МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра системного анализа и
автоматического управления

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ЧАТ-БОТА ДЛЯ СТУДЕНТОВ УНИВЕРСИТЕТА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Черепкова Дмитрия Романовича

Научный руководитель
к. ф.-м. н., доцент _____ И. Е. Тананко

Заведующий кафедрой
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. Современный студент в процессе учебы уже вынужден работать с огромным объёмом информации: учебно-методические пособия на сотни страниц, разноязычные форумы, чужие рукописные лекции. Помимо этого приходится тратить время на бытовые вопросы вроде «Какие документы нужны для заселения в общежитие?» — и почти никогда не удаётся получить ясный ответ с первой попытки. На практике такой запрос оборачивается долгим поиском по сайту университета, звонками и распросами в групповых чатах. Информация разбросана по десяткам источников, у каждого из которых свой формат, своя степень актуальности и своя логика организации. Проблема уже не в отсутствии данных, а в том, что они не работают на человека.

Архитектура Retrieval-Augmented Generation (RAG) придумана как раз для таких ситуаций. В отличие от классических чат-ботов по скриптам или «голых» языковых моделей, которые нередко выдумывают убедительно звучащие ответы, сервис на RAG сначала ищет нужные фрагменты в базе документов и только потом на их основе формулирует ответ. Модель получает конкретный текст в качестве контекста и работает строго в его рамках.

За качество ответов принято винить языковую модель. На практике же первопричина чаще в в поиске и в том, что за данные ему поступают. Модель лишь оформляет то, что нашёл поисковик. Если тот извлек устаревший приказ или бессмысленно нарезанный текст — ответ окажется соответствующе плохим. Так что создание чат-бота начинается не с выбора модели, а с построения пайплайна — конвейера, который превращает разрозненные данные в организованную базу знаний.

Большинство готовых решений для создания чат-ботов предполагают либо идеально структурированные данные (которых у большинства организаций банально нет), либо просто игнорируют проблему их подготовки — и именно это приводит к неудачам при внедрении. Для университета, где информация хаотично разбросана по сайтам, папкам и архивам, это особенно болезненно: собственный цифровой помощник нужен, а сделать его нормально — задача не тривиальная, поскольку упирается не в выбор модели, а в инженерную работу с данными.

Цель бакалаврской работы — спроектировать и описать конвейер

подготовки данных для образовательного RAG-ассистента. Речь идёт о полном цикле: от сбора разрозненных университетских документов до их загрузки в векторное хранилище в виде, пригодном для точного поиска.

Поставленная цель определила **следующие задачи**:

1. Проанализировать проблемы консолидации и подготовки данных в информационной среде университета, а также современные методы их решения в рамках RAG-архитектуры;
2. Спроектировать архитектуру сквозного пайплайна данных для образовательного чат-бота на основе RAG;
3. Реализовать базовую версию пайплайна (MVP) на выбранном стеке технологий и выявить её ограничения;
4. Разработать компоненты улучшенной версии пайплайна: унифицированный парсер, модуль нормализации текста, стратегии семантического чанкования, механизм инкрементального обновления индекса, трансформацию пользовательского запроса;
5. Провести количественную оценку эффективности разработанных компонентов путём сравнения с базовой версией.

Методологические основы построения RAG-систем и информационных конвейеров представлены в работах P.Lewis[1], Y.Gao[2], C.D.Manning[4]. Вопросы построения эффективных пайплайнов данных и применения методов обработки естественного языка рассмотрены в работах N. Reimers, I. Gurevych [5], а также M. Korobov [6]. Технологические решения для парсинга документов и векторного поиска базируются на исследованиях C. Auer [7] и L. Wang [8].

Практическая значимость бакалаврской работы. В ходе выполнения выпускной квалификационной работы был разработан и реализован конвейер подготовки данных для чат-бота NETutor, предназначенного для студентов Саратовского государственного университета. Разработанные компоненты, в числе которых унифицированный парсер, модуль нормализации текста, стратегии чанкования, механизм инкрементального обновления и модуль трансформации запроса, могут быть использованы при построении RAG-систем в других образовательных организациях. Все компоненты реализованы с опорой на локальные инструменты и открытые библиотеки, что обеспечивает независимость системы от внешних сервисов и соответствие требова-

ниям законодательства о персональных данных.

