Содержание

Введение.......................................................................................................3

Постановка задачи.......................................................................................5

Описание используемого стека технологий..............................................7

Описание программы..................................................................................15

Заключение..................................................................................................24

Список использованной литературы.........................................................26

Исходный код консольного приложения..................................................27

ВВЕДЕНИЕ

Вы когда-нибудь читали текст, и при определенной рассеяности внимания вместо вчитывания в слова вы начинали считать слова? Или, быть может, смотрели на то, как часто встречалась буква "Ы" в тексте? Если такое случалось, то не стоит расстраиваться, возможно, в вашей крови течет жила настоящего филолога и криптоаналитика. Наверняка вы слышали, что такое частотный анализ. Если же нет, то давайте разберемся, что это такое.

Частотный анализ, частотный криптоанализ — один из методов криптоанализа, основывающийся на предположении о существовании нетривиального статистического распределения отдельных символов и их последовательностей, как в открытом тексте, так и в шифротексте, которое, с точностью до замены символов, будет сохраняться в процессе шифрования и дешифрования.

Упрощённо, частотный анализ предполагает, что частотность появления заданной буквы алфавита в достаточно длинных текстах одна и та же для разных текстов одного языка. При этом, в случае моноалфавитного шифрования, если в шифротексте будет символ с аналогичной вероятностью появления, то можно предположить, что он и является указанной зашифрованной буквой. Аналогичные рассуждения применяются к биграммам (двубуквенным последовательностям), триграммам и т. д. в случае полиалфавитных шифров.

Ещё проще: используется тот факт, что вероятность появления отдельных букв, а также их порядок в словах и фразах естественного языка подчиняются статистическим закономерностям: например, пара стоящих рядом букв «ся» в русском языке более вероятна, чем «цы», а «оь» в русском языке не встречается вовсе (зато часто встречается, например, в чеченском). Таким образом, подсчитав количество букв, пар букв, можно после этого будет восстановить исходный текст. [1]

1 ПОСТАНОВКА ЗАДАЧИ

Цель учебной практики – разработать программу на Python с графическим интерфейсом для проведения анализа текста.  
 Данная программа позволит пользователям проводить анализ текста как из текстового файла, так и из буфера обмена. Правда, потребуется вставить текст в окно, но это другое дело. Для реализации данной программы необходимо выполнить определенные действия:

1. Выбрать метод подсчета слов. Скажем, что у нас есть два метода: "Костыльный" и "Стандартный". Под "костылями" имеются ввиду подсчеты, которые используют достаточно редко из-за их неточностей. Тоесть, подсчет количества пробелов, который теряет слова при много-абзацном тексте, либо метод, где считается "середина" количества слов в первых 5 предложениях, а потом просто умножается на количество предложений. Мы же будем использовать "стандарт", в котором воспользуемся функциями Python.

2. Частотный анализ. Здесь же все проще. Частотный анализ будет работать так, что мы подсчитываем, как часто встречалась одна буковка.

3. Анализ файлов. Помимо текстового поля, можно открыть файл, и программа его прочитает за нас.

4. Реализация интерфейса. Создать простой интерфейс для программы, отображающего две кнопки для двух разных анализов, информацию о проведенном анализе текста на частотность и слова.

Вообще, говоря про частотный анализ необходимо уточнить, что его часто используют для того, чтобы создать определенный словарь частоты встречи слов, и сам анализ проходит дольше и больше. Частотный анализ предполагает намного большее количество этапов, чем будет использовано в программе, а если указывать точно какие, то:  
 1. Загрузка и обзор данных

2. Очистка и предварительная обработка текста

3. Удаление стоп-слов (удаляются местоимения и подобные слова)

4. Подсчет статистики

5. Визуализация частоты встречи слов. [2]

Еще нужно сделать оговорку по поводу того, какие слова за сколько слов считаются. Часто встречаются числительные, написанные цифрами. Скажем, 2033. Если вспоминать устное ОГЭ по Русскому языку, то такое “слово” засчитывалось за 2 (Две тысячи, тридцать три). У нас же в программе они будут считаться как одно слово. Еще нужно упомянуть про спец. символы. Ровно как вопросительный знак, или тире, при подсчете частотного анализа букв, они, вроде как, указываться не должны.

