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 №2

«Автоматическое распознавание языка текста»

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):**

* Изучить и реализовать основные методы определения языка текстовых документов (N-граммный, алфавитный, нейросетевой).
* Разработать систему для обработки коллекции HTML-документов на Испанском и Английском языках.
* Провести сравнительный анализ реализованных методов по точности и быстродействию.
* Разработать десктопный интерфейс для взаимодействия с системой.

## **Язык текста:**

Испанский, Английский

## **Формат Документа:**

HTML

## **Текстовая коллекция документов:**

Документы на английском и испанских языках были взяты в сайтов <https://lingua.com/> и <https://www.fabulang.com/en> - сайты для изучения языков. Что позволило сформировать качественную основу базу для выполнения лабораторной работы.

## **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.
* **BeautifulSoup4** — библиотека для парсинга HTML и XML документов. В проекте была задействована для извлечения чистого текстового содержимого из HTML-файлов, что является ключевым шагом предобработки перед анализом.
* **langdetect** — библиотека для определения языка, основанная на портированной из Google-кода нейросетевой модели. Использовалась для реализации "нейросетевого" метода, предоставляя быстрый и точный способ определения языка без необходимости самостоятельного обучения сложных моделей.
* **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 **определения языка текста.**

## **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 (нажатие кнопок, открытие файлов) и координацию работы других компонентов.
* **language\_profiler.py (Профилировщик):** Одноразовый скрипт, который анализирует тренировочную коллекцию текстов и создает "профили языков" (language\_profiles.json) для N-граммного метода.
* l**anguage\_detector.py** (Детектор языка): Ядро системы, реализующее три независимых алгоритма определения языка (N-грамм, алфавитный, нейросетевой).
* **watcher.py (Мониторинг ФС):** Фоновый процесс, отслеживающий изменения в файлах.

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

1. Корпус документов (Файловая система):
   1. Тренировочный корпус (training\_corpus/):
      1. Структура:
         1. *training\_corpus/ -> en/ -> file.html*
         2. *training\_corpus/ -> es/ -> file.html*
      2. Назначение: Является источником данных для "обучения" N-граммного метода и построения профилей языков.
   2. Тестовый корпус (corpus\_root/):
      1. Структура: *corpus\_root/ -> file.html*
      2. Назначение: Является источником входных данных для определения языка.
2. Профили языков (language\_profiles.json):
   1. Структура: JSON-файл вида {"код\_языка": ["N-грамма\_1", "N-грамма\_2", ...], …}, рисунок - 1.
   2. Назначение: Хранит "отпечатки" языков — отсортированные списки из 300 самых частотных N-грамм. Это основная структура данных для N-граммного метода, позволяющая избежать повторного анализа тренировочной коллекции при каждом запуске.

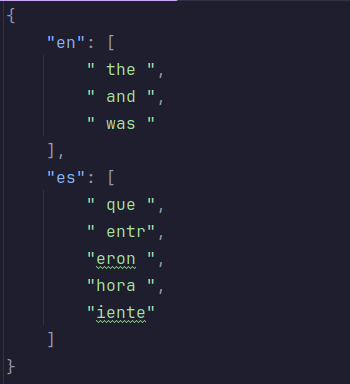


Рис. 1 – Структура language\_profiles.json

1. Кэш для LLM (llm\_lang\_cache.json файл):
   1. Структура: JSON-файл вида {"path": {"hash": “...”, "lang\_code": “ ”}, …}, рисунок - 2.
   2. Назначение: Хранит "кеш" для модели, позволяет избежать повторного анализа тренировочной коллекции при каждом запуске.



Рис. 2 – Структура llm\_lang\_cache.json

1. Выходные данные (.csv файл):
   1. Структура: Текстовый файл с разделителями-запятыми, содержащий колонки: File Path, N-Gram Method, Alphabet Method, Neural Net Method, Ollama

