

## ДОДАТОК А

## Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



Рисунок А.1 – Основна частина звіту (створений самостійно)

5	<a href="https://openarchive.nure.ua/bitstreams/09ae7c02-03cb-40ba-b45a-311ed8b1859d/download">https://openarchive.nure.ua/bitstreams/09ae7c02-03cb-40ba-b45a-311ed8b1859d/download</a>	6 0.07 %
6	СОЦІАЛЬНІ ГРАФИ ЯК ІНСТРУМЕНТ СУЧАСНОЇ КОМУНІКАЦІЇ ТА КОМАНДУВАННЯ 2/7/2022 Donetsk National Technical University (Керівники секцій МАН)	6 0.07 %
<b>3 бази даних RefBooks (0.00 %)</b>		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
<b>3 домашньої бази даних (0.00 %)</b>		
ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
<b>3 програми обміну базами даних (0.17 %)</b>		
НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	СОЦІАЛЬНІ ГРАФИ ЯК ІНСТРУМЕНТ СУЧАСНОЇ КОМУНІКАЦІЇ ТА КОМАНДУВАННЯ 2/7/2022 Donetsk National Technical University (Керівники секцій МАН)	14 (2) 0.17 %
<b>3 Інтернету (0.43 %)</b>		
ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	<a href="https://openarchive.nure.ua/bitstreams/09ae7c02-03cb-40ba-b45a-311ed8b1859d/download">https://openarchive.nure.ua/bitstreams/09ae7c02-03cb-40ba-b45a-311ed8b1859d/download</a>	24 (3) 0.30 %
2	<a href="https://arxiv.org/html/2411.15382v1">https://arxiv.org/html/2411.15382v1</a>	11 (1) 0.14 %
<b>Список прийнятих фрагментів (немає прийнятих фрагментів)</b>		
ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

**ВСТУП**

За останні кілька років наш світ колосально змінився. З появою штучного інтелекту стало зрозуміло що багато речей можна дізнатися значно швидше та написати значно швидше. Дуже важливим аспектом штучного інтелекту є те, що ми не просто читаємо статтю або підручник на конкретну тему, а ми можемо обговорювати нову інформацію. Це можна порівняти як самостійне читання підручника, дослідження та розуміння чому саме така формула наведена, чи немає десь помилки і яка логіка за цим стоїть, та випадком коли вам хтось пояснює цей матеріал. Будь-то ваш друг, колега, або викладач в університеті. Найважливіша різниця в тому, що ви можете задавати зустрічні питання та отримувати на них відповіді. Це значним чином прискорює процес навчання. При правильному використанні, та без логічних помилок, штучний інтелект у сучасному вигляді є корисним для людства.

Щодо логічних помилок, це стється в більшій мірі через те, що модель знає лише про те, що бачила під час тренування. Там може не бути деталей щодо певних доменів, про які ви хочете спитати. Це означає що в модель потрібно зробити ін'єкцію знань конкретного домену, щоб вона краще могла відповідати на поставлені запитання та бути більш корисною. Але у наш час існує багато методів ін'єкції знань у великі лінгвістичні моделі.

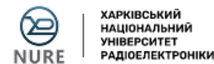
Ця кваліфікаційна робота як раз націлена на дослідження методів ін'єкції знань для великих мовних моделей. Тема дослідження тісно пов'язана з науковим напрямком кафедри ПІ, яка досліджує інноваційні технології та методи їх використання.

Тема дослідження тісно пов'язана з науковим напрямком кафедри ПІ, яка досліджує інноваційні технології та методи їх використання.

Рисунок А.2 – Додаткова частина звіту (створений самостійно)

## ДОДАТОК Б

### Слайди презентації



## Дослідження методів ін'єкції знань для великих мовних моделей

Студент: Флягін Владислав Костянтинович, ІПЗМ-23-4  
Науковий керівник: доцент Турута Олексій Петрович



16 червня 2025

Рисунок Б.1 – Титульний слайд (створений самостійно)

## Аналіз предметної галузі

Сучасні проблеми:

- статичність знань
- галюцинації
- нестача структурованих даних
- обчислювальні витрати
- інтерпретація логіки моделей

Перспективи та тренди:

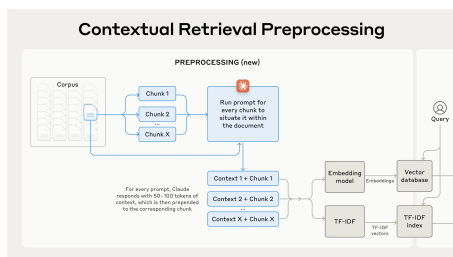
- автоматизація інтеграції знань
- інтерпретованість моделей
- зменшення обчислювальних витрат (для користувачів)
- генерація штучних даних
- агентні системи



2

Рисунок Б.2 – Дослідження (створений самостійно)

# Огляд аналогів



## Parameters in Selected AI Models

Some of these figures are estimates. Newer models are many times larger than their predecessors.

GPT-1	117,000,000
GPT-2	1,500,000,000
Gemini Nano-1	1,800,000,000
Gemini Nano-2	3,250,000,000
Llama 3 8b	8,000,000,000
Llama 3 70B	70,000,000,000
Claude 2	130,000,000,000
GPT-3	175,000,000,000
Gemini Pro	500,000,000,000
Gemini Ultra	1,000,000,000,000
GPT-4	1,760,000,000,000

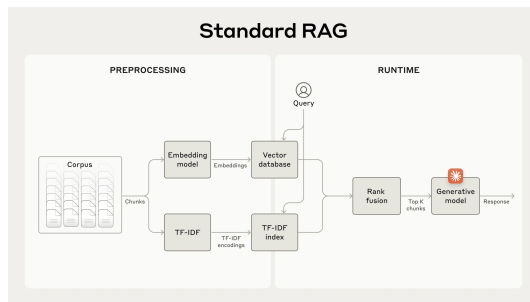
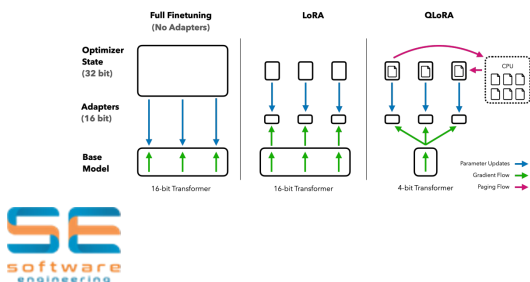


Рисунок Б.3 – Огляд аналогів 1 частина (створений самостійно)

# Огляд аналогів

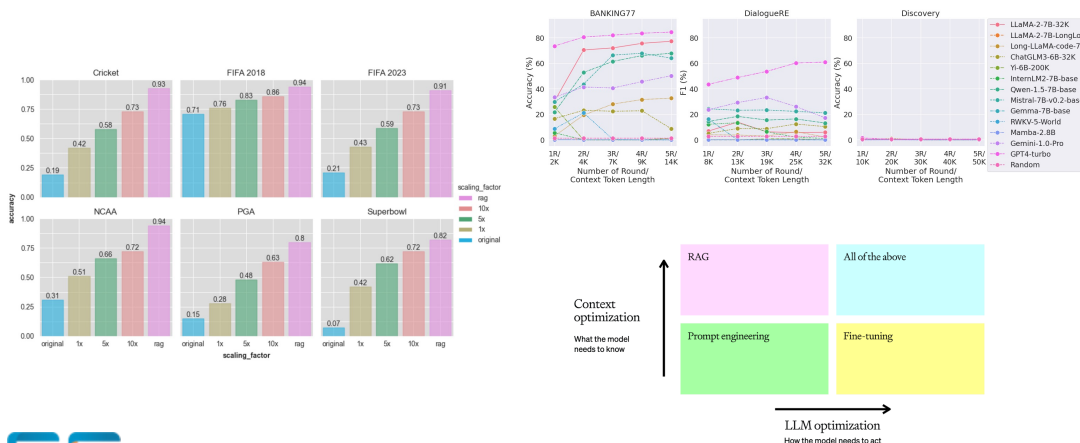


Рисунок Б.4 – Огляд аналогів 2 частина (створений самостійно)

## Постановка задачі

В магістерській роботі планується дослідити задачу ін'єкції знань у великі лінгвістичні моделі з використанням RAG підходу, в особливості з використанням графів знань.

