Министерство образования Республики Беларусь

Учреждение образования “Белорусский государственный университет информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчёт по дисциплине

“Естественно-языковой интерфейс интеллектуальных систем”

Лабораторная работа №1

“Разработка автоматизированной системы формирования словаря естественного языка”

Выполнили студенты Целуйко Д.А.

группы 221702: Гринь Н.А.

Проверил: Крапивин Ю.Б.

Минск 2025

Лабораторная работа №1

## **Цель работы:**

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

## **Задачи лабораторной работы (Вариант 9):**

* Изучить основы построения информационно-поисковых систем.
* Реализовать модель поиска документов по запросу.
* Оценить качество работы системы с использованием метрик.
* Разработать десктопный интерфейс для взаимодействия с системой.

## **Язык текста:**

Английский

## **Предметная область:**

Локальная вычислительная сеть

## **Описание инструментов разработки:** В рамках задачи было создано десктопное приложение на языке Python с применением следующих библиотек и технологий:

* **Tkinter** — стандартная библиотека Python для создания графических пользовательских интерфейсов (GUI). Использовалась для построения основного окна приложения, вкладок, полей ввода, кнопок и таблиц для отображения результатов.
* **spaCy** — библиотека для обработки естественного языка (NLP). В проекте была задействована для выполнения лемматизации — приведения слов к их начальной форме (например, "running", "ran" → "run"). Это обеспечивает более точный семантический поиск по сравнению со стеммингом.
* **Scikit-learn** — ключевая библиотека для машинного обучения. В проекте использовалась для реализации векторной модели поиска:
  + **TF-IDF векторизация:** С помощью TfidfVectorizer текстовые данные (содержимое файлов) преобразовывались в числовые векторы на основе частоты слов (TF), скорректированной на их важность в контексте всего корпуса (IDF).
  + **Расчет релевантности:** Функция cosine\_similarity применялась для вычисления косинусного сходства между вектором запроса и векторами всех документов, что позволило ранжировать результаты по степени их релевантности.
* **Watchdog** — библиотека для мониторинга событий файловой системы. Позволила реализовать **фоновый процесс**, который отслеживает создание, изменение и удаление файлов в указанной директории, обеспечивая автоматическое обновление поискового индекса.
* **Ollama** и библиотека ollama — платформа для локального запуска больших языковых моделей (LLM). В проекте использовалась для реализации AI-компонента **суммаризации текста**:
  + **Генерация текста:** Легковесная модель phi3 применялась для создания краткого содержания (саммари) выбранного документа. Запросы к модели выполнялись асинхронно, чтобы не блокировать интерфейс.

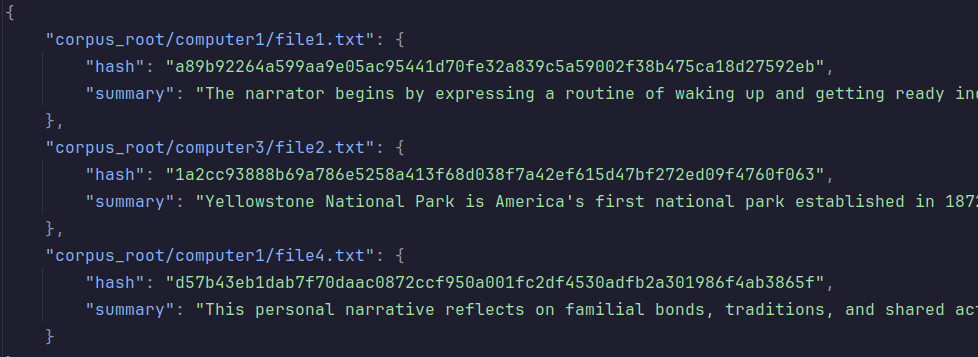
## **Структура разработанной системы:**

Система построена по модульному принципу, где каждый компонент отвечает за свою часть функционала:

* **main.py (GUI и Контроллер):** Основной модуль, отвечающий за графический интерфейс, обработку действий пользователя и координацию работы других компонентов.
* **search\_engine.py (Поисковый движок):** Ядро системы, реализующее индексацию, кэширование и поиск.
* **watcher.py (Мониторинг ФС):** Фоновый процесс, отслеживающий изменения в файлах.
* **summarizer.py (AI Суммаризатор):** Модуль для взаимодействия с Ollama, управляющий генерацией и кэшированием саммари.

**Структуры данных:**  
Вместо традиционной СУБД система использует комбинацию файловой системы и файлов кэша для хранения данных и состояния.

