Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и

Радиоэлектроники

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №7

по дисциплине “Естественно-языковой интерфейс интеллектуальных систем”

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

Вариант 1

Выполнили:

Студенты гр. 221701 Абушкевич А. А.

Робилко Т. М.

Проверил: Крапивин Ю. Б.

Минск

2025

**Цель работы**

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

**Задание**

Разработать поисковую систему, удовлетворяющую следующим требованиям:

* Сфера применения - локальная вычислительная сеть
* Стратегия поиска - логическая
* Язык поиска - русский
* Вход системы - множество естественно-языковых текстов, по которым осуществляется поиск
* Выделение ключевых слов документов осуществляется системой автоматически в соответствие с формулой из методического пособия (применяется метрика TF-IDF)
* Система должна позволять пользователю формулировать ЕЯ-запрос
* Выход системы - список документов, релевантных запросу пользователя, в соответствие с моделью поиска, согласно варианту
* Результаты поиска должны содержать: активную ссылку на документ, список слов запроса присутствующих в документе
* На основании информации о существующих метриках, наиболее часто использующихся для оценки качества работы систем информационного поиска, требуется дать оценку работы системы
* Вычисление оценок, получаемых на основании метрик, реализовать программно, путем вызова соответствующего подменю, и отображать в виде таблиц и графиков
* Интерфейс системы должен быть предельно простым и доступным для пользователей любого уровня, содержать понятный набор инструментов и средств, а также help-средства

