Прототип RAG-системы для анализа нормативных документов

RAG-система: краткое описание

RAG-система сочетает поиск по векторной базе знаний (retrieval) с генерацией ответов LLM (generation).
Это позволяет отвечать на вопросы на основе реальных данных из документов, минимизируя эффект галлюцинации.

Цель: создать RAG-систему для нормативных документов (например, СП 20.13330.2016), чтобы отвечать на технические вопросы в области проектирования, строительства и эксплуатации промышленных объектов (например, вопросы о зонах, коэффициентах, нагрузках и т.д.).

Вопросы для демонстрации:
1 В каких зонах по весу снегового покрова находятся Херсон и Мелитополь?
2 Какие регионы Российской Федерации имеют высотный коэффициент k_h, превышающий 2?
3 Какой коэффициент надежности по нагрузке для металлических конструкций?
4 Как определяется нормативное значение основной ветровой нагрузки w?
5 Что такое коэффициент надежности по нагрузке?
и др.

Результат: система точно отвечает, ссылаясь на источники (таблицы, карты, формулы).
Успешно обрабатывает текст, таблицы, изображения, формулы, в том числе с OCR/анализом.

RAG-система: исходные данные и материалы

Иточник данных

Папка «data» содержит нормативно-техническую документацию.

В прототипе решения в качестве примера использован нормативный документ - "СП 20.13330.2016 2024-09-05.docx", который структурирован в формате .docx, и содержит текст, таблицы, изображения (карты, рисунки), формулы.

Проблемы с данными и нюансы разработки прототипа

1) исходные docx-файлы требуют проверки структурности и полноты содержания. Так например, в документе «СП 20.13330.2016 Нагрузки и воздействия. Актуализированная редакция СНиП 2.01.07-85.docx» содержались не все карты при наличии заголовков на них, что влияло на точность ответов LLM;

2) .docx — это ZIP-архив с XML, изображениями (JPEG, PNG, WMF). Нужно извлекать текст, таблицы, изображения разных форматов;