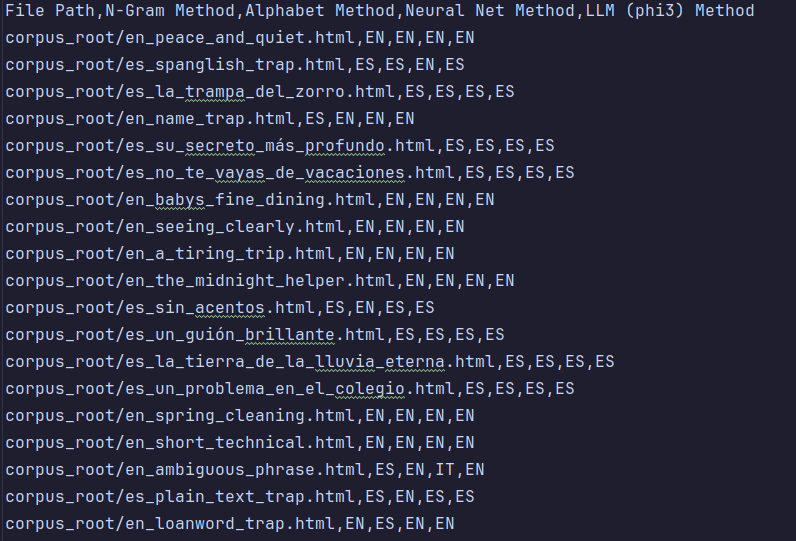


Рис - 3. Структура выходного файла(файл сохранения)

* 1. Назначение: Позволяет сохранить результаты анализа из таблицы приложения для дальнейшего использования или составления отчета.

## **Описание алгоритма определения языка**

В системе реализовано четыре метода для сравнения их эффективности.

1. Алфавитный метод - простой и очень быстрый эвристический метод. Он проверяет наличие в тексте символов, уникальных для испанского алфавита (ñ, á, é, í, ó, ú, ü). Если хотя бы один такой символ найден, язык определяется как испанский (es), в противном случае — как английский (en).
2. Нейросетевой метод использует готовую библиотеку langdetect. Метод передает очищенный текст в предварительно обученную нейросетевую модель, которая возвращает код наиболее вероятного языка. Этот метод служит эталоном для сравнения точности.
3. N-граммный метод (основной) - Алгоритм основан на сравнении частотных профилей N-грамм (в данной работе N=5) и вычислении метрики расстояния "Out-of-Place".

Текстовое описание N-граммного метода:

1. Этап обучения: Для каждого языка из тренировочной коллекции создается профиль языка (ПОЯ) — отсортированный по частоте список из 300 самых популярных N-грамм.
2. Этап детекции:
   1. Для входного HTML-файла извлекается и очищается текст.
   2. Для этого текста строится профиль документа (ПОД) — аналогичный список из 300 самых популярных N-грамм.
   3. Вычисляется метрика расстояния между ПОД и каждым ПОЯ. Расстояние — это сумма модулей разниц позиций одинаковых N-грамм в двух списках. Если N-грамма из ПОД отсутствует в ПОЯ, к расстоянию добавляется максимальный штраф (300).
   4. Языком документа считается тот, до профиля которого расстояние оказалось минимальным.

#### Алгоритм работы (langdetect):

Предобработка и генерация N-грамм:

* + Библиотека берет входной текст и минимально его очищает (например, удаляет URL-адреса).
  + Затем она разбивает текст не на слова, а на символьные N-граммы разной длины (обычно от 1 до 3 символов). Например, для слова "text" N-граммами будут: t, e, x, t, te, ex, xt, tex, ext.

1. Сравнение с профилями языков:
   * У langdetect "внутри" зашиты профили для множества языков. Каждый профиль — это, по сути, таблица частот, показывающая, как часто та или иная символьная N-грамма встречается в данном языке. Например, в английском профиле N-грамма th будет иметь очень высокую частоту, а в испанском — N-грамма es.
   * Алгоритм "пробегается" по всем N-граммам из вашего входного текста.
2. Выбор языка (Наивный Байес):
   * Для каждого языка из своей базы библиотека вычисляет вероятность того, что данный текст принадлежит этому языку.
   * Она делает это, перемножая вероятности встретить каждую N-грамму из текста в профиле данного языка. "Наивность" классификатора заключается в предположении, что все N-граммы независимы друг от друга.
   * В итоге, языком текста считается тот, для которого итоговая вероятность оказалась наивысшей.

