

# System RAG z kwantyzowanym LLM

Jacek Tyszkiewicz  
Bartłomiej Stepniewski

Akademia Górniczo-Hutnicza  
im. Stanisława Staszica w Krakowie (AGH)

## 1 Wstęp

Podczas realizacji projektu, wykonano następujące czynności:

- Stworzono system wykorzystujący mechanizm RAG (Retrieval-Augmented Generation).
- Uwzględniono bazę danych wektorowych (Qdrant) oraz model językowy ().
- Przy użyciu Streamlit, stworzono prosty pulpit z interfejsem czatu.
- Podczas generowania odpowiedzi, podświetlono kontekstowe osadzenia (embeddings), które są najbliższe zapytaniu.

## 2 Szczegóły techniczne

Warstwa	Narzędzia
UI	Streamlit
LLM	DeepSeek-R1 7B - Ollama
Embeddings	Snowflake Arctic-Embed-L (SentenceTransformers)
Vector DB	Qdrant
PDF parsing	PyPDF
Kod	Python 3.10, ollama-python, qdrant-client, sentence-transformers, streamlit
Hardware	GPU RTX 4070 (CUDA), CPU x86-64

## 3 Logika działania systemu RAG

### 1. Dodanie bazy wektorowej (Qdrant)

- Każdy dokument (np. fragment tekstu, strona PDF) zamieniany jest na wektor przy pomocy modelu embeddingowego (np. Snowflake Arctic-Embed-L).
- Wektory wraz z treścią są zapisywane w bazie Qdrant jako dokumenty z metadanymi (text, source, page).

- Baza jest przeszukiwana semantycznie – na podstawie podobieństwa wektorowego do zapytania użytkownika (metryka: COSINE).

## 2. Łączenie z LLM

- Gdy użytkownik zada pytanie, system oblicza embedding pytania i znajduje najbardziej podobne dokumenty (top  $k$ ).
- Najtrafniejsze fragmenty są przekazywane do modelu językowego jako kontekst.
- Model generuje odpowiedź w oparciu o ten kontekst — bez korzystania z własnej wiedzy.

## 3. Budowanie interfejsu w Streamlit

- Interfejs umożliwia: przesyłanie PDF, zadawanie pytań, przegląd trafień i wyświetlenie końcowej odpowiedzi.
- Użytkownik widzi również trafione fragmenty wraz z oceną dopasowania (score), co ułatwia zrozumienie źródła odpowiedzi.

## 4. Wyróżnij kontekst

- System wyraźnie pokazuje, z których dokumentów pochodzi cytowana odpowiedź.
- Trafienia z wysokim score są wyróżniane kolorystycznie (np. jasne tło powyżej 0,8).

Prompt użyty w systemie:





```
from ollama import Client
client = Client(host="http://localhost:11434")

MODEL_NAME = "deepseek-r1:7b"

def generate(question: str, context: list[str]) -> str:
    top_hit = context[0]
    prompt = (
        "You are a **strict factual assistant**.\n"
        "Rules:\n"
        "1. Use ONLY the information in CONTEXT.\n"
        "2. Quote the sentence that contains the answer.\n"
        "3. Give the answer in one short sentence.\n"
        "4. If the answer is missing, say: I don't know.\n"
        "5. Do NOT reveal your reasoning or use external knowledge.\n\n"
        f"### CONTEXT\n{top_hit}\n\n"
        f"### QUESTION\n{question}\n\n"
        "### ANSWER (quote + short answer):"
    )
    resp = client.generate(model=MODEL_NAME, prompt=prompt)
    return resp["response"].strip()
```

Tak sformułowany prompt pozwala ograniczyć halucynacje, wymuszając na modelu językowym (np. `deepseek-r1:7b`) cytowanie tylko tych informacji, które rzeczywiście występują w dokumentach źródłowych. Jeśli brak odpowiedzi w kontekście, model wyświetla "I don't know". Ponadto algorytm był testowany dla  $k=1$ , ponieważ skupiając się na jednym kontekście przynosił najlepszych rezultaty - wyeliminowanie halucynacji.