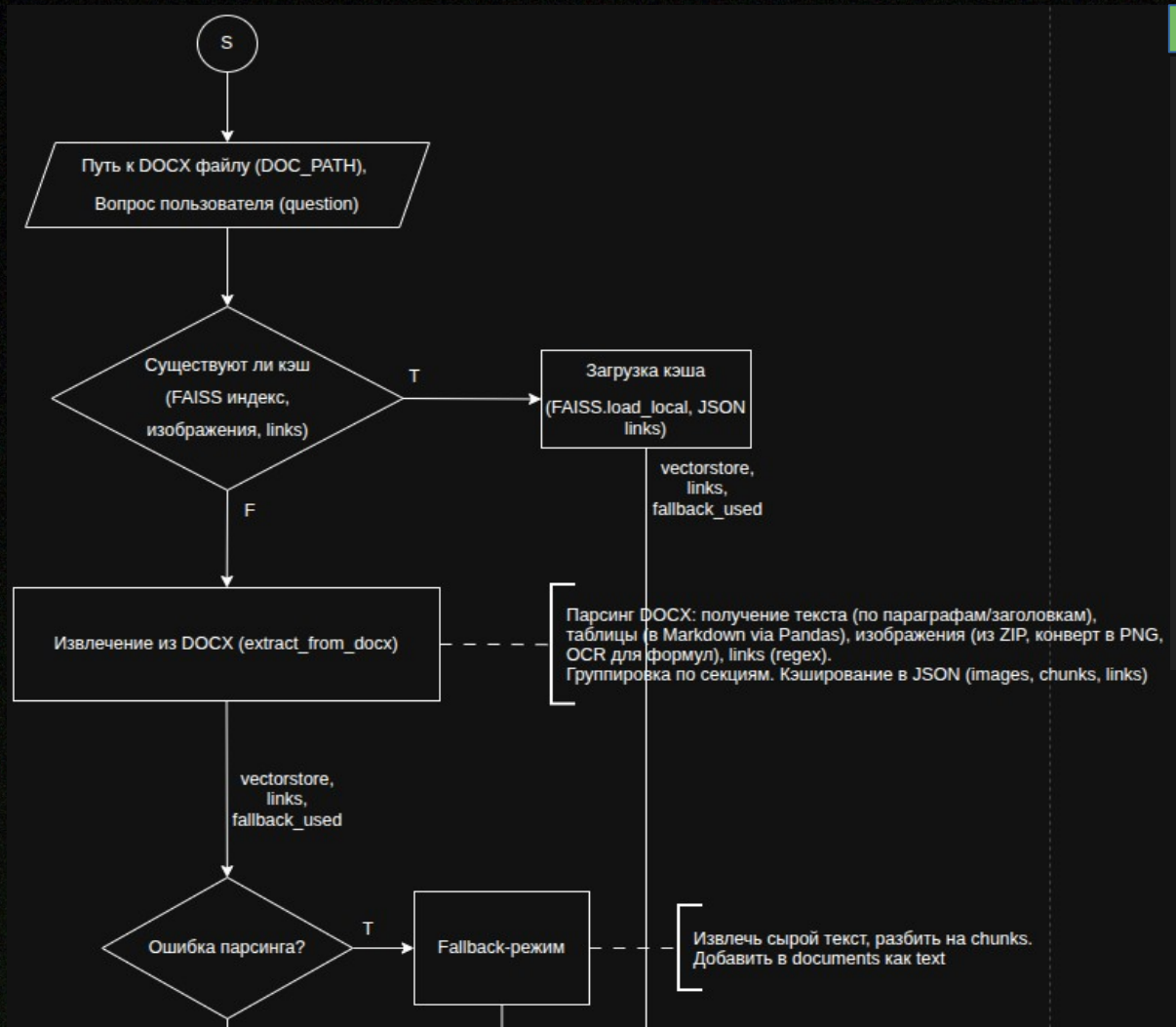
3) с учетом п.2 необходимо обрабатывать возможные ошибки / нештатные ситуации парсинга (fallback-режим);

4) обеспечить логирование ответственных этапов работы алгоритма (в рамках прототипа на локальном уровне);

5) требуется анализировать разную информацию: текст + изображение. Изображение само по себе (особенно карты, схемы) может не нести внутри себя информацию о предметной области, что требует его линковки со связанным по смыслу текстом, например, заголовком и т.п.

6) с учетом п.5 требуется мультимодальная модель (текст + изображение), накладывающая ограничения на вычислительные мощности (CPU/GPU etc.) локальной разработки прототипа.

