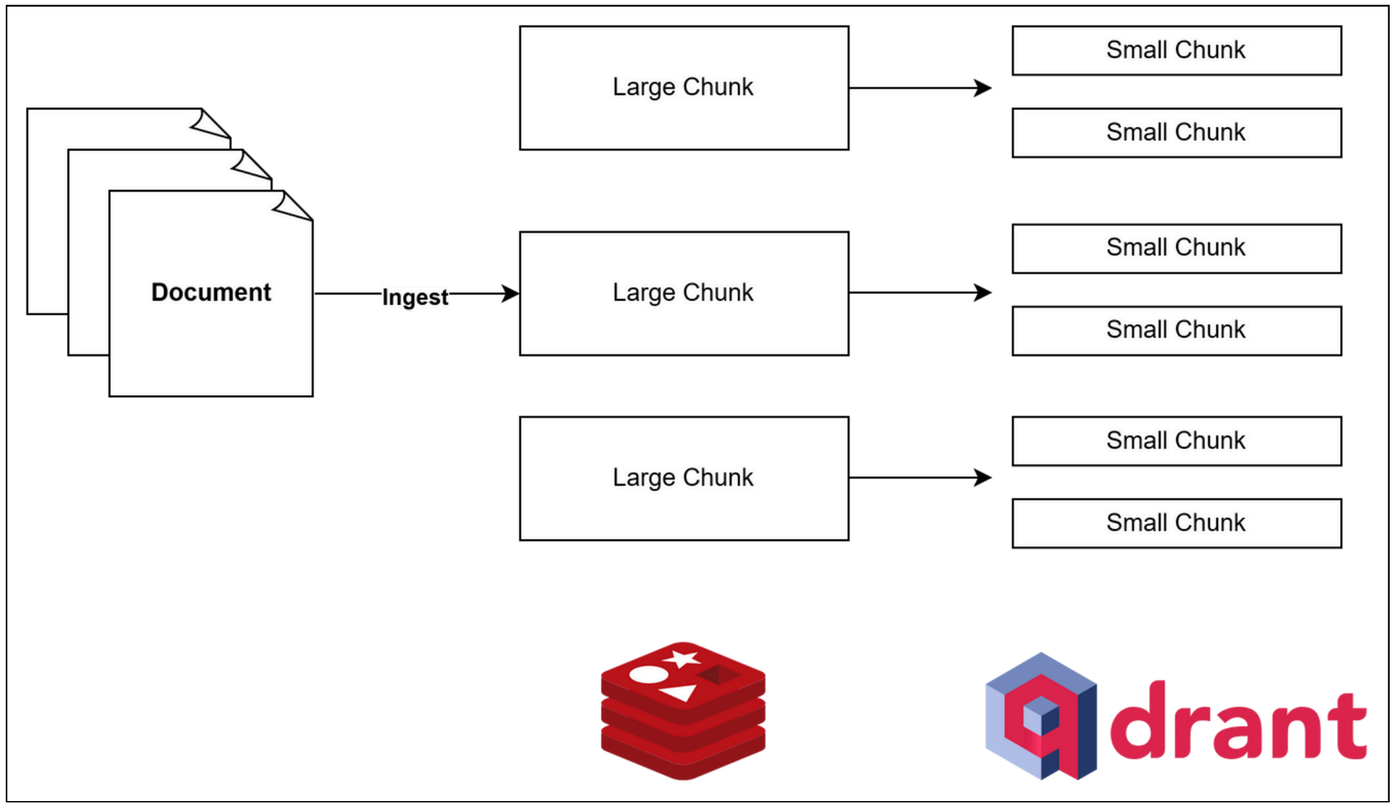
Multimodal Parser



```
<picture><loc_102><loc_85><loc_416><loc_160>
<flow_chart>
<caption>Fig. 3. OTSL description of table structure: A - table example; B - graphical representation of table structure; C - mapping structure on a grid; D - OTSL structure encoding; E - explanation on cell encoding</caption>
</picture>

<otsl><loc_63><loc_78><loc_436><loc_158>
<ched>Data set <ched>Language<ched>TEDs ..<lcel>.....<lcel>.... <ched>mAP(0.75)<ched>Inference ...<nl>
<ecel>.....<ecel>.....<ched>simple<ched>complex<ched>all ..<ecel>.....<ecel>.....<nl>
<rhed>PubTabNet<ched>OTSL....<fcel>0.965 <fcel>0.934..<fcel>0.955<fcel>0.88.....<fcel>2.73.....<nl>
<ucel>.....<ched>HTML....<fcel>0.969 <fcel>0.927..<fcel>0.955<fcel>0.857....<fcel>5.39.....<nl>
...
<caption>Table 2. TSR and cell detection results compared between OTSL and HTML on the PubTabNet [22], FinTabNet [21] and PubTables-1M [14] data sets using TableFormer [9] (with enc=6, dec=6, heads=8).
</caption>
</otsl>
<section_header_level_1><loc_58><loc_343><loc_209><loc_351>5.2 Quantitative Results
</section_header_level_1>

<text><loc_66><loc_360><loc_358><loc_368>The OTSL vocabulary is comprised of the following t...</text>
<unordered_list>
<list_item><loc_53><loc_377><loc_439><loc_385>- "C" cell - a new table cell that either...</list_item>
<list_item><loc_53><loc_387><loc_439><loc_403>- "L" cell - left-looking cell, merging w...</list_item>
<list_item><loc_53><loc_405><loc_439><loc_421>- "U" cell - up-looking cell, merging wit...</list_item>
<list_item><loc_53><loc_423><loc_409><loc_431>- "X" cell - cross cell, to merge with bo...</list_item>
<list_item><loc_53><loc_433><loc_269><loc_441>- "NL" - new-line, switch to the next row...</list_item>
</unordered_list>

<code><loc_58><loc_170><loc_443><loc_266><_Python_>
require(plyr)
df <- data.frame(
  x = runif(120, 1, 168),
  y = runif(120, 7, 334),
  z = runif(120, 1.7, 20.7),
  month = rep(c(5,6,7,8),30),
  week = sample(1:4, 120, TRUE)
)
ddply(df, .(month, week), summarize,
      mean = round(mean(x), 2),
      sd = round(sd(x), 2))
</code>
```