Процесс обучения langdetect:

Профили языков для langdetect были созданы заранее ее разработчиками. Процесс обучения был очень похож на то, что было сделанов language\_profiler.py, но в гораздо большем масштабе:

* Брался огромный корпус текстов с заранее известным языком (например, вся англоязычная Википедия).
* Для этого корпуса подсчитывались частоты всех символьных N-грамм (от 1 до 3 символов).
* Самые частотные N-граммы и их вероятности сохранялись как профиль для английского языка.
* Эта процедура повторялась для десятков других языков.

4. Текстовое описание:

1. Этап 1:Подготовка. 100 Первых символов 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: "Определи язык текста и выведи либо EN для английского или ES для испансокго."
3. Этап 3: Генерация. Модель обрабатывает текст, определяет язык и возвращает ответ в виде кода языка.

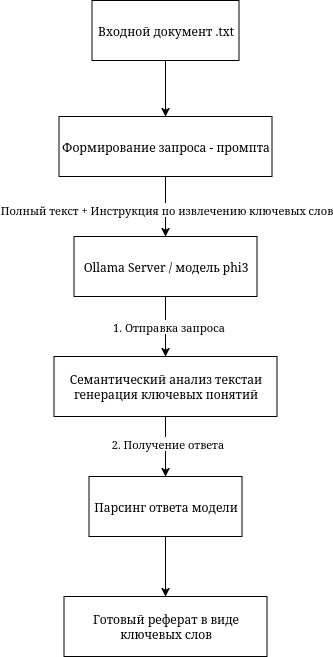


Рис. 4 – Структурная схема AI определения языка

Графическое описание:

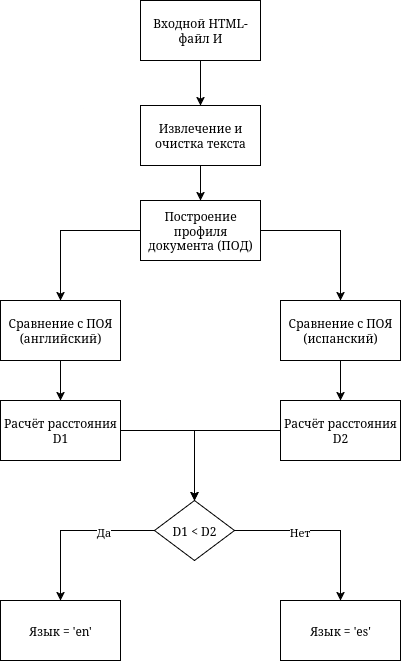


Рис. 5 – Блок-схема N-граммного метода

#### 

#### 

#### 

#### 

#### 

#### ПОЯ (Поисковый Образ Языка)

Это "отпечаток", эталон, или профиль, который однозначно характеризует конкретный язык в рамках нашей системы. ПОЯ — это содержимое файла language\_profiles.json.

* Список из 300 N-грамм для ключа "en" — это ПОЯ для английского языка.
* Список из 300 N-грамм для ключа "es" — это ПОЯ для испанского языка.
* Они создаются один раз скриптом language\_profiler.py и затем только читаются.
* На блок-схеме (рисунок - 3) это соответствует блокам "Сравнение с ПОЯ (английский)" и "Сравнение с ПОЯ (испанский)", которые используют эти заранее созданные данные.

#### ПОД (Поисковый Образ Документа)

Это "отпечаток" конкретного входного документа, который мы анализируем. Важно, что он строится по тем же правилам, что и ПОЯ. ПОД — это временный профиль, который создается в памяти для каждого анализируемого файла внутри метода detect\_by\_ngram в файле language\_detector.py.

* Когда вызывается этот метод, он читает HTML-файл, извлекает и очищает текст, строит для него список из 300 самых частых 5-грамм.
* Этот временный, созданный "на лету" список и есть ПОД.
* На вашей блок-схеме это в точности соответствует блоку "Построение профиля документа (ПОД)".

После создания ПОД система сравнивает его с каждым доступным ПОЯ (английским и испанским), вычисляет расстояние и выбирает язык с минимальным расстоянием. Ваша реализация полностью соответствует этой теоретической модели.