Модифікації RAG для дослідження:

- Гібридний пошук та семантичний пошук
- Кількість актуальних чанків (3 або 5)
- Розмір чанків (128 та 256 токенів)
- Токенайзер на чанки (символи або речення)

Датасет:

Штучно згенерований, 20000+ токенів, 50+ прикладів питань та відповідей



5

Рисунок Б.5 – Постановка задачі (створений самостійно)

## Методологія

	Якість знань	Масштабованість	Довжина контекстного вікна	Мульти-дальність	Відкритість	Корисність
o1-mini	0,82	1,00	0,06	0,25	0,33	0,556
Gemini 1.5 Pro	0,80	1,00	1,00	1,00	0,67	<b>0,885</b>
Llama 3.2 90B	0,67	0,40	0,06	0,50	1,00	0,537
Llama 3.2 11B	0,53	0,80	0,06	0,50	1,00	<b>0,651</b>
Llama 3.1 405B	0,72	0,20	0,06	0,25	1,00	0,459
Gemma 2 27B	0,61	0,80	0,00	0,25	1,00	0,632

```

<start_header_id>-system<end_header_id>
You are an expert in composing functions. You are given a question and a set of possible functions. Based on the question, you will need to make one or more function/tool calls to achieve the purpose. If none of the functions can be used, point it out. If the given question lacks the parameters required by the function, also point it out. You should only return the function call in tools call sections. If you decide to invoke any of the function(s), you MUST put it in the format of [func_name(params_name1=params_value1, params_name2=params_value2,...), func_name2(params)] You SHOULD NOT include any other text in the response. Here is a list of functions in JSON format that you can invoke:
{
  "name": "get_user_info",
  "description": "Retrieve details for a specific user by their unique identifier. Note that the provided function is in python 3 syntax.",
  "parameters": {
    "type": "dict",
    "required": [
      "user_id"
    ],
    "properties": {
      "user_id": {
        "type": "integer",
        "description": "The unique identifier of the user. It is used to fetch the specific user details from the database."
      },
      "special": {
        "type": "string",
        "description": "Any special information or parameters that need to be considered while fetching user details.",
        "default": "none"
      }
    }
  }
}
]
</start_header_id>-<start_header_id>-user<end_header_id>
Can you retrieve the details for the user with the ID 7890, who has black as their special request?</start_header_id>-<start_header_id>-assistant<end_header_id>

```



6

Рисунок Б.6 – Методологія 1 частина (створений самостійно)

# Методологія

**Instruction:** Determine the most cost-effective country to purchase the smartphone model "CodeAct 1". The countries to consider are the USA, Japan, Germany, and India.

**Available APIs**

```

[lookup_exchange_rate: str] -> (float, float)
[convert_and_tax(price: float, exchange_rate: float, tax_rate: float)] -> float
[estimate_final_price(converted_price: float, shipping_cost: float)] -> float
[lookup_phone_price(model: str, country: str)] -> float
[estimate_shipping_cost(destination_country: str)] -> float
    
```

**LLM Agent using [Text/JSON] as Action**

**Think** I should calculate the phone price in USD for each country, then find the most cost-effective country.

**Action** Text: lookup\_rates, Germany  
JSON: {"tool": "lookup\_rates", "country": "Germany"}

**Environment** 1.1, 8.19

**Action** Text: lookup\_phone\_price, CodeAct 1, Germany  
JSON: {"tool": "lookup\_phone\_price", "model": "CodeAct 1", "country": "Germany"}

**Environment** 788

**Fewer Actions Required!**

**Action** Text: convert\_and\_tax, 788, 1.1, 8.19  
JSON: {"tool": "convert\_and\_tax", "price": 788, "exchange\_rate": 1.1, "tax\_rate": 8.19}

**Environment** 916.3

[...] Interactions omitted (lookup shipping cost and calculate final price) [...]

**Action** Text: lookup\_rates, Japan  
JSON: {"tool": "lookup\_rates", "country": "Japan"}

[...] Interactions omitted (calculate final price for all other countries) [...]

**Response** The most cost-effective country to purchase the smartphone model is Japan with price 984.88 in USD.

**CodeAct: LLM Agent using [Code] as Action**

**Think** I should calculate the phone price in USD for each country, then find the most cost-effective country.

**Action**

```

countries = ["USA", "Japan", "Germany", "India"]
FinalPrices = {}

for country in countries:
    exchange_rate, tax_rate = lookup_rates(country)
    local_price = lookup_phone_price("CodeAct 1", country)
    converted_price = convert_and_tax(
        local_price, exchange_rate, tax_rate
    )
    shipping_cost = estimate_shipping_cost(country)
    final_price = estimate_final_price(converted_price, shipping_cost)
    FinalPrices[country] = final_price

most_cost_effective_country = min(FinalPrices, key=FinalPrices.get)
most_cost_effective_price = FinalPrices[most_cost_effective_country]
print(most_cost_effective_country, most_cost_effective_price)
    
```

**Environment** 1.1, 8.19

**Response** The most cost-effective country to purchase the smartphone model is Japan with price 984.88 in USD.

Generate me a private corpus with some details mentioning the imagined NeonByte Interactive – Video game development and VR/AR experiences A list of products, prices, responsible stuff, etc. I want to use it as my private corpus for the LLM fine-tuning You can generate really a lot of the text. The more the better.

generate 20 questions & answers from the dataset above, structure them in the python list with "question" & "answer" fields



Рисунок Б.7 – Методологія 2 частина (створений самостійно)

# Методологія

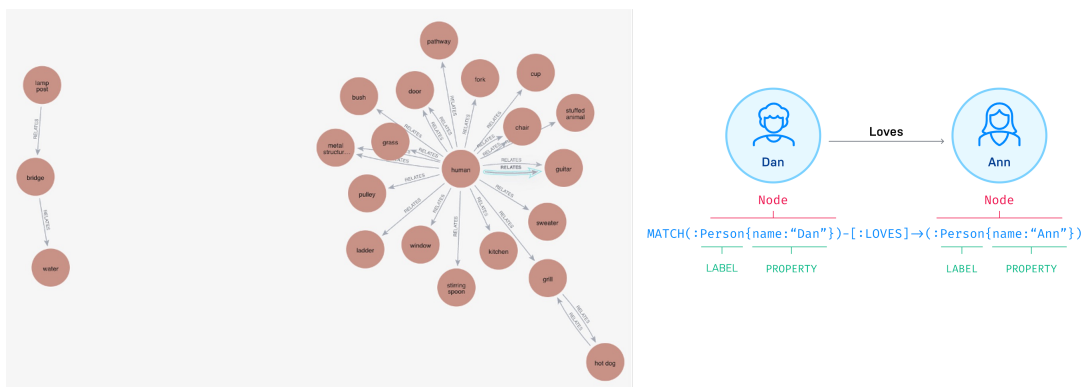
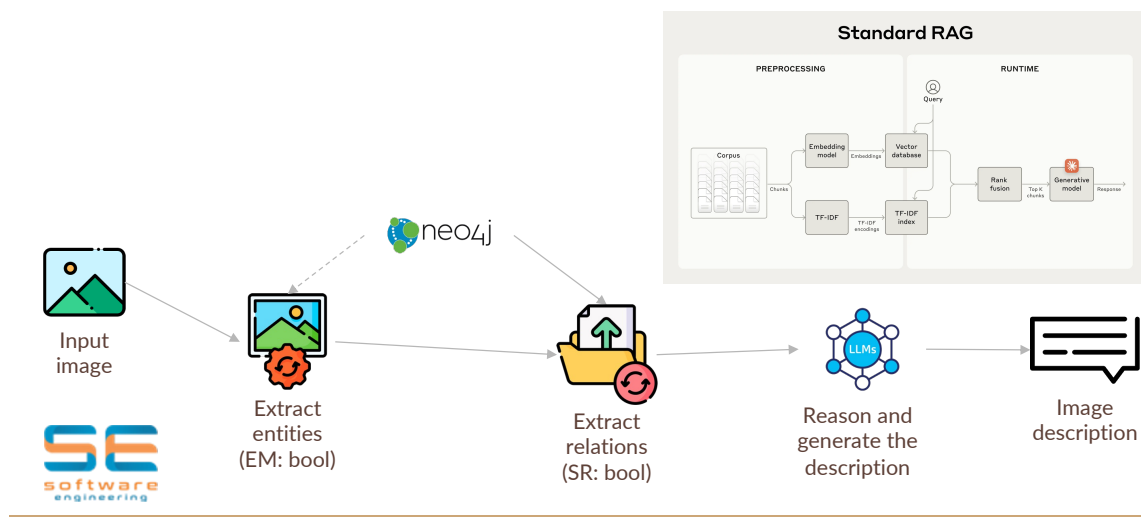


Рисунок Б.8 – Методологія 3 частина (створений самостійно)

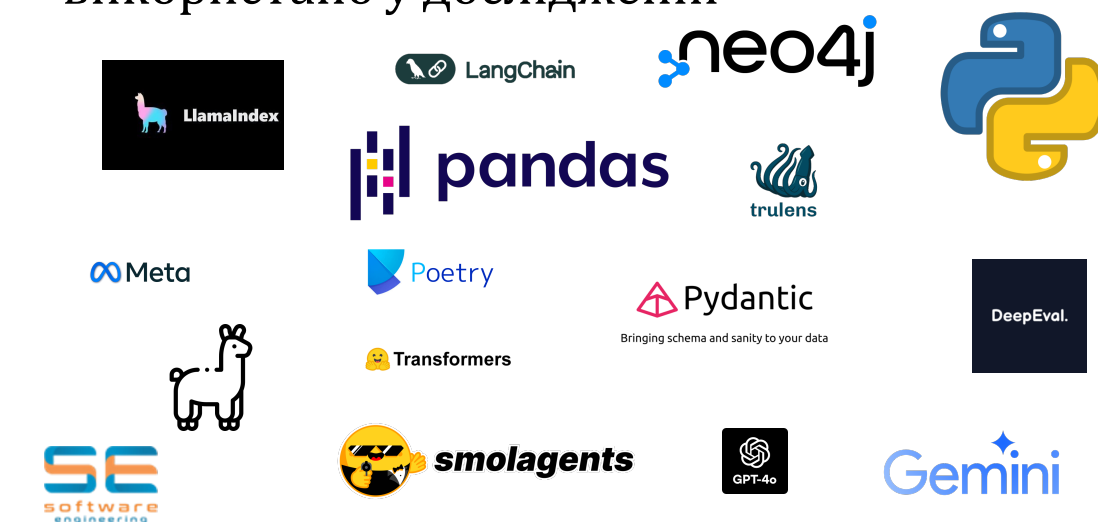
## Архітектурна система для проведення експериментального дослідження



