МЕТОДОЛОГИЯ ПОСТРОЕНИЯ СИСТЕМЫ АНАЛИЗА ДАННЫХ С ИСПОЛЬЗОВАНИЕМ RAG ТЕХНОЛОГИЙ И ГРАФОВОЙ БАЗЫ ДАННЫХ

Кулевич А. О.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Пилецкий И. И. – канд. физ.-мат. наук, доцент

Приводится описание методологии построения системы анализа данных с использованием RAG технологии и графовой базы данных Neo4j. Рассматривается конвейер RAG с использованием LlamaIndex и Neo4j. Описывается методика построения и применения этого конвейера для создания и использования RAG-моделей для обработки и анализа текстовых данных.

В современном мире быстрый и точный анализ данных с различных источников играет ключевую роль, поскольку огромные объемы информации требуют систематизации и извлечения ценной информации. Одной из основных проблем является необходимость обработки и анализа больших массивов текстовых данных. Для этого используются методы искусственного интеллекта, включая **Large Language Models (LLM)** и технологию **Retrieval Augmented Generation (RAG)** [1].

LLM, такие как GPT, обучаются на огромных объемах текстовых данных и способны генерировать качественный текст, понимать естественный язык и выполнять задачи NLP. Они могут использоваться для автоматического создания контента, обработки запросов и генерации текстов. Однако при использовании LLM могут возникать такие трудности, как предоставление ложной или устаревшей информации, создание ответов на основе ненадежных источников, а также неточность из-за использования неправильной терминологии. И тогда на помощь приходит RAG технология.

RAG технологии позволяют улучшить процесс анализа текстовых данных, предоставляя механизмы для поиска и выбора наиболее подходящей информации из обширных источников. Они комбинируют методы поиска информации с возможностью генерации текста, что делает процесс анализа данных более эффективным и точным.

Чтобы продемонстрировать анализ данных с применением RAG технологий использовались документы с платформы Medium [2], в которых предоставлена обширная информация о библиотеке Neo4j Graph Data Science и совместимости Neo4j с Large Language Models (LLM). Для хранения и структурирования данных использовалась графовая база данных Neo4j [3].

Для успешного анализа и работы с данными необходимо эффективно извлекать нужную текстовую информацию из статей. Один из ключевых этапов этого процесса заключается в разделении статей на отдельные части, создании хранилища и векторного представления (embedding) для этих текстовых данных с использованием библиотеки Neo4jVectorStore. После этого, с помощью библиотеки VectorStoreIndex данные загружаются в графовую БД Neo4j и индексируются.

На рисунке 1 слева представлены узлы, в которых содержаться данные статей, а справа свойства одного из узлов.

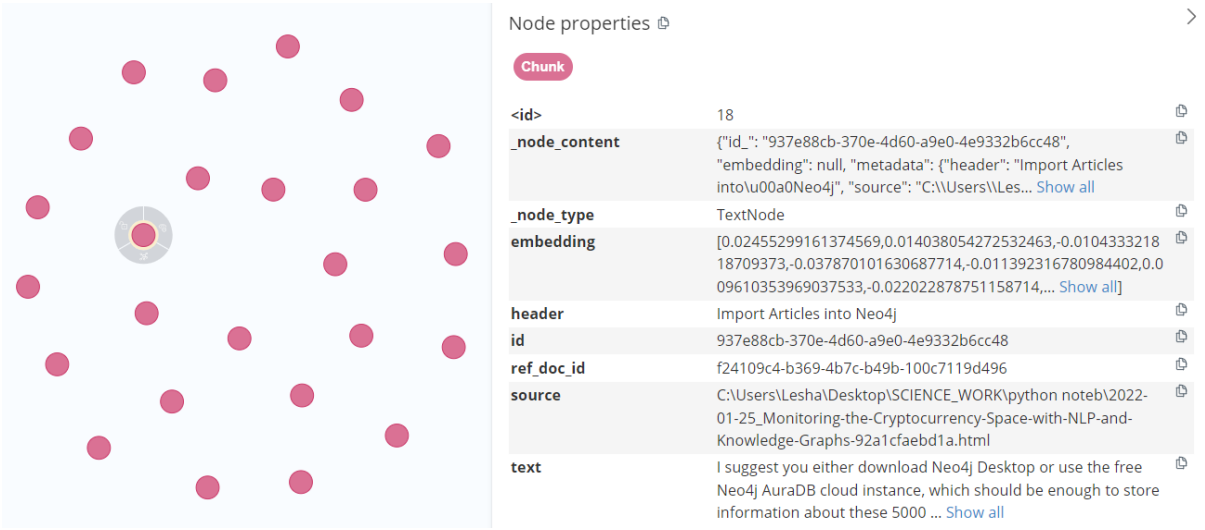

Рисунок 1 – Графовая база данных и свойства узла

Далее для создания мощной языковой модели (LLM) использовалась библиотека llama_index [4]. Эта библиотека предоставляет инструменты для работы с данными и формулирования запросов к

