https://doi.org/10.64142/jeai.2.1.41

Journal of Engineering and Artificial Intelligence

ISSN: 3117-3357

Research Article

JEAI-26-41

Study on Real Estate Search Model using RAG Applied Property Graph Index

Akira Otsuki*

Nihon University College of Economics, Tokyo, Japan, Japan

***Corresponding author:** Akira Otsuki, Nihon University College of Economics, Tokyo, Japan, E-mail: otsuki.akira@nihon-u.ac.jp

Accepted date: January 07, 2026; **Published date:** January 17, 2026

Citation: Otsuki A (2026) Study on Real Estate Search Model using RAG Applied Property Graph Index. J Eng Artif Intell Vol.2 No.1: 41.

Abstract

Retrieval Augmented Generation (RAG) is a text-generative AI model that combines search-based and text-generative-based AI models. Because original data can be used as external search data for RAG, it is not affected by incorrect data from the internet introduced by fine-tuning. Furthermore, it is possible to construct an original generative AI model that has expert knowledge. Although the LlamaIndex library currently exists for implementing RAG, text vectorization is performed using an approach similar to doc2Vec, creating issues that affect the accuracy of the generative AI's answers. Therefore, in this study, we proposed a property graph RAG that can define meaning when indexing text by applying the property graph index to LlamaIndex. For example, if a property was 10 years old, traditional RAG could not determine whether it was relatively new or old. However, by applying the property graph index, it becomes possible to meaning-making "relatively new" and "relatively old," this enables responses that better align with the user's question intent. Evaluation experiments were conducted using 10 real estate datasets and various cases including sales prices, on foot time to nearest station (min) and exclusive floor area (m^2) and the results confirmed that the proposed generative AI model offers more accurate answers than Prompt Refinement and Text-to-SQL for property search indexing.

Keywords: Generative AI; Property Graph Index; Real Estate Search; Retrieval-Augmented Generation (RAG); Property graph indexing

Introduction

Since the emergence of ChatGPT, research has increasingly been focused on techniques for enhancing the accuracy of responses from text-generative AI (hereafter referred to as "generative AI") [1-8]. Examples include RAG [1], chunking strategies [2], metadata filtering [3-4], Prompt Refinement [5], knowledge graph indexing [6], property graph indexing [7] and Text-to-SQL [8]. Among these, RAG is a natural language processing technology that combines the strengths of both search-based and generation-based AI models. Since RAG allows setting custom datasets for its external search, it avoids the bias toward incorrect online information inherent in fine-tuning and enables the construction of original generative AI with specialized knowledge. However, while LlamaIndex currently exists as a tool for implementing RAG, according to its documentation [9], text vectorization is performed using an indexing approach similar to doc2vec. This causes a discrepancy in the cosine similarity score between the user's query (Q) and the external search dataset's Q, resulting in poor accuracy of the generative AI's response (A). Therefore, in this research, we developed Property Graph RAG by applying Property Graph Index to LlamaIndex. This approach resolves the issue by performing semantic definition when indexing text for submission to the LLM. Evaluation experiments confirmed that for generative AI handling relatively structured QA datasets like real estate searches, the proposed method outperforms Prompt Refinement and Text-to-SQL.

Prior Related Research and Methods

Situating the proposed approach within the existing literature

Figure 1 shows the history of text mining technology utilizing deep learning and GPT. Until BERT, the technology

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

involves a deep learning-based text mining model; GPT-2 and later models are GPT-based text mining models. The proposed generative AI is positioned as a GPT-based text mining model. Studies that improve the accuracy of GPT-based text mining models have been actively pursued. Studies on technology used for the improvement of generative AI's response accuracy will be discussed in the next section.

Figure 1: The history of text mining technology utilizing deep learning and GPT.

Studying technologies for improving accuracy of generative AI responses

RAG (Retrieval Augmented Generation) [1]

RAG is a natural language processing technology that combines the strengths of both search-based and generation-based AI models. By configuring custom datasets as external search data for RAG, it eliminates the bias toward incorrect information found online that fine-tuning often introduces. Furthermore, it enables the construction of original generative AI with specialized knowledge.