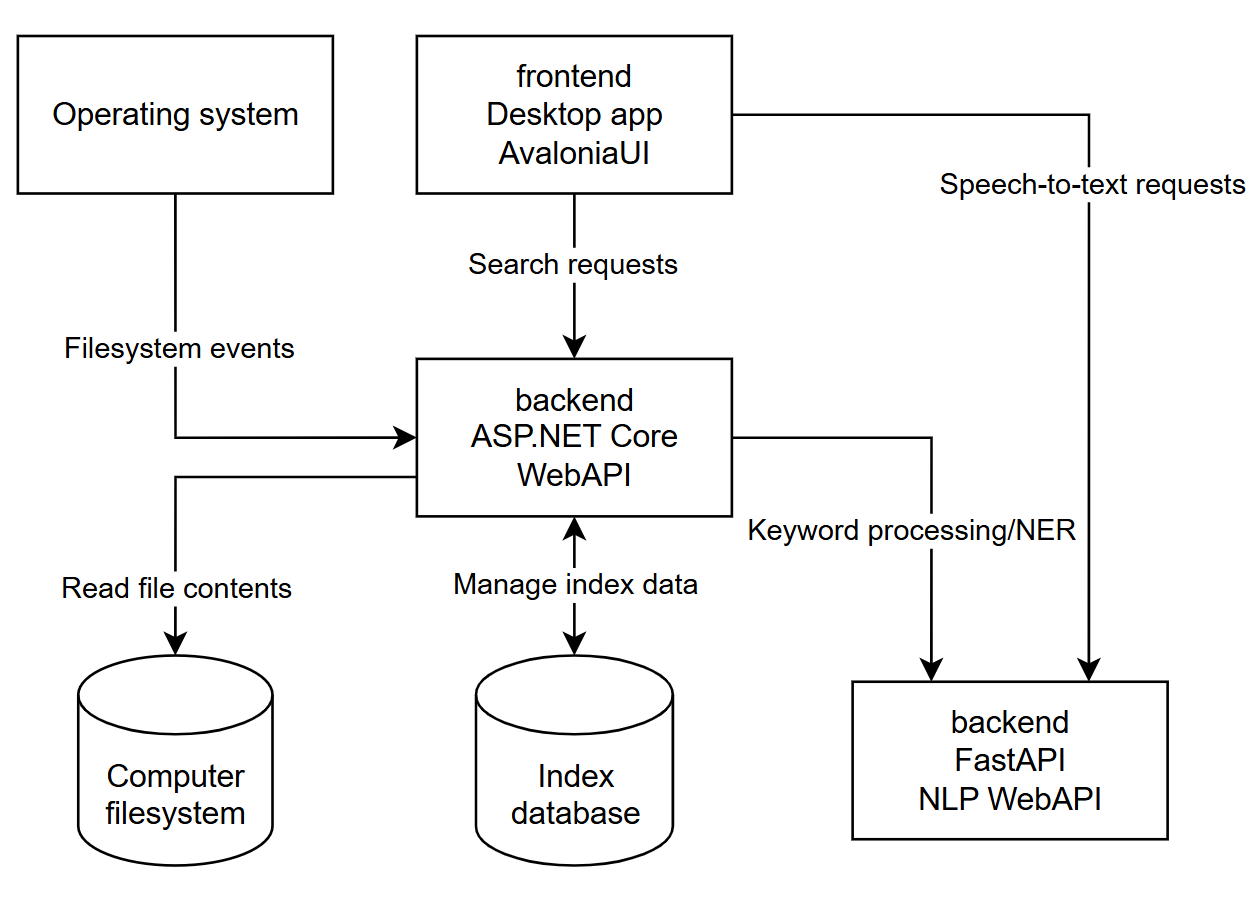
**Структура разработанной системы**

Система состоит из нескольких сервисов, взаимодействующих с операционной системой и пользователем, а также базой данных индексации.

Основные компоненты системы:

1. Пользовательский интерфейс - десктопное приложение на C# (фреймворк Avalonia UI).
2. Сервис индексации на C# (ASP.NET Core). Представляет из себя webAPI для ответа на поисковые запросы, а также индексации файлов в фоне и при запуске сервиса. Сервис подписывается на события операционной системы, связанные с файловой системой.
3. Сервис естественно-языковой обработки. Принимает запросы от пользовательского интерфейса и сервиса индексации. Нормализует формы ключевых слов, извлекает именованные сущности, а также преобразует речь в текст.
4. База данных индекса, где хранятся поисковые образы документов и ключевые слова с количеством содержащих их документов.

Сервис индексации помимо WebAPI содержит паука, который работает в фоне и обрабатывает события из специальной очереди, полученные от операционной системы.

****

**Структура базы данных системы**

База данных индекса содержит две таблицы:

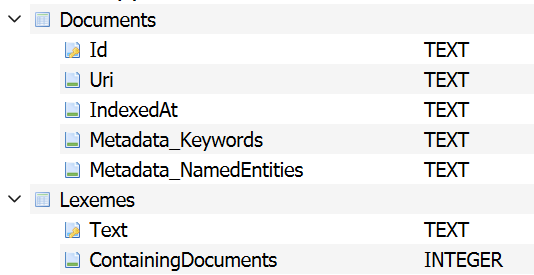


Таблица Documents служит для хранения поисковых образов документов. Содержит уникальный идентификатор (Guid) документов, их URI, дата индексации, а также метаданные: список ключевых слов с их частотой в документе, а также список именованных сущностей.

Таблица Lexemes содержит все используемые в индексированных документах ключевые слова в начальной форме. Для каждого слова хранится количество документов, где оно встречается.

**Основные алгоритмы реализации компонентов системы**

Алгоритм обработки поискового запроса:

* Предобработка полученного запроса
  + Выделение ключевых слов
  + Нормализация ключевых слов
  + Выделение именованных сущностей (местоположения, личности, организации)
* Формирование фильтра для документов
* Получение количества документов в базе данных
* Поиск в базе данных документов, удовлетворяющих фильтру. Для каждого документа применяется полученный фильтр
* Фильтрация документов с релевантностью более 1.5% (получено эмпирически)
* Применение пагинации

Алгоритм получения фильтра документов для конкретного запроса:

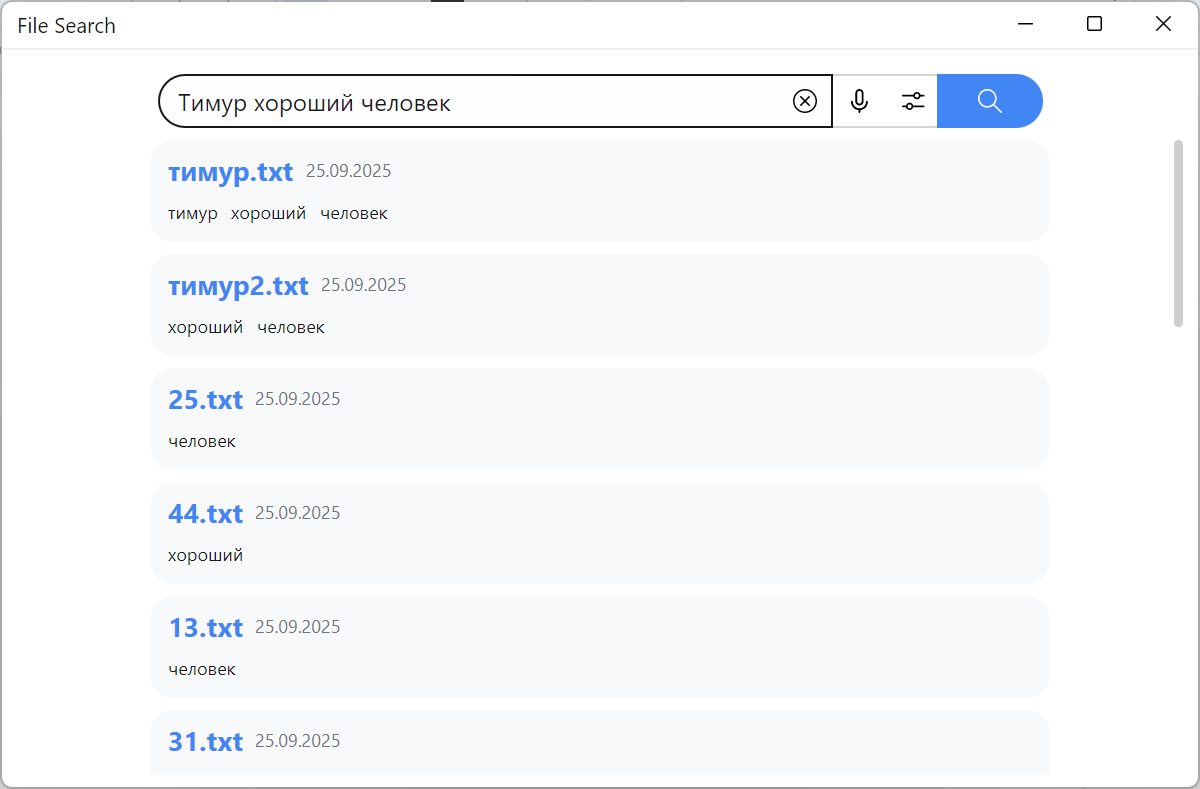
* Нахождение пересечений множества ключевых слов в запросе и в документе
* Расчёт метрики TF-IDF для каждого ключевого слова из найденного множества
* Суммирование всех рассчитанных метрик
* Если в запросе нет именованных сущностей, возврат суммы рассчитанных метрик как меры релевантности документа, а также множество ключевых слов из множества-пересечения
* Нахождение пересечений множества именованных сущностей в документе и в запросе
* Расчет метрики именованных сущностей: деление количества найденных сущностей на общее количество сущностей в запросе
* Расчет и возврат итогового значения релевантности документа как взвешенная сумма метрики по ключевым словам и метрики по именованным сущностям, а также множество ключевых слов из множества-пересечения