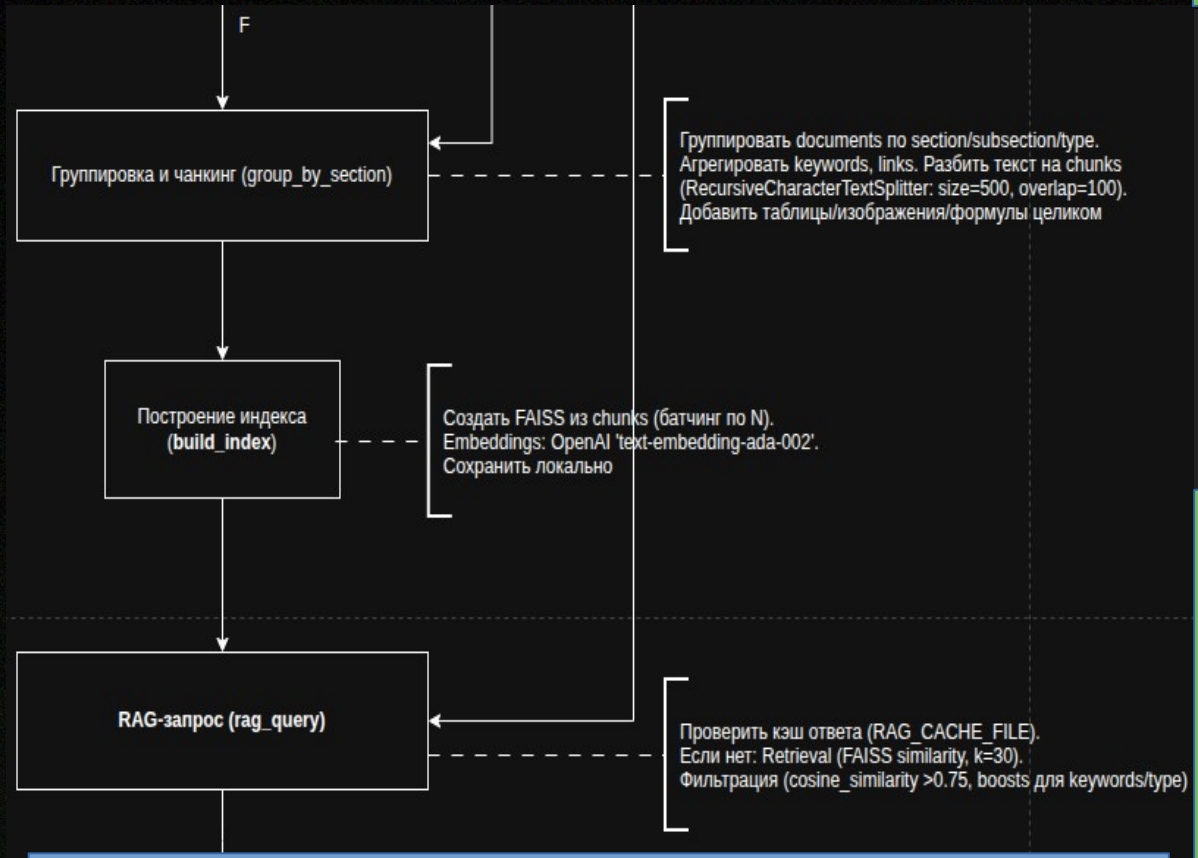
RAG-система: алгоритм предобработки данных



Ключевой код (extraxt_from_docx — упрощённо)

```python
def extract_from_docx(doc_path: str, ...) -> Tuple[List[LangchainDocument], [
    doc = docx.Document(doc_path)
    for para in doc.paragraphs:
        # Обработка текста, заголовков
        if style_name.startswith("Heading"):
            current_section = text.strip()
        # Извлечение изображений из runs
        for run in para.runs:
            for blip in run._element.xpath('.//a:blip'):
                # Конверт в PNG, OCR если формула
                ocr_text = ocr_image(image_path) if area < FORMULA_AREA_THRES
    for table in doc.tables:
        # Конверт в Markdown
        df = pd.DataFrame([[cell.text.strip() for cell in row.cells] for row
        table_md = df.to_markdown(index=False)
    # Fallback
    except Exception:
        text = process(doc_path)  # docx2txt
        chunks = [c.strip() for c in text.split('\n\n') if c.strip()]
    return documents, links, fallback_used
```

RAG-система: чанкинг, построение индекса и RAG-пайплайн



НТД имеют структуру (разделы, таблицы, изображения, приложения).

Маленькие чанки теряют контекст, большие — превышают лимит токенов LLM.

Батчинг для больших документов при индексации. Лог чанков в JSON.

