Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления

Кафедра Интеллектуальных информационных технологий

**ОТЧЁТ**

**По лабораторной работе № 1**

по дисциплине «Естественно-языковой интерфейс интеллектуальных систем»

по теме: “ Разработка информационно-поисковой системы и методы оценки качества ее работы”

Выполнил: Веркович Е. В.

Лукашевич К. С.

гр. 221703

Проверил: Крапивин Ю. Б.

Минск 2025

**Цель:** Освоить на практике основные принципы реализации информационно-поисковых систем и методы оценки качества их работы.

**Вариант задания:** № 21. Индексация документов - векторная - локальная сеть.

**Структура разработанной системы и основные алгоритмы реализации компонентов системы**

Структуру разработанной системы можно представить, как слоистую архитектуру, со следующими ключевыми модулями:

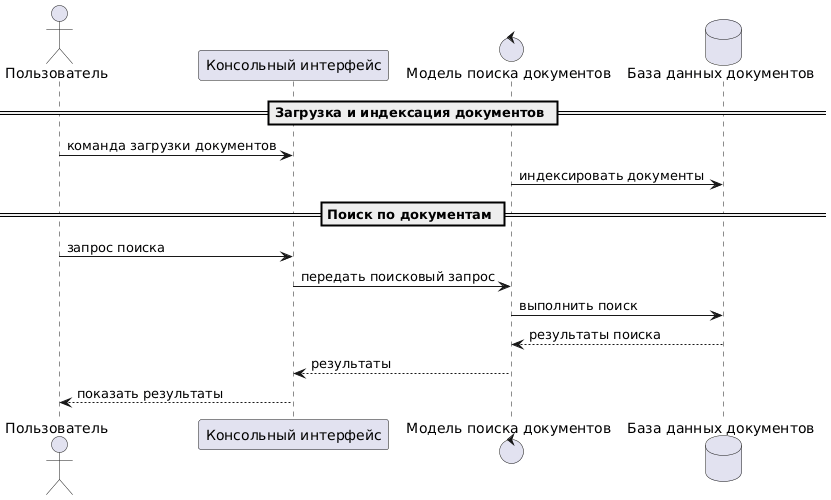


Рисунок 1 - Структура системы поиска

В рамках лабораторной работы был реализован модуль **processors**, который обеспечивает базовую инициализацию, предобработку и поиск по корпусу текстовых документов. В него входят три основных файла, реализующих:

1. **Класс VectorIndexer** — модуль сбора и индексирования документов. Модуль отвечает за формирование базы документов и их преобразование в векторную модель:

\_\_init\_\_(...).Инициализация. Это конструктор, который запускается при создании объекта. Он загружает AI-модель, подготавливает переменные для индекса и карт, а затем вызывает load\_index(), чтобы попытаться загрузить данные с диска. Если данных нет, запускает initial\_crawl().

load\_index(). Загрузка. Этот метод загружает ранее сохранённый FAISS-индекс и словари (path\_to\_id, id\_to\_path) с диска. Это позволяет приложению "вспомнить" своё состояние после перезапуска.

save\_index(). Сохранение. Делает обратное — сохраняет текущее состояние индекса и словарей на диск, чтобы не потерять проделанную работу по индексации.

initial\_crawl(). Первичное сканирование. Запускается только один раз (если индекс пуст). Метод обходит все указанные папки, находит все файлы и поочерёдно добавляет каждый из них в индекс с помощью метода add\_file.

add\_file(...). Добавление / Обновление файла. Это одна из главных рабочих функций. Она принимает путь к файлу, читает его содержимое, превращает текст в вектор с помощью AI-модели, нормализует его и добавляет в FAISS-индекс. Если файл уже существует, метод сначала удаляет старую версию, а затем добавляет новую.

remove\_file(...). Удаление файла. Принимает путь к файлу и удаляет информацию о нём из словарей-карт (path\_to\_id, id\_to\_path), тем самым исключая его из результатов поиска.