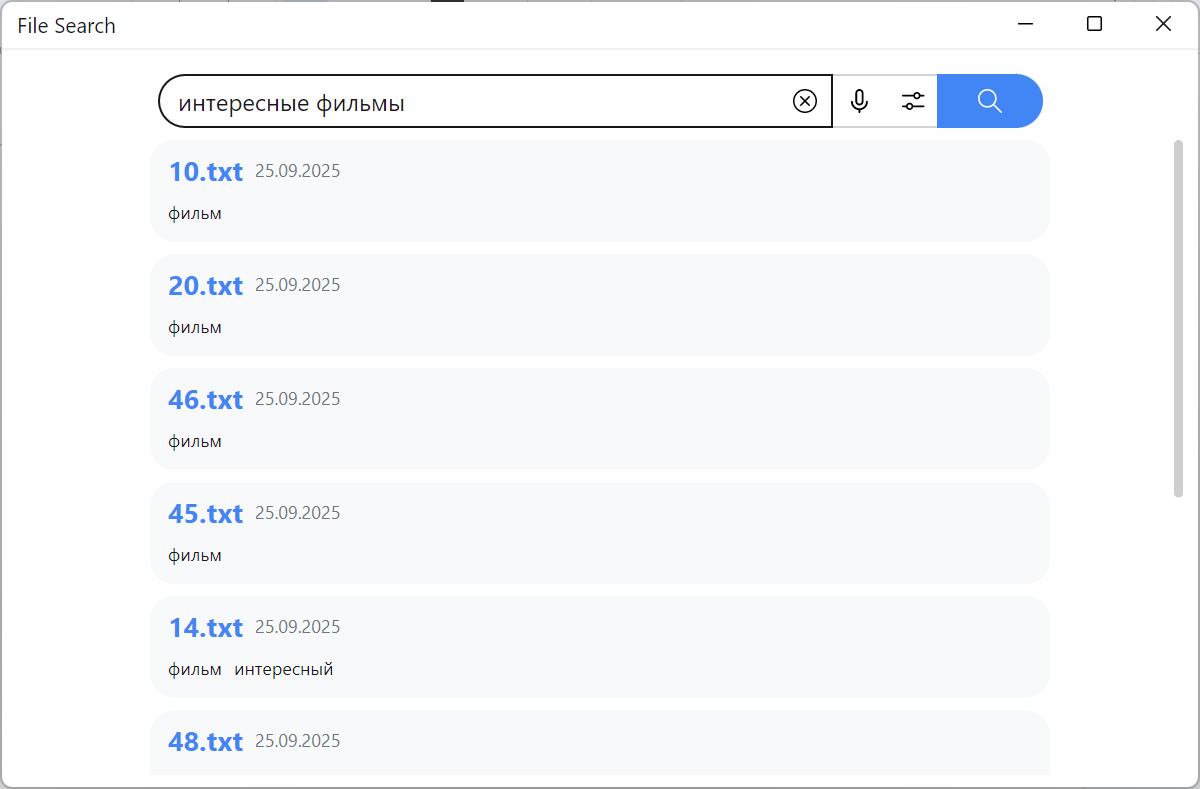
Алгоритм индексации документа:

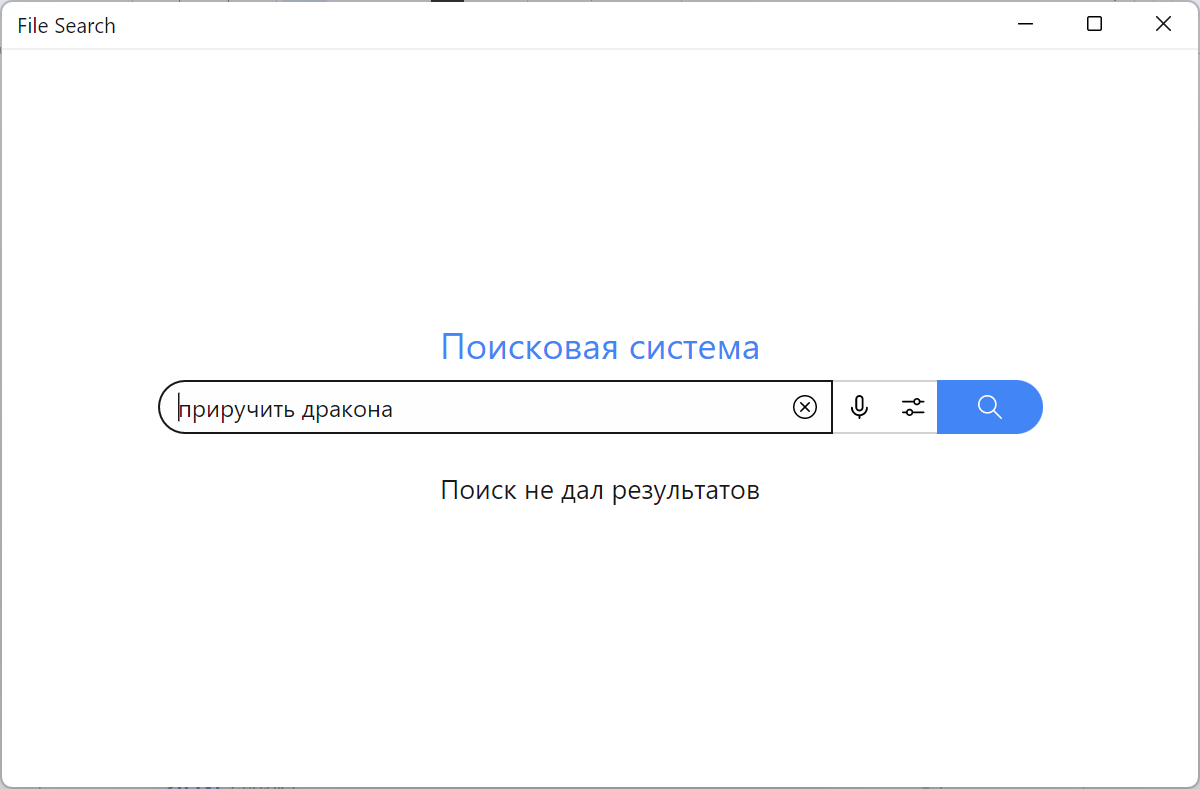
* Если документ уже проиндексирован, конец алгоритма
* Чтение содержимого документа
* Запрос обработанного текста документа у естественно-языкового сервиса
* Для каждого из ключевых слов документа:
  + Если слово уже содержится в базе данных, увеличить количество содержащих слово документ на единицу
  + Если слова еще нет, создать запись с количеством, равным 1
* Добавление образа документа в базу данных индекса