Структура и объем работы. Бакалаврская работа состоит из введения, 4 разделов, заключения, списка использованных источников и 2 приложений. Общий объем работы — 68 страниц, из них 47 страниц — основное содержание, включая 10 рисунков и 1 таблицу, список использованных источников информации — 32 наименования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические основы построения информационных конвейеров для RAG-систем» состоит из трёх подразделов и посвящен анализу архитектуры Retrieval-Augmented Generation и роли этапа извлечения релевантной информации.

В подразделе 1.1 рассматривается устройство RAG-архитектуры. Поступающий от пользователя запрос на естественном языке преобразуется в векторное представление (эмбеддинг), по которому выполняется семантический поиск в предварительно построенной векторной базе данных. Найденные текстовые фрагменты (чанки) формируют контекст, который вместе с исходным запросом передаётся языковой модели для синтеза ответа. Единственный канал, по которому информация поступает к языковой модели — это компонент поиска, что делает качество подготовки данных критическим фактором. Анализируется опыт компании DoorDash, которая, не меняя языковую модель, снизила частоту галлюцинаций на 90% за счёт замены эмбеддинг-модели, контекстуализации запросов и итеративной доработки поиска. Подчёркивается, что улучшение этапа извлечения информации на 20–30% приводит к пропорциональному росту точности системы.

В подразделе 1.2 проводится сравнительный анализ проблем консолидации и подготовки данных в корпоративных и образовательных RAG-проектах. На примере телекоммуникационной компании Bell показана проблема поддержания актуальности данных: задержка синхронизации даже на неделю создавала операционный риск. Решение было найдено в модульном пайплайне с автоматическими триггерами и системой метаданных. На примере NASA рассмотрена проблема семантической несвязности архива: документы разных эпох и проектов путались при простом векторном поиске, что потребовало интеграции гибридного поиска и GraphRAG. Опыт Гарвардской школы бизнеса (ChatLTV) продемонстрировал проблему консолидации

чрезвычайно разнородных источников: структурированные лекции, слайды, неформальные обсуждения. Выделены три общие проблемы: актуальность данных, семантическая несвязность и консолидация. Показано, что в университете эти проблемы стоят острее, чем в корпорациях, из-за децентрализации, хаотичного обновления и отсутствия единых стандартов.

В подразделе 1.3 описываются методологические подходы к обработке неструктурированных текстовых данных. Первая задача — извлечение текстового содержимого из всех потенциальных источников (PDF, DOCX, HTML, сканы) с помощью специализированного инструментария. Вторая задача — очистка от артефактов форматов, ошибок распознавания и приведение к единому формату: токенизация, лемматизация, удаление стоп-слов. Отдельное внимание уделяется обработке специальных терминов и аббревиатур. Третья задача — чанкование: разбиение текста на фрагменты для поиска. Рассматриваются два подхода: наивное чанкование (нарезка на куски фиксированного размера, которое игнорирует языковую и логическую структуру документа) и семантическое чанкование (выделение логически завершённых блоков на основе смысловой целостности). Описывается процесс векторизации текста с помощью нейросетевых моделей эмбеддингов и обосновывается необходимость событийно-управляемой архитектуры конвейера для поддержания актуальности базы знаний.

Таким образом, в первом разделе обосновано, что качество RAG-системы определяется прежде всего этапом извлечения информации, а не мощностью языковой модели. Выявлены три ключевые проблемы университетских данных (актуальность, семантическая несвязность, консолидация) и определены направления их решения: модульный пайплайн, инкрементальное обновление и семантическое чанкование.

Второй раздел «Технологии и архитектурные решения для организации хранения и поиска в RAG-системах» состоит из четырёх подразделов и посвящен выбору и обоснованию технологического стека для проектируемой системы.