1. Корпус документов (Файловая система):
   1. Структура: *corpus\_root/ -> computer\_N/ -> file.txt.*
   2. Назначение: Имитирует распределенную файловую систему в локальной сети. Является источником данных для индексации.
2. Кэш поискового индекса (vector\_index.pkl):
   1. Структура: Сериализованный Python-объект, содержащий:
      1. Обученный TfidfVectorizer.
      2. Готовую TF-IDF матрицу.
      3. Словари для сопоставления файлов, хешей и метаданных.
   2. Назначение: Обеспечивает практически мгновенный запуск приложения, избавляя от необходимости перестраивать индекс при каждом старте.
3. Кэш суммаризаций (summary\_cache.json):
   1. Структура: JSON-файл вида {"путь\_к\_файлу": {"hash": "...", "summary": "..."}}, рисунок - 1.

Рисунок - 1. Структура кэша суммаризаций (summary\_cache.json)

* 1. Назначение: Хранит сгенерированные саммари, чтобы избежать повторных дорогостоящих запросов к LLM для неизмененных файлов.

**Факторы влияющие на релевантность:**

1. **Частота терминов** (Term Frequency)
   * Чем чаще искомые слова встречаются в документе, тем выше релевантность
   * Документы с большим количеством вхождений поисковых терминов получают более высокий ранг
2. **Обратная частота документа** (Inverse Document Frequency)
   * Редкие слова в общей коллекции документов имеют больший вес
   * Распространенные слова (артикли, предлоги) влияют на релевантность меньше
3. **Близость терминов** (Proximity)
   * Чем ближе друг к другу находятся поисковые слова в тексте, тем выше релевантность
   * Фразовый поиск учитывает порядок и расстояние между словами
4. **Вес полей** (Field Weights)
   * Разные поля документа могут иметь разную важность (например, заголовок весомее основного текста)
   * Веса назначаются по шкале A (самый важный) до D (наименее важный)
5. **Нормализация**
   * Учитывается длина документа (более длинные тексты нормализуются)
   * Предотвращает предвзятость в пользу очень коротких или очень длинных документов

