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 №3

«Автоматическое реферирование документов»

Выполнили студенты Целуйко Д.А.

группы 221702: Гринь Н.А.

Проверил: Крапивин Ю.Б.

Минск 2025

## **Цель работы:**

Изучить и отработать практические навыки применения методов автоматического распознавания языка текстовых документов.

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

1. Изучить и реализовать два основных метода автоматического реферирования:
   1. **Классический реферат** на основе метода извлечения предложений (Sentence Extraction) с вычислением весов по формуле TF\*IDF.
   2. **Реферат в виде ключевых слов** с использованием локальной языковой модели (Ollama).
2. Разработать систему для обработки коллекции текстовых документов (.txt) на Испанском и Английском языках.
3. Провести сравнительный анализ реализованных методов, оценив их сильные и слабые стороны.
4. Разработать десктопный интерфейс для взаимодействия с системой.

## **Язык текста:**

Испанский, Английский

## **Формат Документа:**

TXT

## **Текстовая коллекция документов:**

Для проведения тестирования была сформирована коллекция текстовых документов на английском и испанском языках. Источниками послужили открытые библиотеки и ресурсы, что позволило обеспечить разнообразие тематик в соответствии с вариантом задания:

* Научные статьи по Computer Science: Тексты были взяты с портала [Google Scholar](https://scholar.google.com/).
* Сочинения по литературе: Были использованы произведения классической литературы из [Project Gutenberg](https://www.gutenberg.org/).

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

* **Tkinter**: Стандартная библиотека Python для GUI, выбрана за простоту и отсутствие внешних зависимостей. Использовалась для создания основного окна, меню, кнопок и таблицы ttk.Treeview.
* **watchdog**: Библиотека для мониторинга файловой системы. Позволила реализовать асинхронное отслеживание изменений в фоновом потоке, что делает приложение отзывчивым и динамичным.
* **Ollama** и библиотека ollama — платформа для локального запуска больших языковых моделей (LLM). В проекте использовалась для генерации реферата в виде ключевых слов.
* **spacy:** Мощная библиотека для NLP. В данном проекте использовалась ее ключевая возможность — высокоточное разбиение текста на предложения (sentence tokenization), что является критически важным шагом для экстрактивного метода.
* **hashlib, json:** Стандартные библиотеки Python, использованные для реализации надежной и эффективной системы кэширования путем вычисления хеш-сумм файлов и хранения результатов в формате JSON.

## 

## **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 пользовательский интерфейс на базе Tkinter. Он управляет отображением списка файлов, текстовых полей для рефератов, обрабатывает действия пользователя (выбор файла) и координирует работу других компонентов.
* **summarizer.py (Референт):** Ядро системы. Этот модуль инкапсулирует всю логику создания рефератов. Он содержит реализации обоих методов: классического экстрактивного и генерацию ключевых слов через Ollama. Также в этом модуле реализована **система кэширования** результатов в JSON-файл для значительного ускорения повторной обработки неизмененных документов.
* **watcher.py (Мониторинг ФС):** Фоновый процесс на базе библиотеки watchdog, отслеживающий изменения (создание, изменение, удаление) файлов в рабочей директории. При обнаружении изменений он инициирует обновление списка файлов в основном приложении, обеспечивая актуальность данных без перезапуска программы.

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

1. Корпус документов (Файловая система):
   1. Тестовый корпус (corpus\_root/):
      1. Структура: *corpus\_root/ -> file.txt*
      2. Назначение: Является источником входных текстовых документов для анализа и реферирования. Система рекурсивно сканирует эту директорию и все ее подпапки.
2. Кэш рефератов (summaries\_cache.json):
   1. Структура: {"path": {"hash": "...", "keyword\_summary": "...", "classic\_summary": "..."}, …}, рисунок - 1.
   2. Хранит ранее сгенерированные рефераты для каждого файла. Ключом является полный путь к файлу. Для каждого файла сохраняется его хеш-сумма (SHA256) и результаты работы обоих методов реферирования. При запросе реферата для файла система сначала вычисляет его текущий хеш и сравнивает с сохраненным. Если хеши совпадают, результаты мгновенно берутся из кэша. Если нет (файл был изменен), рефераты генерируются заново, и кэш обновляется.



Рис. 1 – Структура summaries\_cache.json

**Описание алгоритма построения реферата**

### Классический реферат (Метод извлечения предложений).

Алгоритм основан на оценке "важности" каждого предложения в тексте и последующем извлечении наиболее значимых.