Ключевой код (build_index— упрощённо)

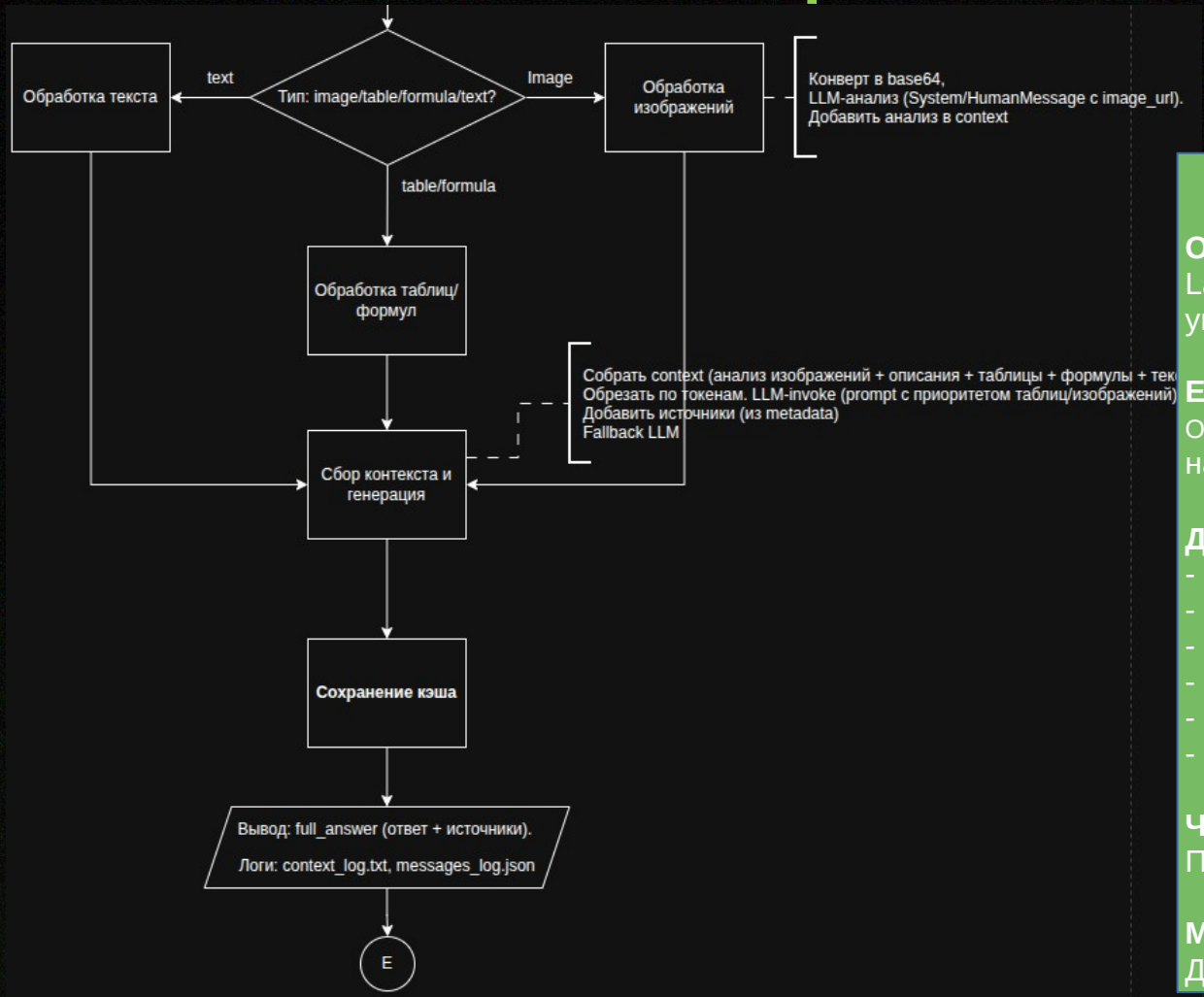
```python
def build_index(documents: List[LangchainDocument], ...):
    text_docs = [d for d in documents if d.metadata["type"] == "text"]
    grouped_text = group_by_section(text_docs)
    splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=1
    chunked_texts = splitter.split_documents(grouped_text)
    other_docs = [d for d in documents if d.metadata["type"] != "text"]
    final_docs = chunked_texts + other_docs
    # Батчинг для FAISS
    for i in range(0, len(final_docs), batch_size):
        batch = final_docs[i:i + batch_size]
        if vectorstore is None:
            vectorstore = FAISS.from_documents(batch, embeddings)
        else:
            vectorstore.add_documents(batch)
    return vectorstore
```

Retrieval:
Similarity search в FAISS (k=30) с фильтрацией по similarity.