2 ОПИСАНИЕ ИСПОЛЬЗУЕМОГО СТЕКА ТЕХНОЛОГИЙ И БИБЛЕОТЕК

1 Python

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++. [3]

Преимущества языка программирования Python:

– Простота и воспринимаемость.

Язык Python специально создан похожим по своему синтаксису на естественные языки (прежде всего английский). Блоки кода в нем отделяются друг от друга пробельными отступами. Это делает код, написанный на Python, более удобочитаемым и понятным для программистов. Минималистичный синтаксис также позволяет реже обращаться к документации и писать меньше строк кода, что, в свою очередь, ускоряет работу над проектом. [4]

– Обширная библиотека.

Программисты и сообщество пользователей Python создали для этого языка большую библиотеку, в которой содержатся оптимизированные и многократно используемые фрагменты кода для решения практически любых задач. Благодаря этому программисту не нужно писать код полностью с нуля, что также ускоряет работу над проектом или анализом данных. Кроме того, сообщество пользователей Python создало множество сторонних библиотек с дополнительными материалами, а также обширную сеть форумов, на которых делятся опытом и решениями.

– Совместимость.

Интерпретатор Python может интегрироваться с другими языками программирования — например, с Java, C и C++. Благодаря этому Python-разработчики могут расширять функциональность своего проекта, подключая к нему их возможности. Верно и обратное: программы, написанные на указанных языках программирования, можно сочетать с модулями, созданными с помощью «Питона».

– Мультиплатформенность.

Программы, написанные на Python, адаптированы ко всем существующим аппаратным и программным платформам. Язык используется для работы на разных устройствах, от мейнфреймов до смартфонов и карманных компьютеров под управлением операционных систем Microsoft Windows, MacOS, iOS, Android, UNIX, Linux, Symbian, PalmOS и т.д. Это создает широчайшие возможности для разработки программных продуктов различного назначения, адаптированных под разные платформы.

– Мультипарадигменность.

На Python можно программировать в различных парадигмах: объектно-ориентированной, функциональной, императивной, процедурной, структурной и т.д. Это достигается за счет подключения дополнительных библиотек, интерпретаторов и фреймворков. Мультипарадигменность позволяет расширить выразительные способности языка программирования, решать с его помощью больший спектр задач в рамках одного проекта.

– Динамическая типизация.

В отличие от C-подобных языков программирования, в Python переменные связываются с типом в момент присваивания в них конкретных значений. [4]  
 Для чего нужен Python

– Data Science и машинное обучение.

Эти два направления IT тесно связаны друг с другом. Наука о данных заключается в обработке больших массивов информации из базы данных, а машинное обучение — в разработке компьютерных алгоритмов, способных учиться на ней и делать точные прогнозы. В Data Science используют Python для включения очистки и разметки данных, поиска и обработки статистической информации, ее визуализацию в виде диаграмм, графиков и т.д. С помощью библиотеки Python ML классифицируются изображения, тексты, поисковый трафик, осуществляется распознавание лиц и речи, глубинное машинное обучение.

– Веб-разработка.

Многие крупные интернет-компании, такие как Google, Facebook\*, программируют на Python свои самые известные проекты, например, Instagram\*, YouTube, Dropbox и т.д. Этот язык позволяет вести веб-разработку на стороне сервера, потому что его обширная библиотека включает множество решений как раз для реализации сложных серверных функций. За счет своей простоты использования Python широко применяется небольшими командами и одиночными разработчиками для создания сайтов, десктопных и мобильных веб-приложений.

– Разработка и тестирование ПО.