**Скриншоты разработанной системы:**

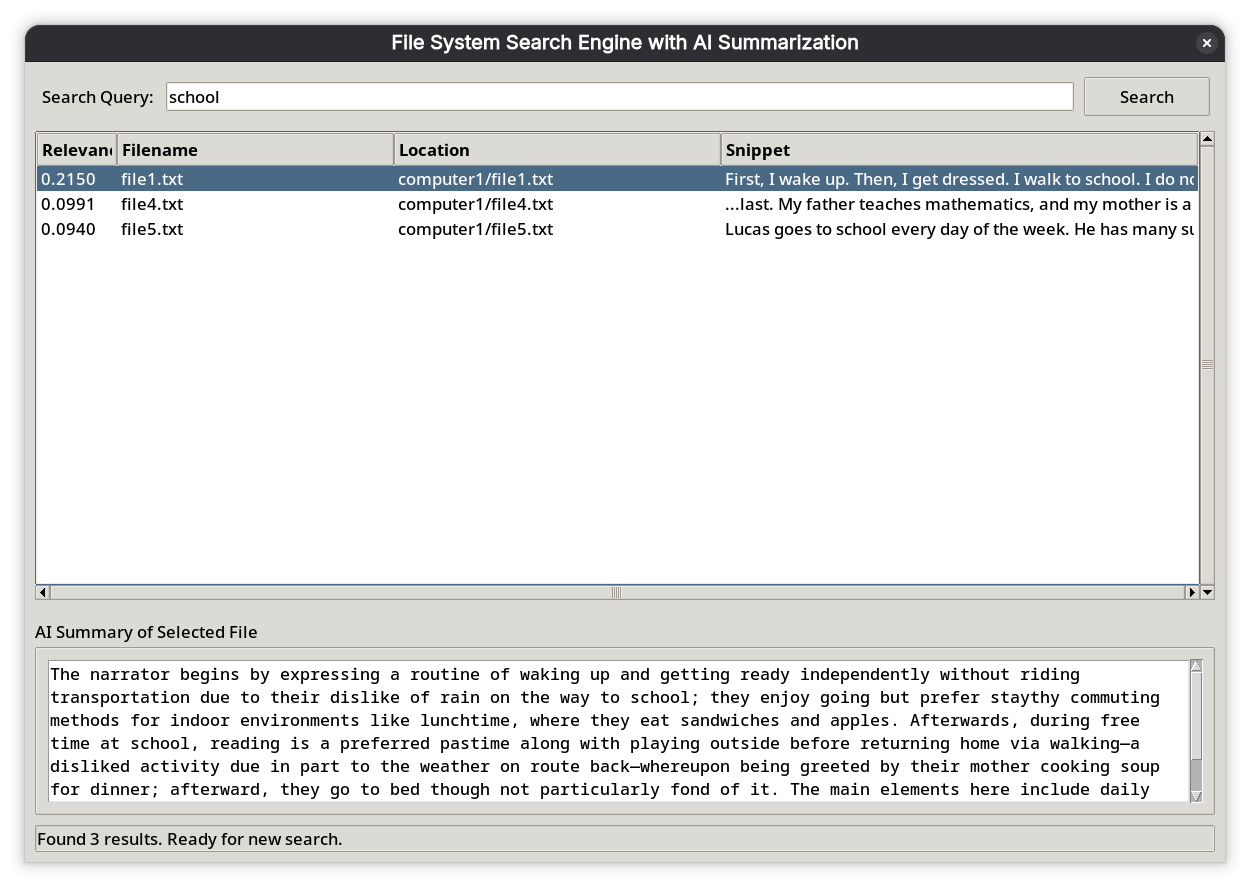


Рис. 2 – Скриншот №1

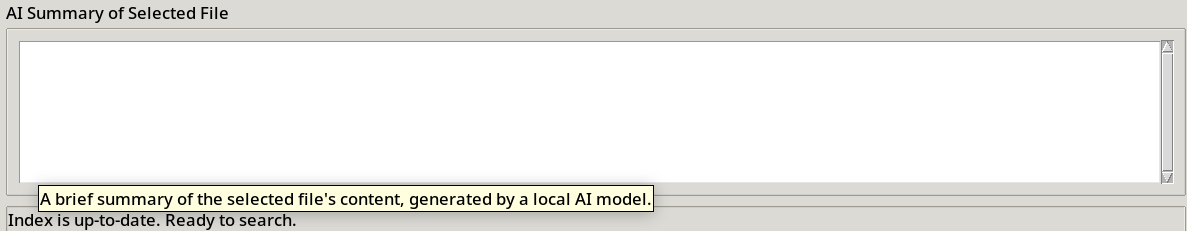


Рис. 3 – Скриншот №2

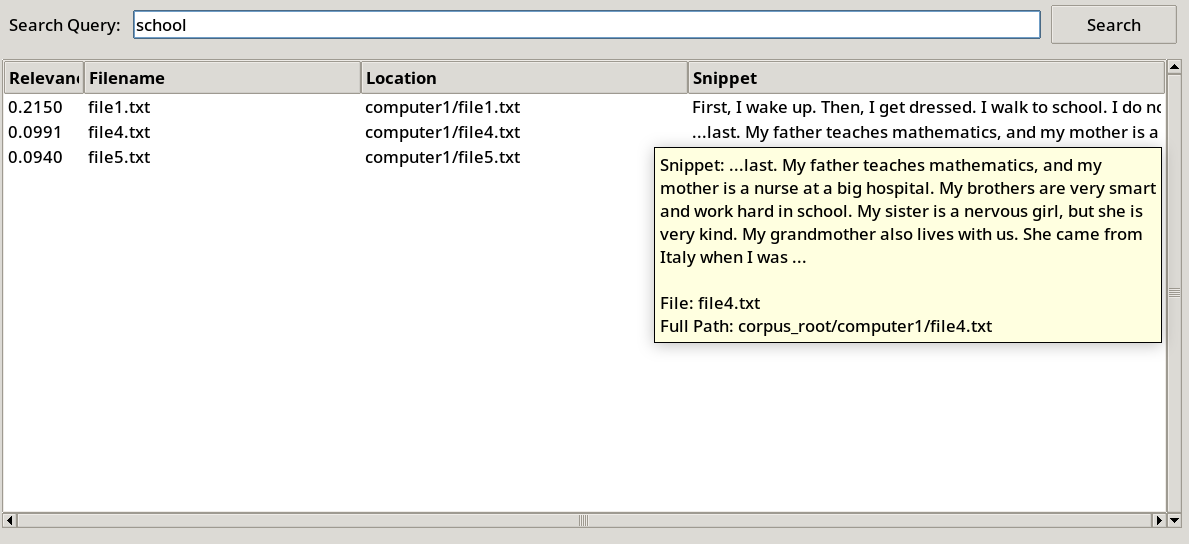


Рис. 4 – Скриншот №3

## **Структурно-функциональная схема приложения:**

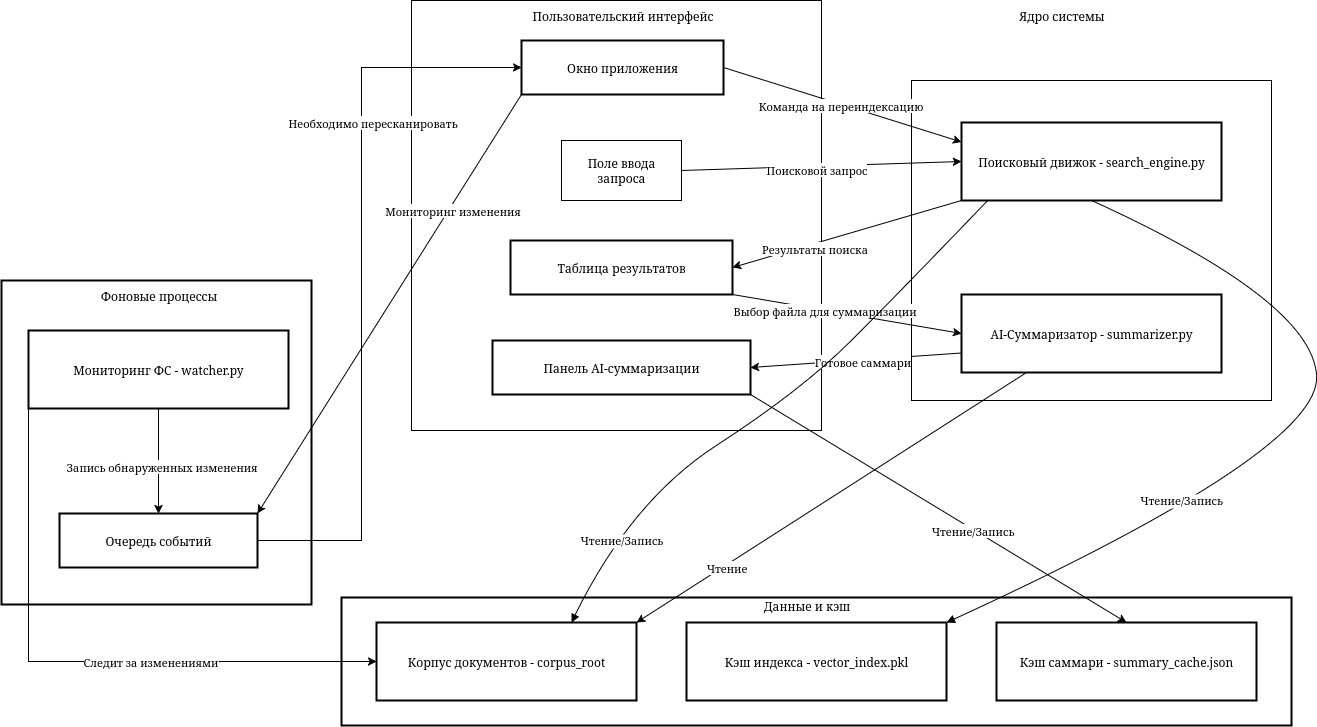


Рис. 5 - структурно-функциональная схема приложения