**Результаты тестирования системы**

Система показала приемлемые результаты при ручном тестировании. Были проверены случаи поиска по именованным сущностям и по обычным словам. Система позволила получать желаемые документы в числе первых в поисковой выдаче.

****

****

****

**Информация о тестовой коллекции документов**

Тестовая коллекция документов представляет собой набор текстов из случайно выбранных слов по теме длиной 50-300 слов каждый.

Количество текстов - 500.

Количество уникальных ключевых слов - 98.

Общее количество слов в текстах составило 86278.

Тексты соответствуют разным тематикам, таким как образование и обучение, медицина и здоровье, наука и исследования, бизнес и управление, технологии и программирование, искусство и культура, городское развитие и инфраструктура, спорт и активный образ жизни.

Метрики, по которым проводилось тестирование - mean\_precision, mean\_recall, mean\_accuracy.

Для ручного тестирования были также добавлены несколько дополнительных документов различного содержания: тексты отзывов о фильмах и сериалах, единичные документы с конкретными словами.

**Результаты оценки по каждой из метрик**

Тестирование информационной системы производилось на 25-ти произвольных запросах. Каждый запрос состоит из случайных слов, используемых в документах, в количестве от 1 до 3. Документ считается релевантным, если содержит хотя бы одно слово из запроса.

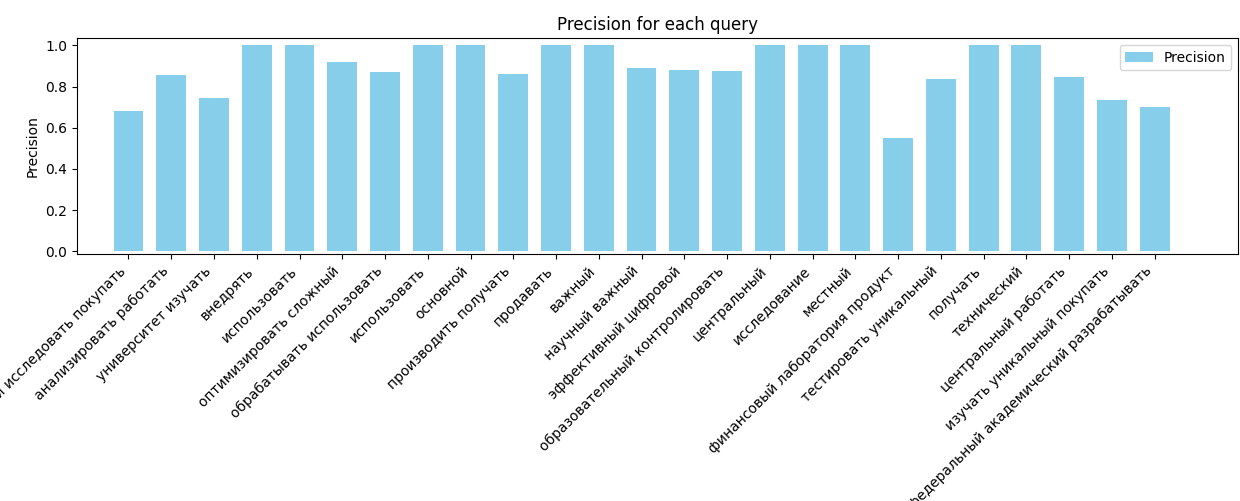
Метрики и результаты приведены в таблице ниже:

| Метрика | Значение |
| --- | --- |
| mean\_precision | 0.890 |
| mean\_recall | 0.343 |
| mean\_accuracy | 0.318 |

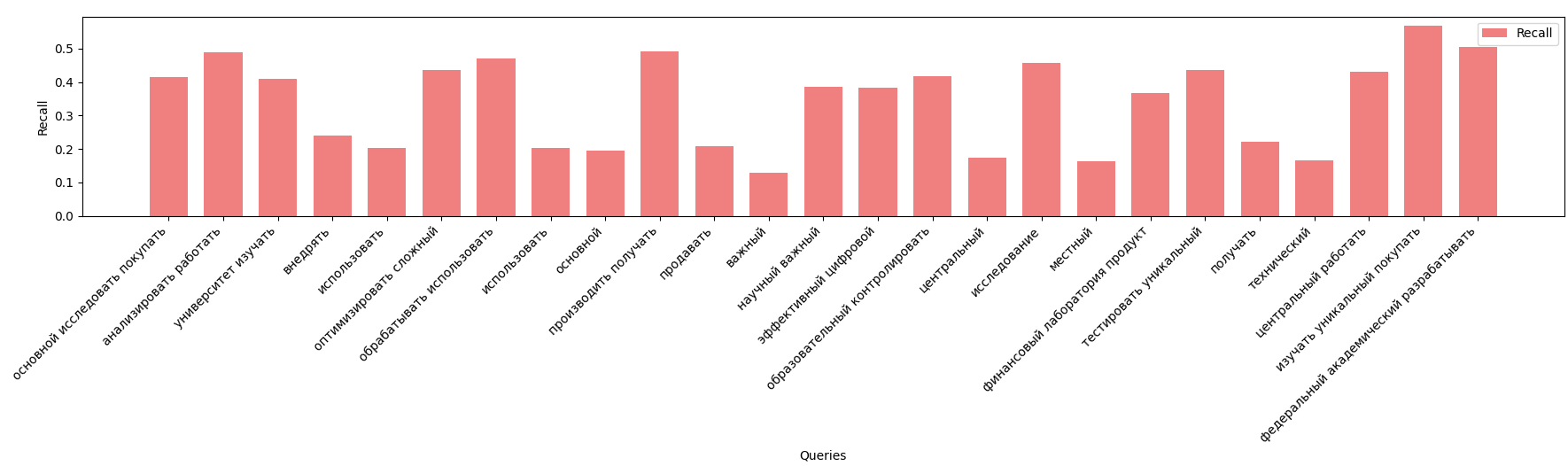
**Результаты анализа полученных данных, предложения по улучшению системы**

Тестирование информационно-поисковой системы проводилось по трем метрикам.