9

Рисунок Б.9 – Архітектура системи (створений самостійно)

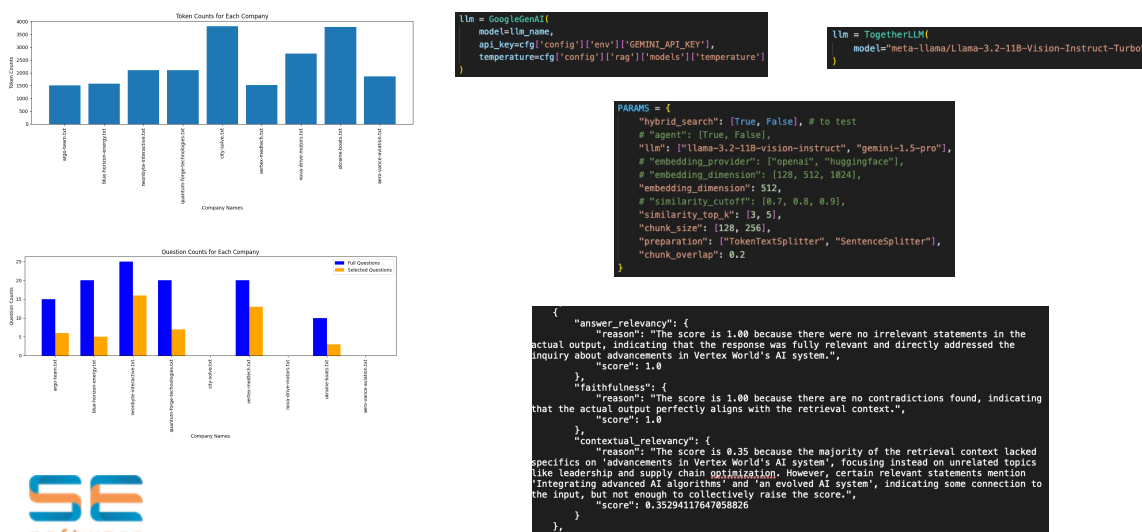
## Опис програмного забезпечення, що було використано у дослідженні



10

Рисунок Б.10 – Опис використаних інструментів (створений самостійно)

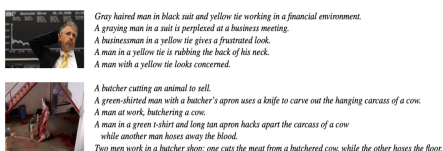
## Зміст проведеного експерименту



11

Рисунок Б.11 – Зміст основного експерименту (створений самостійно)

## Зміст проведеного експерименту



Relevance [0 - 10] - вимірює, наскільки добре згенерована відповідь відповідає очікуваному змісту зображення.  
Groundedness [0 - 10] - оцінює ступінь, до якого результати роботи моделі ґрунтуються на отриманих знаннях, а не на галюцинаційній інформації.  
BERTScore [0 - 1] - широко використовувана метрика схожості тексту, заснована на контекстних вбудовуваннях з попередньо навченої BERT-моделі.

```
{
  "start": "human",
  "relation": "standing near",
  "end": "gate"
},
{
  "start": "gate",
  "relation": "surrounded by",
  "end": "bush"
},
{
  "start": "bush",
  "relation": "growing near",
  "end": "tree"
},
{
  "start": "tree",
  "relation": "casting shade on",
  "end": "grass"
},
{
  "start": "grass",
  "relation": "covering ground near",
  "end": "bush"
},
{
  "start": "human",
  "relation": "operating",
  "end": "crane"
},
{
  "start": "human",
  "relation": "standing on",
  "end": "crane"
},
{
  "start": "human",
  "relation": "climbing",
  "end": "steps"
},
}
```

```
config = {
  "ARAG": [
    "gpt-4o-mini",
    "gpt-4o-mini-ARAG",
    "gemini-2.0-flash-lite",
    "gemini-2.0-flash-lite-ARAG"
  ],
  "ARAG-EM": [
    "gpt-4o-mini-ARAG-EM",
    "gemini-2.0-flash-lite-ARAG-EM"
  ],
  "ARAG-SR": [
    "gpt-4o-mini-ARAG-SR",
    "gemini-2.0-flash-lite-ARAG-SR"
  ],
  "ARAG-EM-SR": [
    "gpt-4o-mini-ARAG-EM-SR",
    "gemini-2.0-flash-lite-ARAG-EM-SR"
  ]
}
```

12

Рисунок Б.12 – Зміст експерименту статті (створений самостійно)

## Результати експерименту

setup	answerc_relevance	faithfulness	contextual_relevance
L16_hn_True_Itn_gemini-1.5-pro_topk_3_cs_256_prep_TokenTextSplitter	0.844286	0.925478	0.130450
L17_hn_True_Itn_gemini-1.5-pro_topk_3_cs_256_prep_TokenTextSplitter	0.847867	0.972667	0.170579
L18_hn_True_Itn_gemini-1.5-pro_topk_3_cs_256_prep_SentenceSplitter	0.946500	0.860000	0.131643
L4_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_SentenceSplitter	0.910000	0.910000	0.197912
L7_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_TokenTextSplitter	0.903333	0.914000	0.139580
L8_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_SentenceSplitter	0.896667	0.941667	0.139540
L20_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_TokenTextSplitter	0.897667	0.863667	0.122358
L5_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_TokenTextSplitter	0.890000	0.780000	0.164850
L10_hn_True_Itn_gemini-1.5-pro_topk_3_cs_128_prep_SentenceSplitter	0.884667	0.924333	0.214884
L32_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_256_prep_SentenceSplitter	0.882857	0.861667	0.134109
L31_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_256_prep_TokenTextSplitter	0.878667	0.839667	0.121231
L13_hn_True_Itn_gemini-1.5-pro_topk_3_cs_128_prep_TokenTextSplitter	0.876667	0.848444	0.167451
L11_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_TokenTextSplitter	0.871667	0.856667	0.210987
L12_hn_True_Itn_gemini-1.5-pro_topk_3_cs_256_prep_SentenceSplitter	0.861061	0.876910	0.188790
L20_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_SentenceSplitter	0.860667	0.840000	0.170301
L14_hn_True_Itn_gemini-1.5-pro_topk_3_cs_128_prep_SentenceSplitter	0.858500	0.879810	0.163078
L3_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_TokenTextSplitter	0.850000	0.886667	0.162243
L11_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_SentenceSplitter	0.839667	0.880000	0.227976
L6_hn_True_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_SentenceSplitter	0.833333	0.890000	0.150708
L21_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_TokenTextSplitter	0.823333	0.860000	0.124607
L9_hn_True_Itn_gemini-1.5-pro_topk_3_cs_128_prep_TokenTextSplitter	0.822976	0.826333	0.203043
L27_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_256_prep_TokenTextSplitter	0.810500	0.855000	0.157657
L27_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_TokenTextSplitter	0.806905	0.891000	0.184476
L22_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_SentenceSplitter	0.805510	0.878000	0.181172
L28_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_256_prep_SentenceSplitter	0.794952	0.911333	0.184109
L28_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_128_prep_SentenceSplitter	0.768500	0.852000	0.204230
L28_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_128_prep_TokenTextSplitter	0.763667	0.810286	0.158947
L19_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_256_prep_TokenTextSplitter	0.750000	0.938333	0.153895
L20_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_128_prep_SentenceSplitter	0.729167	0.866667	0.145131
L25_hn_Fake_Itn_gemini-1.5-pro_topk_3_cs_128_prep_TokenTextSplitter	0.753005	0.859143	0.213084
L17_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_TokenTextSplitter	0.741667	0.866667	0.195462
L18_hn_Fake_Itn_Iama-3.2-11B-vision-instruct_topk_3_cs_128_prep_SentenceSplitter	0.726667	0.910000	0.181072



13

Рисунок Б.13 – Результати експериментів 1 частина (створений самостійно)

## Результати експерименту

Approach	Relevance	Groundedness	BERTScore
gpt-4o-mini	8.72	5.0	0.560
gpt-4o-mini-ARAG	7.06	6.0	0.538
gpt-4o-mini-ARAG-EM	6.74	6.34	0.542
gpt-4o-mini-ARAG-SR	6.44	5.88	0.536
gpt-4o-mini-ARAG-EM-SR	6.14	6.18	0.536
gemini-2.0-flash-lite	8.52	5.92	0.607
gemini-2.0-flash-lite-ARAG	6.40	6.82	0.592
gemini-2.0-flash-lite-ARAG-EM	5.82	7.68	0.587
gemini-2.0-flash-lite-ARAG-SR	6.20	6.60	0.593
gemini-2.0-flash-lite-ARAG-EM-SR	5.94	6.92	0.589