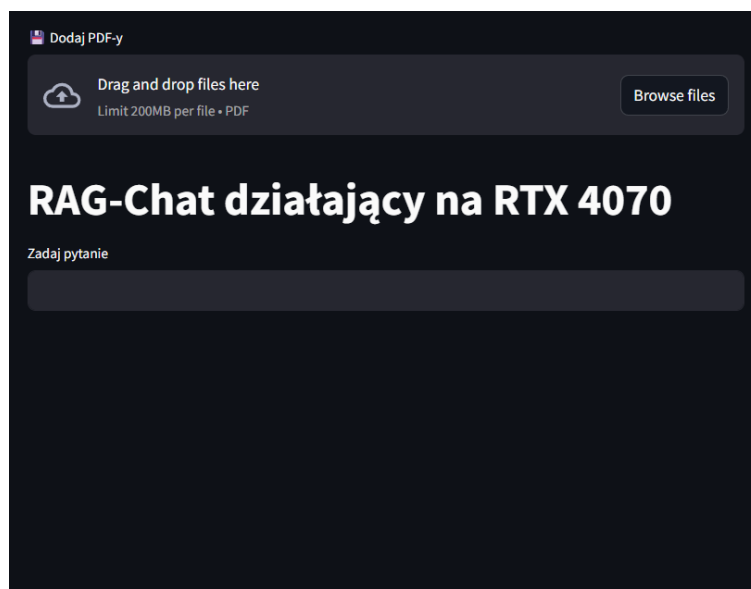
W projekcie zastosowano silnik wektorowy Qdrant, uruchomiony lokalnie w środowisku kontenerowym Docker. Dzięki temu możliwe było szybkie uruchomienie i utrzymanie bazy wektorowej.

<input type="text" value="Search"/>		<input type="checkbox"/> Only show running containers					
<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	qdrant	7123ed291ea2	<a href="#">qdrant/qdrant</a>	<a href="#">6333:6333</a>  <a href="#">Show all ports (2)</a>	0.33%	6 hours ago	 

## 4 Interfejs

Poniżej znajduje się interfejs, który został przygotowany w oparciu o streamlit. Zawiera on pole do zadania oraz możliwość wrzucenia pliku pdf.

Po dodaniu pliku PDF jego zawartość jest automatycznie przetwarzana tzn. tekst z każdej strony jest wyodrębniany, następnie zamieniany na wektory przy użyciu modelu embeddin-gowego i finalnie wektory trafiają do bazy Quadrant.



## 5 Eksperymenty

Podczas realizacji projektu, eksperymenty będą przeprowadzał zadając pytania odnośnie dwóch artykułów [1], [2], które są przetwarzane jako pliki pdf, ale również testowana będzie funkcja pozwalająca na dodanie danych tekstowych bezpośrednio z poziomu kodu źródłowego.

### 5.1 Pierwsze pytanie brzmiało: What is the difference between a multimodal algorithm and a quality diversity algorithm?

<think> Alright, so I'm trying to figure out what the difference is between a multimodal algorithm and a quality diversity algorithm. I've heard both terms before in the context of optimization algorithms, especially in machine learning or AI. Let me break down my understanding step by step. First, let's tackle "multimodal." From what I remember, modality refers to having multiple peaks or modes in a function. So, a multimodal algorithm is one that deals with problems where there are multiple local optima—places where the function reaches a peak but not necessarily the highest possible value (the global optimum). These algorithms must be able to find not just any peak but potentially all relevant ones if needed.

On the other hand, quality diversity sounds like it has something to do with both the quality of solutions and their diversity. Quality here probably refers to how good the solution is in terms of the objective function, while diversity would mean exploring different types of solutions that vary in some meaningful way beyond just being better or worse.

Wait, but I think I'm mixing up some concepts here. Let me see if I can find any references or definitions. Oh yeah, quality diversity algorithms are part of a broader category called optimization algorithms, which aim to explore the search space thoroughly to find not just one good solution but multiple useful ones that represent different trade-offs between objectives.

So putting it together: A multimodal algorithm is focused on finding multiple local optima in problems where there are many peaks. It's about exploring various regions of the search space to locate all significant local maxima or minima, which can be important when you want to choose among several good solutions based on different criteria.

In contrast, a quality diversity algorithm goes beyond just finding multiple local optima by also ensuring that each solution is of high quality and distinct in terms of certain traits. This makes them suitable for scenarios where having a diverse set of well-performing solutions is valuable, even if some are not the absolute best but represent different aspects or trade-offs.

