ЛАБОРАТОРНА РОБОТА № 6

Оцінювання якості RAG (Evaluation Framework)

Мета роботи. Формалізувати методологію оцінювання RAG-систем через автоматичні метрики якості та статистичну верифікацію результатів.

Науково-теоретичне обґрунтування

Метрики оцінювання RAG

Метрика	**Що вимірює**	**Формула / діапазон**
Faithfulness	Частка фактів з контексту	підтверджених тверджень / всіх тверджень → [0,1]
Context Recall	Частка знайдених золотих фактів	gold facts у контексті / всіх gold facts → [0,1]
Hallucination Rate	Частка відповідей з вигаданими фактами	hallucinated / total responses → [0,1]
Exact Match	Точний збіг відповіді з еталоном	1 якщо normalize(answer) == normalize(gold), інакше 0
F1-score	Баланс Precision та Recall токенів	2×P×R / (P + R), де P/R - по токенах відповіді

Зміст роботи

Завдання 1. *Створити gold-dataset із 20 контрольних питань.*

Методичні рекомендації

1. *Gold-dataset* - це набір пар (питання, еталонна відповідь), де правильна відповідь відома заздалегідь. Для датасету *Netflix*: оберіть 20 конкретних фільмів, складіть питання типу «Про що фільм ….?» і запишіть точну відповідь з поля *description*.

2. Розподіліть питання за складністю: 7 простих (факт у одному реченні), 7 середніх (потрібно синтезувати 2–3 факти), 6 складних (абстрактна тема, кілька фільмів).

3. Збережіть *gold-dataset* у форматі CSV:

columns = [question, gold_answer, relevant_doc_ids, difficulty].

Файл буде використовуватись для всіх наступних завдань.

4. Переконайтесь, що еталонні відповіді є у датасеті: пошукайте документи за ключовими словами і скопіюйте відповідний текст з description - без домислювань.

Створення даних на основі метаданих Netflix (**ПРИКЛАД**):

```
data = {
    'question': [
```

```
        # Easy (Simple facts)
        "Про що фільм 'Inception'?",
        "Хто головний герой серіалу 'Stranger Things'?",
        "Який сюжет мультфільму 'Spirited Away'?", …
        # Medium (Synthesis of 2-3 facts)
        "Які спільні риси сюжетів 'Inception' та 'Interstellar' у контексті
ролі батька?",…
        "Порівняйте атмосферу серіалів 'Dark' та 'Stranger Things'.",
        # Hard (Abstract/Multiple films)
        "Як тема штучного інтелекту розкривається у 'The Matrix' та 'Ex
Machina'?",
        "Аналіз соціальної нерівності на прикладі 'Parasite' та
'Snowpiercer'.",…
    ],
    'gold_answer': [
        "Злодій, який краде корпоративні секрети через технологію обміну
снами, отримує завдання вкорінити ідею в розум дитини.",
        "Дівчинка потрапляє у світ духів і намагається врятувати батьків,
перетворених на свиней.",
        "Бетмен бореться з хаосом, який сіє Джокер у Готем-сіті.",…
    ],
    'relevant_doc_ids': [[1], [2], [3], [4], [5], [6], [7], [1, 5], [8, 2],
[4], [9], [10], [11], [12], [6, 13], [9, 14], [1, 4, 5], [15, 16], [4, 17],
[3, 18]],
    'difficulty': ['easy']*7 + ['medium']*7 + ['hard']*6
}
```

Збереження дата сету:

```
df = pd.DataFrame(GOLD_DATA)
    df.to_csv("gold_dataset.csv", index=False, encoding="utf-8")
```

Завдання 2. *Реалізувати автоматичну оцінку через cosine similarity та semantic equivalence, для порівняння згенерованої відповіді з еталоном.*

Методичні рекомендації

На цьому етапі потрібно кількісно виміряти, наскільки «близько» згенерована відповідь до еталону.

Cosine Similarity (Косинусна схожість). Тексти перетворюються на вектори (у просторі ознак). Косинусна схожість вимірює косинус кута між двома векторами. Якщо вектори дивляться в одному напрямку, косинус дорівнює 1 (максимальна схожість). Це ***лексична*** метрика, якщо використовувати TF-IDF (враховує спільні слова), або ***семантична***, якщо використовувати Embeddings (враховує зміст).

Token F1-Score - це гармонійне середнє між *Precision* (точність) та *Recall* (повнота) на рівні окремих слів (токенів), вона показує, наскільки повно модель відтворила ключові слова з еталону і чи не «нафантазувала» зайвого. На відміну від косинуса, вона жорстко прив'язана до фактичної наявності конкретних слів.

Faithfulness (Вірність контексту) - це ключова метрика RAG, яка перевіряє, чи не вигадала модель щось від себе (галюцинації). Ми розбиваємо відповідь на твердження і перевіряємо, чи кожне з них логічно випливає зі знайденого документа. У коді ми використали проксі-метод через схожість речень відповіді з контекстом.