## **Скриншоты разработанной системы:**

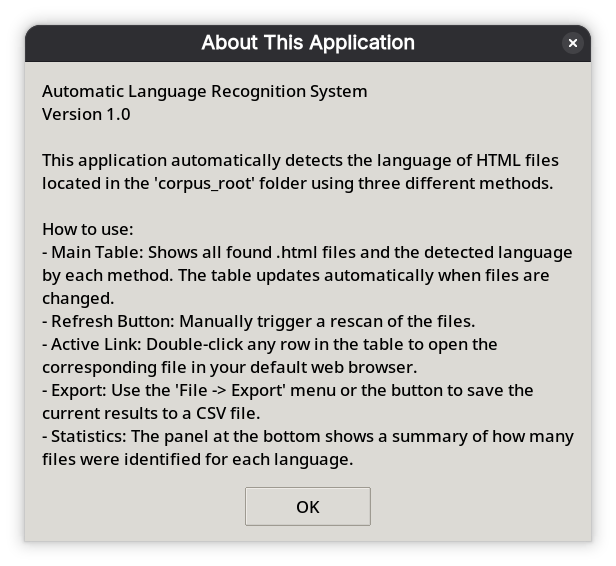


Рис. 6 – Скриншот №1

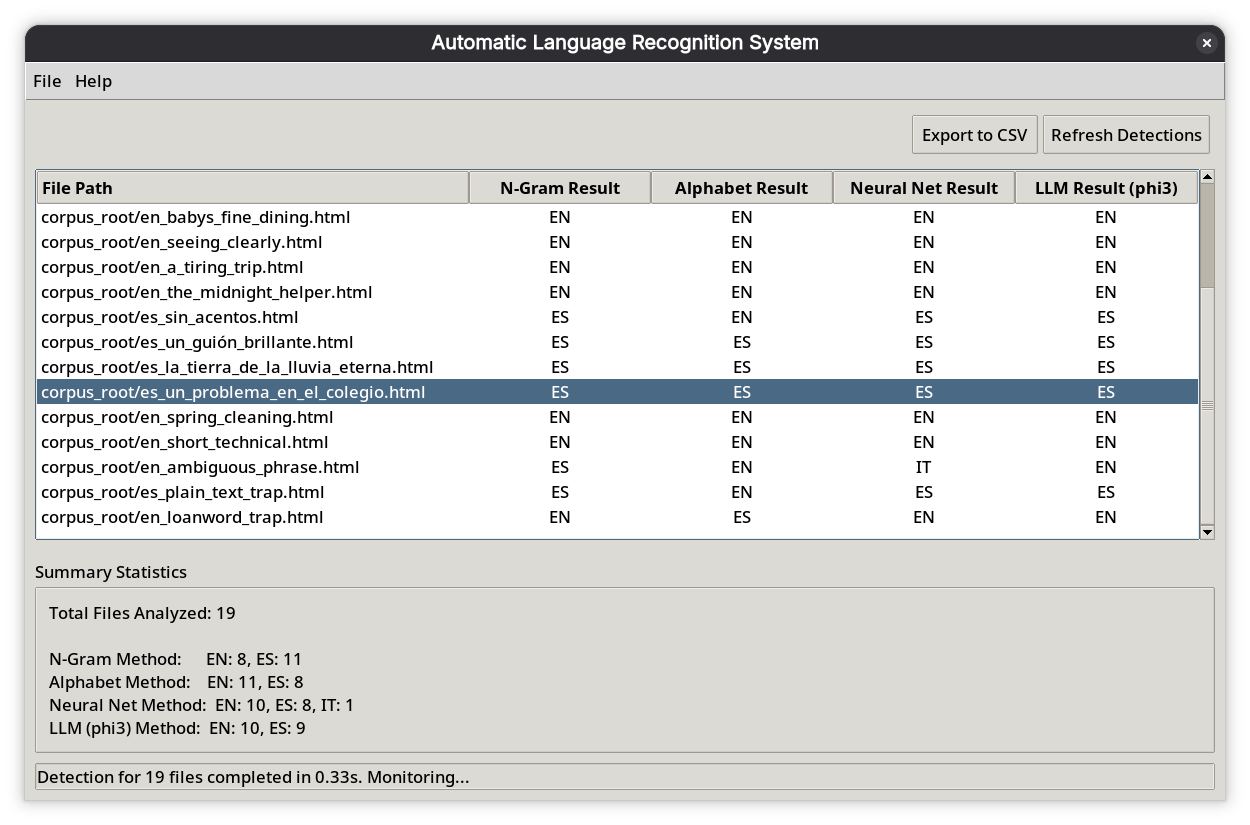


Рис. 7 – Скриншот №2

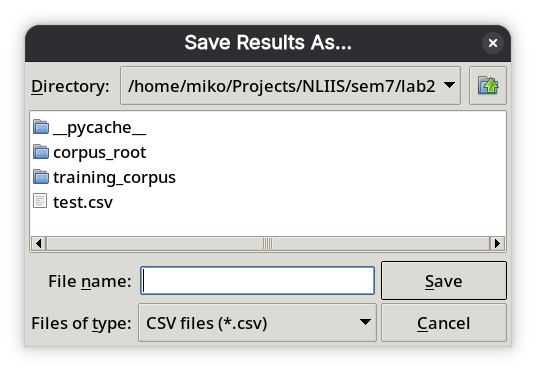


Рис. 8 – Скриншот №2

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

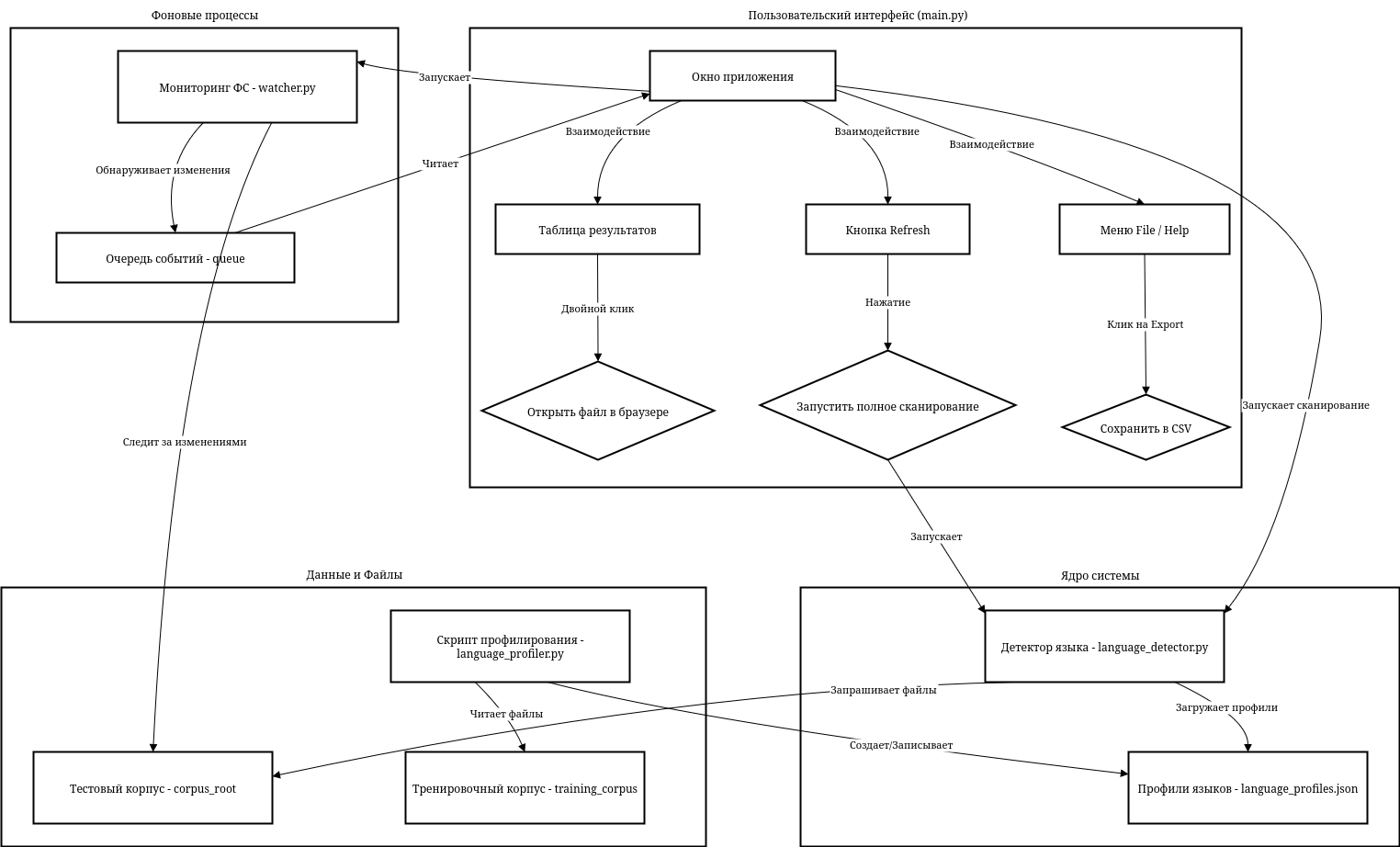


Рис. 9 - структурно-функциональная схема приложения

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

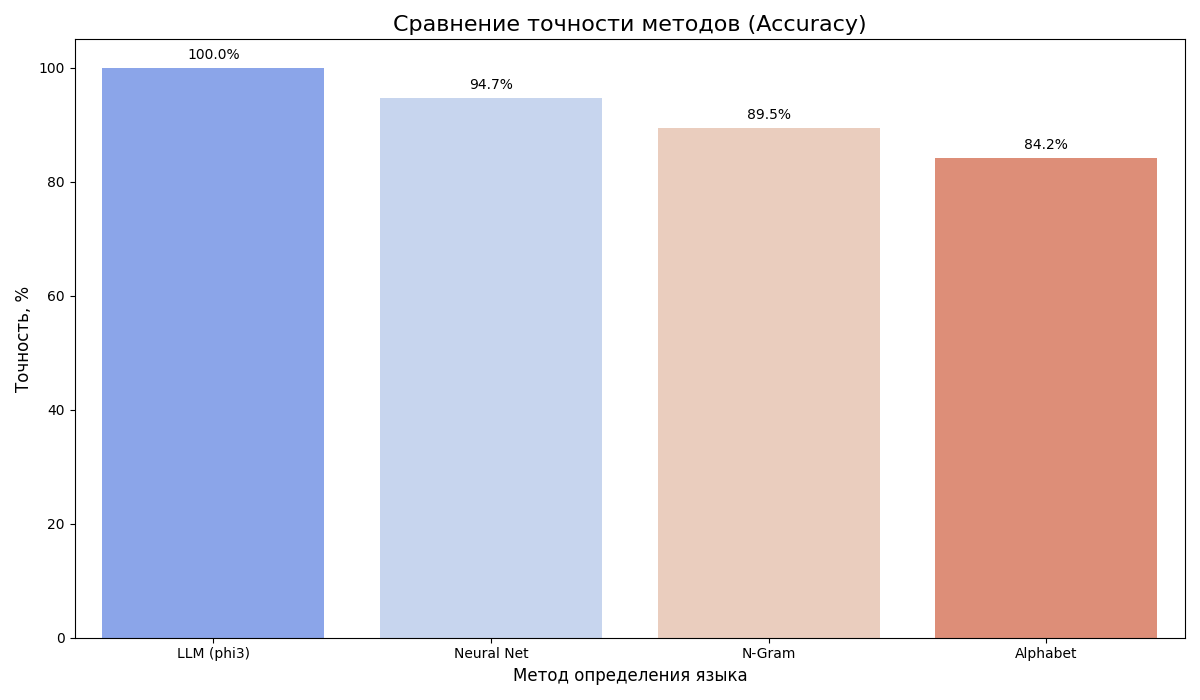


Рис. 10 - График сравнения точности методов

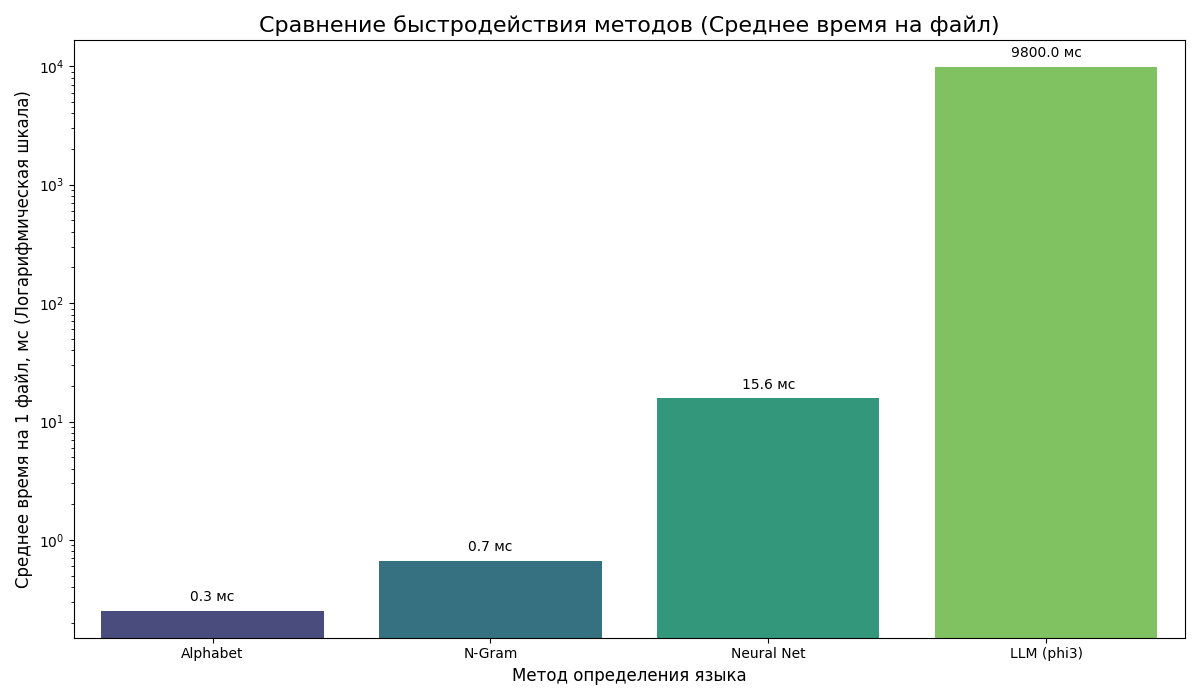


Рис. 11 - График сравнения быстродействия методов

**Описание и особенности применения готовых компонент:**

* **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.