I think I'm getting closer to understanding how these two concepts differ. Multimodal is about finding multiple local optima focusing on quality in each region, while quality diversity emphasizes both high-quality solutions and their diversity across different traits. </think>

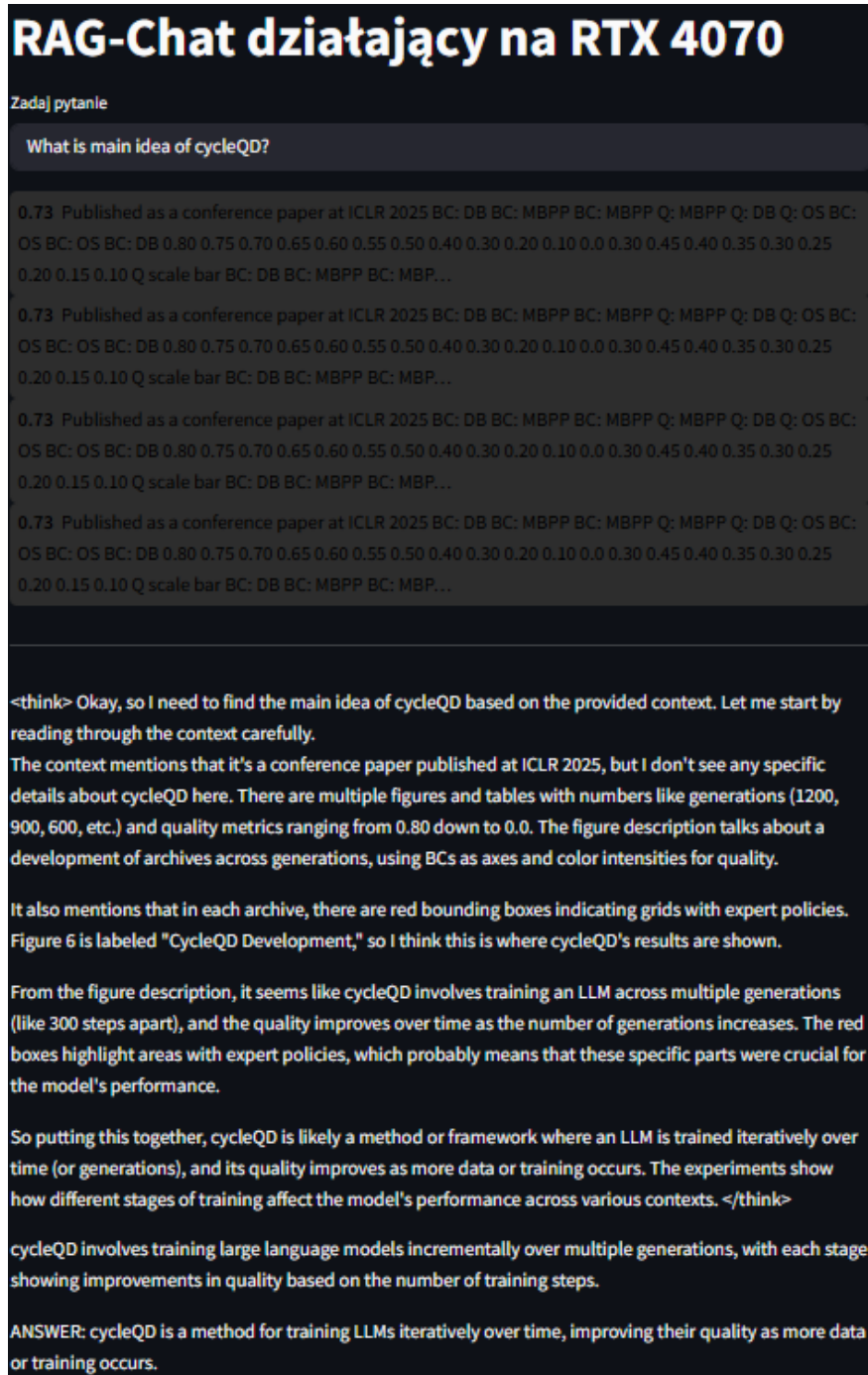
The difference between a multimodal algorithm and a quality diversity algorithm lies in their objectives:

- 1. Multimodal Algorithm:**
  - **Objective:** Focuses on identifying multiple local optima within the search space where there are several peaks or modes. These algorithms are designed to explore various regions of the problem space, each potentially containing a different peak.
  - **Focus:** Primarily concerned with finding diverse solutions that represent significant points in the solution landscape.
- 2. Quality Diversity Algorithm:**
  - **Objective:** Aims to generate not only multiple local optima but also ensures that these solutions are high quality and distinct in terms of specific traits or characteristics. It balances both performance (quality) and diversity.
  - **Focus:** Prioritizes having a diverse set of well-performing solutions, each representing different trade-offs or aspects within the solution space.