Chunking strategy [2]

Chunking in RAG refers to the process of dividing long text into smaller segments (chunks). Since RAG searches the resulting chunks rather than the original text, it often generates responses using only chunks relevant to the input query. Various chunking methods have been proposed. For example, Fixed-Size Chunking determines the number of tokens per chunk and allows arbitrary overlap selection, resulting in lower computational costs compared to other chunking methods. Semantic Chunking divides text into semantically complete chunks by considering relationships within the text. This method allows for generating more accurate and contextually appropriate responses by preserving information integrity during search.

Metadata filtering [3-4]

This technique improves search performance by efficiently narrowing down only the necessary text without referencing the full text. This is achieved by pre-processing documents using text metadata to filter them. Examples of metadata include text titles, authors, creation dates, keywords and categories. PDFTriage [4] extracts metadata about text structure and then uses functions like fetch_pages to extract context necessary for responses and utilize it in generation. This achieves better performance than existing methods.

Prompt refinement [5]

Prompt Refinement can be achieved by setting the response_mode parameter in query_engine to refine. In RAG, the system extracts the dataset most similar to the user's query (Q) from the external search dataset using cosine similarity and then generates the answer (A) based on this dataset. For example, setting similarity_top_k=2 means repeating this process twice. In the second pass, the dataset with the second-highest cosine similarity score is extracted. This dataset is then used to rewrite the first A, aiming to improve its accuracy.

Knowledge graph index [6]

This technique applies knowledge graph representations, such as knowledge triples (subject, predicate, object), to queries sent to the LLM. This enables the LLM to interpret the meaning of the queries. However, according to LlamaIndex documentation [10], knowledge triples (subject, predicate, object) have limitations in expressiveness—such as the inability to assign labels and properties to relationships between nodes—and are therefore deprecated in LlamaIndex. Property Graph Index [7] addresses this issue. The LlamaIndex documentation states that using labeled property graph representations significantly improves knowledge graph modeling, storage and querying capabilities. Therefore, this research proposes Property Graph RAG, which applies this Property Graph Index to RAG.

Text-to-SQL [8]

This technique converts queries submitted to LLMs into SQL queries. Specifically, LlamaIndex connects to a database (DB) to retrieve schema information and associates it with queries submitted to the LLM. This enables the DB schema to be reflected in the queries and makes the basis for the answer visible. However, since DB column names are typically in

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

English, this approach has drawbacks for Japanese queries, including reduced accuracy or the need for translation.

Property Graph RAG (Proposed Method)

The proposed generative AI (Property Graph RAG) consists of the following four components:

- LLMs (Large Language Models): Section 3.1.
- LlamaIndex: Section 3.2.
- External information md file configured for RAG: Section 3.3.
- Property Graph Index: Section 3.4.

LLMs (Large Language Models)

For the model proposed in this study, we adopted Llama-3-ELYZA-JP-8B-gguf, which is built on Meta's "Llama3" architecture. Llama3 has 70 billion parameters and a demonstrated performance surpassing major global models at the time, such as GPT-4 and Google's Gemini 1.5 Flash. A lightweight 8-billion-parameter version is publicly available for commercial use by individuals. Therefore, in this study, we utilized this lightweight Japanese model.

LlamaIndex

In this study, we adopted llama_cpp_python-0.3.2-cp313-cp313-win_amd64.whl, which is a library for implementing RAG, primarily responsible for the following two functions:

- Creating structured data to pass external information to the LLM;
- Requesting the LLM to process and answer questions based on the created structured data.

The environment details of the implemented RAG are as follows:

- CPU/GPU: GPU
- RAM: 32GB
- OS (ver): Windows 11
- Python (ver): Python 3.13
- LlamaIndex (ver): llama_cpp_python-0.3.2-cp313.

Overall architecture of the proposed generative AI

The overall architecture of the proposed generative AI is shown in **Figure 2**. The folder .venv represents the Python virtual environment. The models folder contains the LLM from Section 3.1 and the 'inputs' folder contains the md files from the next section. LlamaIndex_refine_prompt.py is the script that controls the entire proposed generative AI.

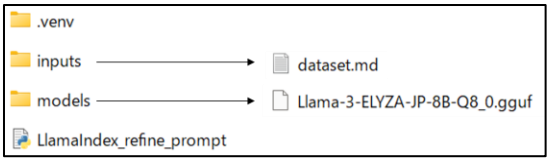

Figure 2: Overview of the proposed generative AI.