Текстовое описание:

1. Этап 1: Вычисление весов слов. Для всего корпуса документов вычисляется Document Frequency (DF) для каждого слова. Затем для конкретного документа вычисляется вес каждого слова по модифицированной формуле TF\*IDF, указанной в задании: *w(t,D) = 0.5 \* (1 + tf(t,D)/tf\_max(D)) \* log(|DB|/df(t))* .
2. Этап 2: Вычисление весов предложений. Текст разбивается на предложения. Вес каждого предложения Si вычисляется как произведение трех функций:
   1. Score(Si): Сумма весов всех значимых слов в предложении.
   2. Posd(Si): Позиция предложения в документе (предложения в начале текста получают больший вес).
   3. Posp(Si): Позиция предложения в абзаце (в данной реализации аппроксимирована позицией в общем списке предложений).
3. Этап 3: Генерация реферата. Выбираются 10 предложений с наибольшим итоговым весом. Для сохранения читаемости они сортируются в том порядке, в котором они изначально находились в тексте, и объединяются в один абзац.

Графическое описание (Блок-схема):

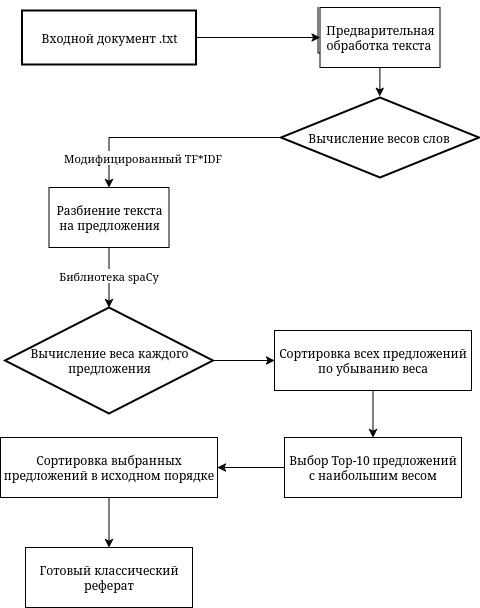


Рис. 2 – Структурная схема метода извлечения предложений

### Реферат в виде ключевых слов (Ollama).

Этот метод использует возможности большой языковой модели (LLM) для семантического анализа текста.

Текстовое описание:

1. Этап 1:Подготовка. Полный текст входного документа передается в качестве контекста локально запущенной модели phi3 через сервер Ollama.
2. Этап 2: Запрос (Промпт)**.** Вместе с текстом модели отправляется четкая инструкция: "Проанализируй текст и извлеки 10-15 самых важных ключевых слов и фраз. Представь их в виде простого, разделенного запятыми списка."имых слов в предложении.
3. Этап 3: Генерация. Модель обрабатывает текст, семантически определяет ключевые темы и сущности, после чего генерирует ответ в заданном формате. Этот ответ и является рефератом.