In summary, while multimodal algorithms focus on exploring various local optima, quality diversity algorithms emphasize creating a collection of high-quality, diverse solutions that capture different dimensions of the problem.


System oparty na RAG trafnie wychwycił różnice między quality diversity, a algorytmami multimodalnymi, opierał się o dane z artykułów (pdf), jak również dane tekstowe.


## 6 Drugie pytanie: What is main idea of cycleQD?



Widzimy, że model krótko ale snesownie odpowiedział na pytanie, jednak patrząc na proces myślowy jaki przeszedł, można wnioskować, że poprawnie przebiegał proces myślowy.

## 7 Provide the next steps of the cycleQD algorithm

 Dodaj PDF-y

 Drag and drop files here  
Limit 200MB per file • PDF

Browse files

# RAG-Chat działający na RTX 4070

Zadań pytanie

Provide the next steps of the cycleQD algorithm

0.79 Published as a conference paper at ICLR 2025 The  
AAAB6nicbZC5TgMxEIznOUO4wtHRWCRIVNEuRaAjEgWUQZBDSlaR1/EmVmzvyvYihVUegYYChGipqXgSOkr

0.79 Published as a conference paper at ICLR 2025 The  
AAAB6nicbZC5TgMxEIznOUO4wtHRWCRIVNEuRaAjEgWUQZBDSlaR1/EmVmzvyvYihVUegYYChGipqXgSOkr

0.79 Published as a conference paper at ICLR 2025 The  
AAAB6nicbZC5TgMxEIznOUO4wtHRWCRIVNEuRaAjEgWUQZBDSlaR1/EmVmzvyvYihVUegYYChGipqXgSOkr

0.79 Published as a conference paper at ICLR 2025 The  
AAAB6nicbZC5TgMxEIznOUO4wtHRWCRIVNEuRaAjEgWUQZBDSlaR1/EmVmzvyvYihVUegYYChGipqXgSOkr

<think> Okay, so I'm trying to figure out the next steps of the CycleQD algorithm based on the information given in the provided text. Let me read through the text carefully and break it down.

From what I understand, CycleQD stands for Cyclic Quality Diversity, which seems to be an algorithm used for training multiple agents or models with diverse skills. The user has included a description of how CycleQD works, focusing on model merging, mutation, and aggregation.

First, the text mentions that CycleQD maintains K archives, each containing LLMs trained for one agent skill. So, the initial step is probably setting up these archives to handle different tasks or skills.

Then, it talks about a process where models are merged using crossover operations based on singular value decomposition (SVD). This crossover combines two parent models into a child model by taking linear combinations of their parameters after SVD-based mutations. The mutation involves decomposing parameter matrices into components and scaling them with weights sampled from a uniform distribution.

After merging, the child model is trained to acquire one task skill. Once all K skills are learned, CycleQD returns multiple agents with specialized skills.

Now, looking ahead, the next steps after these K archives have been filled should involve combining or aggregating these models into a single high-performing model since the user's focus seems to be on creating a single model that excels across multiple tasks.

The text also mentions an 'Model Aggregation' section where CycleQD aggregates LLMs from each archive. So, probably, after all archives are filled with trained agents for each skill, they combine these into one model.

So the next steps would likely involve:

1. Training each of the K archives to specialize in their respective skills.
2. Once all archives are populated, merging their models or aggregating them to create a single strong model that covers multiple tasks.

I need to structure this as clear steps without too much jargon but ensure it's accurate based on the provided information. </think>