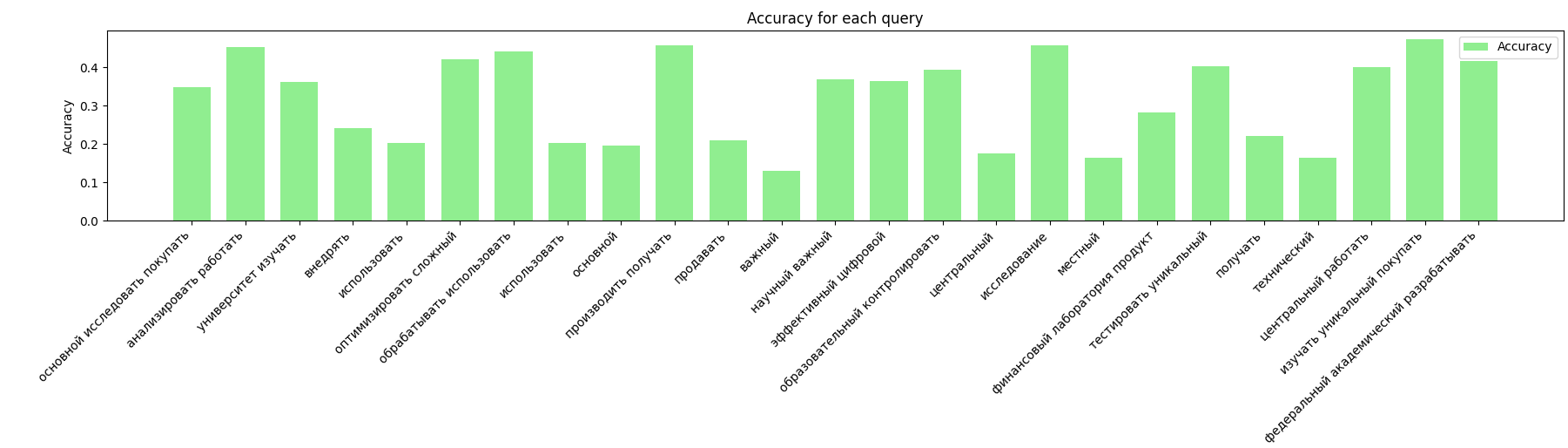
Исходя из показателей метрик, можно сделать вывод, что система с большой долей вероятности верно классифицирует объекты по заданному классу - за это отвечает метрика mean\_precision, показатель которой равен 0.890.

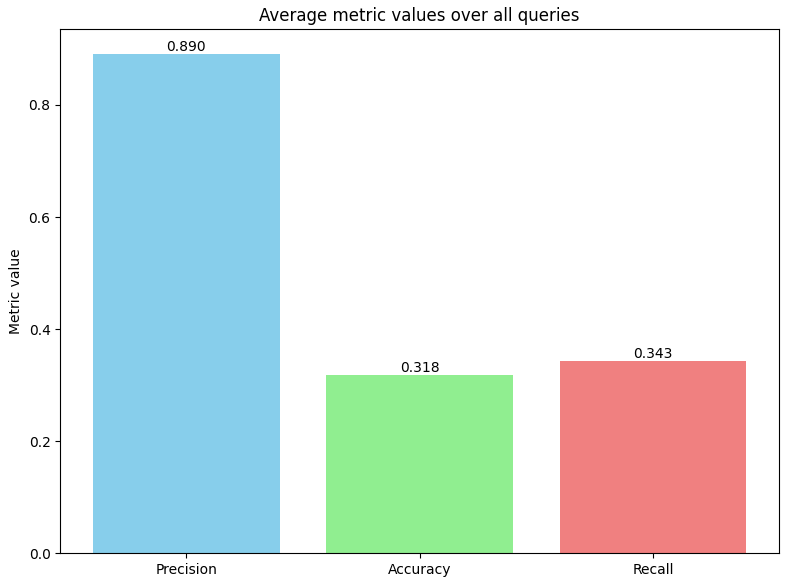


Метрика recall означает полноту - какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Здесь видно, что у метрики достаточно низкий показатель ввиду логической стратегии поиска, не допускающей синонимии и серьезных отклонений документов от запроса.



Метрика accuracy отвечает за долю правильных ответов алгоритма. Показатель метрики равен 0.32. Стоит отметить, что данная метрика не применима к системам с сильным дисбалансом классов.



****

Повысить метрики возможно с помощью внедрения более совершенных моделей поиска для обработки естественного языка и голосовых запросов, а также расширение базы данных для увеличения покрытия и релевантности результатов.

Кроме того, целесообразно проведение дополнительной оценки с использованием разнообразных наборов данных и сценариев использования, а также применение методов кросс-валидации для более надежной оценки качества. Оптимизация скорости обработки и масштабируемости сервиса обеспечит комфортное взаимодействие пользователей с системой при росте нагрузки.

**Описание использованных готовых компонент в системе**

## **1. Avalonia UI**

### Общее описание

Avalonia UI - это кроссплатформенный фреймворк для построения пользовательских интерфейсов на .NET, вдохновленный WPF (Windows Presentation Foundation). Позволяет создавать нативные приложения для Windows, Linux, macOS, Android и iOS из единой кодовой базы.