- Інноваційна ідея динамічного узгодження моделі в залежності від контексту зображення
- Експериментально доведено корисність впроваджених методів узгодження
- Можливе використання в реальних системах
- Є великий потенціал для масштабування



14

Рисунок Б.14 – Результати експериментів 2 частина (створений самостійно)

## Висновки

- Було проведено серію з 32 експериментів, по результатам яких було досліджено ефективність налаштування конкретних параметрів у реальних RAG системах, що дає змогу краще зрозуміти природу параметрів RAG та їх правильного налаштування
- Було впроваджено новий метод узгодження великих мовних моделей в залежності від змісту зображення та користувацьких обмежень, який є інноваційним та може бути масштабований до реальних систем



15

Рисунок Б.15 – Підсумки (створений самостійно)

## Апробація результатів дослідження



16

### Approaching LLM alignment using Agents with RAG

Vladyslav Fliahin<sup>[ORCID:1528-0131]</sup> and Oleksii Turuta<sup>[ORCID:0009-0076-9471]</sup>

<sup>1</sup> Kharkiv National University of Radio Electronics, Nasky Ave. 14, Kharkiv, 61000, Ukraine

**Abstract.** This paper focuses on contributing scalable LLM alignment approaches to generate outputs close to human goals and values. As AI models become more advanced, alignment becomes increasingly critical. This article explores a novel approach using agents and Retrieval-Augmented Generation (RAG) for alignment. We create a custom knowledge graph based on the Flickr30k subset. Leverage a Neo4j database to store predefined entities and their relationships, which serve as constraints for model outputs. We use RAG to guide the model's generation by focusing only on the relevant entities and relationships detected in an image, ensuring alignment with structured knowledge while ignoring irrelevant details.

**Keywords:** LLM, Alignment, RAG, Knowledge Graph, Multimodal Data.

### 1 Introduction

#### 1.1 Problem statement

As large language models (LLMs) become more capable [1], the challenge of aligning their outputs with human intent, ethical constraints, and factual accuracy becomes increasingly important. While traditional alignment methods focus on reinforcement learning with human feedback (RLHF) or prompt engineering, these approaches often lack fine-grained control over specific aspects of model behavior. In particular, ensuring that an LLM only considers predefined entities and relationships, especially when processing complex multimodal data like images, remains an open challenge.

This paper focuses on contributing scalable LLM alignment approaches to generate outputs close to human goals and values.

#### 1.2 Overview of our method

In this work, we propose a novel approach to LLM alignment that leverages agents, Retrieval-Augmented Generation (RAG), and knowledge graphs to enforce controlled generation. We store a structured representation of allowed entities and their relationships in a Neo4j knowledge graph, which acts as a constraint system. When analyzing an image, our method first detects the entities present, retrieves only the correspond-

Рисунок Б.16 – Апробація результатів дослідження (створений самостійно)

# Дякую за увагу!

17

Рисунок Б.17 – Дякую за увагу (створений самостійно)

## Sample 58

**gpt-4o-mini:** A dog is joyfully running through a snowy landscape, with snowflakes flying around it. The dog is predominantly white with brown patches and has a playful expression. It is positioned in the center of the image, slightly to the right, with its body angled forward as it leaps through the snow. The background features a blurred, dark area, likely trees, contrasting with the bright white snow.

**gpt-4o-mini-ARAG-EM:** In the image, a dog is running on the snow, which covers the ground.

**gemini-2.0-flash-lite:** A white dog with tan markings is running through the snow. The dog is in the center of the image, with its body angled towards the center-right. Snow is kicked up around the dog's paws, indicating movement. The background consists of a snow-covered field and a blurred, dark forest.

**gemini-2.0-flash-lite-ARAG-EM:** A dog is running on snow. The dog is in the center of the image.



18

Рисунок Б.18 – Приклад на 58 зображені з датасету Flickr30k (створений самостійно)

## Sample 83

**gpt-4o-mini:** The image depicts a snowy street scene. In the center-left, a man is shoveling snow from the sidewalk onto the street. He is wearing a dark jacket and a hat. To his right, in the center, there is a partially cleared area of the street where the snow has been removed. The background features a row of buildings with shops, visible in the top-left and top-center areas. The buildings have a mix of architectural styles, with some graffiti visible on one of them. Several parked cars are lined up along the street, mostly in the bottom-left and bottom-center areas, with a few more visible in the bottom-right. The overall atmosphere is cold and wintry, with a significant amount of snow covering the ground.

**gpt-4o-mini-ARAG-EM:** The image shows a streetlight in the top-left corner and buildings in the top-center. In the center-left, a human is shoveling snow, with snow covering the ground in the center. To the center-right, there is a car parked on the road.

**gemini-2.0-flash-lite:** A man is shoveling snow on a city street. Cars are parked along the side of the road, and buildings line the street. The scene is covered in snow, and the road is wet.

**gemini-2.0-flash-lite-ARAG-EM:** A human is wearing a hat and holding a shovel. The human is moving snow. A car is in the background. A building is in the background.



19

Рисунок Б.19 – Приклад на 83 зображені з датасету Flickr30k (створений самостійно)

## Sample 82

**gpt-4o-mini:** The image depicts three individuals walking away from the viewer on a grassy path in a natural landscape. The scene is set under a clear blue sky, with a few trees visible on the left side and a distant view of rolling hills in the background. The individuals are positioned in the center of the image, with the first person on the left, the second person in the middle, and the third person on the right. The foreground is filled with lush green grass and some plants, while the background features a horizon line with trees and hills.

**gpt-4o-mini-ARAG-EM:** The image features three humans walking in the center-right area, surrounded by grass in the center. Trees are present in the top-left and surrounding areas, with a clear blue sky above.

**gemini-2.0-flash-lite:** Three people are walking on a grassy path towards a distant mountain range under a clear blue sky. The path is surrounded by lush green grass and trees. The person in the center is wearing a white shirt, the person on the right is wearing a blue shirt, and the person on the left is wearing a gray shirt. The scene is bright and sunny, suggesting a pleasant day for a hike.

**gemini-2.0-flash-lite-ARAG-EM:** The image shows a sky above a human, a tree, and grass. A tree is near a human. The grass is in front of trees and surrounded by trees. A human is walking on the grass.



20

Рисунок Б.20 – Приклад на 82 зображені з датасету Flickr30k (створений самостійно)

## ДОДАТОК В

### Апробація результатів роботи

Матеріали публікації «Approaching LLM alignment using Agents with RAG», яку було презентовано на воркшопі MIT@AIS-2025.

Vladyslav Fliahin<sup>1</sup>[0009-0000-1520-0310] and Oleksii Turuta<sup>1</sup>[0000-0002-0970-8617]

<sup>1</sup> Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, 61000, Ukraine

**Abstract.** This paper focuses on contributing scalable LLM alignment approaches to generate outputs close to human goals and values. As AI models become more advanced, alignment becomes increasingly critical. This article explores a novel approach using agents and Retrieval-Augmented Generation (RAG) for alignment. We create a custom knowledge graph based on the Flickr30k subset. Leverage a Neo4j database to store predefined entities and their relationships, which serve as constraints for model outputs. We use RAG to guide the model's generation by focusing only on the relevant entities and relationships detected in an image, ensuring alignment with structured knowledge while ignoring irrelevant details.

**Keywords:** LLM, Alignment, RAG, Knowledge Graph, Multimodal Data.

## 1 Introduction

### 1.1 Problem statement

As large language models (LLMs) become more capable [1], the challenge of aligning their outputs with human intent, ethical constraints, and factual accuracy becomes increasingly important. While traditional alignment methods focus on reinforcement learning with human feedback (RLHF) or prompt engineering, these approaches often lack fine-grained control over specific aspects of model behavior. In particular, ensuring that an LLM only considers predefined entities and relationships,

especially when processing complex multimodal data like images, remains an open challenge.

This paper focuses on contributing scalable LLM alignment approaches to generate outputs close to human goals and values.

## 1.2 Overview of our method

In this work, we propose a novel approach to LLM alignment that leverages agents, Retrieval-Augmented Generation (RAG), and knowledge graphs to enforce controlled generation. We store a structured representation of allowed entities and their relationships in a Neo4j knowledge graph, which acts as a constraint system. When analyzing an image, our method first detects the entities present, retrieves only the corresponding allowed relationships from the database, and then guides the LLM's generation using RAG. This ensures that the model adheres strictly to the predefined knowledge constraints, avoiding irrelevant or undesired outputs.

Our contributions are as follows:

- A knowledge-constrained alignment framework using Neo4j and RAG to regulate model outputs.
- A multimodal alignment strategy that ensures only predefined entities and relations are described from an image.
- Empirical validation demonstrates how this approach improves alignment precision while reducing hallucination.

The rest of this paper is structured as follows: Section 2 discusses related work on LLM alignment, knowledge-grounded generation systems, and agents. Section 3 describes our methodology, including the role of LLMs, agents, RAG, and Neo4j. Section 4 presents our experiments and results. Section 5 concludes with key insights and future research directions. Section 6 outlines the limitations of the developed framework.