В подразделе 2.1 проводится сравнительный анализ трёх инструментов парсинга документов. Docling (IBM Research, 2024) учитывает макет страницы, определяет заголовки, основной текст, таблицы, подписи к рисункам, восстанавливает строки и колонки таблиц, работает локально, лицензия Apache

2.0. LlamaParse (LlamaIndex) — облачный сервис, показывающий лучшее качество на сложных многоколоночных PDF с вложенными таблицами и формулами, но его облачная архитектура исключает использование в университете из-за требований 152-ФЗ о персональных данных. Unstructured.io — наиболее гибкий по спектру форматов (более 25 типов), построен на модульной архитектуре, полностью локальное развёртывание, но уступает Docling в качестве работы с таблицами и сложными PDF-макетами. Обосновывается выбор стратифицированной схемы: Docling как основной парсер для PDF, Unstructured для DOCX, собственные веб-парсеры (beautifulsoup4) для HTML-источников.

Подраздел 2.2 посвящен концепции векторных баз данных и стратегий поиска. Каждый текстовый фрагмент преобразуется в числовой вектор, кодирующий его смысл. Таким образом семантически похожие тексты оказываются геометрически близки в многомерном пространстве. Чистый семантический поиск хорош на однородных текстах, но на разнородных университетских документах сбоит. Гибридный поиск решает эту проблему, совмещая семантический и лексический каналы: результаты объединяются через взвешенное суммирование рангов. В качестве векторного хранилища выбран Qdrant — open-source решение с поддержкой локального развёртывания, фильтрации по метаданным, гибридного поиска и высокой производительностью (время поиска менее 0.05 с).

В подразделе 2.3 рассматриваются методы трансформации пользовательского запроса для компенсации разрыва между разговорным языком студента и официальным языком университетских документов. Query Rewriting — переформулирование запроса в официальном регистре с помощью языковой модели. Multi-Query — генерация нескольких вариантов запроса, каждый с иным акцентом, для увеличения полноты поиска. HyDE (Hypothetical Document Embeddings) — генерация гипотетического документа-ответа, чей вектор естественно ложится в область пространства реальных документов базы. Step-Back Prompting — переформулирование слишком конкретного вопроса в более абстрактный для поиска общих регламентных документов.

Подраздел 2.4 формулирует принципы проектирования сквозных пайплайнов данных для RAG-систем, выработанные на основе анализа реальных внедрений: модульность (смена модели эмбеддингов не затрагивает логику

парсинга, добавление нового источника не требует переписывания стратегии чанкования), инкрементальное обновление (пайплайн отслеживает изменения и обновляет только затронутые фрагменты), обработка ошибок (система не должна падать при встрече с одним плохим файлом), мониторинг и трассировка (каждый результат поиска и источник каждого чанка должны быть записаны), приоритизация источников (более свежие и авторитетные документы имеют больший вес при ранжировании), учёт перспективных расширений (модульная архитектура не должна закрывать возможность внедрения GraphRAG).

На основе сравнительного анализа во втором разделе обоснован технологический стек: Docling для PDF, Unstructured для DOCX, веб-парсеры для HTML, Qdrant как векторное хранилище с гибридным поиском. Сформулированы принципы проектирования пайплайна (модульность, инкрементальное обновление, обработка ошибок, мониторинг), которые легли в основу практической реализации.

Третий раздел «Архитектура пайплайна данных чат-бота» состоит из двух подразделов и описывает место пайплайна в общей архитектуре образовательного чат-бота NETutor.

Подраздел 3.1 описывает разграничение зон ответственности между двумя дипломными работами. Пайплайн данных — предмет данной работы — охватывает всё, что происходит с документом от появления в источнике до загрузки в векторное хранилище: парсинг, предобработка, чанкование, векторизация, загрузка в Qdrant с определённой структурой метаданных. Система поиска и генерации (вторая дипломная работа) отвечает за приём запроса, поиск релевантных фрагментов и сборку ответа с помощью языковой модели. Граница между блоками проходит по интерфейсу Qdrant: пайплайн записывает чанки с определённой структурой метаданных, поисковый компонент их читает.

Подраздел 3.2 содержит классификацию источников данных факультета КНиИТ, которая легла в основу проектных решений. Выделены пять категорий: нормативная документация преимущественно в PDF, часть ее обновляется нерегулярно, но значимо. Расписание — HTML-страницы , обновляется каждый семестр с активной корректировкой первые 2–4 недели. Страницы сайта факультета — HTML sgu.ru с неоднородной структурой, где со-

держательный текст соседствует с навигационными блоками. Методические материалы в DOCX и PDF форматах, степень структурированности сильно варьируется. Оперативные объявления находятся одновременно в нескольких местах (сайт, соцсети), теряют актуальность быстрее всех остальных. На основе анализа формулируются требования к пайплайну: покрытие пяти форматов, инкрементальное обновление, управление версиями чанков через метаданные, нормализация терминологии, устойчивость к низкому качеству входных данных.