Возможности Python используются тестировщиками и разработчиками для поиска и исправления ошибок, автоматической сборки, разработки прототипов программного обеспечения, управления проектами и т.д. Кроме того, с помощью сред модульного тестирования Python осуществляется проверка функций. Также на этом языке создаются тестовые скрипты, имитирующие различные сценарии использования ПО. Разработчики аппаратных платформ (например, IBM, Hewlett-Packard, Intel) тоже используют Python для тестирования своей продукции. [4]

Python – это мощный и, в то же время, легкий в использовании язык программирования, который позволяет создавать разнообразные проекты и находить решения для большого спектра задач.

Его отличает простота и универсальность, что позволяет применять Python в различных областях, в том числе и начинающими программистами. Большое сообщество разработчиков обеспечивает помощь и поддержку при работе с языком.

Возможность запускать Python на различных операционных системах позволяет создавать приложения и скрипты, которые могут быть запущены в разных окружениях без необходимости переписывать код. Дополнительные инструменты и библиотеки позволяют использовать Python для анализа данных, машинного обучения, веб-разработки и во многих других областях. [4]

2 Tkinter

Tkinter — это кроссплатформенный графический интерфейс Python, позволяющий работать с библиотекой Tk. Он содержит элементы графического интерфейса пользователя (GUI — Graphical User Interface), с помощью которых можно создавать различные приложения. Изначально Tkinter был разработан в 1988 году профессором математических наук в университете Беркли Джоном Остерхаутом для собственного языка программирования Tcl. Однако впоследствии он был адаптирован для многих других ЯП, таких как Ruby, Perl и т.д. С 1994 года этот программный пакет входит в стандартный дистрибутив языка программирования Python.

В большинстве пользовательских приложений, которыми мы пользуемся на мобильных и десктопных устройствах, используется графический интерфейс пользователя. Это система интерактивных элементов, обеспечивающих взаимодействие человека с программой. Чтобы программа выполнила нужное вам действие, не нужно вводить соответствующую команду в строку — достаточно кликнуть или коснуться пальцем виртуальной кнопки, поставить флажок или галочку, перетащить ползунок и т.д. Этим программы с графическим интерфейсом отличаются от консольных приложений, в которых любую команду нужно вбивать текстом в командную строку. Преимущества GUI очевидны:

– управлять приложением с помощью интерактивных элементов проще, чем вводить каждую команду текстом;

– графический интерфейс интуитивно понятен — даже пользователь, впервые открывший приложение или не знающий языка, сможет разобраться хотя бы в основном функционале.

Очевидно, что любой визуальный интерактивный элемент — не просто картинка. Изображение должно быть сопряжено с программной частью приложения, чтобы обеспечивать взаимодействие пользователя с функциями программы. Только в этом случае совокупность визуальных интерактивных элементов образует пользовательский интерфейс. Для этой цели и предназначены различные библиотеки и GUI-фреймворки, одним из которых является Tkinter.

По сути, программный пакет представляет собой интерпретатор, переводящий код приложений с Python на Tcl — высокоуровневый скриптовый язык, который используется в быстром прототипировании, создании графических интерфейсов, веб-разработке. Программист пишет приложение на Python, а встроенный модуль Tkinter автоматически переводит его на Tcl.

В Tkinter реализовано событийно-ориентированное и объектно-ориентированное программирование.

– Та или иная часть программного кода начинает работать тогда, когда случилось определенное событие — в данном случае это либо действия пользователя в приложении, такие как клики или нажатия на кнопки, внесение данных в онлайн-форму, прокручивание и т.д., либо события, заложенные в самой программе, например завершение загрузки, срабатывание таймера.

– Интерфейс приложения представляет собой взаимодействующие друг с другом и пользователем элементы (виджеты), каждый из которых является объектом-экземпляром, создаваемым от встроенных классов.

Tkinter позволяет создать полноценный работающий интерфейс пользователя мобильного или десктопного приложения с помощью уже встроенных в программный пакет классов и наследующих им объектов. Это довольно универсальный инструмент, у которого есть свои преимущества и недостатки, определяющие сферу его применения.

