

Эффективность поиска и генерации с дополненным извлечением в корпоративных информационных системах управления знаниями

Лапушкина Елена Павловна

*Приамурский государственный университет имени Шолом-Алейхема
студент*

Аннотация

В статье рассматривается методика проектирования и оценки эффективности системы Retrieval-Augmented Generation для автоматизации доступа сотрудников к корпоративной базе знаний. Описаны ключевые компоненты на языке Python с использованием библиотеки LangChain и векторного хранилища FAISS. Представлены результаты повышения точности и достоверности ответов, снижение доли галлюцинаций генеративной модели и обеспечение прозрачности источников информации.

Ключевые слова: Retrieval-Augmented Generation, большие языковые модели, управление знаниями, векторный поиск.

The effectiveness of search and generation with enhanced extraction in corporate knowledge management information systems.

Lapushkina Elena Pavlovna

*Sholom-Aleichem Priamursky State University
Student*

Abstract

The article discusses the methodology of designing and evaluating the effectiveness of the Retrieval-Augmented Generation system for automating employee access to the corporate knowledge base. The key components in Python are described using the LangChain library and the FAISS vector storage. The results of improving the accuracy and reliability of responses, reducing the proportion of hallucinations of the generative model and ensuring transparency of information sources are presented.

Keywords: Retrieval-Augmented Generation, large language models, knowledge management, vector search.

1 Введение

1.1 Актуальность

Объём внутренней документации современных организаций - регламентов, инструкций, политик, отчетов и технической документации - растёт быстрее, чем способность сотрудников её обрабатывать вручную. Традиционные системы полнотекстового поиска, по ключевым словам, не учитывают смысловую близость запроса и документа, а прямое использование

больших языковых моделей для ответов на вопросы о внутренних процессах ограничено фиксированным объёмом знаний, зафиксированных на момент обучения, и риском генерации недостоверных сведений. В этих условиях особую значимость приобретает подход Retrieval-Augmented Generation (RAG), позволяющий совмещать генеративные способности LLM с оперативным извлечением актуальной информации из корпоративной базы знаний без переобучения модели.

1.2 Обзор исследований

Основы подхода заложены в работе Р. Lewis и др., где предложена архитектура, объединяющая параметрическую память пред обученной модели с непараметрической памятью векторным индексом внешних документов; это позволило существенно повысить точность решения знание ёмких задач без увеличения числа параметров модели [1]. И.В. Антонов и Ю.В. Бруттан провели масштабное сравнительное исследование эффективности 26 больших языковых моделей в качестве генеративного компонента RAG-архитектуры применительно к корпоративной информационной системе поиска по университетскому документальному фонду, показав методом экспертных оценок существенный разброс качества и времени отклика моделей и сформулировав практические рекомендации по выбору LLM для внедрения в корпоративные системы [2]. А.В. Мельников, И.Е. Николаев, М.А. Русанов, В.Р. Аббазов провели сравнительный анализ методов извлечения контекста RAG применительно к русскоязычным данным, отметив, что большинство существующих оценок ориентировано на английский язык, а качество эмбединг-моделей существенно влияет на точность ответов для русскоязычных сервисов [3]. А.Е. Мащенко исследовал применение RAG-подхода для автоматизации конфигурации корпоративных информационных систем. В работе показано, что использование RAG позволяет повысить точность генерации ответов и упростить адаптацию систем под специфику задач организации, однако отмечается необходимость тщательной настройки компонентов индексации и поиска для достижения стабильного качества [4]. А.О. Науменко проанализировал архитектуру RAG, ключевые компоненты индексации и поиска, а также сферы практического применения технологии, отметив зависимость эффективности системы от качества базы знаний и сложности настройки компонентов [5].

1.3 Цель исследования

Цель данного исследования - разработка и оценка эффективности архитектуры RAG-системы для автоматизации доступа сотрудников к корпоративной базе знаний.

2 Материалы и методы

В исследовании использован комплекс программных средств и компонентов, образующих экспериментальный стенд для оценки подходов к построению систем генерации ответов на основе корпоративной

документации. Реализация стенда выполнена на языке Python версии 3.11, в качестве основного фреймворка для построения RAG-конвейера выбран LangChain 0.2. Для генерации векторных представлений текстов на русском языке применена модель эмбедингов sentence-transformers с архитектурой paraphrase-multilingual-mpnet-base-v2.