2. **Класс VectorSearch** — модуль поиска документов. Задача модуля — обработка пользовательских запросов и поиск релевантных документов:

\_\_init\_\_(...). Инициализация. Этот конструктор очень простой. Он не создаёт ничего нового, а лишь получает доступ к уже готовым компонентам из объекта VectorIndexer:

self.model: AI-модель для превращения запроса в вектор.

self.index: Готовая база данных FAISS для поиска.

self.id\_to\_path: Словарь для связи ID векторов с путями к файлам.

search(...). Основной метод поиска. Он выполняет поиск по шагам:

1. Векторизация запроса: Принимает текстовый запрос пользователя (query), превращает его в числовой вектор с помощью AI-модели и нормализует его.

2. Поиск в FAISS: Отправляет этот вектор в FAISS-индекс (self.index.search). Это самый важный этап, где происходит "магия" семантического поиска. FAISS мгновенно возвращает ID самых похожих документов и их оценку сходства (similarities).

3. **text\_processing.py** — модуль текстовой нормализации. Отвечает за предварительную обработку текста:

1. Используется TweetTokenizer из библиотеки NLTK для разбиения строки на токены.
2. Для каждого слова определяется часть речи (nltk.pos\_tag). В анализ оставляются только существенные категории: существительные, глаголы, прилагательные и числительные.
3. Далее применяется стеммер Lancaster для приведения слов к базовой форме.
4. Все токены переводятся в нижний регистр.
5. Результат работы функции normalize() — список нормализованных токенов, пригодных для дальнейшей индексации и поиска.

**Результаты тестирования системы**

Рассмотрим работу системы с точки зрения пользователя. Проверим выполнение основного функционала системы – поиска релевантных документов по запросу на естественном языке. А также оценим дружелюбность интерфейса.