Достоинства Tkinker:

– Функциональность.

Tkinter предоставляет достаточно широкий встроенный набор интерактивных элементов и их атрибутов, который позволяет создавать полноценный работающий интерфейс пользователя. Кроме того, он позволяет использовать внешние подключаемые модули, которые расширяют возможности пользователя за счет дополнительных элементов и их параметров.

– Кроссплатформенность.

Приложения, созданные с помощью Tkinter, могут работать на любой распространенной платформе (Windows, [Linux](https://blog.skillfactory.ru/glossary/linux/), MacOS). Это обусловлено тем, что Tkinter написан на высокоуровневом языке Tcl, который слабо привязан к техническим аспектам конкретной операционной системы или процессора. Кроссплатформенность Tkinter потенциально позволяет разработчику охватить максимальную аудиторию для продвижения своего продукта.

– Простота.

Библиотека Tkinter проста и удобна в установке, настройке и использовании, чему способствует и ограниченный набор возможностей. Это отличный инструмент для новичков, желающих начать первые шаги в создании пользовательских интерфейсов. С помощью Tkinter они могут понять базовые принципы функционирования GUI, научиться создавать простейшие приложения, чтобы затем перейти к использованию более мощных профессиональных инструментов.

– Широкая поддержка.

Вокруг Tkinter сформировалось достаточно большое сообщество профессиональных разработчиков и энтузиастов-любителей. Благодаря им библиотека имеет большой объем документации, различных инструкций, примеров созданных приложений, разборов ошибок и проблем — все это помогает пользователям быстро освоить данный инструмент.

– Включение в стандартный дистрибутив Python.

С 1994 года Tkinter по умолчанию устанавливается вместе с инсталляционным пакетом Python. Это позволяет не искать и скачивать библиотеку отдельно, а быстро импортировать ее и начать работать.

– Свободное использование.

Tkinter распространяется под Python-лицензией, то есть является свободным программным обеспечением. Это значит, что с ее помощью можно создавать, распространять, копировать и изменять как бесплатные (любительские), так и коммерческие приложения. Это существенно расширяет ее применение и одновременно снижает порог вхождения со стороны новичков.

Чаще всего Tkinter используется как инструмент обучения разработке графических интерфейсов новичками, делающими первые шаги в этом направлении. Благодаря простоте и лаконичности библиотека действительно позволяет быстро освоить основные принципы создания GUI, научиться работать с интерактивными элементами, понять механику эффективного взаимодействия между человеком и программой. С помощью Tkinter можно создавать интерфейсы приложений различных типов, например:

– вспомогательных программ — установщиков приложений, несложных фоторедакторов, калькуляторов, утилит для чистки компьютера, виджетов погоды и т.д.;

– различных электронных справочников, словарей, энциклопедий, учебников, программ для обучения иностранным языкам;

– несложных десктопных и мобильных игр — особенно квестов, пошаговых стратегий, в которых не требуется высокое качество анимации и сложные визуальные эффекты.

Кроме того, с помощью Tkinter можно создавать интерактивные интерфейсы веб-приложений и сайтов любой сложности, от простых лендингов до социальных сетей, информационных порталов и интернет-магазинов.

Также Tkinter используется в прототипировании и тестировании интерфейсов приложений. С его помощью можно быстро создавать рабочие «эскизы» (макеты) будущих приложений, чтобы с минимальными издержками отработать на них основные механики, выявить ошибочные и правильные решения, возможные программные баги и т.д. А затем на основе полученных данных разработать полноценный интерфейс.

Таким образом, несмотря на ограниченную функциональность, устаревший интерфейс и другие недостатки, Tkinter продолжает широко использоваться как программистами-любителями, так и профессиональными разработчиками. Хотя у него есть более функциональные и современные аналоги, этот графический пакет хорошо подходит для быстрой разработки несложных приложений, веб-страниц и прототипов.