В третьем разделе спроектирована архитектура пайплайна данных чатбота NETutor, определены границы ответственности компонентов и проведена классификация пяти типов источников данных факультета КНиИТ. Сформулированные требования задали спецификацию для реализации.

Четвертый раздел «Реализация компонентов пайплайна данных» является центральным и состоит из семи подразделов, описывающих практическую реализацию всех компонентов системы. Разработка велась итеративно: от минимально работающей версии (MVP) к последовательному устранению её ограничений.

В подразделе 4.1 описана базовая версия системы (MVP), реализованная на стеке paraphrase-multilingual-MiniLM-L12-v2, llama3, Qdrant. Базовая версия подтвердила жизнеспособность выбранной архитектуры, но выявила ряд существенных ограничений. Текст попадал в индекс без какой-либо лингвистической обработки — для лексической части поиска это означало, что разные формы одного слова считались разными токенами, а аббревиатуры обрабатывались непредсказуемо. Функция create_collection при каждом запуске удаляла коллекцию и создавала её с нуля — весь корпус переиндексировался целиком, в период переиндексации система была недоступна. Лексическая составляющая поиска (hybrid_search.py) использовала прямой буст по ключевым словам: функция get_keyword_score считала долю совпавших слов и добавляла коэффициент от 1.0 до 1.3, что давало неоправданное преимущество коротким чанкам с высокой плотностью терминов. Запрос пользователя уходил в поиск без каких-либо преобразований — разрыв между разговорным и официальным языком не компенсировался. Парсеры работали исключительно с HTML-источниками (расписание и сайт факультета), оставляя за пределами системы нормативные PDF-документы и методиче-

ские материалы.

Подраздел 4.2 содержит описание унифицированного парсера, реализованного как единый модуль с общим интерфейсом. На входе — путь к файлу или URL, на выходе — структурированный объект ParsedDocument с извлечённым текстом, типом документа и базовыми метаданными. Внутри модуль определяет тип входящего документа через документированный enum (*DocumentType*) и направляет его к соответствующему обработчику. DoclingHandler инициализируется с включённым OCR, распознаванием таблиц и сохранением иерархии заголовков, экспортирует результат в Markdown для сохранения структурной разметки. UnstructuredHandler использует partition_docx, разбивающий документ на структурные элементы (абзацы, заголовки, списки, таблицы). WebParserHandler выбирает специализированный парсер в зависимости от URL (schedule_parser для расписания, netutor_parser для страниц факультета). Обработка ошибок построена по принципу «не останавливаться»: если документ не поддаётся парсингу, исключение логируется с указанием источника и причины сбоя, после чего пайплайн продолжает работу со следующим документом.

В подразделе 4.3 описывается модуль предобработки и нормализации документов (normalizer.py), выполняющий три последовательных прохода. Первый проход — механическая очистка типовых артефактов с помощью регулярных выражений: схлопывание множественных переносов строк, склеивание слов, разорванных дефисом при переносе, удаление одиночных номеров страниц, лишних пробелов и OCR-специфичных ошибок. Второй проход — лингвистическая нормализация (лемматизация) с помощью библиотеки pymorphy2. Лемматизация применяется только к тексту, по которому строится вектор эмбеддинга (поле text_normalized); оригинальный текст сохраняется отдельно (поле text_original) и возвращается пользователю в ответе для цитирования документа в исходной форме. Третий проход — обработка аббревиатур и разговорных алиасов через два набора словарей: PROTECTED_TERMS (СГУ, ВКР, КНиИТ, ФГОС — пропускаются без изменений) и TERM_ALIASES (общага→общежитие, зачетка→зачётная книжка, академ→академический отпуск). Защита от дубликатов реализована на двух уровнях: на уровне документа (сравнение SHA-256 хеша содержимого перед индексацией) и на уровне отдельных чанков (обнаружение частич-

ных дубликатов). Результат работы модуля — объект NormalizedDocument, содержащий text_normalized, text_original, source, doc_type, content_hash, indexed_at, source_date, ocr_quality.