External search file for RAG

The Q&A below shows an example of an MD file for external search in the proposed generative AI. This MD file is written in Markdown format. The first level (#) is for linking to Llamaindex.py, while the second level (##) describes what dataset it is. The third level (###) describes the dataset in a Q&A format, as given below. Q is written under ### and A is written under.

Q&A Resources

Real Estate Search Data

¥87.8 million, 2 Hongo, Bunkyo-ku, Tokyo, 5 min walk from Suidobashi Station (Toei Mita Line), Monthly maintenance fee: ¥11,100, Monthly repair reserve fund: ¥24,000, Exclusive area: 53.32m², Floor plan: 1LDK, Floor location: SRC 13-story building, Age: 21 years, Corner unit facing 3 directions (South, West, North), Walk-in closet, Multi-purpose closet included, Windows in kitchen and bathroom, User-friendly L-shaped counter kitchen, Floor heating in living/dining area, 24-hour parcel delivery box available, Accessible via 2 or more train lines / Close to downtown / System kitchen / Bathroom dryer / Corner unit / Storage in all rooms / Vanity with shower / Balcony on two or more sides / South-facing balcony / Warm-water bidet toilet seat / Window in bathroom / TV monitor intercom / Suburban location / Good ventilation / Flooring in all rooms / Walk-in closet / Pets negotiable / Floor heating / Elevator / Package locker

-

https://suumo.jp/ms/chuko/tokyo/sc_bunkyo/nc_74818252/, ¥87.8 million, 5 min walk from Suidobashi Station (Toei Mita Line), Monthly maintenance fee ¥11,100, Repair reserve fund: ¥24,000/month, Exclusive area: 53.32m², Layout: 1LDK, Floor: SRC 13-story building, Age: 21 years old. Highly recommended!

(Omitted)

¥24.8 million, 1 Hongo, Bunkyo-ku, Tokyo, 5 min walk from Suidobashi Station (Toei Mita Line), Management fee: ¥13,005/month, No repair reserve fund, Exclusive area 28.29m², Floor plan 1LDK, Location 3rd floor, Built 48 years ago, ˜Property currently undergoing renovation˜, Convenient location accessible *via* 2 stations

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

and 3 lines, Supermarkets and commercial facilities within walking distance, Excellent living environment ~We accept inquiries even during renovation work~ 《Access》 ・Approx. 6 min walk to Tokyo Metro Marunouchi Line [Hongo Sanchome] Station ・Approx. 8 min walk to Toei Oedo Line [Hongo Sanchome] Station ・Approx. 5 min walk to Toei Mita Line [Suidobashi] Station《Nearby Facilities》 ・Maibasket Hongo 2-chome Store ・Approx. 370m (approx. 5 min walk) ・Maruetsu Petit Hongo 2-chome Store ・Approx. 470m (approx. 6 min walk) ・Lawson Hongo 1-chome Store ・approx. 290m (approx. 4 min walk) ・Kusuri no Fukutaro Hongo Store ・approx. 500m (approx. 7 min walk) ・Tokyo Dome City LaQua ・approx. 910m (approx. 12 min walk), Accessible via 2 or more train lines / Supermarket within 10 min walk / Close to downtown / Storage in all rooms / Vanity with shower / Flooring replaced / Renovated / Flooring in all rooms / Pets negotiable / BS/CS/CATV / Elevator / Movable partitions

GS Heim Hongo (https://suumo.jp/ms/chuko/tokyo/sc_bunkyo/nc_74543313/), ¥24.8 million, 5 min walk from Toei Mita Line "Suidobashi" Station, Monthly maintenance fee ¥13,005, No reserve fund, Exclusive area 28.29m², Floor plan 1LDK, Located on the 3rd floor, Built 48 years ago - Highly recommended!

(Omitted)

The QA dataset for this MD file was created based on the AtHome dataset provided by National Institute of Informatics (NII). The metadata for this dataset can be found at the given DOI link. This dataset was extracted in August 2024. To evaluate the improvement in the accuracy of the generative AI designed for routine Q&A scenarios, we randomly extracted 10 property listings from the AtHome dataset to create the database shown in **Table 1** (Details of **Table 1** are provided in Appendix A). From this database, we generated the QA dataset (MD file). For the QA dataset, all data from **Table 1** was set as the A (answer) and all data except the apartment name (Name column) from **Table 1** was set as the Q (question). The rationale is that the proposed generative AI assumes users may pose questions spanning multiple Qs within the QA dataset. Therefore, Prompt Refinement (Section 2.4) was adopted with similarity_top_k=3.