3 Результаты

Разработанный экспериментальный стенд реализует типовой двухэтапный конвейер RAG: офлайн-индексацию базы знаний и онлайн-обработку пользовательского запроса. Общая последовательность этапов показана в таблице 1.

Таблица 1 – Этапы работы RAG-конвейера

№	Этап	Назначение
1	Загрузка документов	Извлечение текста из файлов корпоративной базы знаний (docx, pdf, html)
2	Чанкинг	Разбиение документов на смысловые фрагменты с перекрытием
3	Векторизация	Построение эмбедингов фрагментов моделью sentence-transformers
4	Индексация	Сохранение эмбедингов в векторном индексе FAISS
5	Поиск (retrieval)	Извлечение top-k фрагментов, релевантных запросу
6	Генерация	Формирование ответа LLM на основе запроса и найденного контекста

На этапе индексации документы разбиваются на фрагменты фиксированного размера с перекрытием, после чего для каждого фрагмента вычисляется векторное представление и сохраняется в индексе FAISS (рис.1).

```
from langchain_community.document_loaders import (
    | DirectoryLoader
    | )
from langchain.text_splitter import (
    | RecursiveCharacterTextSplitter
    | )
from langchain_huggingface import (
    | HuggingFaceEmbeddings
    | )
from langchain_community.vectorstores import FAISS

# 1. Загрузка документов базы знаний
loader = DirectoryLoader(
    | "knowledge_base/", glob="**/*.txt"
    | )
documents = loader.load()

# 2. Разбиение на фрагменты (чанки)
splitter = RecursiveCharacterTextSplitter(
    | chunk_size=800,
    | chunk_overlap=120,
    | )
chunks = splitter.split_documents(documents)

# 3. Модель эмбедингов для русского языка
embeddings = HuggingFaceEmbeddings(
    | model_name=(
    | | "sentence-transformers/"
    | | "paraphrase-multilingual-mpnet-base-v2"
    | )
    | )

# 4. Построение и сохранение векторного индекса
vector_store = FAISS.from_documents(
    | chunks, embedding=embeddings
    | )
vector_store.save_local("faiss_index")
```

Рисунок 1 – Индексация документов и построение векторного индекса FAISS

На онлайн-этапе запрос пользователя преобразуется в вектор той же моделью эмбедингов, после чего из индекса извлекается группа наиболее релевантных фрагментов. Найденный контекст вместе с исходным вопросом передаётся в шаблон и направляется генеративной модели (Рис.2).

Для количественной оценки качества ответов использовались метрики, близкие по смыслу к подходу RAG: точность ответа доля вопросов, на которые дан фактически верный ответ, достоверность - доля утверждений ответа, подтверждённых извлечённым контекстом, и доля галлюцинаций. Упрощённая процедура расчёта достоверности ответа приведена в (рис.3).

```

from langchain_core.prompts import (
    ChatPromptTemplate
)
from langchain_core.output_parsers import (
    StrOutputParser
)
from langchain_core.runnables import RunnablePassthrough

retriever = vector_store.as_retriever(
    search_type="similarity",
    search_kwargs={"k": 4},
)

PROMPT = ChatPromptTemplate.from_template(
    "Ответь на вопрос сотрудника, используя\n"
    "только приведённый контекст. Если ответа\n"
    "в контексте нет, сообщи об этом.\n\n"
    "Контекст:\n{context}\n\n"
    "Вопрос: {question}\n\n"
    "Укажи документ-источник в конце ответа."
)

def format_docs(docs):
    parts = []
    for d in docs:
        src = d.metadata.get("source", "н/д")
        parts.append(f"[{src}] {d.page_content}")
    return "\n\n".join(parts)

rag_chain = (
    {
        "context": retriever | format_docs,
        "question": RunnablePassthrough(),
    }
    | PROMPT
    | llm
    | StrOutputParser()
)

answer = rag_chain.invoke(
    "Каков порядок согласования отпуска?"
)
print(answer)