Виконання завдання:

1. *Cosine similarity* між *embedding(gold_answer)* та embedding(generated_answer) є швидкою проксі-метрикою семантичної схожості. Значення >0.85 зазвичай означає семантичну еквівалентність.

2. Для точнішої оцінки використовуйте *BERTScore* або саме *RAGAS*. RAGAS використовує LLM як суддю (LLM-as-Judge): передає пару (відповідь, gold) до іншого LLM і отримує структуровану оцінку.

3. Реалізуйте F1-score на рівні токенів, токенізуйте обидві відповіді, порахуйте *Precision* та *Recall* за унікальними токенами, обчисліть *F1*. Це класична метрика з *QA-benchmarks (SQuAD).*

– Порівняйте три метрики для кожної пари: *Cosine Sim*, *F1-Score*, *RAGAS Faithfulness*. Виявіть розбіжності - це допоможе зрозуміти обмеження кожної метрики.

ПРИКЛАД виконання

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from collections import Counter

def get_cosine_sim(str1, str2):
    vec = TfidfVectorizer().fit_transform([str1, str2])
    return cosine_similarity(vec[0:1], vec[1:2])[0][0]

def get_f1_score(gold, gen):
    gold_tokens = gold.lower().split()
    gen_tokens = gen.lower().split()
    common = Counter(gold_tokens) & Counter(gen_tokens)
```

```
    num_same = sum(common.values())
    if num_same == 0: return 0
    precision = num_same / len(gen_tokens)
    recall = num_same / len(gold_tokens)
    return 2 * (precision * recall) / (precision + recall)
```

Імітація згенерованих відповідей (для прикладу)

```
gold_df['generated_answer'] = gold_df['gold_answer'].apply(lambda x:
x if "easy" in x else x[:len(x)//2])
```

Розрахунок метрик

```
results = []
for idx, row in gold_df.iterrows():
    cos = get_cosine_sim(row['gold_answer'],
row['generated_answer'])
    f1 = get_f1_score(row['gold_answer'], row['generated_answer'])
```

Faithfulness proxy: спрощена логіка

```
    faithfulness = cos * 0.95 # Умовний коефіцієнт для демонстрації
    results.append({'cos': cos, 'f1': f1, 'faithfulness':
faithfulness})

metrics_df = pd.concat([gold_df, pd.DataFrame(results)], axis=1)
print(metrics_df[['difficulty', 'cos', 'f1',
'faithfulness']].head())
```

Завдання 3. *Провести статистичну перевірку: t-test, p-value, для того, щоб підтвердити, що різниця між двома системами (наприклад, LLM-only та RAG) не є випадковою.*

Методичні рекомендації

Нульова гіпотеза (H0) - припущення про те, що нововведення (RAG) насправді нічого не змістило і різниця в результатах є чисто випадковою.

Парний t-тест (Paired t-test) використовується, коли ми тестуємо *одні й ті самі питання* двома різними методами. Це прибирає фактор "складності питання", оскільки ми порівнюємо результати попарно для кожного кейсу.

P-value (Значущість) - це ймовірність отримати такий (або більший) відрив у результатах, якщо насправді системи однакові. Поріг 0.05, якщо $p < 0.05$, ми кажемо: "Ймовірність випадковості менше 5%, отже, RAG дійсно працює краще".

Тест Вілкоксона використовується, якщо дані розподілені нерівномірно (не за Гаусом). Це "запасний план" для невеликих або специфічних датасетів.

Виконання завдання:

1. Сформулюйте нульову гіпотезу H_0: «RAG не покращує Answer Relevancy порівняно з LLM-only». Якщо p-value < 0.05 - відхиляємо H_0 і стверджуємо статистично значуще покращення.

2. Використайте парний t-тест (*scipy.stats.ttest_rel*): він порівнює метрики по одних і тих самих запитах між двома системами. Це правильніше за незалежний t-тест, оскільки складність запитів може варіюватись.

3. Якщо дані не відповідають нормальному розподілу (перевірте через *scipy.stats.shapiro*), використайте критерій Вілкоксона (*scipy.stats.wilcoxon*) - непараметричний аналог парного t-тесту.

4. Зафіксуйте у звіті: *t-statistic*, *p-value*, ступені свободи. Ефект вважається значущим при $p < 0.05$ і великим при $|t| > 2$.

ПРИКЛАД виконання

```
from scipy import stats
```

Створимо синтетичні дані оцінок для двох систем. Припустимо, ми оцінювали Answer Relevancy (0-1)

```
llm_only_scores = np.random.normal(0.6, 0.15, 20).clip(0, 1)
rag_scores = np.random.normal(0.75, 0.1, 20).clip(0, 1)
```

Перевірка на нормальність (Shapiro-Wilk)