* **BeautifulSoup4**: Ключевая библиотека для веб-скрапинга, примененная для надежного извлечения текста из HTML-тегов. Это позволило работать с "грязными" данными, как в реальных условиях.
* **langdetect**: Готовая библиотека для определения языка, выбранная для реализации "нейросетевого" метода. Ее особенность — высокая точность и скорость "из коробки", так как она использует сложную, предварительно обученную Google модель. Это позволило получить эталонный результат для сравнения.
* **Ollama** и библиотека ollama — платформа для локального запуска больших языковых моделей (LLM). В проекте использовалась для реализации AI-компонента **определения языка текста.**

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

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

1. N-граммный метод показал себя как надежный, полностью контролируемый алгоритм, точность которого напрямую зависит от качества и объема тренировочной коллекции.
2. N-грамный (langdetect) продемонстрировал наивысшую точность и скорость, подтвердив эффективность современных NLP-моделей.
3. Алфавитный метод является простейшим и самым быстрым, но его применимость ограничена парами языков с уникальными алфавитами.
4. Нейросетевой метод (Ollama/phi3) является самым ресурсо и время затранным для определения языка и для данной задачи этот метод является перебором.

Система полностью соответствует требованиям задания, включая наличие GUI, фоновый мониторинг, экспорт результатов и справочную информацию.

## 

## 

## **Перспективы развития приложения:**

1. Расширение языковой поддержки: Архитектура N-граммного метода позволяет легко добавлять новые языки путем создания для них профилей в training\_corpus.
2. Поддержка других форматов: Систему можно доработать для поддержки .pdf и .docx, интегрировав соответствующие библиотеки для извлечения текста (например, PyPDF2, python-docx).
3. Создание гибридного метода: Для оптимизации можно разработать гибридный алгоритм, который сначала применяет быстрый алфавитный метод, а в случае неоднозначности (например, если уникальные символы не найдены) передает текст более точному, но ресурсоемкому нейросетевому методу.
4. Интеграция: Разработанный детектор может быть использован как независимый модуль в более крупных системах, например, для автоматической сортировки входящих документов или для предварительной языковой фильтрации в поисковых системах.