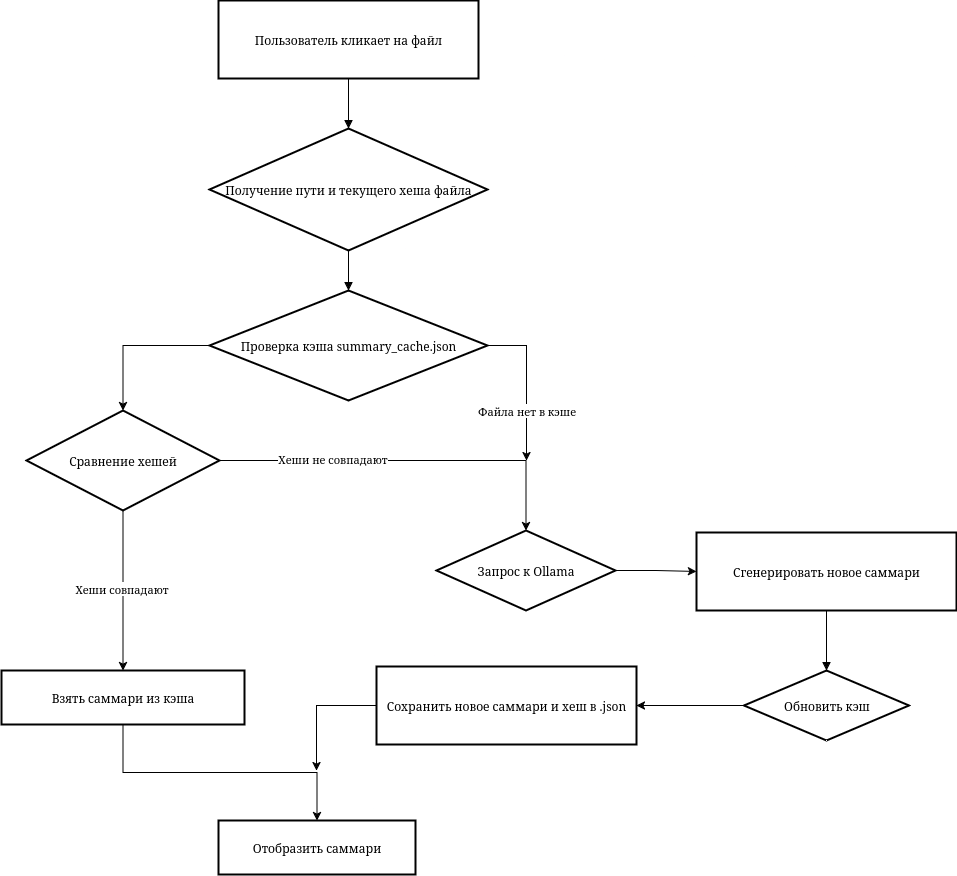


Рис. 6 - структурная схема AI суммаризации

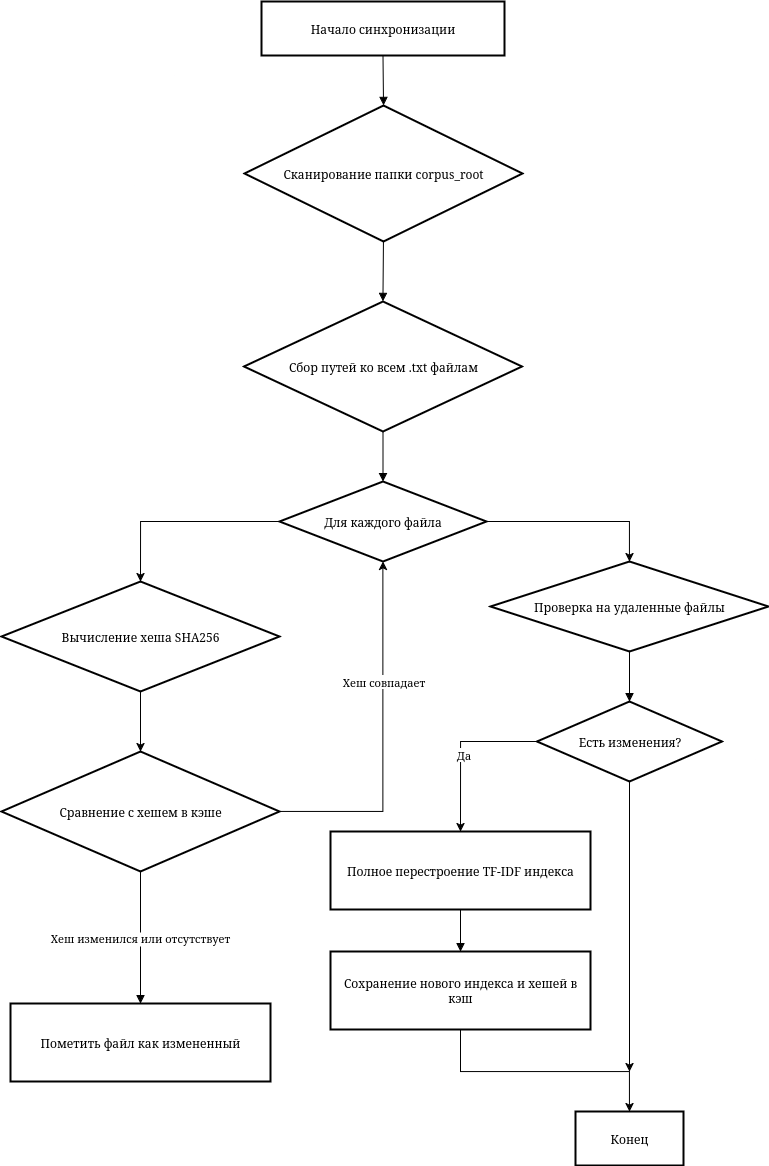


Рис. 7 - структурно-функциональная схема запуска приложения

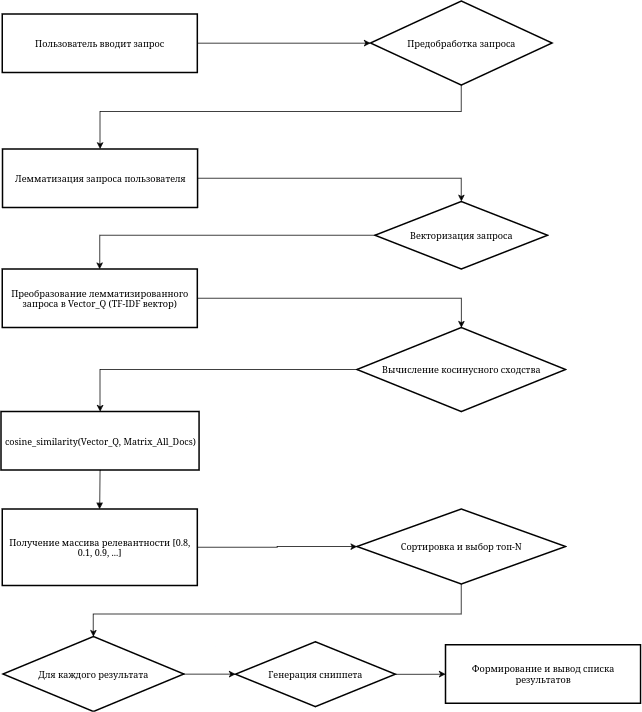


Рис. 8 - структурная схема алгоритма поиска

**Оценка точности поиска приложения**

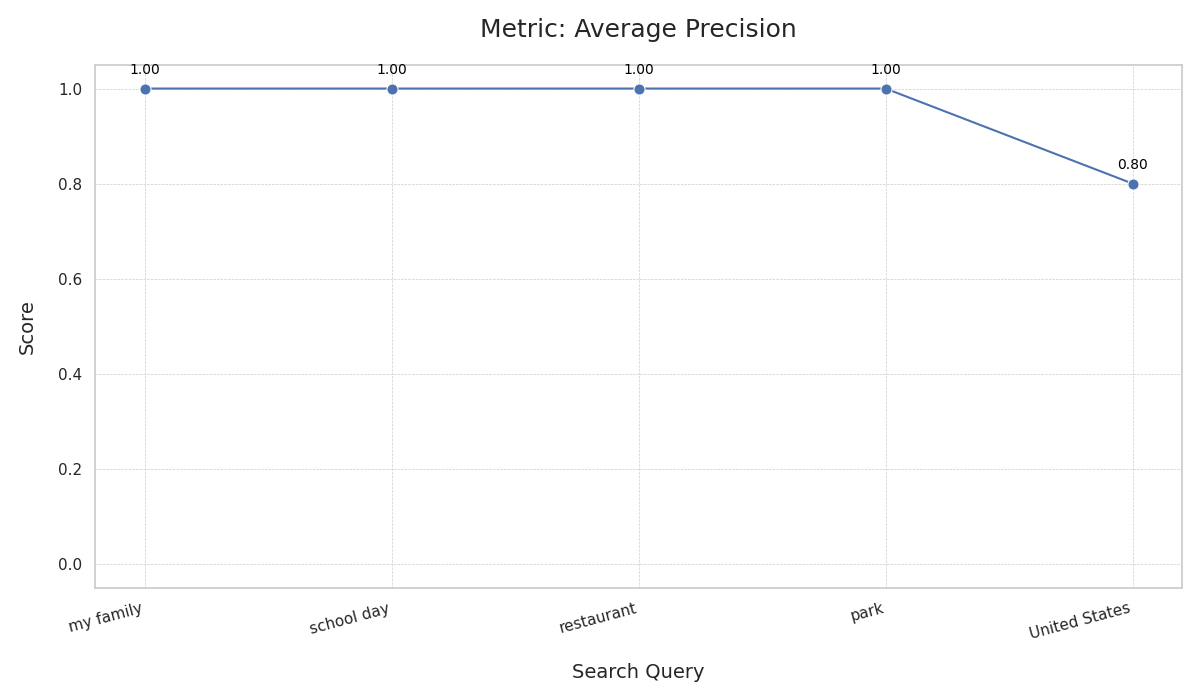