```
_, p_norm_llm = stats.shapiro(llm_only_scores)
_, p_norm_rag = stats.shapiro(rag_scores)

print(f"Нормальність (p-value): LLM={p_norm_llm:.3f}, RAG={p_norm_rag:.3f}")
```

Вибір тесту

```
if p_norm_llm > 0.05 and p_norm_rag > 0.05:
    t_stat, p_val = stats.ttest_rel(llm_only_scores, rag_scores)
    test_name = "Paired T-test"
else:
    t_stat, p_val = stats.wilcoxon(llm_only_scores, rag_scores)
    test_name = "Wilcoxon Signed-Rank Test"

print(f"Статистика: {t_stat:.4f}")
print(f"p-value: {p_val:.4f}")
```

```
if p_val < 0.05:
    print("Результат: Відхиляємо H0. RAG значущо покращує якість.")
else:
    print("Результат: Не вдалося відхилити H0. Різниця не є значущою.")
```

Завдання 4. *Побудувати довірчі інтервали, для наочного представлення стабільності системи.*

Методичні рекомендації

Якщо *t-test* каже "чи є різниця", то довірчий інтервал каже "наскільки ми впевнені в результаті".

Довірчий інтервал (95% CI) - це діапазон значень, у якому з імовірністю 95% знаходиться справжнє середнє значення якості системи. Якщо ми проведемо тест ще 100 разів на інших питаннях, у 95 випадках результат потрапить у цей проміжок.

Error Bars (Вусики похибки) на графіках. Короткі вусики - система стабільна. Вона видає передбачувану якість на всіх питаннях. Довгі вусики - систему "штормить". На одні питання вона відповідає ідеально, а на інших - повністю провалюється.

Якщо довірчі інтервали двох систем на графіку сильно перекриваються (overlap), то навіть при різниці в середніх показниках ми не можемо впевнено сказати, яка система краща.

Виконання завдання:

1. 95% довірчий інтервал для середнього значення метрики:

$$CI = mean \pm t^*(s/\sqrt{n}),$$

де t* - критичне значення t-розподілу при α=0.05, s - стандартне відхилення, n - кількість спостережень.

2. Використайте *scipy.stats.t.interval()* або *stats.sem()* для обчислення. Вузький CI означає стабільну поведінку системи; широкий - велику варіативність відповідей.

3. Побудуйте графік: *grouped barplot* з *error bars* (довірчими інтервалами) для LLM-only та RAG за кожним рівнем складності (easy/medium/hard). Це наочно показує статистичну значущість різниці.

ПРИКЛАД виконання

```
import matplotlib.pyplot as plt
import seaborn as sns

def calculate_ci(data):
    mean = np.mean(data)
    sem = stats.sem(data)
    ci = stats.t.interval(0.95, len(data)-1, loc=mean, scale=sem)
    return mean, ci[1] - mean #повертаємо середнє та розмір помилки
```

Групування за складністю

```
summary = metrics_df.groupby('difficulty')['cos'].agg(['mean', 'sem',
'count'])
summary['ci_yerr'] = summary.apply(lambda x: stats.t.ppf(0.975, x['count']-
1) * x['sem'], axis=1)
```

Побудова графіка

```
plt.figure(figsize=(10, 6))
plt.bar(summary.index, summary['mean'], yerr=summary['ci_yerr'], capsize=10,
color='skyblue', edgecolor='black')
plt.title('Середня Cosine Similarity з 95% CI за рівнем складності')
plt.ylabel('Score')
plt.ylim(0, 1.1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Завдання 5. *Індивідуальні завдання*

№	Завдання
1, 6, 11	Інтегрувати RAGAS (pip install ragas) і порівняти автоматичні оцінки з ручними
2, 7, 12	Побудувати confusion matrix для класифікації відповідей: correct / hallucinated / irrelevant
3, 8, 13	Дослідити залежність метрик від розміру gold-dataset (5, 10, 15, 20 питань)
4, 9, 14	Реалізувати bootstrap confidence intervals (1000 ітерацій) замість аналітичних
5, 10, 15	Порівняти F1-score та BERTScore: при яких типах відповідей вони розходяться?

Контрольні запитання

1. Чим Gold-dataset відрізняється від звичайної бази знань у RAG?
2. Чому Cosine Similarity на основі TF-IDF може бути неточною метрикою для оцінки "Faithfulness"?
3. Що означає метрика F1-score у контексті QA систем?
4. Чому для порівняння двох версій RAG-системи краще використовувати парний (paired) t-test, а не незалежний?
5. Яка роль p-value у статистичному висновку?

6. Що таке LLM-as-a-Judge і які переваги він має перед автоматичними метриками?
7. Чому важливо розділяти тестові питання за рівнями складності?
8. Яку інформацію надає ширина довірчого інтервалу (CI) на графіку?
9. У якому випадку замість t-тесту слід використовувати тест Вілкоксона?
10. Які 3 основні компоненти оцінює фреймворк RAGAS??