В рамках данной лабораторной фреймворк выполнял роль клиентской части информационной системы.

### Ключевые характеристики

* Единый код для всех поддерживаемых платформ;
* Использует XAML для описания UI, аналогично WPF/UWP;
* Полная поддержка Model-View-ViewModel архитектуры;
* Мощная система привязки данных;
* Гибкая система стилизации элементов;
* Возможность создания пользовательских контролов;

## **2. SQLite + Entity Framework Core**

SQLite - встраиваемая реляционная база данных, не требующая отдельного серверного процесса. В данном приложении выполняла роль базы данных для хранения индексов документов.

Характеристики:

* хранение базы данных в одном файле;
* простая конфигурация;
* кроссплатформенность;
* высокая производительность для локального хранения;

EF Core - это ORM (Object-Relational Mapper) для .NET, который упрощает работу с базами данных.

Ключевые возможности:

* Code First подход: Сначала создаются классы моделей, затем генерируется БД;
* Migrations: Система управления версиями схемы базы данных;
* LINQ поддержка: Запросы к БД с использованием Language Integrated Query;
* Tracking изменений: Автоматическое отслеживание изменений объектов;
* Настройка связей между сущностями через код;

## **3. ASP.NET Core**

### Общее описание

ASP.NET Core - кроссплатформенный фреймворк для построения веб-приложений и API на .NET. Представляет из себя webAPI для работы с индексацией и ответами на запросы.

### Архитектурные компоненты

* Web API: Создание RESTful API сервисов;
* Middleware: Конвейер обработки HTTP-запросов;
* Dependency Injection: Встроенная система внедрения зависимостей;

## **4. FastAPI**

FastAPI - современный, быстрый веб-фреймворк для построения API на Python 3.6+. В данной лабораторной работе применялся в качестве серверной части информационной системы.

### Ключевые особенности

* высокая производительность;
* автодокументация Swagger UI и ReDoc;
* валидация данных на основе аннотаций типов;
* поддержка async/await;

## **5. Модель wav2vec2-large-xlsr-53-russian**

Модель wav2vec2-large-xlsr-53-russian от Jonatas Grosman - предобученная модель для распознавания русской речи, основанная на архитектуре wav2vec 2.0. Модель применялась как компонент обработки речи пользователя при вводе поискового запроса голосом.

### Технические характеристики

* Архитектура: wav2vec 2.0 Large
* Язык: Русский
* Тренировочные данные: Common Voice, Multilingual LibriSpeech
* Размер модели: ~1.1 GB
* Требуемая частота дискретизации: 16 kHz

**Вывод**

Целью данной лабораторной работы было практическое освоение основных принципов реализации информационно-поисковых систем (ИПС) и методов оценки качества их работы. В рамках работы была разработана и реализована информационно-поисковая система с многослойной архитектурой, включающей клиентскую и серверную части.

Клиентская часть выполнена на основе Avalonia UI, что обеспечивает кроссплатформенный и удобный интерфейс для пользователей. Серверная часть разработана с использованием Python и фреймворка FastAPI, что позволило реализовать высокопроизводительный и масштабируемый backend. Для расширения функционала системы была интегрирована модель обработки голосового ввода, что значительно облегчает и ускоряет поиск информации пользователем.

В ходе тестирования системы были проведены замеры производительности и качества работы по основным метрикам информационного поиска: точности (accuracy), полноте (recall) и precision. Высокое значение precision свидетельствует о корректности выданных системой релевантных результатов, однако сравнительно низкие показатели accuracy и recall указывают на необходимость улучшения полноты и общей сбалансированности поиска.

Перспективы развития системы связаны с оптимизацией и доработкой алгоритмов ранжирования и обработки запросов, что позволит повысить полноту поиска без значительной потери точности. Рекомендуется внедрение более совершенных моделей машинного обучения для обработки естественного языка и голосовых запросов, а также расширение базы данных для увеличения покрытия и релевантности результатов.

Кроме того, целесообразно проведение дополнительной оценки с использованием разнообразных наборов данных и сценариев использования, а также применение методов кросс-валидации для более надежной оценки качества. Оптимизация скорости обработки и масштабируемости сервиса обеспечит комфортное взаимодействие пользователей с системой при росте нагрузки.