В подразделе 4.4 описана реализация чанкования — ключевого этапа, определяющего качество поиска. В MVP чанкование было реализовано тремя отдельными функциями под каждый тип источника, что неприменимо для PDF и DOCX. В улучшенной версии реализованы три стратегии. Фиксированная (Baseline) — RecursiveCharacterTextSplitter с chunk_size=500 и chunk_overlap=50, используется как запасной вариант. Семантическая — SemanticChunker из LlamaIndex с моделью intfloat/multilingual-e5-base и порогом breakpoint_percentile_threshold=85; граница чанка фиксируется там, где падение косинусного сходства между соседними предложениями попадает в верхние 15% наблюдаемых падений. Иерархическая — HierarchicalNodeParser с chunk_sizes=[1024, 256]; поиск выполняется по мелким дочерним чанкам, а в языковую модель передаётся крупный родительский контекст. Стратегии распределены по типам документов: иерархическое чанкование для нормативных документов и учебных планов, семантическое для методических материалов, предметно-ориентированное для HTML-источников и расписания. Каждый чанк оформляется в структуру *ChunkNode*, содержащую нормализованный и оригинальный текст, chunk_id, parent_id, источник, тип документа, стратегию чанкования, хеш содержимого, даты и оценку качества OCR.

Подраздел 4.5 посвящен модулю трансформации пользовательского запроса. В MVP запрос уходил в поиск без изменений, что приводило к провалам на разговорных формулировках. Модуль построен на абстрактном классе *QueryTransformer* с методом *transform*, возвращающим список вариантов запроса. Реализованы два трансформера. *QueryRewriter* передаёт запрос языковой модели с инструкцией переформулировать его в официальном регистре и сгенерировать несколько вариантов (Multi-Query); исходный запрос включается в список первым — поиск всегда выполняется и по оригинальной формулировке. *HyDETransformer* реализует метод Hypothetical Document Embeddings: модель генерирует гипотетический документ, который мог бы содержать ответ на вопрос, и поиск выполняется по его вектору. Оба трансформера имеют защиту от ошибок — при сбое соединения или некорректном ответе модели возвращается только исходный запрос, и поиск продолжается

без трансформации.

Подраздел 4.6 описывает переход от полной переиндексации к инкрементальному обновлению. В MVP функция create_collection выполняла delete_collection и создавала коллекцию заново при каждом запуске — в период переиндексации система была недоступна, а информация об изменениях терялась. В улучшенной версии каждый чанк записывается в Qdrant как точка с payload, содержащим: text, text_original, source, doc_type, chunk_strategy, parent_id, position, content_hash (SHA-256 от нормализованного текста), source_date, indexed_at, ocr_quality и флаг is_outdated. При проверке пайплайн вычисляет хеш текущего документа, сравнивает с хешами существующих точек: чанки с новыми хешами добавляются (*upsert*), чанки с исчезнувшими хешами помечаются is_outdated=True (не удаляются физически, что страхует от ошибочной переиндексации). Каждая операция с индексом журналируется через *IndexingReport*: источник, количество добавленных и устаревших чанков, ошибки, длительность.

В подразделе 4.7 приведена количественная оценка эффективности. Методология тестирования включала десять реальных студенческих запросов, покрывающих разные сценарии: чёткие формулировки, сленг, орфографические ошибки, аббревиатуры. Каждый вопрос подавался трём конфигурациям: MVP (MiniLM, llama3), NEWutor без рерайта (e5-base, qwen2.5:3b, с нормализацией и инкрементальным обновлением) и полная версия NEWutor с Query Rewriting.

Среднее время ответа: MVP — 27.1 с (разброс 12.0–50.7 с), NEWutor без рерайта — 13.8 с (разброс 7.2–22.1 с), NEWutor с рерайтом — 20.2 с. Снижение почти в два раза и сокращение разброса в 2.5 раза. Покомпонентный анализ показал, что в варианте без рерайта 93% времени уходит на генерацию ответа языковой моделью, а поиск в Qdrant занимает менее 0.05 с. С рерайтом трансформация запроса занимает 31% времени, генерация ответа — 60%.