In other words, by running prompt refinement multiple times, the model generates the optimal A from multiple Qs in the QA dataset. Therefore, we reasoned that having more information in the Qs of the QA dataset would make it easier to match user Qs with the Qs in the QA dataset.

Table 1: Original DB for .md file of proposed generative AI.

	High City Futakotamagawa	**Brands Minami-Azabu**	**Lions Mansion Plaza Odakyu Sagamihara**
Selling Price (JPY)	9,980,000	74,800,000	31,800,000
Location	3-24-13 Tamagawa, Setagaya-ku, Tokyo	1-27-13 Minami-Azabu, Minato-ku, Tokyo	1-22-13 Sounami, Minami Ward, Sagamihara City, Kanagawa Prefecture
Nearest Station	Futakotamagawa Station (Tokyu Den-en-toshi Line)	Azabujuban Station (Toei Oedo Line)	Odakyu Odawara Line Odakyu Sagamihara Station
On Foot Time to Nearest Station (min)	5	8	5
Management Fee (JPY per month)	9,000	13,000	14,600
Building Age (Years)	41	13	21
Floor Level	1	8	5
Floor Plan Type	1LDK	2LDK	3LDK
Exclusive Floor Area (m²)	13.43	38.38	77.53
Parking	None	Available	Available

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

Property graph RAG

Our proposed model adopts the property graph index for the query engine index fed to the LLM. Property graph index allows entities and relations to hold attribute information, similar to graph data, enabling the assignment of meaning to data and relations. For this study, meanings were defined as shown in **Figure 3**. In the graph on the right side of **Figure 3**, the apartment complex (e.g., Minami-Osawa Residence) represents an entity and the Minami-Osawa Residence entity possesses the attribute information "0-minute walk." Furthermore, since attribute information can be defined in a dictionary format, it is possible to assign attributes like "near station" in addition to "0-minute walk." This enables the semantic definition "Minami-Osawa Residence 0-minute walk close to station." The definitions for "close to station" and "far from station" were determined by calculating quartiles for 10 properties: the first quartile (25%) or below is defined as "close to station," and the third quartile (75%) or above as "far from station." Similarly, this study performed semantic definitions for property price and room size.

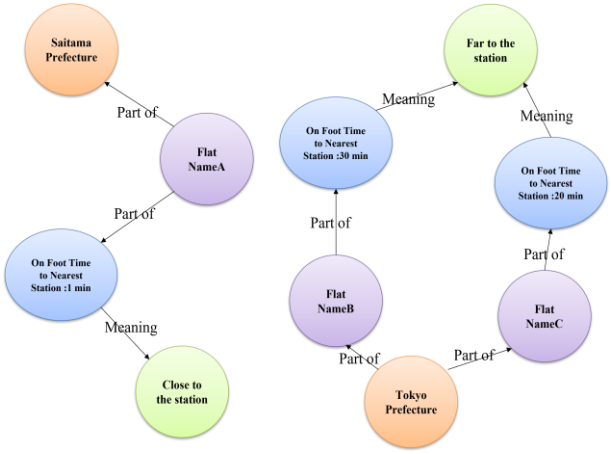

Figure 3. Conceptual diagram of the proposed generative AI's property graph RAG.

Next, we describe the specific implementation method for property graph index. As shown in Algorithm 1 (Python source code), we define the basic property information in "entitiesn" and perform the semantic definitions from **Figure 3** in "relationsn". Then, we insert entities as nodes and relations as relations into the graph_store.

Algorithm 1: Semantic definition using property graphs.

```
graph_store = SimplePropertyGraphStore()
entities1 = [
EntityNode(label="FLAT",name="Flat NameA",properties={" Address": "3-24-13 Tamagawa, Setagaya-ku, Tokyo","Management Fee (¥/month)":9000," Building Age (Years) ": 41," Floor Level": 1," Floor Plan Type": "1LDK","Parking": None}),
]
(Omitted)
relations1 = [
  Relation(label="CLOSEST", source_id=entities1[0].id,target_id=entities2[0].id, properties={"Walking Time (min)":5," Close from the station"}),
]
(Omitted)
graph_store.upsert_nodes(entities1)
graph_store.upsert_relations(relations1)
```

Then, as shown in Algorithm 2, the graph_store from Algorithm 1 is set as the PropertyGraphIndex for the index. This is set as the query_engine and the user's query (req_msg) is processed using this query_engine to generate the answer.