## 2 Relevant work

Large Language Models (LLMs) have advanced in structured reasoning through techniques like Chain-of-Thought (the authors showed how such reasoning abilities

emerge naturally in sufficiently large language models via a simple prompting [2]), Self-Consistency (the authors propose a new decoding strategy, self-consistency, to replace the naive greedy decoding used in chain-of-thought prompting; it first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths [3]), and Tree-of-Thought (authors introduced a new framework for language model inference, Tree of Thoughts (ToT), which generalizes over the popular Chain of Thought approach to prompting language models, and enables exploration over coherent units of text (thoughts) that serve as intermediate steps toward problem solving [4]), improving inference by generating intermediate steps rather than relying on greedy decoding [5, 6, 7, 8, 9, 10]. In case the amount of information is too big to fit into a prompt, prior works have used knowledge storage, such as knowledge graphs.

Knowledge Graphs (KGs) are structured repositories of interconnected entities and relationships, offering efficient graph-based knowledge representation and retrieval [11, 12, 13].

Prior work combining KGs with LLMs has primarily focused on tasks such as knowledge-based question answering [14, 15, 16, 17], entity-centric retrieval [18, 19, 20], and fact-checking [21, 22, 23].

However, in the described applications, the obtained data was mainly used to extract the correct answer or infer the answer from it. Our work focuses on utilizing the extracted data as supporting information.

### 3 Agents with RAG (ARAG)

Our approach combines Neo4j knowledge graphs, Retrieval-Augmented Generation (RAG), and agent-based processing to enforce alignment constraints in large language models (LLMs) (see Fig. 1). This section details our framework, outlining how entities and relationships are stored, retrieved, and used to control LLM-generated descriptions of images.

### 3.1 System Overview

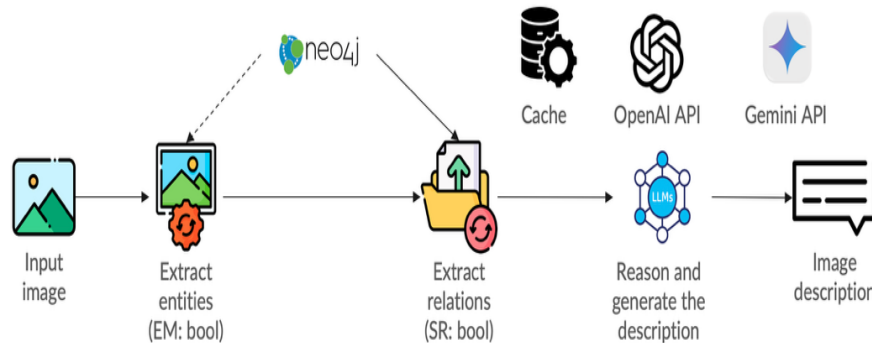


Fig. 1. System overview diagram

The proposed system consists of three main components:

- Knowledge Graph: Stores predefined entities and relationships that outline the allowed knowledge constraints.
- Image Processing Module: Extracts entities from the image using the VLM.
- RAG-Enhanced LLM Agent: Retrieves relevant entities and relationships from Neo4j and conditions the model's output within those constraints.

### 3.2 Knowledge graph

Representation. To build the alignment graph, we incorporated the Flickr30k dataset [24], containing 29000 train, 1014 validation, and 1000 test samples. Each sample from the dataset has the respective image and 5 English captions. Due to resource limitations, we built the graph on top of the first 100 train samples from the dataset. We have checked two different setups:

- Extracting entities and relations from the image captions (this turned out to provide a small number of entities and relations and was not used for the full-size experiments)
- Extracting entities and relations directly from images (was used for the full-size experiments)

The first approach led to the 184 entities and 226 relations, while the second one led to the 231 entities and 404 relations present in the DB. In both cases, GPT-4o was used as a base LLM to generate the DB.

We structure our Neo4j database as a directed graph where:

- Nodes represent entities (e.g., "Person," "Vehicle").
- Edges define relationships between entities (e.g., "drives," "owns").

A sample knowledge graph structure:

```
(:Person)-[:OWNS]->(:Vehicle)
(:Vehicle)-[:IS_LOCATED_AT]->(:Building)
```

In the schema above, there are different relations: OWNS and IS\_LOCATED\_AT. We decided to store the relations under the single RELATES relation and to store the exact relation type as a type attribute. The same procedure was performed with the Person, Vehicle, and Building combined into a single Entity with the name attribute. The actual knowledge graph snapshot is depicted in Fig. 2.

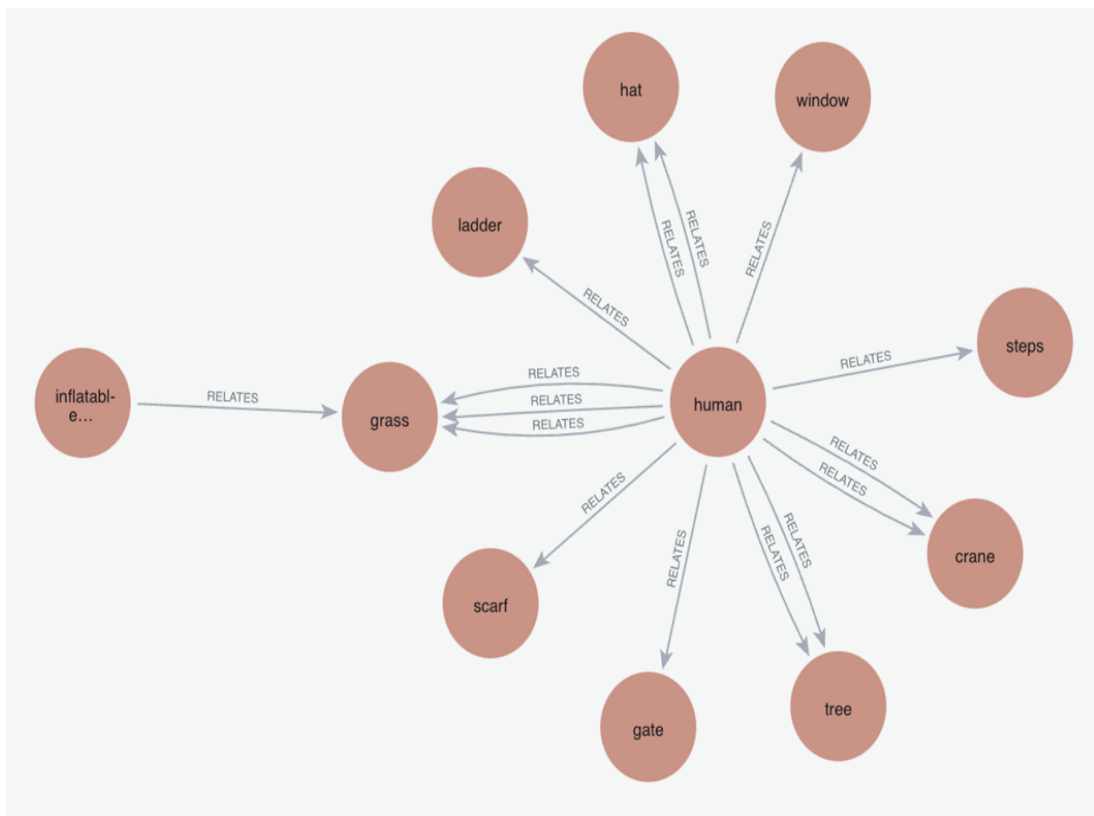


Fig. 2. Knowledge graph snapshot

Such a structure ensures that if a "Person" and "Vehicle" appear in an image, only the predefined "OWNS" relationship is considered during LLM generation. There could also be introduced a lot of additional attributes for each of the entities and relations, but for the sake of simplicity, we did not incorporate much metadata.

Querying. The system queries Neo4j to retrieve only the relations that match the list of entities. We have implemented an optional strict mode to customize the retrieval process. The strict case is when we extract two entities, cat and dog, then we extract only relations between them, like dog->bark->cat. An unstrict case would also return all the relations that match only one entity, like dog->eat->food, dog->is playing->ball.

Extracted entities matching. This step is optional and is used only during inference. By utilizing the vector embeddings, we match the extracted entities for a particular image with the list of entities present in the DB. This step is required because some entities may not be consistently extracted every time (e.g., windows/window, man/human, building/house). Without handling such situations, the results may be much worse than expected. We utilize the vectors obtained using the OpenAI Embeddings API. It provides us with 64-dimensional normalized embeddings that are further compared using a cosine similarity score and a 0.7 threshold.

In addition, we incorporated a caching procedure that pre-calculates embeddings for all the new strings and then uses them in case we face the same entity. Caching significantly reduces the actual costs for this operation. This is a local version of the semantic search that could be used on the Neo4j side, but was simplified to the local version. In the production scenario, it should be performed by a single operation in Neo4j.

### 3.3 Image Processing

The image processing module detects objects and extracts their textual representations (e.g., "car," "building"). Right now, we utilize the VLM for this task, including the input image and the prompt. This works fine for now, but can be substituted by any Zero-Shot model like Grounding DINO or something similar in the future, in case cost reduction is needed for production applications.