Качество поиска: MVP находил релевантный контекст в половине случаев, NEWutor — в девяти из десяти. На запросе «чё по общагам?» MVP и NEWutor без рерайта находили нерелевантные фрагменты, тогда как полная версия с рерайтом вернула корректную информацию о студенческих общежитиях — сработала цепочка: лемматизация, алиасы, переформулирование запроса. Единственный запрос, с которым не справилась ни одна версия («что

такое ВКР?»), выявил ограничение исходных данных, а не архитектуры поиска.

В четвёртом разделе реализованы все компоненты пайплайна: унифицированный парсер, модуль нормализации текста, три стратегии чанкования, трансформация запроса (Query Rewriting, HyDE) и инкрементальное обновление Qdrant. Количественная оценка подтвердила эффективность подхода: среднее время ответа снизилось с 27.1 до 13.8 секунды, доля успешного поиска выросла с 50% до 90%.

ЗАКЛЮЧЕНИЕ

В работе рассмотрены теоретические основы и практическая реализация пайплайна данных для чат-бота на RAG-архитектуре NETutor, предназначенного для студентов СГУ.

Анализ показал, что точность RAG-системы ограничивает не столько выбранная языковая модель, сколько организация данных и поискового этапа — единственный канал передачи информации к модели. В образовательном контексте это особенно критично: неверная информация о сроках аттестации или порядке оформления документов влечёт конкретные потери для студента.

Разработка велась итеративно. Базовая версия (MVP) подтвердила жизнеспособность архитектуры, но выявила ряд ограничений: отсутствие нормализации текста, полную переиндексацию при каждом обновлении, упрощённый лексический поиск и отсутствие трансформации запроса. Каждое из них последовательно устранялось в практической части работы.

В ходе работы получены следующие результаты:

— реализован унифицированный парсер на основе стратифицированной схемы: Docling для PDF-документов, Unstructured для DOCX, собственные парсеры для HTML-источников;

— разработан модуль нормализации текста, включающий механическую очистку, лемматизацию (pymorphy2), обработку аббревиатур и разговорных алиасов, дедупликацию по SHA-256;

— реализованы и сравнены три стратегии чанкования — фиксированная (500 символов), семантическая (SemanticChunker, порог 85-й перцентиль) и иерархическая ([1024, 256] символов), распределённые по типам документов;

— внедрён механизм инкрементального обновления, заменивший полную переиндексацию: сравнение хешей, точечное обновление затронутых фрагментов, мягкая пометка устаревших;
— реализованы методы трансформации пользовательского запроса — Query Rewriting и HyDE — с защитой от ошибок;
— проведена количественная оценка на десяти реальных запросах: среднее время ответа снизилось с 27.1 с (MVP) до 13.8 с (улучшенная версия), разброс сократился с 12–50 с до 7–22 с, доля успешного нахождения релевантной информации выросла с 50 % до 90 %.

Все компоненты реализованы с опорой на локальные инструменты и открытые библиотеки, что обеспечивает независимость системы от внешних сервисов и соответствие требованиям законодательства о персональных данных. Дальнейшая работа предполагает расширение покрытия источников (подключение нормативных PDF-документов и методических материалов в полном объёме) и внедрение детектора формальности запроса для выборочного применения рерайта.

Основные источники информации:

1. Lewis P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis [и др.] // arXiv:2005.11401, 2020.
2. Gao Y. Retrieval-Augmented Generation for Large Language Models: A Survey / Y. Gao [и др.] // arXiv:2312.10997, 2023.
3. Manning C. D. Introduction to Information Retrieval / C. D. Manning, P. Raghavan, H. Schütze. — Cambridge: Cambridge University Press, 2008. — 506 с.
4. Reimers N. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks / N. Reimers, I. Gurevych // arXiv:1908.10084, 2019.
5. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages / M. Korobov // Analysis of Images, Social Networks and Texts: 4th International Conference, AIST 2015. — Cham: Springer, 2015. — P. 320–332.
6. Auer C. Docling Technical Report / C. Auer [и др.] // arXiv:2408.09869, 2024.
7. Wang L. Multilingual E5 Text Embeddings: A Technical Report / L. Wang [и др.] // arXiv:2402.05672, 2024.