INGESTION - 101

Parent-Child



RETRIEVAL - 101

FIRST ITER.

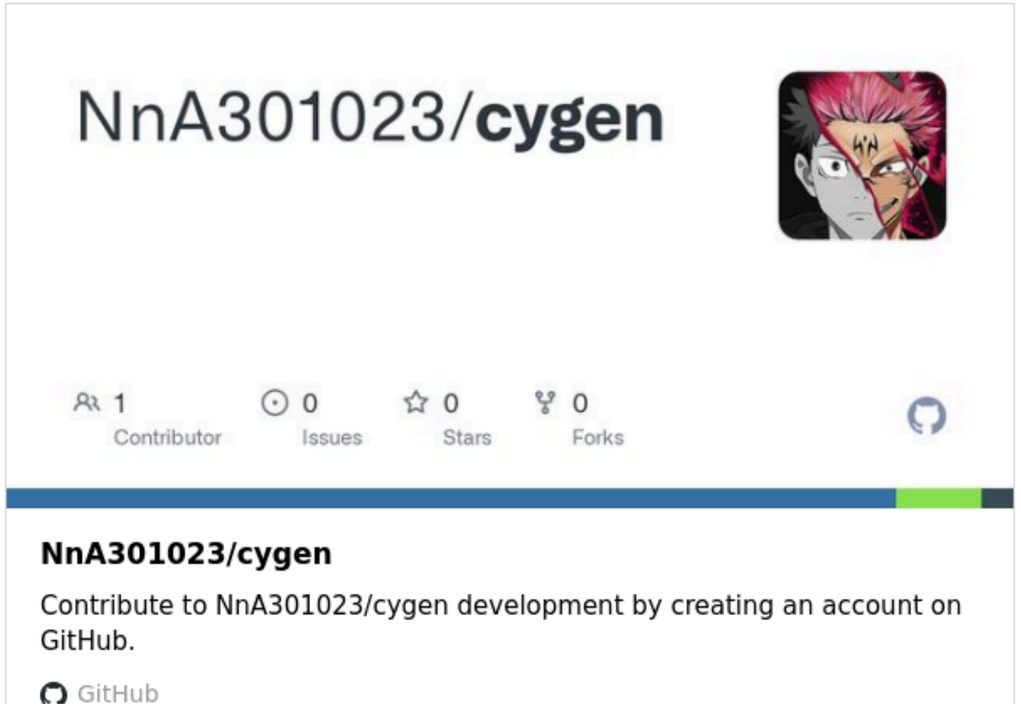
- **Go To MTEB Benchmark** + Experiment (Language Type)
- Setup LLM Monitoring