### 3.4 Vanilla LLM

In our setup, Vanilla LLM calls do not use DB or any other tools to perform the task.

### 3.5 Agent Processing

There are two different types of agents that we could use: tool-calling agents and code agents. After preliminary experiments, we found out that code agents perform better in planning and aligning with the step-by-step nature of the instructions.

During the implementations, we utilized the smolagents framework as a core of the agent backend. ReAct was used as the agent’s planning strategy.

RAG-Enhanced Generation. The agent conditions the LLM using retrieved knowledge, ensuring it describes only the allowed entities and relationships. Furthermore, we prompted the model to identify the exact positions of the entities in the image. There are nine available values for position: top-left, top-center, top-right, center-left, center, center-right, bottom-left, bottom-center, and bottom-right. After the initial description generation, the agent defines the positions for each entity and verifies its answer with the DB to minimize the possible hallucinations.

## 4 Experiments

To accept or reject the alignment improvements, we compared 10 head-to-head approaches utilizing 50 samples and three different metrics.

### 4.1 Data

We build the knowledge graph from the 100 train samples of the Flickr30k dataset. For the benchmarks, we selected 50 samples from the 100 train samples used during the knowledge graph generation.

Actually, the system can perform on any image with any entity-relation constraints.

### 4.2 Models

We have utilized two different API providers: OpenAI and Google. GPT-4o-mini and Gemini-2.0-flash-lite were selected as candidates. On top of vanilla LLMs, we introduced different modifications for them. ARAG suffix means that the models

utilize Agentic flow with RAG. The EM suffix means this setup uses the entities matching option, and the SR means that we included the strict relations option.

Regular LLM calls (without agentic flow) did not have access to the tools/DB. All the agentic flow setups were equipped with three tools (`load_initial_image`, `extract_entities`, and `get_data_from_neo4j`), a code interpreter, and a maximum of 7 steps to accomplish the task.

For the LLM-as-a-judge purpose, we utilize the same model we used during the knowledge graph creation – `gpt-4o`. This will mitigate the bias of inference and evaluation using the same model. To calculate the BERTScore, we incorporate `bert-base-uncased`.

### 4.3 Metrics

To evaluate the experiment results, we need to be able to understand how good the model performs at describing the image and to what extent it aligns with the DB reference. We focused on the following metrics:

- Answer Relevance [0 - 10] – Measures how well the generated response aligns with the expected content of the image. It is assessed based on the LLM judgment of the generated descriptions of the image. A higher score indicates that the response remains applicable and contextually appropriate despite alignment constraints.
- Groundedness [0 – 10] – Evaluates the extent to which the model’s output is based on retrieved knowledge rather than hallucinated information. A higher grounded-ness score indicates better alignment with structured knowledge and reduced model hallucination.
- BERTScore [0 – 1] – A widely used text similarity metric based on contextual embeddings from a pre-trained BERT model. It compares the generated description with reference captions by computing cosine similarity between token embeddings.

### 4.4 Results

The overall prediction time took ~4.5 hours to accomplish. The evaluation took ~14.5 hours due to the GPT-4o’s degraded performance stated by OpenAI in the last

few days. We have highlighted the best results per model in bold because we have to evaluate the models separately compared to the vanilla LLM setup. The final results are depicted in Table 1 below.

Table 1. ARAG vs vanilla LLM performance

Approach	Relevance	Groundedness	BERTScore
GPT-4o-mini	<b>8.72</b>	5.0	<b>0.560</b>
GPT-4o-mini-ARAG	7.06	6.0	0.538
GPT-4o-mini-ARAG-EM	6.74	<b>6.34</b>	0.542
GPT-4o-mini-ARAG-SR	6.44	5.88	0.536
GPT-4o-mini-ARAG-EM-SR	6.14	6.18	0.536
Gemini-2.0-flash-lite	<b>8.52</b>	5.92	<b>0.607</b>
Gemini-2.0-flash-lite-ARAG	6.40	6.82	0.592
Gemini-2.0-flash-lite-ARAG-EM	5.82	<b>7.68</b>	0.587
Gemini-2.0-flash-lite-ARAG-SR	6.20	6.60	0.593
Gemini-2.0-flash-lite-ARAG-EM-SR	5.94	6.92	0.589

As we can see, all the somehow aligned setups obtained higher groundedness scores compared to the vanilla LLM approach. The final groundedness score also significantly depends on the base model capabilities. In our results, we can state the significant difference between the Gemini-2.0-flash-lite and the gpt-4o-mini alignment capabilities.

Meanwhile, answer relevance dropped significantly during alignment, the answers provided by the aligned models are still valid (but, of course, less detailed). The most important thing is that we see a correlation between the extent of alignment and the answer relevance.

BERTScore is almost the same, meaning both models provide captions that partially align with the labels. This can be explained by low-detail captions describing the Flickr30k dataset. They are right to the point and were initially focused on much less capable models.

From the above experiments, we can say that for a more optimal performance enti-ties matching should be set to True while the strict relations should be set to False.

## 5 Conclusion

The scientific novelty of the presented system lies in contributing scalable LLM alignment approaches to generate outputs close to human goals and values. We introduced an approach for improving LLM alignment using Neo4j knowledge graphs, Retrieval-Augmented Generation (RAG), and agent-based processing. Our method ensures that an LLM describes only predefined entities and relationships extracted from an image, enforcing structured alignment constraints. Using vector-based entity matching, strict relationship retrieval, and agentic execution, we successfully constrained the model’s output while maintaining relevant and coherent responses.

Our experimental results demonstrate that the aligned agent-based approach significantly improves groundedness compared to a vanilla LLM with prompting. The trade-off is a slight drop in answer relevance, but overall, the method remains effective for structured and controlled generation. Importantly, BERTScore results suggest that despite these constraints, the aligned model’s responses still align with human-annotated captions at approximately the same level.

While our approach has proven effective, there are several avenues for future work:

- Scaling predictions to larger models such as GPT-4o or fine-tuned open-weight LLMs to reduce hallucinations.
- Expanding the dataset beyond the first 100 samples of Flickr30k to increase generalization.
- Verifying the performance using the Ukrainian multi30k dataset [25].
- Verify the framework performance on the specific domains, like Autonomous Driving.
- Integrating Neo4j-side semantic search for more efficient and scalable entity matching.
- Exploring techniques to enhance the alignment of vision-language models with structured knowledge.

- Researching the capabilities of image-based alignment instead of the entities and relations DB [26, 27, 28].

By combining custom constraints from knowledge graphs with retrieval-enhanced generation, our work demonstrates a promising pathway for more controllable, aligned, and factually grounded LLMs. This methodology can be extended to other alignment-sensitive applications, such as medical AI, legal document analysis, and explainable AI systems.

Source code is available on GitHub [29].

## 6 Limitations

This research was conducted as part of the master’s thesis at the Kharkiv National University of Radio Electronics. That is the reason why the KG sizes and the evaluation sizes are not that big. The overall experiment budget was around 30\$. We tested only the GPT-4o, GPT-4o-mini, and the Gemini-2.0-flash-lite models in our tests because of their affordability. We expect our approach to scale even better using the more prominent models like Gemini-2.0-pro and Claude-sonnet-3.7.

The obtained framework sometimes faces the output token limits during the evaluations. These are just being retried for now.

We have conducted full-size experiments only using the Code Agent. After the preliminary tests, the Tool Calling Agent almost always showed worse results.

We did not evaluate our framework on the new data samples that were not used to build the KG.

Image size is limited to 20MB (current OpenAI limitation).

## Acknowledgments

This publication is based upon work from COST Action GOBLIN - Global Network on Large-Scale, Cross-domain and Multilingual Open Knowledge Graphs (CA23147), supported by COST (European Cooperation in Science and Technology).

## References

1. Erkut Erdem, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii, Oleksii Turuta, Aykut Erdem, Iacer Calixto, Elena Lloret, Elena-Simona Apostol, Ciprian-Octavian Truică, Branislava Šandrih, Sanda Mar-tinčić-Ipšić, Gábor Berend, Albert Gatt, and Grăzina Korvel, Neural Natural Language Generation: A Survey on Multilinguality, Multimodality, Controllability and Learning. *J. Artif. Int. Res.* 73 (2022).
2. J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou et al., Chain-of-thought prompting elicits reasoning in large language models. In: *Advances of the 36th Neural Information Processing Systems (NeurIPS)*, Curran Associates, Inc., pp. 24824-24837 (2022).
3. X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, Self-consistency improves chain of thought reasoning in language models. In: *Proceedings of the 11th International Conference on Learning Representations (ICLR)* (2023).
4. S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, Tree of thoughts: Deliberate problem solving with large language models. In: *Advances of the 37th Neural Information Processing Systems (NeurIPS)*, pp. 11809-11822 (2023).
5. S. Zhou, U. Alon, F. F. Xu, Z. Jiang, and G. Neubig, Docprompting: Generating code by retrieving the docs. In: *Proceedings of the 11th Conference on Learning Representations (ICLR)* (2023).
6. T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, Large language models are zero-shot reasoners. In: *Advances of the 36th Neural Information Processing Systems (NeurIPS)*, Curran Associates, Inc., pp. 22199–22213 (2022).
7. A. Creswell, M. Shanahan, and I. Higgins, Selection-inference: Exploiting large language models for interpretable logical reasoning. In: *Proceedings of the 11th Conference on Learning Representations (ICLR)* (2023).
8. N. Shinn, B. Labash, and A. Gopinath, Reflexion: an autonomous agent with dynamic memory and self-reflection. In: *Proceedings of the 37th International*

Conference on Neural Information Processing Systems (NIPS), Curran Associates Inc., Red Hook, NY, USA, Article 377, pp. 8634–8652 (2023).