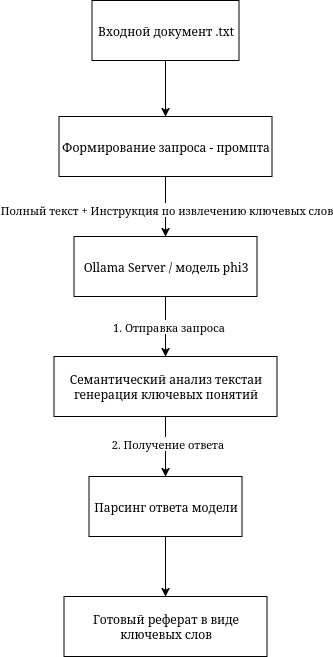


Рис. 3 – Структурная схема AI реферирования

## **Скриншоты разработанной системы:**

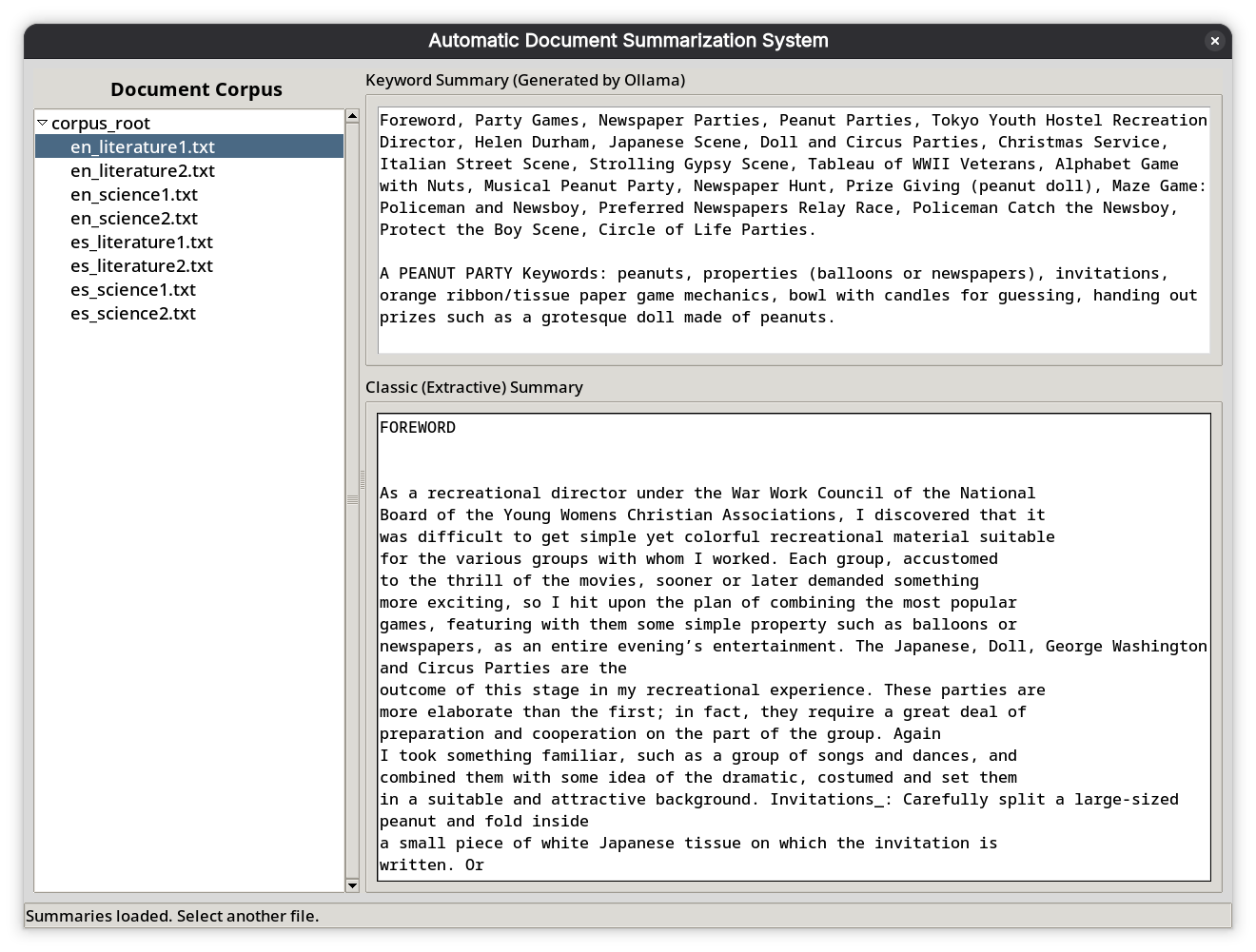


Рис. 4 – Скриншот №1

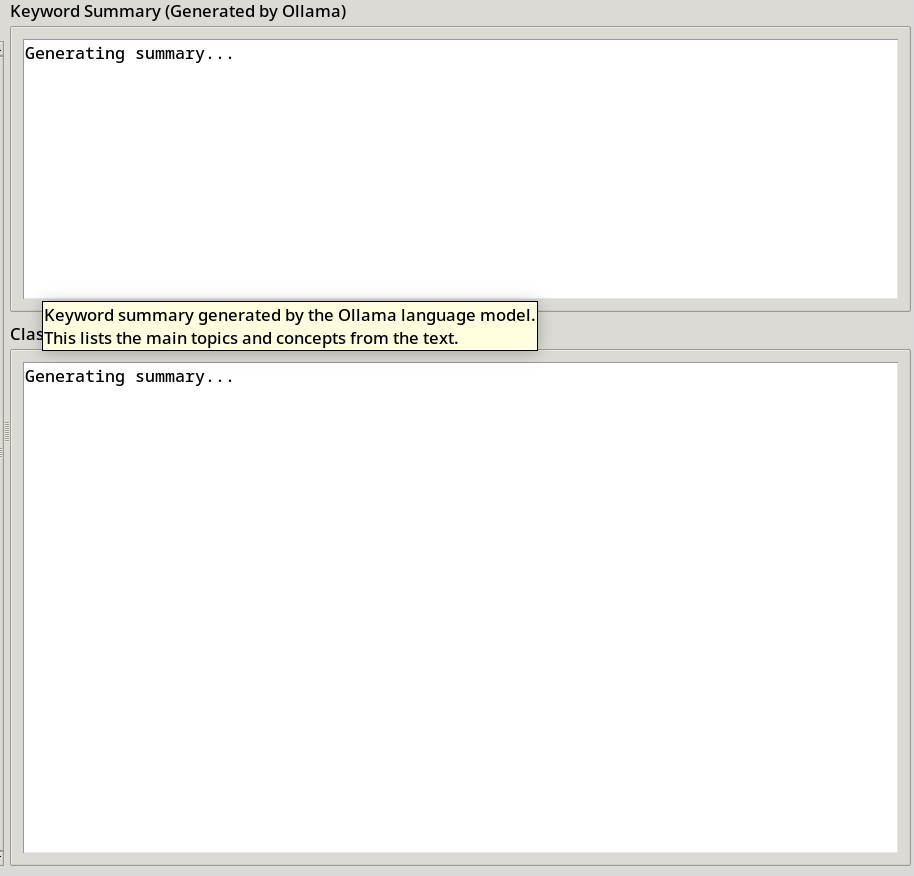


Рис. 5 – Скриншот №2

**Структурно-функциональная схема приложения:**

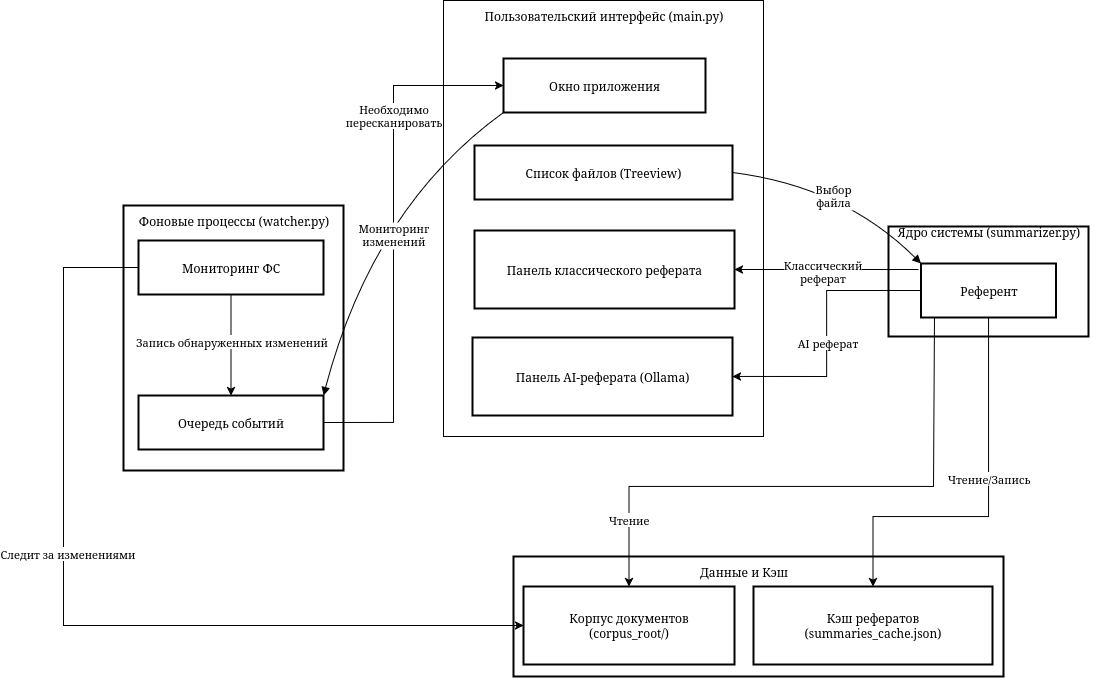


Рис. 6 - Структурно-функциональная схема приложения

## **Оценка полученных результатов (Точность и Время)**

Вместо количественной оценки точности (которая трудно формализуема для задачи реферирования) был проведен качественный анализ результатов работы двух методов на одном и том же фрагменте текста.