Algorithm 2: Reflect the property graph index in the query engine.

```
index = PropertyGraphIndex.from_existing(
  llm=llm,
  property_graph_store=graph_store,
  # optional, neo4j also supports vectors directly
  #vector_store=vector_store,
  #embed_kg_nodes=True,
  embed_model=embed_model
)
query_engine = index.as_query_engine(
  include_text=False,
  llm=llm,
)
print("================================")
while True:
  req_msg = input('\n Question: ')
  req_msg = req_msg.strip()
  if req_msg == "":continue
  if req_msg == "\q":break
  res_msg = query_engine.query(req_msg)
  print('\n', str(res_msg).strip())
```

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

Evaluation Experiment

Outline of evaluation experiment

This study focuses on indexes such as a real estate search; therefore, we examine RAG Prompt Refinement and Text-to-SQL to verify the effectiveness of the proposed model. Since all three were implemented using the AtHome dataset in **Table 1**, we decided to use data from this dataset for the evaluation experiments. Various evaluation metrics exist for generative AI, such as the following:

Matching-based, which includes the following:

- F1 Score: Harmonic mean of Recall and Precision;
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Calculated using Recall, Precision and F1 Score;
- BLEU (Bilingual Evaluation Understudy): Calculated based on the matching rate of n-grams (one word at a time, two words at a time, etc.) between the original text and the generated text;
- BERTScore (Bidirectional Encoder Representations from Transformers Score): Calculates similarity by vectorizing generated and reference sentences using a large language model (BERT).

Generation-based, including the following:

- LLM-as-a-Judge: Evaluates the validity of LLM outputs using the LLM itself.
- Human-based.

Evaluation standard

Five test subjects considering a real estate purchase participated in the study. The subjects prepared five questions for the evaluation experiment, which are presented in column "Q" of **Table 2**. The five test subjects evaluated the responses of the proposed generative AI and the comparison generative AI using the scoring as follows: 1 point, 0.5 point and 0 point. To scoring with 1 point when they accurately reproduced the answer of test subjects were seeking. The requirements of the answer that subjects were seeking is shown as underline in **Table 3**. And to scoring with 0.5 point when does not meet some of the requirements or if the answer is partially incorrect. Finally, to scoring with 0 point when does not fulfill the requirements or causes hallucination.

Table 2: Summary of test subjects' evaluation scores and reasoning.

		Score	Reason
Proposed generative AI	Q1	1	Because it proposed the property with the lowest price.
	Q2	1	Because it proposed the property with the lowest price.
	Q3	0	Because it proposed the property for which management fee was the third lowest.
	Q4	1	Because it proposed the property with the biggest room and closest to the station.
	Q5	1	Because it proposed a property with low management fees and a spacious room.
	Total	4	
RAG Refine Prompt	Q1	1	Because it proposed the property with the lowest price.
	Q2	1	Because it proposed the property with the lowest price.
	Q3	1	Because it proposed the property for which management fee was lowest.
	Q4	0	Because it proposed the property with the third-most spacious rooms and ninth-longest time on foot.
	Q5	0.5	Because it proposed the property that had the lowest price, but ninth-largest room and fifth-lowest management fee.
	Total	3.5	
Text-to-SQL	Q1	0	Because it proposed the property with the second-lowest price.
	Q2	0	Because it could not propose a property.

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

	Q3	0	Because it proposed the property with the second-lowest management fee.
	Q4	0	Because it could not propose a property.
	Q5	0	Because it could not propose a property.
	Total	0	

Table 3: Detailed results of the evaluation experiment ("○" indicates a correct answer (1 point), "×" indicates an incorrect answer (0 point) and "△" indicates 0.5 point).