9. M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk et al., Graph of thoughts: Solving elaborate problems with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2024).

10. E. Zelikman, Y. Wu, J. Mu, and N. Goodman, STaR: Bootstrapping Reasoning With Reasoning. In: Advances of the 36th Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., pp. 15476–15488 (2022).

11. H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods. In: 17th international semantic web conference (ISWC), IOS Press, NLD, pp. 489–508 (2017).

12. Q. Wang, Z. Mao, B. Wang, and L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 12, pp. 2724–2743 (2017).

13. Y. Jing, Y. Yang, X. Wang, M. Song, and D. Tao, Meta-aggregator: Learning to aggregate for 1-bit graph neural networks. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), pp. 5281–5290 (2021).

14. D. Sanmartin, Kg-rag: Bridging the gap between knowledge and creativity, arXiv pre-print (2024).

15. Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr, Knowledge graph prompting for multi-document question answering. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI), pp. 19206–19214 (2024).

16. X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In: Proceedings of the Advances in 37th Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., pp. 132876–132907 (2024).

17. X. Li, R. Zhao, Y. K. Chia, B. Ding, S. Joty, S. Poria, and L. Bing, Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In: Proceedings of the 12th International Conference on Learning Representations (ICLR) (2024).
18. L. Luo, Y.-F. Li, G. Haffari, and S. Pan, Reasoning on graphs: Faithful and interpretable large language model reasoning. In: Proceedings of the 12th International Conference on Learning Representations (ICLR) (2024).
19. J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, H.-Y. Shum, and J. Guo, Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. In: Proceedings of the 12th International Conference on Learning Representations (ICLR) (2024).
20. H. Liu, S. Wang, Y. Zhu, Y. Dong, and J. Li, Knowledge graph-enhanced large language models via path selection. In: Findings of the 62nd Association for Computational Linguistics (ACL), Bangkok, Thailand, pp. 6311-6321 (2024).
21. R.-C. Chang and J. Zhang, CommunityKG-RAG: Leveraging Community Structures in Knowledge Graphs for Advanced Retrieval-Augmented Generation in Fact-Checking, arXiv preprint (2024).
22. Y. Mu, P. Niu, K. Bontcheva, and N. Aletras, Predicting and analyzing the popularity of false rumors in weibo, *Expert Systems with Applications*, vol. 243, p. 122791 (2024).
23. A. Kau, X. He, A. Nambissan, A. Astudillo, H. Yin, and A. Aryani, Combining knowledge graphs and large language models, arXiv preprint (2024).
24. Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier, From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions, *Transactions of the Association for Computational Linguistics*, pp. 67–78 (2014).
25. Nataliia Saichyshyna, Daniil Maksymenko, Oleksii Turuta, Andriy Yerokhin, Andrii Babii, and Olena Turuta, Extension Multi30K: Multimodal Dataset for Integrated Vision and Language Research in Ukrainian. In: Proceedings of the 2nd Ukrainian Natural Language Processing Workshop (UNLP), pp. 54–61 (2023).

26. Kyrychenko, I., Tereshchenko, G., & Smelyakov, K., Optimized Indexing Method in a Hybrid Image Storage Model for Efficient Storage and Access in Big Data Environ-ments. In: Proceedings of the 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), 1-4 (2024).
27. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I. and Kobylin, O., Reducing compu-tational costs by compressing the structural description in image classification meth-ods, Advanced Information Systems 9, 5-12 (2025).
28. Gorokhovatskyi, Volodymyr, et al. "Search for visual objects by request in the form of a cluster representation for the structural image description." Advances in Electri-cal and Electronic Engineering 21.1 (2023).
29. Implementation of the paper, <https://github.com/Vlad-Fliahin/LLM-alignment-with-ARAG>, last accessed 2025/04/20.

## ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на  
відповідність оформлення вимогам ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент  
(посада)

програмної інженерії  
(кафедра)

ПЗМ-23-4  
(група)

**Флягін Владислав Костянтинович**

(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні позначки	
	7.15 Додатки	

зауважень немає

Експерт

\_\_\_\_\_

(підпис)

**Олена ОЛІЙНИК**

(прізвище, ініціали)

03.06.2025

Рисунок Г.1 – Висновок перевірки на відповідність вимогам ДСТУ

## ДОДАТОК Д

### Приклади запитань та відповідей

На рисунку Д.1 наведено приклад питань та відповідей з тестового датасету.

```
{
  "question": "What is a standout feature of the product, My Monster Friend?",
  "answer": "Expanded options allow players to personalize their virtual pets with diverse appearances.",
  "src": "neonbyte-interactive"
},
{
  "question": "In which area has Vertex MedTech partnered with the University of Health Sciences?",
  "answer": "Research initiatives and educational programs.",
  "src": "vertex-medtech"
},
{
  "question": "What is the price of the EnergyVault X storage solution per unit?",
  "answer": "$6000 per unit.",
  "src": "blue-horizon-energy"
},
{
  "question": "Which company is partnered with Vertex MedTech for developing biocompatible materials?",
  "answer": "BioInnovate Solutions",
  "src": "vertex-medtech"
},
{
  "question": "What does QubitSim Pro offer to universities?",
  "answer": "A quantum simulation software designed for education, including classroom integration and demonstrations.",
  "src": "quantum-forge-technologies"
},
{
  "question": "In what year was NeonByte Interactive founded?",
  "answer": "2017",
  "src": "neonbyte-interactive"
},
{
  "question": "What is the price of the AgroPrime Soybean AS-215 per unit?",
  "answer": "$295.00",
  "src": "agro-team-inc"
},
{
  "question": "What is a strategic initiative of NeonByte Interactive?",
  "answer": "Community engagement with regular developer live streams and feedback forums.",
  "src": "neonbyte-interactive"
},
{
  "question": "What features does the LungGuard Pulmonary Device include?",
  "answer": "Real-time data analytics, portable design, and cloud-based patient records.",
  "src": "vertex-medtech"
},
{
  "question": "Who is the lead developer for Fantasy Realms: The Awakening?",
  "answer": "Claire Nakamura",
  "src": "neonbyte-interactive"
},
}
```

Рисунок Д.1 – 10 перших запитань тестового датасету (створений самостійно)

## ДОДАТОК Е

Основний код для проведення експериментів

```

import traceback
import json
import tomlib as tomlib
from pathlib import Path
from dotenv import load_dotenv
from tqdm import tqdm

import numpy as np
from llama_index.core import (
    Settings,
    VectorStoreIndex,
    get_response_synthesizer,
    SimpleDirectoryReader,
    StorageContext,
)
from llama_index.core.node_parser import import
TokenTextSplitter, SentenceSplitter
from llama_index.core.query_engine import import
RetrieverQueryEngine
from llama_index.core.indices.vector_store.retrievers
import VectorIndexRetriever
from llama_index.embeddings.openai import import
OpenAIEmbedding
from llama_index.vector_stores.neo4jvector import
Neo4jVectorStore
from llama_index.llms.google_genai import GoogleGenAI
from deepeval.metrics import (
    AnswerRelevancyMetric,
    FaithfulnessMetric,
    ContextualRelevancyMetric,
)
from deepeval.test_case import LLMTTestCase
from deepeval import evaluate
from llama_index.llms.together import TogetherLLM
from neo4j import GraphDatabase

load_dotenv()

def id_func(index, document):
    """Creates a specific chunk id"""

```

```

    document_name =
Path(document.metadata["file_name"]).stem
    return f"{document_name}-{index}"

def clear_neo4j(cfg):
    """
    Clears all nodes and relationships from a Neo4j
    database.

    Args:
        uri (str): The URI of the Neo4j database.
        user (str): The username for connecting to the
    database.
        password (str): The password for connecting to
    the database.
    """
    driver = GraphDatabase.driver(
        cfg["config"]["env"]["NEO4J_URI"],
        auth=(
            cfg["config"]["env"]["NEO4J_USER"],
            cfg["config"]["env"]["NEO4J_PASSWORD"],
        ),
    )
    with driver.session() as session:
        session.run("MATCH (n) DETACH DELETE n")
        # session.run(f"DROP INDEX
`{cfg['config']['rag']['setup']['index_name']}`")
    driver.close()

PARAMS = {
    "hybrid_search": [True, False], # to test
    # "agent": [True, False],
    "llm": ["llama-3.2-11B-vision-instruct", "gemini-1.5-
pro"],
    # "embedding_provider": ["openai", "huggingface"],
    # "embedding_dimension": [128, 512, 1024],
    "embedding_dimension": 512,
    # "similarity_cutoff": [0.7, 0.8, 0.9],
    "similarity_top_k": [3, 5],
    "chunk_size": [128, 256],
    "preparation": ["TokenTextSplitter",
"SentenceSplitter"],
    "chunk_overlap": 0.2,