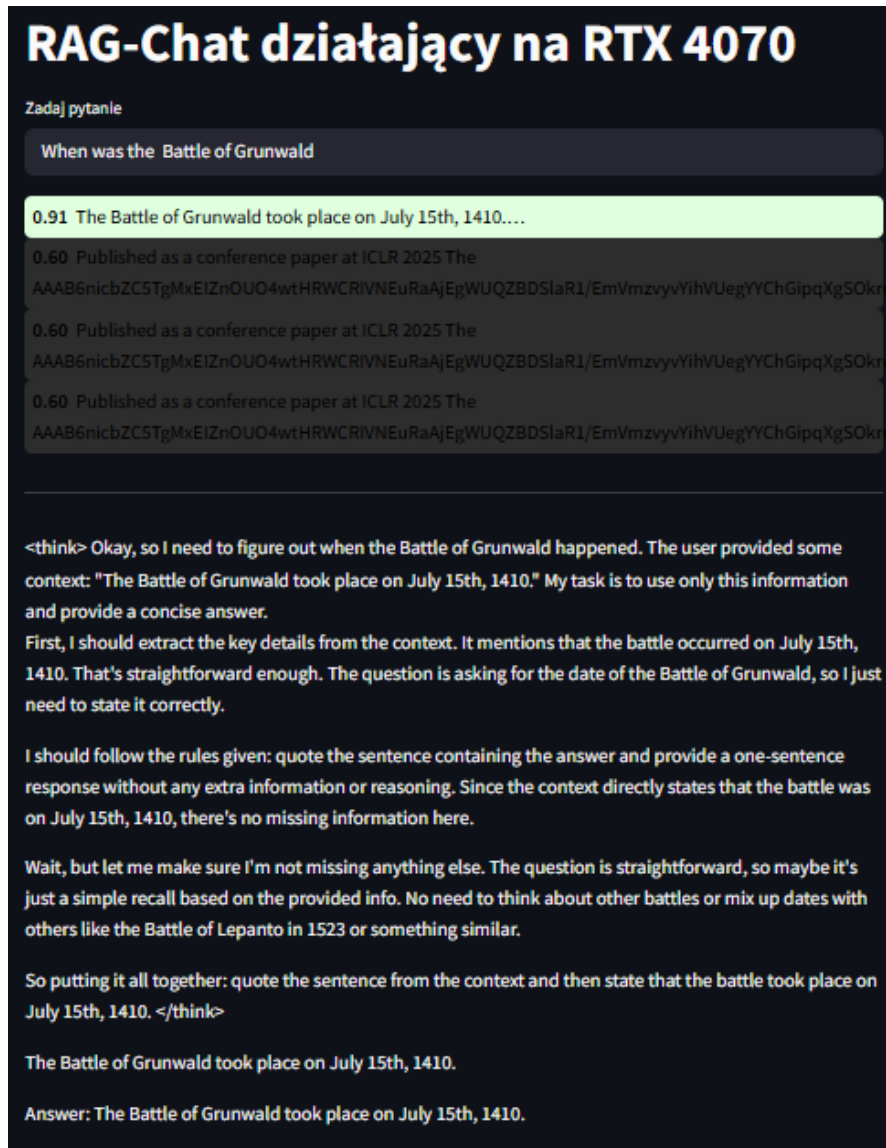
The next steps of the CycleQD algorithm involve:

1. **Establishing K Archives:** Create K separate archives, each dedicated to training models for distinct skills or agent specialties.
2. **Merging Models via Crossover:** Utilize crossover operations combined with SVD-based mutations to merge models from parent agents into child models. This process involves linear combinations of parameter matrices after decomposing them into meaningful directions using singular value decomposition.
3. **Specialized Training:** Train each merged child model to specialize in acquiring one specific task or skill, ensuring that each archive is filled with models optimized for their respective skills.
4. **Model Aggregation and Optimization:** Once all K archives are populated with specialized agents capable of handling various tasks, aggregate these models into a single comprehensive model. This aggregation process combines the strengths of all trained agents to create a versatile, high-performing model that excels across multiple tasks.

By following these steps, CycleQD effectively trains a diverse set of agents and then synthesizes them into a unified, multi-task-capable model.

Jako osoba, która przeczytała cały artykuł odnośnie algorytmu CycleQD, mogę z całą pewnością podkreślić, że model dobrze opisał kolejne kroki algorytmu, rozumowanie, którym się kierował też jest poprawne.