Возможности Tkinter

Работа в этом графическом пакете организована вокруг набора визуальных интерактивных элементов управления — виджетов. Именно их совокупность образует пользовательский интерфейс приложения. В Tkinter представлена большая палитра виджетов — перечислим некоторые из них.

* Toplevel/Tk (окно верхнего уровня) — корневой виджет, в котором группируются все остальные, визуальная оболочка интерфейса.
* Frame (рамка) — с помощью этого виджета осуществляется группировка других элементов в пределах общей визуальной оболочки.
* Button (кнопка) — простейший элемент, клик на который вызывает определенное изменение (открытие нового окна, загрузку элемента и т.д.).
* Label (этикетка) — виджет, показывающий некоторый текст и/или изображение.
* Canvas (холст) — элемент, используемый для вывода простейших графических фигур, таких как стрелки, многоугольники, круги, линии. Также с его помощью отображаются тексты, окна и изображения.
* Text (текст)— виджет в виде поля, в котором можно вводить, редактировать и форматировать текст с использованием различных стилей, а также внедрять в него окна и изображения.
* Scale (шкала)— ползунок, с помощью которого можно плавно выбрать определенное числовое значение (например, стоимость товара или яркость изображения).

[6]

3 ОПИСАНИЕ ПРОГРАММЫ

Разработанное приложение имеет следующие функции:

– Проведение анализа текста из окошка внутри приложения.

– Проведение анализа текста из файла.

– Вывод результатов анализа текста.

СТРУКТУРА ПРОГРАММЫ

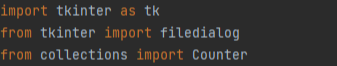
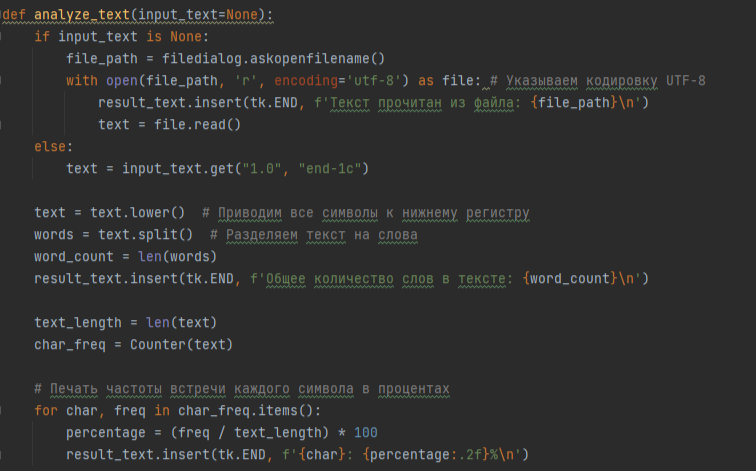


Рис. 1 – Подключенные библиотеки

Итак, начнем с того, что разберемся какие библиотеки мы подключили. На рисунке 1 представлены сами библиотеки. Tkinter – знаем, требуется для GUI. Из него мы используем filedialog, что позволяет реализовать нам проведение анализа текста из файлов. Из collections же используется Counter, далее он потребуется нам для подсчета частотности текста.

Рис. 2 – Анализ текста



Далее в программе встречаем анализ текста, видим его здесь как рисунок 2. Так как у нас возможен анализ текста и из файла, и из текстового окна, то видим здесь две вариации ввода текста. Вопрос по поводу того, зачем указана кодировка UTF-8: это сделано для того, чтобы гарантировать правильное чтение текста из файла с учетом специфических символов и языковых особенностей. Кодировка UTF-8 позволяет корректно обрабатывать символы различных языков, включая кириллицу, латиницу, китайские и японские иероглифы, и многие другие.

3 1 Чтение из файла

Первая if отвечает за чтение текста из файла. Когда мы будем запускать анализ текста из файла, первым делом программа попросит указать откуда мы забираем наш файл, причем сделано достаточно просто. Вместо указания пути к файлу мы буквально находим тот, что нам нужен, на рисунке 3 представлен выбор нашего файла “Арбузики”, что мы используем для демонстрации работы программы.