Augmentation:
Текст/Таблицы/Формулы: добавляем в контекст.
Изображения: анализ LLM (vision mode), извлечение данных.
Лимит токенов: 250k, обрезка контекста.
Generation: LLM с промптом.
Кэширование: ответы в JSON для повторных вопросов.

Result:
Точные ответы с источниками. Обработка изображений улучшает ответы на визуальные данные (карты).

RAG-система: сбор контекста и генерация ответа



Выбранные решения в рамках прототипа

Основное:
LangChain (для Documents, Splitter, VectorStore, Retriever) - упрощает RAG.

Embeddings/LLM:
OpenAI (text-embedding-ada-002, gpt-4o-mini) - надёжные модели.

Другие:
- FAISS (индекс),
- python-docx/docx2txt (парсинг),
- pytesseract (OCR),
- PIL/Wand (изображения),
- pandas (таблицы),
- scikit-learn (cosine_similarity) etc.

Что получилось:
Полный пайплайн от извлечения данных до ответа.

Масштабируемость:
Добавление других файлов НТД в load_or_create_all.

Ответы RAG-системы

```
================================================================================
❓ Вопрос: В каких зонах по весу снегового покрова находятся Херсон и Мелитополь?
--------------------------------------------------------------------------------
Херсон находится в зоне I, а Мелитополь в зоне II по весу снегового покрова. Источник: КАРТА 1, В.

Источники:
- СП 20.13330.2016 2024-09-05.docx: КАРТА 1, в. РАЙОНИРОВАНИЕ ТЕРРИТОРИИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ, ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ, ЗАПОРОЖСКОЙ ОБЛАСТИ, ХЕРСОНСКОЙ ОБЛАСТИ ПО ВЕСУ СНЕГОВОГО ПОКРОВА (ДОПОЛНЕНИЕ К КАРТЕ 1. РАЙОНИРОВАНИЕ ТЕРРИТОРИИ РОССИЙСКОЙ ФЕДЕРАЦИИ ПО ВЕСУ СНЕГОВОГО ПОКРОВА) (image, релевантность: 0.873);

================================================================================
❓ Вопрос: Какие регионы Российской Федерации имеют высотный коэффициент k_h, превышающий 2?
--------------------------------------------------------------------------------
Регионы Российской Федерации с высотным коэффициентом \( k_h \), превышающим 2, включают:

1. Краснодарский край (Адлерский район) - \( k_h = 2,15 \)
2. Республика Адыгея (Адыгея) - \( k_h = 2,15 \)
3. Красноярский край (Кемеровская область - Кузбасс, Кузнецкий Алатау, Горная Шория) - \( k_h = 2,25 \)
4. Красноярский край (Саянский хр., Куртушибинский хр.) - \( k_h = 3,15 \)
5. Норильский промышленный район (до 20 км к югу от г. Норильска) - \( k_h = 2,5 \)
6. Норильский промышленный район (к северо-востоку от г. Норильска) - \( k_h = 3,75 \)
7. Республика Бурятия (Байкальский хр.) - \( k_h = 2,3 \)

Источник: Таблица 49.

Источники:
- СП 20.13330.2016 2024-09-05.docx: Таблица 49 (table, релевантность: 0.964); КАРТА 1. РАЙОНИРОВАНИЕ ТЕРРИТОРИИ РОССИЙСКОЙ ФЕДЕРАЦИИ ПО ВЕСУ СНЕГОВОГО ПОКРОВА (formula, релевантность: 0.936); Таблица 11.2 ( formula, релевантность: 0.930)

================================================================================
```