Входной текст (фрагмент из "Phunology: A collection of tried and proved plans for play, fellowship and profit"):

*As a recreational director under the War Work Council of the National Board of the Young Womens Christian Associations, I discovered that it was difficult to get simple yet colorful recreational material suitable for the various groups with whom I worked. Each group, accustomed to the thrill of the movies, sooner or later demanded something more exciting, so I hit upon the plan of combining the most popular games, featuring with them some simple property such as balloons or newspapers, as an entire evening’s entertainment. The Japanese, Doll, George Washington and Circus Parties are the outcome of this stage in my recreational experience.*

**Результат 1: Классический реферат (Метод извлечения предложений)**

*As a recreational director under the War Work Council of the National Board of the Young Womens Christian Associations, I discovered that it was difficult to get simple yet colorful recreational material suitable for the various groups with whom I worked. Each group, accustomed to the thrill of the movies, sooner or later demanded something more exciting, so I hit upon the plan of combining the most popular games, featuring with them some simple property such as balloons or newspapers, as an entire evening’s entertainment. The Japanese, Doll, George Washington and Circus Parties are the outcome of this stage in my recreational experience.*

**Результат 2: Реферат в виде ключевых слов (Ollama)**

*movies, popular games, young women'de Christian Associations, War Work Council, National Board, balloons, newspapers, Japanese Party, Doll Party, George Washington Party, Circus Party, evening’s entertainment.*

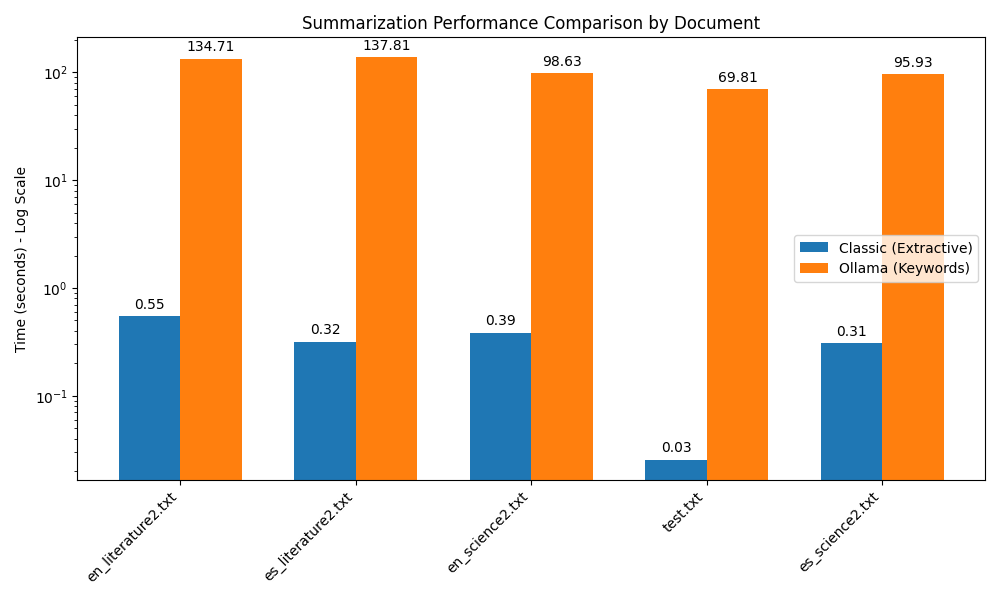


Рис. 7 - График сравнения быстродействия методов

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

В ходе лабораторной работы была успешно разработана система для автоматического определения языка, реализующая и сравнивающая три различных метода.

1. Классический экстрактивный метод показал себя как быстрый и полностью контролируемый алгоритм, результат которого предсказуем, но может страдать от потери связности.
2. Метод на основе LLM (Ollama) продемонстрировал способность к семантическому анализу текста, предоставляя высокоуровневое понимание его содержания в виде ключевых тем. Его медлительность была успешно компенсирована системой кэширования.
3. Система полностью соответствует требованиям задания, включая наличие GUI, фоновый мониторинг и сравнение методов.

## **Перспективы развития приложения:**

1. Создание гибридного реферата: Можно объединить оба подхода, сгенерировав экстрактивный реферат и дополнив его списком ключевых слов для лучшего понимания контекста.
2. Поддержка других языков: Хотя у метода реферирования нет проблем с генерацией реферата на других языках, метода классического реферирования отвечает только на
3. Поддержка других форматов: Систему можно доработать для поддержки .pdf и .docx, интегрировав соответствующие библиотеки для извлечения текста (например, PyPDF2, python-docx).
4. Полноценный абстрактивный реферат: С использованием более мощных локальных моделей (например, Llama 3, Mistral) можно реализовать генерацию не просто ключевых слов, а полноценного, связного и уникального краткого пересказа текста.
5. Настраиваемый размер реферата: Добавить в интерфейс возможность пользователю самому задавать желаемое количество предложений для классического реферата.