	Q	A		
		Proposed generative AI	**RAG Refine Prompt**	**Text-to-SQL**
1	(Include 1 Item of Table 1) Please tell me the property with the lowest price. → Correct Answer: Rubelle Kumagaya	○ Rubelle Kumagaya at ¥9,800,000 is the lowest property price.	○ The property with the lowest price is Rubelle Kumagaya, with a selling price of ¥9,800,000.	× High City Futakotamagawa at ¥9,980,000 is the lowest price. →Second-lowest property price.
2	(Include 1 Item of Table 1) Please tell me the property with the lowest property price. → Correct: Rubelle Kumagaya	○ Rubelle Kumagaya at ¥9,800,000 is the lowest.	○ The property with the lowest price is Rubelle Kumagaya, with a selling price of ¥9,800,000.	× We apologize, but there is no property information available around the specified station.
3	(Include 1 Item of Table 1) Please tell me the property with the lowest management fee. → Correct: Minami-Osawa Residence	× Palette Studio Ginza 2-chome at ¥10,820. → Management fee of ¥10,820 is the third-lowest property → Proposed generative AI did not define meaning in Management fees	○ The property with the lowest management fee is Minami-Osawa Residence, with a management fee of ¥6,000 per month	× High City Futakotamagawa at ¥9,980,000 has the lowest management fee. →Second-lowest property price.
4	(Include 2 Item of Table 1) Please tell me about properties near the station with spacious rooms. → Correct: Minami-Osawa Residence	○ The following properties are near the station and have spacious rooms. 1. Park House Tsunashima (omitted) 67.82 m² and 9 minutes on foot from the nearest station. 2. Minami-Osawa Residence (omitted) 85.97 m² and 1 minutes on foot from the nearest station	× The older apartment "Queen City Ichikawa Fierte" is a 2SLDK on the 2nd floor with a floor area of 70.94m², located 13 minutes on foot from the nearest station, JR Sobu/Chuo Slow Line Ichikawa Station. → 70.94m² is the third-most spacious room. → 13 minutes is the rooms that ninth-longest time on foot	× We apologize, but no matching properties were found.

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

5	(Include 3 Item of Table 1) Please tell me about properties with low prices and management fees and spacious rooms. → Correct answer: Minami-Osawa Residence. Because two out of three items are correct as shown below ・× JPY 30,900,000 is sixth-lowest price ・〇 85.97 m² is most large room ・×JPY 30,900,000 is sixth-lowest price ・〇 85.97 m² is most large room ・〇 JPY 6000/month is lowest price →△ evaluation answer: Rubelle Kumagaya. Because one out of three items is correct as shown below →〇 JPY 9,800,000 is lowest price →× 53.43m² is nineth-largest room →× JPY 12,290 per month is fifth-lowest price	〇 The following property matches your criteria. Minami-Osawa Residence (omitted) This property has low management fees and a spacious room.	△ The used condominium “Rubelle Kumagaya” is priced at ¥9,800,000 with an exclusive area of 53.43m². Management fees are ¥12,290 per month.	× We apologize, but no matching properties were found.

Implementation of the comparison generative AI

RAG prompt refinement

As mentioned in Section 2.4, we implemented the model with QueryEngine=response_mode and similarity_top_k=3. By setting response_mode, the text is split into chunks, processed in chunk order to generate the response and then refined using a refine prompt. This results in a separate LLM call for each acquired chunk.

Text-to-SQL

This was implemented based on reference [11]. The SQL search table (DB schema) used **Table 1**. Specifically, Algorithm 3 creates the DB (SAMPLE.db) from **Table 1**.

As shown in Algorithm 4, SAMPLE.db is then configured in the query_engine. User queries (req_msg) are processed using this query_engine to generate answers.

Algorithm 3: Create a database for Text-to-SQL from **Table 1**.

```
import sqlite3
import pandas as pd
df = pd.read_csv("table1.csv",encoding='utf-8')
dbname = 'SAMPLE.db'
conn = sqlite3.connect(dbname)
cur = conn.cursor()
df.to_sql(name='manssion', con=conn, if_exists='replace')
```

Algorithm 4: Implementation of Text-to-SQL.

```
engine = create_engine("sqlite:///SAMPLE.db")
sql_database = SQLDatabase(engine=engine)
query_engine = NLSQLTableQueryEngine(
```

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

```
    sql_database=sql_database,
    tables=["manssion"],
    llm=llm,
    embed_model=embed_model,
)
while True:
    req_msg = input('\n Question: ')
    req_msg = req_msg.strip()
    if req_msg == "":continue
    if req_msg == "\q":break
    res_msg = query_engine.query(req_msg)
    print('\n', str(res_msg).strip())
```

Evaluation experiment results and discussion

Table 2 shows the summary (score and the reason) of the experimental results and **Table 3** shows the detailed experimental results. At first, the total score for Q1 to Q5 of the proposed generative AI was 4.0, with 3.5 for RAG Refine Prompt and 0.0 for Text-to-SQL; thus, the proposed generative AI showed the highest accuracy.