Ответы RAG-системы

```
================================================================================
❓ Вопрос: Какой коэффициент надежности по нагрузке для металлических конструкций?
--------------------------------------------------------------------------------
Коэффициент надежности по нагрузке для металлических конструкций составляет 1,05 (Таблица 1).

Источники:
- СП 20.13330.2016 2024-09-05.docx: Таблица 1 (table, релевантность: 0.980); Таблица 3 (table, релевантность:
0.964); Таблица 26 (table, релевантность: 0.959); Для различных участков конструкций (рисунок В.1)
коэффициент cx определяется по таблице В.1; (image, релевантность: 0.886)

================================================================================
❓ Вопрос: Как определяется нормативное значение основной ветровой нагрузки w?
--------------------------------------------------------------------------------
Нормативное значение основной ветровой нагрузки w определяется как сумма средней составляющей ветровой
нагрузки wm и пульсационной составляющей wg, по формуле:

w = wm + wg. (11.1)

Источник: СП 20.13330.2016, раздел 11.1.2.

================================================================================
```

Ответы RAG-системы

```
================================================================================
❓ Вопрос: Что такое коэффициент надежности по нагрузке?
--------------------------------------------------------------------------------
Коэффициент надежности по нагрузке – это коэффициент, учитывающий в условиях нормальной эксплуатации
сооружений возможное отклонение нагрузок в неблагоприятную (большую или меньшую) сторону от нормативных
значений (СП 20.13330.2016, п. 3.2).

Источники:
- СП 20.13330.2016 2024-09-05.docx: Таблица 3 (table, релевантность: 0.951); Таблица 1 (table, релевантность:
0.951); Рисунок 11.1 - Коэффициенты динамичности
(в ред. Изменения N 1, утв. Приказом
Минстроя России от 05.07.2018 N 402/пр) (formula, релевантность: 0.918)
================================================================================
```

RAG-система: результаты

В ходе поисковых исследований апробированы:

1) различные ***open-source мультимодальные модели*** ***(text + image)***: llava:7b, bakllava:7b, moondream:1.8b и другие, в том числе квантизованые версии. Качество ответов моделей напрямую зависит от используемых ресурсов;

2) **условно *бесплатные версии API***, которые как правило предоставляют доступ к «only text» моделям, что ограничивает заложенный функционал предлагаемой RAG-системы, где в том числе требуется работа с изображениями;

3) ***коммерческие версии API***. Например, модель gpt-4o-mini от OpenAI имеет приемлемые качество/стоимость и скорость работы, надежность, быстрый старт на первом этапе разработки;

4) масштабируемость на другие файлы НТД (код легко адаптируется);

5) установлено, что не всегда и не все картинки следует преобразовывать по типу image-to-text, особенно для карт/графиков/схем, информация по которым нужна в контексте конкретного вопроса, который заранее не известен. Кроме того изображение само по себе может не нести внутри себя информацию без линковки его со связанным по смыслу текстом, например, заголовком.

В рамках прототипа:

✅ **Реализована RAG-система**

✅ **Точность:** Ответы соответствуют документу, ссылаются на источники.

✅ **Скорость:** С кэшем — мгновенно; без — 1-2 мин на полную индексацию.

✅ **Проблемы:** OCR формул неидеален (Tesseract), но vision LLM компенсирует.

✅ **Настраиваемые ограничения:** лимит изображений (5), токенов LLM (250k).

✅ **Улучшения / TODO:** Добавить мульти-файлы, fine-tune embeddings, улучшенный кеш, перевод на offline-LLM (open-source) etc

RAG-система: код решения и полезная информация

►**Алгоритм «RAG_system_prototype.drawio»**:
https://drive.google.com/file/d/1NKXJC81wfimvfwyhNi-EXmaEvut1k-01/view?usp=sharing

►**Код и доплнительные файлы, GIT-repo:**
https://github.com/nicholasid7/rag-system-prototype

Спасибо за внимание!