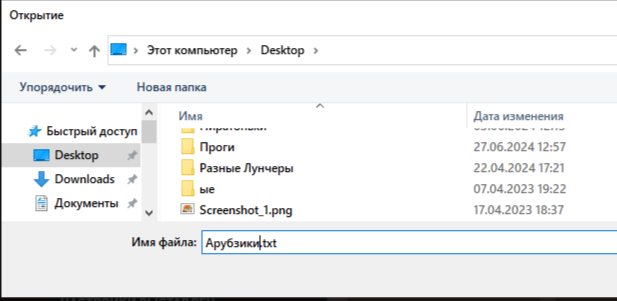
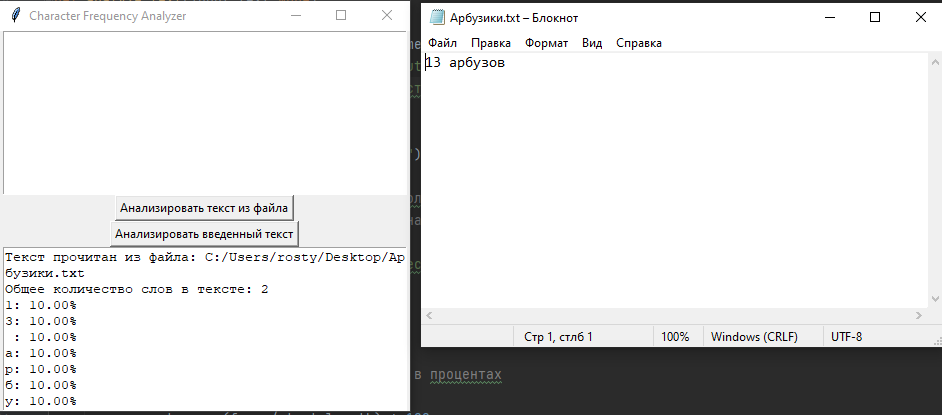


Рис. 3 – Поиск файла внутри компьютера

Далее, на рисунке 4, мы увидим, что наш код указывает то, откуда мы прочитали текст, а точнее весь путь до файла, и показывает частотный анализ. Из него вполне вероятно будет восстановить исходный текст, пускай он и не такой большой.

Рис. 4 – Анализированный файл “Арбузики”



3 2 Чтение предложений из текстового окна.

Итак, теперь смотрим работоспособность анализатора текста, если вставить сам текст внутрь окошка. На рисунке 5 пустое окно только что запущенного приложения. Далее пишем наш текст, пускай это будут всё те же самые 13 арбузов. На следующем рисунке под номер 6 увидим результат.

Рис. 5 – Запущенное приложение

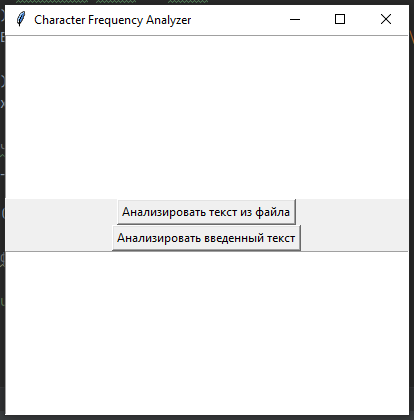
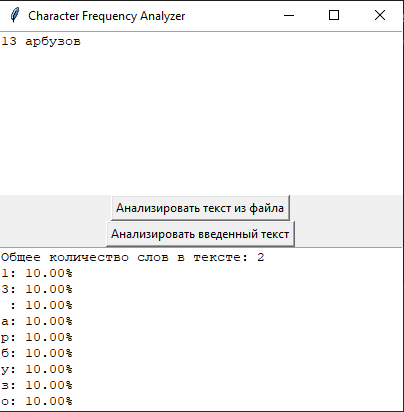


Рис. 6 – Проверка анализа введеного текста



Это мы написали малый текст, теперь можно проверить как программа отреагирует на немного более обширный текст. Предположим, что у нас есть много-абзацный текст, и как раз таки напишем что такое текст на рисунке 7.