моделям LLM. В качестве модели LLM была выбрана последняя доступная версия - модель gpt-4-turbo-preview.

После создания модели и определения шаблона запроса к ней создается объект механизма запросов. Данный процесс показан на рисунке 2. Этот объект используется для формулирования запросов к индексу текстовых данных и получения ответов от модели языкового моделирования. Таким образом, можно эффективно проводить анализ текстовой информации и получать точные и информативные ответы от LLM модели.

```
from llama_index.core import PromptTemplate
from llama_index.core.query_engine import SimpleMultiModalQueryEngine

openai_mm_llm = OpenAIMultiModal(
    model="gpt-4-turbo-preview", max_new_tokens=3000
)

qa_tmpl_str = (
    "Context information is below.\n"
    "---------------------\n"
    "{context_str}\n"
    "---------------------\n"
    "Given the context information and not prior knowledge, "
    "answer the query.\n"
    "Query: {query_str}\n"
    "Answer: "
)
qa_tmpl = PromptTemplate(qa_tmpl_str)

query_engine = index.as_query_engine(
    multi_modal_llm=openai_mm_llm, text_qa_template=qa_tmpl
)
```

Рисунок 2 – Создание языковой модели и объекта механизма запросов

После этого созданной нами модели задаются вопросы, на которые построенная модель генерирует ответы. Результат работы модели представлен на рисунке 3.

```
query_str = "What is RAG?"
response = query_engine.query(query_str)
print(response)
```

```
RAG stands for Retrieval-Augmented Generation. It is an approach that involves using large language models to generate answers
based on externally provided information or relevant documents, rather than relying solely on the internal knowledge of the lan
guage model. This approach allows for more accurate and up-to-date responses by leveraging external sources of information.
```

```
query_str = "How do vector RAG application work?"
response = query_engine.query(query_str)
print(response)
```

```
Vector RAG applications work by providing additional context at query time to large language models (LLMs) in order to generate
accurate and up-to-date answers. This is achieved by leveraging a smart search tool, such as Neo4j's vector index, which allows
for the retrieval of additional information based on user input. By combining structured and unstructured data, vector RAG appl
ications can move beyond simple information retrieval and enable more effective question-answering capabilities.
```

```
query_str = "What is LLM?"
response = query_engine.query(query_str)
print(response)
```

```
LLM stands for Large Language Models. They are models that are trained on vast amounts of text data to understand and generate
human-like text.
```

Рисунок 3 – Результаты выполнения запросов к построенной модели

Полученные данные показывают, что созданная модель функционирует корректно, а сгенерированные ответы являются содержательными и логичными.

В результате была разработана тематическая модель, содержащая данные с платформы Medium, которая с помощью RAG технологии способна генерировать ответы на заданные вопросы с высокой точностью и полнотой. В дальнейшем возможно использование дополнительных библиотек, таких как LangChain, для расширения возможностей модели, что позволит разрабатываемой модели удерживать контекст ранее заданных вопросов, понимать и **уточнять суть вопроса и обсуждаемую тему** без явного указания контекста, что значительно повысит качество и эффективность ее работы.

Список использованных источников:

1. What is Retrieval-Augmented Generation? [Электронный ресурс] / Режим доступа: https://aws.amazon.com/what-is/retrieval-augmented-generation/?nc1=h_ls Дата доступа: 25.03.24

2. Medium [Электронный ресурс] / Режим доступа: https://medium.com Дата доступа: 25.03.24.

3. Neo4j [Электронный ресурс] / Режим доступа: https://neo4j.com/labs/neosemantics/ Дата доступа: 25.03.24

4. LlamaIndex [Электронный ресурс] / Режим доступа: https://www.llamaindex.ai/ Дата доступа: 25.03.24