```

```

}

number_of_experiments = 1
for key, value in PARAMS.items():
    if isinstance(value, list):
        number_of_experiments *= len(value)
print(f"{number_of_experiments=}")

CLEAR_DB = False
UPDATE_DB = False
PREDICT = False
EVALUATE = True

DATASET_SIZE = 50

i = 0
##### setup experiments
for hs in PARAMS["hybrid_search"]:
    for llm_name in PARAMS["llm"]:
        for top_k in PARAMS["similarity_top_k"]:
            for chunk_size in PARAMS["chunk_size"]:
                for preparation in PARAMS["preparation"]:
                    print(
                        f"{i=}      {hs=},      {llm_name=},      {top_k=},
{chunk_size=}, {preparation=}"
                    )
                    i += 1

##### parse configuration
with open("./config.toml", "rb") as file:
    cfg = tomlib.load(file)

cfg["config"]["rag"]["setup"]["hybrid_search"]
= hs
#   cfg['config']['rag']['models']['llm_openai']
= llm_name

cfg["config"]["rag"]["setup"]["similarity_top_k"] = top_k
cfg["config"]["rag"]["setup"]["chunk_size"]      =
chunk_size
cfg["config"]["rag"]["setup"]["preparation"]     =
preparation

# print(f'{cfg=}')

```

```

##### load data
dataset = json.load(open("./data/q&a.json",
"r"))

base_path = Path("./data/companies")
doc_paths = [
    path.as_posix()
    for path in base_path.iterdir()
    if path.is_file()
]

##### load models
# initialize LLM
if llm_name == "gemini-1.5-pro":
    llm = GoogleGenAI(
        model=llm_name,

api_key=cfg["config"]["env"]["GEMINI_API_KEY"],
    temperature=cfg["config"]["rag"]["models"]["temperatur
e"],
    )
elif llm_name == "llama-3.2-11B-vision-
instruct":
    llm = TogetherLLM(
        model="meta-llama/Llama-3.2-11B-Vision-
Instruct-Turbo"
    )
    # time.sleep(60)

# initialize EM
embed_model = OpenAIEmbedding(
    model=cfg["config"]["rag"]["models"]["em_openai"],
    api_key=cfg["config"]["env"]["OPENAI_API_KEY"],
    dimensions=cfg["config"]["rag"]["setup"]["embedding_di
mension"],
)

Settings.llm = llm
Settings.embed_model = embed_model

##### generate embeddings & fill the DB
if CLEAR_DB:
    clear_neo4j(cfg)

if UPDATE_DB:

```

```

# initialize a file reader
reader
SimpleDirectoryReader(input_files=doc_paths)

# load documents into LlamaIndex Documents
documents = reader.load_data()

vector_store = Neo4jVectorStore(
    username=cfg["config"]["env"]["NEO4J_USER"],
    password=cfg["config"]["env"]["NEO4J_PASSWORD"],
    url=cfg["config"]["env"]["NEO4J_URI"],
    embedding_dimension=cfg["config"]["rag"]["setup"][
        "embedding_dimension"
    ],
    distance_strategy=cfg["config"]["rag"]["setup"][
        "distance_strategy"
    ],
    index_name=cfg["config"]["rag"]["setup"]["index_name"]
,
    text_node_property=cfg["config"]["rag"]["setup"][
        "text_node_property"
    ],
    hybrid_search=cfg["config"]["rag"]["setup"][
        "hybrid_search"
    ],
)

# setup context
storage_context
StorageContext.from_defaults(
    vector_store=vector_store
)

if preparation == "TokenTextSplitter":
    parser = TokenTextSplitter(
        chunk_size=cfg["config"]["rag"]["setup"]["chunk_size"]
,
        chunk_overlap=int(
            np.ceil(
                cfg["config"]["rag"]["setup"]["chunk_overlap"]
                *
                cfg["config"]["rag"]["setup"]["chunk_size"]
            )
        ),

```

```

separator=cfg["config"]["rag"]["setup"]["separator"],
        id_func=id_func,
    )
    elif preparation == "SentenceSplitter":
        parser = SentenceSplitter(

chunk_size=cfg["config"]["rag"]["setup"]["chunk_size"],
        chunk_overlap=int(
            np.ceil(
                cfg["config"]["rag"]["setup"]["chunk_overlap"]
                *
cfg["config"]["rag"]["setup"]["chunk_size"]
            )
        ),

separator=cfg["config"]["rag"]["setup"]["separator"],
        id_func=id_func,
    )

        # parse documents into nodes (chunks)
        nodes =
parser.get_nodes_from_documents(documents)

        db_index = VectorStoreIndex(
            nodes,
            storage_context=storage_context,
            embed_model=embed_model,
            show_progress=False,
        )
    else:
        vector_store = Neo4jVectorStore(
            username=cfg["config"]["env"]["NEO4J_USER"],

password=cfg["config"]["env"]["NEO4J_PASSWORD"],
            url=cfg["config"]["env"]["NEO4J_URI"],

embedding_dimension=cfg["config"]["rag"]["setup"][
            "embedding_dimension"
        ],

distance_strategy=cfg["config"]["rag"]["setup"][
            "distance_strategy"
        ],

```

```

index_name=cfg["config"]["rag"]["setup"]["index_name"]
',
text_node_property=cfg["config"]["rag"]["setup"][
    "text_node_property"
],
hybrid_search=cfg["config"]["rag"]["setup"][
    "hybrid_search"
],
)

db_index =
VectorStoreIndex.from_vector_store(vector_store)

##### configure different tools
# custom retriever
retriever = VectorIndexRetriever(
    index=db_index,
    similarity_top_k=cfg["config"]["rag"]["setup"][
        "similarity_top_k"
    ],
    vector_store_query_mode=cfg["config"]["rag"]["setup"][
        "vector_store_query_mode"
    ],
)

# custom response synthesizer
response_synthesizer =
get_response_synthesizer(

response_mode=cfg["config"]["rag"]["setup"]["response_mode"
]

)

# combine custom query engine
query_engine = RetrieverQueryEngine(
    retriever=retriever,
response_synthesizer=response_synthesizer
)

##### configure setup
setup = query_engine

##### predict
if PREDICT:
    predictions = []

```

```

        for item in tqdm(dataset[:DATASET_SIZE]):
            try:
                response =
                setup.query(item.get("question"))

                output = response.response
                context = [
                    node.get_content() for node in
                response.source_nodes
                ]
                ids = [node.id_ for node in
                response.source_nodes]

                predictions.append(
                    {
                        "source_nodes_ids": ids,
                        "source_nodes": context,
                        "response": output,
                    }
                )
            except Exception as e:
                print(e)
                traceback.format_exc()

            ##### save the results
            with open(
                f"./predictions/i_{i}_hs_{hs}_llm_{llm_name}_topk_{top
                _k}_cs_{chunk_size}_prep_{preparation}.json",
                "w",
            ) as f:
                json.dump(predictions, f)
            else:
                predictions = json.load(
                    open(

                f"./predictions/i_{i}_hs_{hs}_llm_{llm_name}_topk_{top_k}_c
                s_{chunk_size}_prep_{preparation}.json",
                "r",
                )
                )

            ##### evaluate
            if EVALUATE:
                test_cases = []

```

```

        for j, item
enumerate (tqdm(dataset[:DATASET_SIZE])):
    # define test case
    test_case = LLMTTestCase(
        input=item.get("question"),
        actual_output=predictions[j]["response"],
        expected_output=item.get("answer"),
        retrieval_context=predictions[j]["source_nodes"],
    )
    test_cases.append(test_case)

    answer_relevancy_metric =
AnswerRelevancyMetric(
    model=cfg["config"]["rag"]["models"]["llm_openai"]
)
    faithfulness_metric = FaithfulnessMetric(
    model=cfg["config"]["rag"]["models"]["llm_openai"]
)
    contextual_relevancy_metric =
ContextualRelevancyMetric(
model=cfg["config"]["rag"]["models"]["llm_openai"]
)

    # time.sleep(60)

    results = evaluate(
        test_cases=test_cases,
        metrics=[
            answer_relevancy_metric,
            faithfulness_metric,
            contextual_relevancy_metric,
        ],
        max_concurrent=50,
    )

    evaluations = []
    for item in results.test_results:
        evaluations.append(
            {
                "answer_relevancy": {
                    "reason": item.metrics_data[0].reason,
                    "score": item.metrics_data[0].score,
                },
                "faithfulness": {

```

```
        "reason": item.metrics_data[1].reason,
        "score": item.metrics_data[1].score,
    },
    "contextual_relevancy": {
        "reason": item.metrics_data[2].reason,
        "score": item.metrics_data[2].score,
    },
    }
)

##### save the results
with open(
    f"./evaluations/i_{i}_hs_{hs}_llm_{llm_name}_topk_{top
_k}_cs_{chunk_size}_prep_{preparation}.json",
    "w",
) as f:
    json.dump(evaluations, f)
```