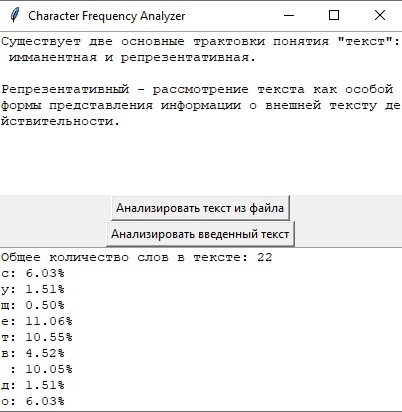


Рис 7 – Проверка много-абзацного текста

Как видно выше, программа спокойно прочитала абзацный текст.

3 3 Проверка спец. Символов

Во время работы с текстом было подмечено, что в каких-то определенных случаях программа считает спец. символы, а в каких-то нет. На рисунке 8 приведен подобный пример нестандартной работы программы.

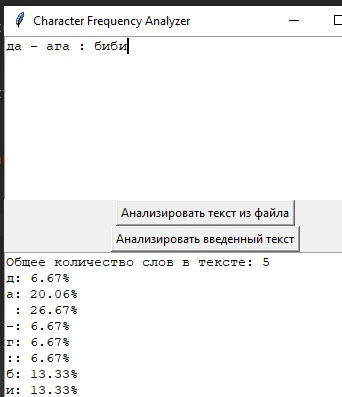


Рис. 8 – Ошибка в подсчете слов

Эту ошибку можно было починить достаточно легко, добавив функцию подсчета “правильных” символов. Это было бы выражение *words = re.findall(r'\b\w+\b', text).* Но на данный момент всё оставлено как есть для дальнейшей доработки.

3 4 Ознакомление с графическим интерфейсом

На рисунке 9 представлен блок с графикой. Быстро пробежимся по нему



Рис. 9 – GUI программы



Рис. 10 – Заголовок окна

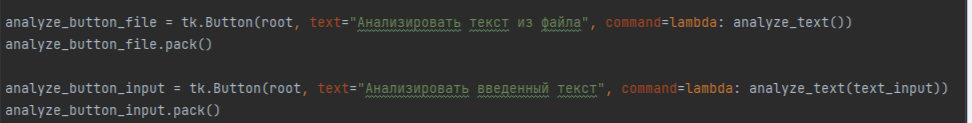
На рисунке выше указано само создание окна и его название, которое мы видим при запуске. Тоже самое что и название окна в браузере или где-нибудь еще.

Рис. 11 – Текстовое поле



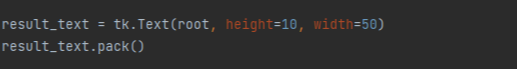
Здесь указано созданное текстовое поле, куда мы после пишем наш текст для анализа из самого приложения.

Рис 12 – Кнопки для анализа



Кнопки для работы указаны выше. Здесь первые две строки отвечают за кнопку с надписью "Анализировать текст из файла" в главном окне, при нажатии на которую будет вызываться функция analyze\_text() для анализа текста из файла., а вторые две – за кнопку с надписью "Анализировать введенный текст" в главном окне, при нажатии на которую будет вызываться функция analyze\_text() с аргументом text\_input, который содержит введенный пользователем текст.

Рис. 13 – Поле результатов



Выше – поле, в котором выводятся результаты выводятся для обозрения.

ЗАКЛЮЧЕНИЕ

В результате работы было разработана программа для анализа текста.

Все поставленные задачи были выполнены. Программа позволяет:

– Анализировать текст не только написанный в текстовом окне, но и из файла;

– Получить результат об анализе;

Думая о том, что можно было бы улучшить в программе стоило бы выделить:

1. Улучшение визуала.

Сейчас это ничем не примечательная программа, можно даже предположить что это сделано на коленке студентом в последний день.