## 8 When was the Battle of Grunwald?

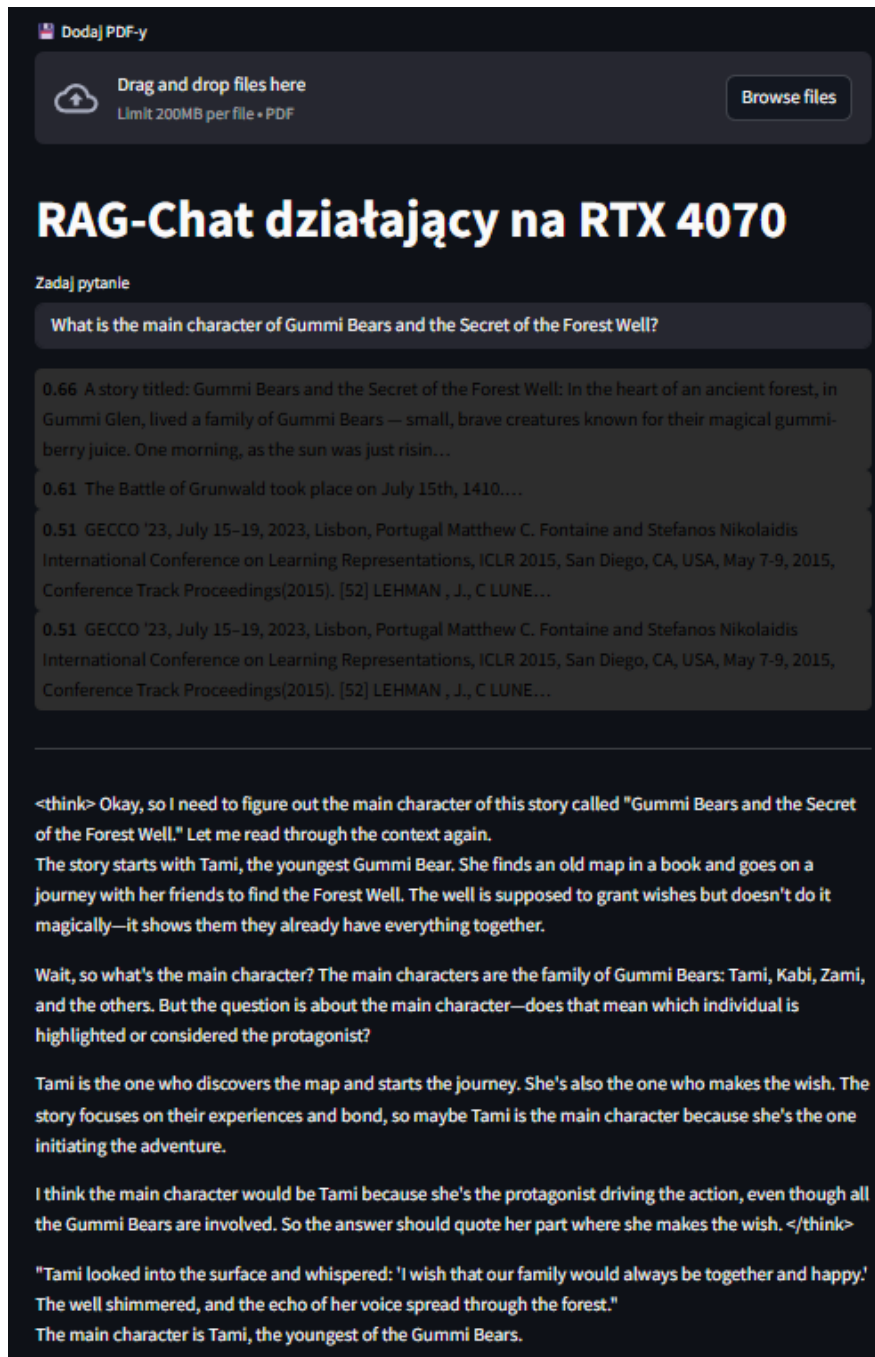


Widzimy, że model poprawnie odpowiedział, a także przypisał wagę 0.91 do tekstu, co znaczy o poprawności przepływu informacji.



## 9 Bajka

Kolejnym testem dla systemu RAG było wzbogacenie bazy danych o bajkę wygenerowaną przez model gpt-4o. Test ma zagwarantować, że model korzysta z wiedzy pochodzącej z bazy qdrant.



Model poprawnie wywnioskował kto jest głównym bohaterem bajki.

## Literatura

- [1] Kuroki, So, et al. (2024). *Agent Skill Acquisition for Large Language Models via CycleQD*. arXiv preprint arXiv:2410.14735v4.
- [2] Balam, N. R., Tjanaka, B., Lee, D. H., Fontaine, M. C., & Nikolaidis, S. (2023). *Generating Tom Cruise Images with DQD Algorithms*. <https://pyribs.org>
- [3] Packt Publishing. (2024). *RAG-Driven Generative AI: Build custom retrieval augmented generation pipelines with LlamaIndex, Deep Lake, and Pinecone*. Dostępne na platformie Packt, 2024-11-27.