```

Рисунок 2 – Построение генеративной RAG-цепочки

```

def faithfulness_score(answer, context_docs):
    """Доля предложений ответа, подтверждённых
    хотя бы одним извлечённым фрагментом.
    """
    context_text = " ".join(
        d.page_content.lower() for d in context_docs
    )
    sentences = [
        s.strip() for s in answer.split(".")
        if s.strip()
    ]
    if not sentences:
        return 0.0

    supported = 0
    for sentence in sentences:
        words = sentence.lower().split()
        overlap = sum(
            1 for w in words if w in context_text
        )
        if words and overlap / len(words) > 0.5:
            supported += 1

    return supported / len(sentences)

```

Рисунок 3 – Упрощённый расчёт метрики достоверности ответа

По результатам тестирования тестового набора из 50 вопросов по трём конфигурациям получены усреднённые показатели, представленные в таблице 2. Результаты носят демонстрационный характер и получены на ограниченном корпусе документов одной организации, однако отражают общую закономерность, отмечаемую в рассмотренных источниках [2; 3; 5]: RAG-система превосходит генерацию без извлечения по точности и достоверности ответов и превосходит классический поиск по удобству восприятия результата, ценой некоторого увеличения времени отклика.

Таблица 2 – Сравнение показателей эффективности трёх конфигураций (пилотное тестирование, n = 50 вопросов)

Показатель	BM25-поиск	LLM без RAG	RAG-система
Точность ответа, %	58	61	84
Достоверность (faithfulness), %	н/п	47	89
Доля галлюцинаций, %	н/п	35	9
Среднее время ответа, с	0,3	1,8	2,6
Наличие ссылки на источник	да	нет	да

Наибольший прирост RAG-система демонстрирует по показателю достоверности ответа и доле галлюцинаций: наличие явного контекста в промптов вынуждает модель опираться на предоставленные фрагменты документов, а не на параметрические знания, зафиксированные при обучении. Увеличение среднего времени ответа по сравнению с генерацией без извлечения (на 0,8 с) обусловлено дополнительным этапом векторного поиска и может быть снижено за счёт кэширования частых запросов и предварительного вычисления эмбеддингов.

4 Выводы

В результате исследования спроектирован и программно реализован экспериментальный стенд RAG-системы для корпоративного управления знаниями, включающий индексацию документов, векторный поиск на базе FAISS. Пилотное сравнительное тестирование показало, что применение RAG существенно повышает точность и достоверность ответов по сравнению с генерацией LLM без привлечения внешних источников и обеспечивает более удобное для пользователя представление результата по сравнению с классическим полнотекстовым поиском. Полученные результаты подтверждают выводы рассмотренных исследований о целесообразности

применения RAG в корпоративных информационных системах управления знаниями.

Библиографический список

1. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W., Rocktäschel T., Riedel S., Kiela D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems (NeurIPS). 2020. Vol. 33. P. 9459–9474. URL:<https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.htm>
2. Антонов И. В., Бруттан Ю. В. Применение больших языковых моделей для интеллектуального поиска и извлечения информации в корпоративных информационных системах // Компьютерные исследования и моделирование. 2025. Т. 17 № 5. С. 871–888. URL:<https://crm.ics.org.ru/journal/article/3656/>
3. Мельников, А. В., Николаев, И. Е., Русанов, М. А., Аббазов, В. Р. Сравнительный анализ методов RAG для построения русскоязычных интеллектуальных сервисов // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2025. Т. 25 №. 2. С. 5-18. URL: <https://cyberleninka.ru/article/n/sravnitelnyy-analiz-metodov-rag-dlya-postroeniya-russkoyazychnyh-intellektualnyh-servisov>
4. Мащенко А. Е. Использование RAG-подхода для автоматизации конфигурации корпоративных информационных систем // Вестник науки. 2026. Т. 2 №. 4 (97). С. 762-768. URL: <https://cyberleninka.ru/article/n/ispolzovanie-rag-podhoda-dlya-avtomatizatsii-konfiguratsii-korporativnyh-informatsionnyh-sistem>
5. Науменко А. О. Технология RAG как инновационный подход в LLM // Вестник науки. 2025. Т. 5. №. 8 (89). С. 280-289. URL:<https://cyberleninka.ru/article/n/tehnologiya-rag-retrieval-augmented-generation-kak-innovatsionnyy-podhod-v-llm>