2. Добавить возможность "попробовать расшифровать" текст.

Кажется, что эта возможность достаточно хорошо вписалась бы в общую картину программы. Расшифровать симметричные шифры: шифр Цезаря, или шифр замещения ROT13, (американский шифр, где если буква находится в первой половине, то она менялась на соответствующую этому месту во второй половине. Т.е. A-N, I-V, E-R). К сожалению, другие шифры, несимметричные, либо которые заменяют буквы на цифры, не смогут быть расшифрованы данной программой, так как в них с высокой вероятностью могут быть нарушены частотные соотношения. Если же действительно добавлять эту функцию, то стоит сказать, что при такой дешифровке вполне вероятны “дефекты речи”. В тексте будет появляться “картавость”, или определенные слова вообще будут слипаться между собой, но общую суть внутри длинного текста можно будет уловить. Шифры Вижинера и Плейфера вполне можно расшифровать при помощи частотного криптоанализа, хотя это и займет достаточно долгое время.

3. Проверка шрифта в файле.  
 Это, скорее, личное пожелание. Вдруг понравился шрифт, и, по какой-либо причине, глазками увидеть эту самую одну строчку не получается. Вполне можно реализовать через семейство библиотек fontTools, а точнее библиотеку fontTools.ttlib . Она может сверить метаданные о шрифтах и сказать, какой из них используется в файле.[8]  
 Правила пользования программой опубликованы в качестве проекта с открытым исходным кодом, и доступны в Git-репозитории по адресу вместе с отчетом и презентацией: https://github.com/BSIW2ARNI/Practics-Frequency

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Частотный анализ // <https://ru.wikipedia.org/wiki/Частотный_анализ>  
 2. Частотный анализ // https://habr.com/ru/articles/517410/

3. Язык программирования Python // <https://ru.wikipedia.org/wiki/Python>

4. Преимущества языка <https://skysmart.ru/articles/programming/preimushestva-i-nedostatki-python>

5. Язык программирования Python // <https://www.nic.ru/help/osnovy-yazyka-programmirovaniya-python_11662.html>

6. Библиотека Tkinter // <https://proglib.io/p/samouchitel-po-python-dlya-nachinayushchih-chast-20-graficheskiy-interfeys-na-tkinter-2023-05-02>  
 7. Частотный анализ с методами шифрования  
 7. Библиотека fontTools // <https://fonttools.readthedocs.io/en/latest/>

Приложение 1 - Исходный код приложения

import tkinter as tk

from tkinter import filedialog

from collections import Counter

def analyze\_text(input\_text=None):

if input\_text is None:

file\_path = filedialog.askopenfilename()

with open(file\_path, 'r', encoding='utf-8') as file: # Указываем кодировку UTF-8

result\_text.insert(tk.END, f'Текст прочитан из файла: {file\_path}\n')

text = file.read()

else:

text = input\_text.get("1.0", "end-1c")

text = text.lower() # Приводим все символы к нижнему регистру

words = text.split() # Разделяем текст на слова

word\_count = len(words)

result\_text.insert(tk.END, f'Общее количество слов в тексте: {word\_count}\n')

text\_length = len(text)

char\_freq = Counter(text)

# Печать частоты встречи каждого символа в процентах

for char, freq in char\_freq.items():

percentage = (freq / text\_length) \* 100

result\_text.insert(tk.END, f'{char}: {percentage:.2f}%\n')

#Создаем графический интерфейс

root = tk.Tk()

root.title("Character Frequency Analyzer")

text\_input = tk.Text(root, height=10, width=50)

text\_input.pack()

analyze\_button\_file = tk.Button(root, text="Анализировать текст из файла", command=lambda: analyze\_text())

analyze\_button\_file.pack()

analyze\_button\_input = tk.Button(root, text="Анализировать введенный текст", command=lambda: analyze\_text(text\_input))

analyze\_button\_input.pack()

result\_text = tk.Text(root, height=10, width=50)

result\_text.pack()

root.mainloop()