LangSmith opik

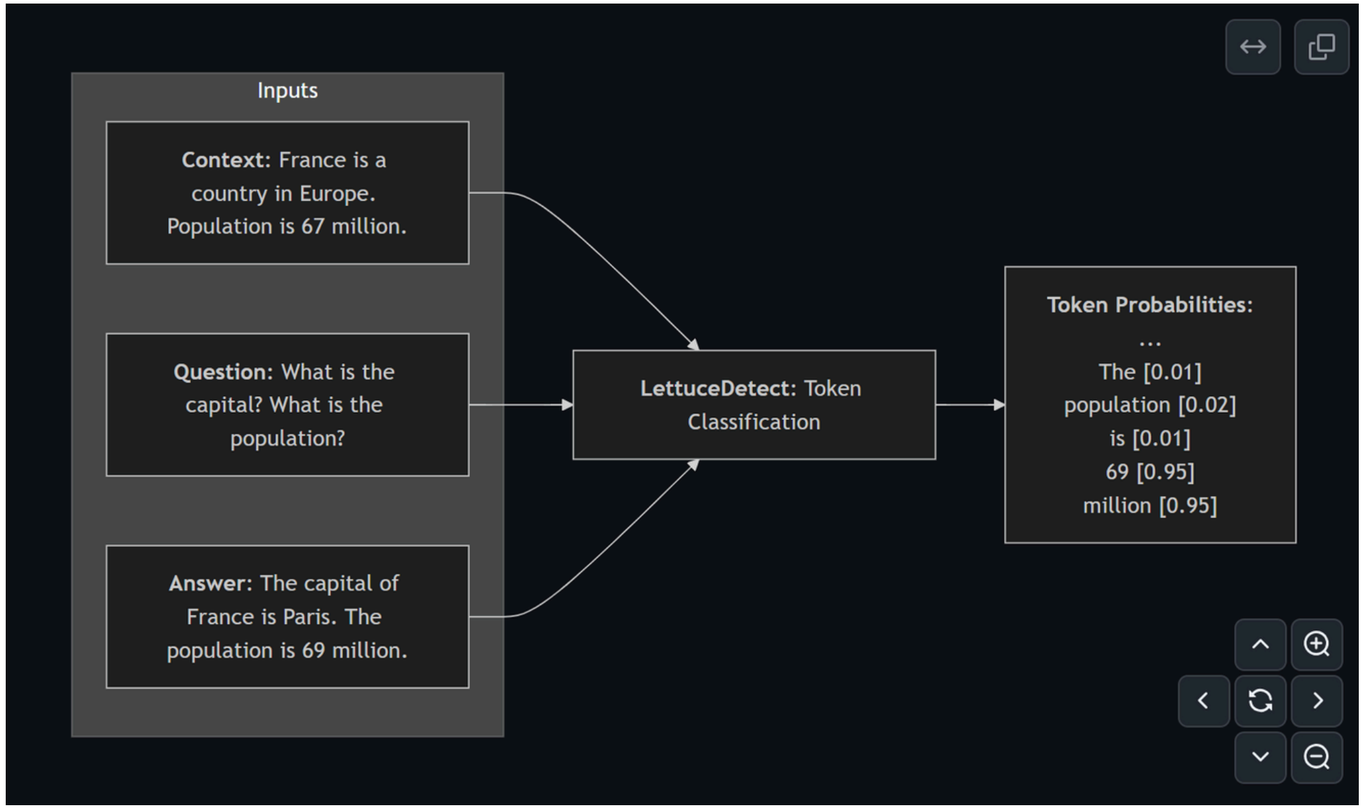
Langfuse

***note: tools above might not stable yet on “production”**


RAG Chat System
Upload Documents
Upload PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
Conversations
New Conversation
Select Conversation
Collaboration in DevSecOps Security Framework (2025-03-19 03:06)
New Conversation (2025-03-19 03:06)
Hai, Let's Start a Conversation (2025-03-19 03:02)
Hai! Let's Start a Conversation (2025-03-18 22:20)
Deploy
RAG Chat System
Conversation: Collaboration in DevSecOps Security Framework
What role does collaboration play in achieving security in a DevSecOps framework?
Based on the provided context, it is clear that collaboration plays a significant role in achieving security in a DevSecOps framework. The concept of DevSecOps is built around the idea of integrating security into the development and operational processes through modern integrated security paradigms.
Pillar 1: Collective Responsibility emphasizes the importance of changing the organization's mindset, ideas, customs, and behaviors regarding software security. It highlights the need for security to be seen as a collective responsibility, rather than someone else's responsibility. This suggests that collaboration among different stakeholders, including development, operations, and security teams, is essential for achieving security in a DevSecOps framework.
The text also mentions the importance of "collective collaboration" in the context of DevOps, which implies that collaboration is a key aspect of the DevOps practice. This collaboration can help to improve outcomes by integrating existing security practices into the development and operational processes.
Furthermore, the text emphasizes the need for a "Pragmatic" approach to security in DevOps, which involves connecting all stakeholders (development, operations, and security) in a manner that ensures security is built into applications and the software lifecycle. This approach requires collaboration among different stakeholders to ensure that security is integrated into the development and operational processes.
In summary, collaboration plays a crucial role in achieving security in a DevSecOps framework by integrating security into the development and operational processes, changing the organization's mindset regarding software security, and ensuring that security is built into applications and the software lifecycle.
Thank you for your feedback!


NnA301023/cygen
1 Contributor
0 Issues
0 Stars
0 Forks
NnA301023/cygen
Contribute to NnA301023/cygen development by creating an account on GitHub.
GitHub

EVALUATION - 101



source - https://github.com/KRLabsOrg/LettuceDetect

PYTHON ID SHARING SESSION

CONCLUSION

- INGESTION
- RETRIEVAL
- EVALUATION

Alibaba Cloud

indigo

Sesi QnA Dibuka

THANKS