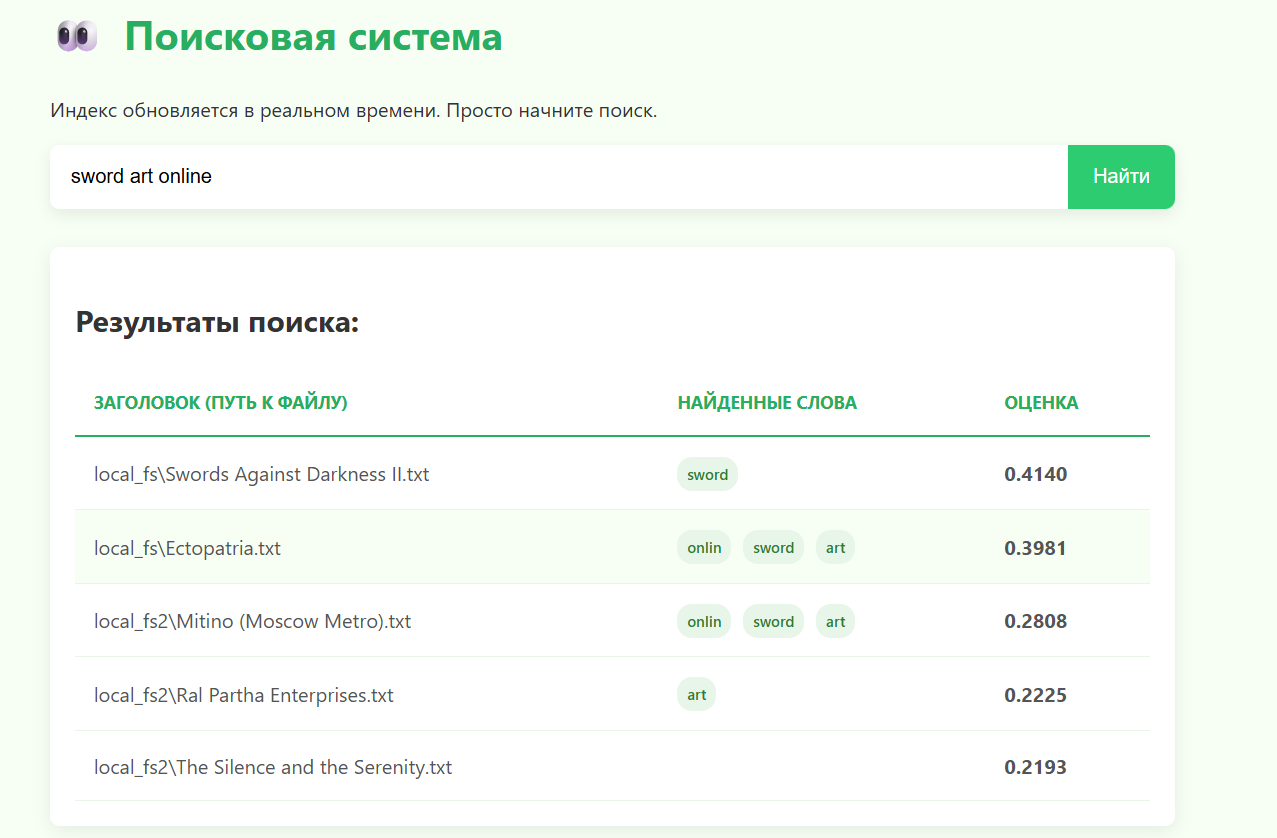


Рисунок 2 - главное меню

**Информация о тестовой коллекции документов**

При создании или обновлении поискового индекса программа загружает случайные статьи из википедии и парсит их краткое содержание, сохраняя в текстовом формате .txt в файлах. Затем данные индексируются и загружаются в таблицу, предварительно получив разметку в соответствии со своими весами.

**Результаты оценки по каждой из метрик (аналитически и графически)**

Для оценки качества работы программы, был реализован отдельный модуль, который запускает тест программы и вычисляет стандартные метрики для оценки релевантности поиска.

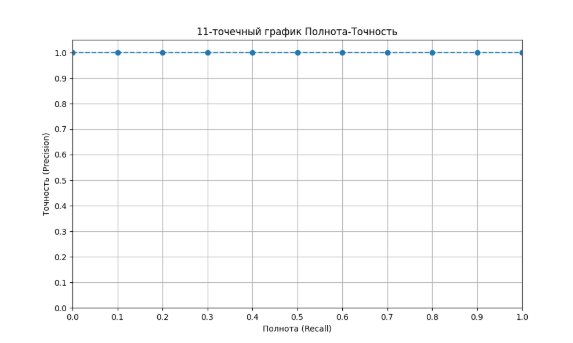
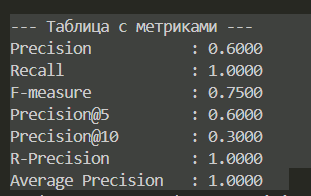


Рисунок 6 - Вычисленные метрики

Интерпретировать можно следующим образом:

Recall: 1.0 значит, что система нашла все релевантные документы из вашего эталонного списка. Ни один правильный документ не был пропущен.

Average Precision: 1.0 метрика говорит о том, что система не просто нашла все нужные документы, а поставила их в самый топ списка результатов, один за другим, без "мусорных" ссылок между ними.

График это прямая линия на уровне Точность = 1.0 — это визуальное подтверждение, что система ранжирует релевантные документы максимально высоко. **Описание и особенности применения готовых к использованию компонентов**

При разработке поисковой системы были применены готовые программные библиотеки, которые существенно упростили реализацию отдельных этапов.

1. wikipedia - предоставляет API-доступ к англоязычной Википедии. Модуль парсера использует библиотеку для генерации случайных статей (wikipedia.random) и получения их содержания (wikipedia.page). Библиотека автоматически обрабатывает исключения (например, неоднозначные запросы или несуществующие страницы), что облегчает загрузку текстов.

2. nltk (Natural Language Toolkit) - библиотека для обработки естественного языка. Использование NLTK позволило быстро внедрить базовую текстовую предобработку без реализации собственных алгоритмов NLP.

3. numpy - библиотека для работы с многомерными массивами и линейной алгебры. Использование numpy существенно ускоряет вычисления по сравнению с чистым Python, что важно для обработки даже небольших корпусов документов. Реализует расчёт норм векторов документов (евклидова длина), вычисление косинусного сходства между вектором запроса и документами, векторизованные операции при ранжировании результатов.

4. watchdog - библиотека для мониторинга файловой системы. Она следит за указанными папками и сообщает, когда в них создаётся, изменяется или удаляется файл.

5. Faiss - библиотека от Meta (Facebook) AI для сверхбыстрого поиска по сходству. Она берёт миллионы числовых векторов и организует их в специальный индекс. Главная задача — по одному новому вектору мгновенно находить в этом индексе самые похожие на него.

6. sentence-transformers - библиотека превращает текст (предложения, абзацы) в числовые векторы, которые отражают его семантический смысл. Она использует мощные AI-модели, чтобы тексты с похожим значением получали математически близкие векторы.