For Q1 to Q3 in **Table 2**, Q was created by including one item from the column names in **Table 1**; Q4 included two items and Q5 included three items. First, the proposed model scored × for Q3, △ for Q4 and ○ for all others. Note that Q4 answered two properties, one of which was correct, hence the △ rating. Since Q3 pertains to an item not semantically defined in the property graph RAG, the result of Q3 being the only × and most others being ○ indicates the proposed model's high accuracy.

The RAG Refine Prompt, as evident from Q4 and Q5 being incorrect, demonstrated good accuracy for single items but increasingly failed to provide correct answers as the number of items increased. This is likely because LlamaIndex performs text vectorization using a doc2vec-like approach [9].

For example, if a user's Q is the "Question:" section and the Q in the .md file is the "###" section, the only matching part is the underlined section below. When vectorizing the entire document, this causes the cosine similarity score to decrease. Finally, Text-to-SQL correctly answered with the property with the second-lowest price in Q1, but when the subject was changed from "lowest price" to "lowest property price" as in Q2, it failed to generate a response.

The SQL statement at this point was "SELECT * FROM flats WHERE nearest_station = 'Shinjuku Station';". This indicates that Text-to-SQL cannot generate correct SQL unless the user's query matches the column names in **Table 1**.

Discussion on comparing proposed generative AI and existing real estate search websites

Existing real estate search websites can only perform searches by narrowing down the conditions. Thus, it is not possible to perform ambiguous searches such as "Please tell me about properties like ~". In contrast, the proposed generative AI enables these ambiguous searches, leading us to believe that the proposed generative AI has the potential to replace existing property search sites. Although not implemented in this paper, the original database also contains textual data describing property characteristics, making it possible to realize the aforementioned fuzzy search.

Conclusion and Limitations of This Study

In this study, we proposed the Property Graph RAG, which combines RAG with a Property Graph Index. Through evaluation experiments, we confirmed that for indexes designed for relatively standardized queries like property searches, Property Graph RAG achieves a higher accuracy than RAG Prompt Refinement or Text-to-SQL. Future challenges include evaluating the accuracy of the proposed generative AI through quantitative evaluation experiments. Furthermore, since the proposed generative AI is expected to be effective for indexes beyond property searches, we aim to verify its effectiveness in these cases. But because this study is preliminary methodological studies, so there are study limitations. Specifically, experiments are conducted on a subset of about 10 properties with a limited number of fixed query scenarios.

So, there is no substantial extension in data scale or task diversity, which makes it difficult to draw general and convincing conclusions about the effectiveness of the proposed method. Furthermore, these results of evaluation experiment are evaluated by five respondents with subjective ratings (○△×). so not evaluated statistical significance tests, confidence intervals or inter rater agreement measures. But the current of this study represents an initial sample analysis and will plan to expand the dataset and quantitative analysis method in the future research to increase statistical reliability.

https://doi.org/10.64142/jeai.2.1.41
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

References

1. Gao Y, Xiong Y, Gao X, Jia K, Pan J, et al. (2023) Retrieval-augmented generation for large language models: A survey. arXiv. [Crossref] [Google Scholar]
2. Kumar V (2024) Chunking strategies for rag in generative AI. ADaSci.
3. Kaur I (2024) AI metadata extraction and filtering from scientific research articles.
4. Saad-Falcon J, Barrow J, Siu A, Nenkova A, Yoon S, et al. (2023) Pdftriage: Question answering over long, structured documents. arXiv. [Crossref] [Google Scholar]
5. Møller AG, Aiello LM (2025) Prompt refinement or fine-tuning? best practices for using llms in computational social science tasks. arXiv. [Crossref] [Google Scholar]
6. (2025) Knowledge Graph Index. LlamaIndex.
7. (2025) Using a property graph index. LlamaIndex.
8. (2025) Text_to_sql. LlamaIndex.
9. llamaindex Docment.
10. (2024) Introducing the property graph index: A powerful new way to build knowledge graphs with llms. LlamaIndex.
11. Ouyang L, Wu J, Jiang X, Almeida D, L Carroll, et al. (2022) Training language models to follow instructions with human feedback. arXiv. [Crossref] [Google Scholar]