Рис. 9 - график оценки точности приложения. Средняя точность

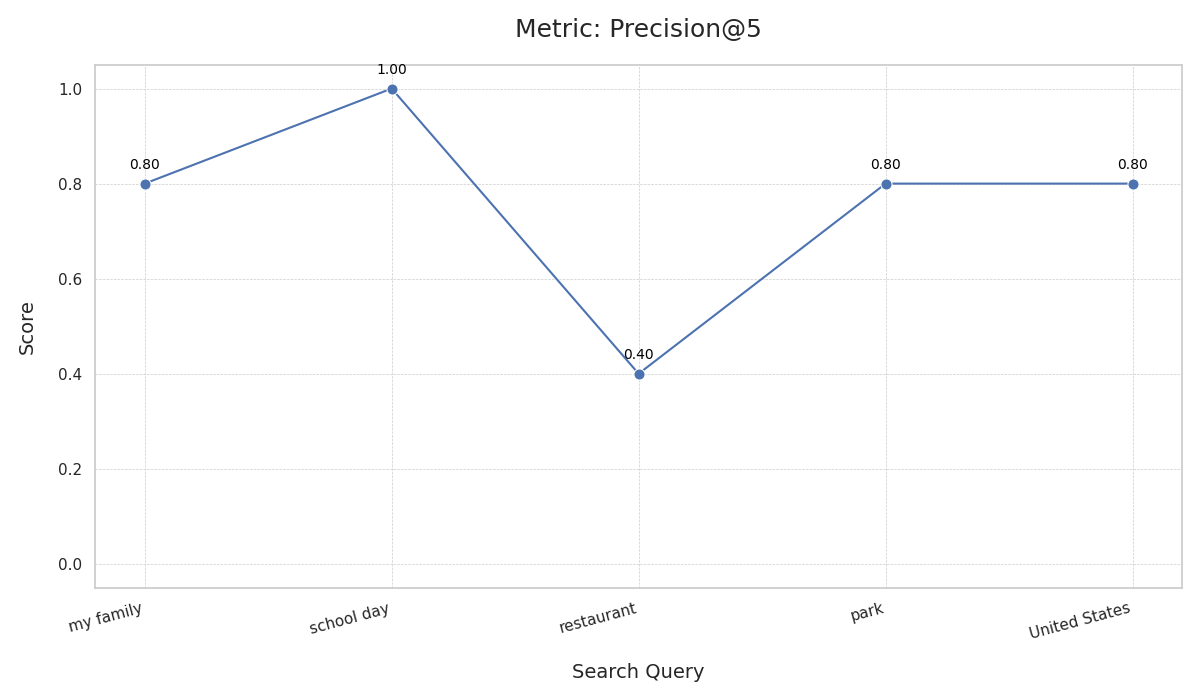


Рис. 10 - график оценки быстродействия приложения. Точность при 5

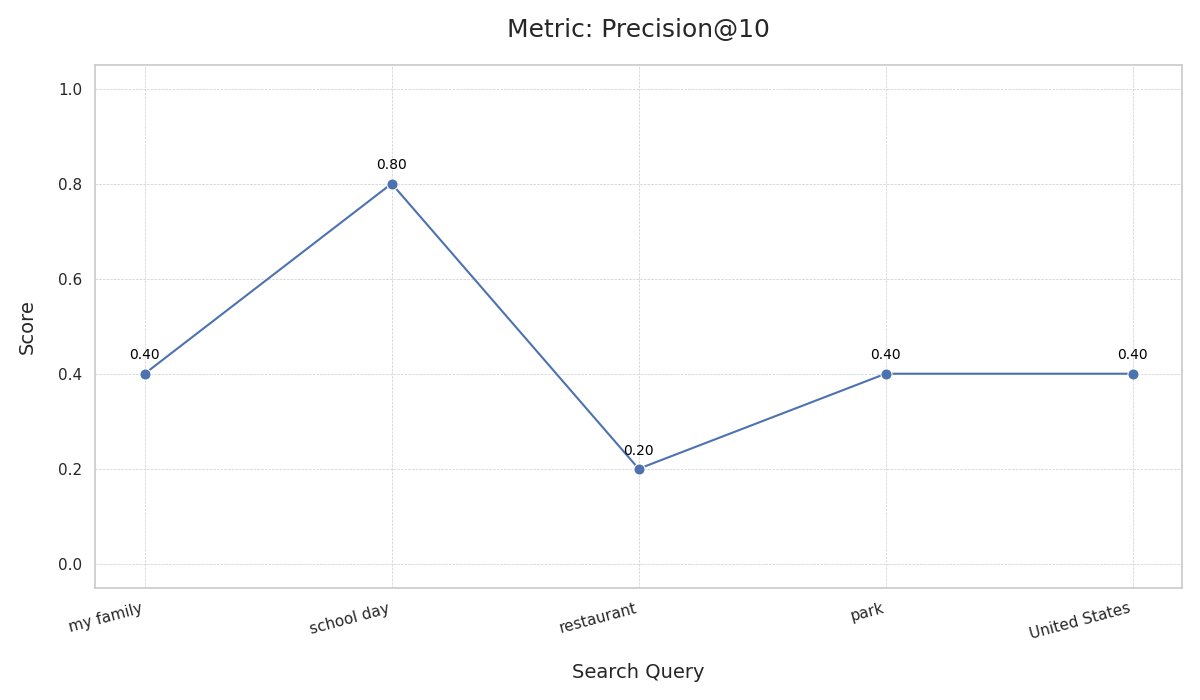


Рис. 11 - график оценки быстродействия приложения. Точность при 10

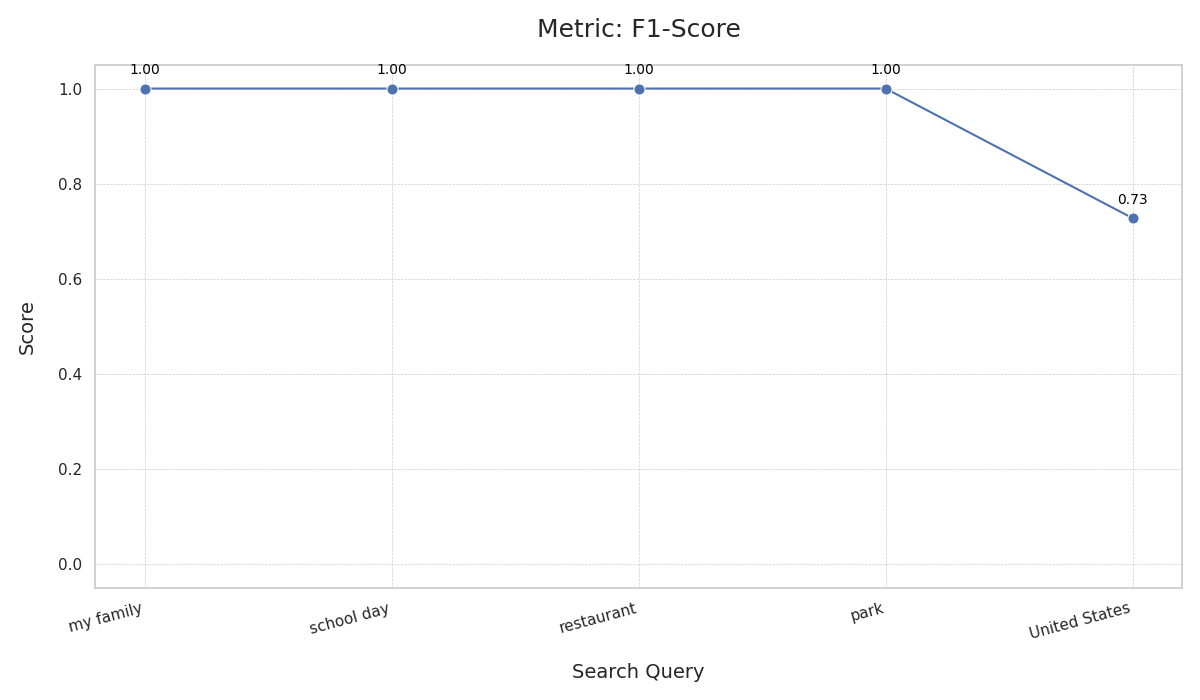


Рис. 12 - график оценки быстродействия приложения. F1-score

## **Выводы по работе и по перспективам развития приложения:**

В ходе выполнения лабораторной работы была успешно разработана информационно-поисковая система, реализующая векторный поиск по локальной коллекции документов. Реализованы ключевые функции: индексация, ранжированный поиск, фоновый мониторинг изменений, а также интегрированы современные AI-компоненты для исправления опечаток и суммаризации текста. Проведена оценка качества системы с помощью стандартных метрик.

## **Перспективы развития приложения:**

1. Интеграция семантического поиска: Вместо TF-IDF использовать более современные подходы, основанные на нейросетевых эмбеддингах (например, Sentence Transformers), чтобы система понимала семантическую близость, а не только совпадение ключевых слов.
2. Улучшение UI/UX: Добавить пагинацию, подсветку ключевых слов в сниппете и возможность открывать файлы прямо из приложения.
3. Интеллектуальная кластеризация: Добавить AI-модуль, который после поиска будет автоматически группировать найденные документы по скрытым темам (например, с помощью алгоритма K-Means).
4. Расширение поддерживаемых форматов: Добавить парсеры для других форматов файлов, таких как .pdf или .docx.