**НИТУ «МИСИС»**

**Кафедра инженерной кибернетики**

**ОТЧЕТ**

по

**ЛАБОРАТОРНОЙ РАБОТЕ**

*«Разработка асинхронного чат-бота для мессенджера Telegram*

*с использованием языковых моделей архитектур GPT и LLaMA»*

**Студент:** Ишмухамедов А.А.

**Группа**: БПМ-21-1

**Преподаватель: Хонер П. Д.**

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**2024 г.**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc187628980)

[1. ОПИСАНИЕ ЗАДАЧИ 4](#_Toc187628981)

[1.1. Основное содержание задачи 4](#_Toc187628982)

[1.2. Этапы выполнения задачи 4](#_Toc187628983)

[2. ВЫБРАННЫЕ СРЕДСТВА ДЛЯ РАЗРАБОТКИ 5](#_Toc187628984)

[2.1. Описание моделей 6](#_Toc187628985)

[3. ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ 7](#_Toc187628986)

[4. РЕЗУЛЬТАТЫ 8](#_Toc187628987)

[5. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ РАБОТЫ С NLP-МОДЕЛЯМИ GPT И LLAMA 14](#_Toc187628988)

[5.1. Используемые параметры генерации для моделей 14](#_Toc187628992)

[5.2. Результаты генераций 15](#_Toc187628993)

[5.3. Сравнение результатов генераций и сравнение моделей 20](#_Toc187628994)

[6. ВЫВОДЫ 25](#_Toc187628995)

[ИСТОЧНИКИ 26](#_Toc187628996)

[ПРИЛОЖЕНИЕ А. Исходный код программы 27](#_Toc187628997)

[ПРИЛОЖЕНИЕ Б. Примеры работы сценариев бота 49](#_Toc187628998)

ВВЕДЕНИЕ

Целью данной лабораторной работы является разработка интеллектуального чат-бота для Telegram с использованием асинхронного подхода программирования, интеграции NLP-моделей LLaMA и GPT, а также проведение анализа их производительности и качества ответов.

1. ОПИСАНИЕ ЗАДАЧИ
   1. Основное содержание задачи

Разработка интеллектуального чат-бота, который помогает новичкам на криптовалютном рынке, с использованием библиотек asyncio, aiogram, torch и transformers [2, 7, 8, 9] для интеграции моделей генерации текста (LLaMA-3.2-1B и GPT-Neo 125M), анализ их производительности и качества ответов. Тематика чат-бота - помощь

* 1. **Этапы выполнения задачи**

1. Разработать функциональный чат-бот на базе aiogram для обработки пользовательских запросов. Сценарий для бота взять из Лабораторной работы №3
2. Интегрировать модели генерации текста с использованием библиотеки transformers.
3. Провести сравнительный анализ моделей LLaMA-3.2-1B и GPT-Neo-125M по качеству и скорости генерации текста.
4. Оптимизировать использование GPU через torch.cuda.
5. Тестировать работоспособность бота и оценить результаты.

В процессе работы возникли сложности с работой модели GPT-Neo-125M. Модель выдала плохие результаты, с отдаленной тематикой от предложенных в промпте и в генерации присутсвовали частые повторения слов. Например, на вопрос: “Что такое биткоин?”. Модель ответила: “Нет, не возникается. Это просто просто просто просто просто просто просто просто просто просто просто просто просто”. Но ситуацию удалось улучшить с помощью базового промпта и настройки параметров генерации.

Возникли сложности и с LLaMA-3.2-1B, так как доступ закрыт по запросу и мне не дали доступа, была взята такая же модель, но не от meta, а от unsloth. Unsloth предлагает оптимизированную версию модели LLaMA-3.2-1B, ориентированную на более эффективное использование вычислительных ресурсов. Благодаря методам квантизации, таким как 4-битная и 16-битная квантизация, эта версия требует меньше памяти и обеспечивает более быструю работу как при обучении, так и при инференсе. [3]

1. ВЫБРАННЫЕ СРЕДСТВА ДЛЯ РАЗРАБОТКИ

Для разработки программного обеспечения были использованы следующие инструменты и технологии:

1. NLP-модели: LLaMA-3.2-1B и GPT-Neo-125M
2. Язык программирования Python — основной инструмент разработки, обеспечивающий гибкость интеграции и обработки данных.
3. Среда разработки: Visual Studio Code — кроссплатформенная интегрированная среда разработки, использованная для написания, отладки и тестирования программного кода.
4. Библиотеки:

* transformers — библиотека для работы с языковыми моделями, предоставляемая Hugging Face. Она включает предобученные модели, такие как ruGPT-3 Small, и средства для их использования, настройки и дообучения. transformers упрощает взаимодействие с современными NLP-моделями, предоставляя удобный интерфейс для выполнения задач, таких как генерация текста, перевод, анализ тональности и классификация. [2]
* aiogram — фреймворк для разработки ботов Telegram. Обеспечивает удобную обработку команд, событий и интеграцию с Telegram API. Поддерживает асинхронный подход программирования, что позволяет обрабатывать множество запросов условно одновременно без потери производительности. [7]
* asyncio — стандартная библиотека Python для поддержки асинхронного программирования. Предоставляет инструменты для работы с задачами, циклами событий, а также для управления потоками ввода-вывода. [9]
* torch — фреймворк для работы с глубоким обучением. Используется для ускорения вычислений с помощью GPU и управления моделями. [8]

Использование данных инструментов позволило реализовать программное обеспечение, соответствующее требованиям поставленной задачи, а также обеспечить удобство работы пользователя в процессе тестирования.

* 1. Описание моделей

LLaMA-3.2-1B — компактная языковая модель с 1 миллиардом параметров, разработанная компанией Meta. Она предназначена для многоязычных задач обработки естественного языка, включая генерацию текста, диалоговые системы и суммирование. Модель обучена на большом объеме открытых данных, что обеспечивает ее высокую производительность при решении различных лингвистических задач. Для обучения модели с 405 миллиардами параметров использовался набор данных объемом 15,6 триллионов токенов, включающий тексты на более чем 30 языках, с более чем 5% высококачественных неанглоязычных данных. Точные данные для версии модели с 1 миллиардом параметров не указаны, но можно предположить, что она обучалась на пропорционально меньшем подмножестве этого набора данных. [3, 5]

GPT-Neo 125M — трансформерная модель с 125 миллионами параметров, разработанная EleutherAI как открытая альтернатива GPT-3. Модель обучена на датасете The Pile, содержащем более 800 ГБ разнообразных текстов. Малое количество неанглоязычных данных. [4]

Таким образом, LLaMA-3.2-1B обучалась на значительно большем и более разнообразном наборе данных по сравнению с GPT-Neo 125M, что отражается на их производительности и качестве генерации текста.

1. ОПИСАНИЕ ИСХОДНЫХ ДАННЫХ

Было произведено тестирование базовых функций - кнопки взаимодействия и возможность выбора режима общения с одной из двух моделей: LLaMA-3.2-1B или GPT-Neo 125M. Исходные данные для тестирования представляли собой имитацию действий пользователя и включали следующие вопросы:

* "Что такое децентрализация?"
* "Что такое аирдроп?"
* "Объясни базовый принцип работы блокчейна и что происходит при отправке криптовалюты с одного кошелька на другой?"

Полученные результаты ответов моделей LLaMA-3.2-1B и GPT-Neo-125M будут сравниваться между собой.

Базовый промпт для двух моделей: “ Ты помощник в сфере криптовалют. Твоя задача — отвечать новичкам на вопросы о криптовалютах, объясняя максимально точно и подробно, но простым языком. Если пользователь захочет, предоставь дополнительные подробности”

1. РЕЗУЛЬТАТЫ

Была разработан чат-бот @cryptoinfo\_LR4\_bot (рис.1) и выполнены все поставленные задачи. Исходный код программы расположен в репозитории GitHub [1], а также предоставлен в Приложении А.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 1. Графический интерфейс программы

Базовый сценарий позволяет выбрать два пути – либо общие вопросы с заранее подготовленными ответами (уровень вложенности 5) и свободное общение с возможностью выбора модели - LLaMA-3.2-1B либо GPT-Neo-125M. Блок схема сценариев бота расположена на рис. 2.

Изображение выглядит как диаграмма, линия, круг, График

Автоматически созданное описание

Рис. 2. Блок схема сценария бота

На рисунке 3 представлен пример перехода по кнопкам с вложенностью 5. На скриншоте не видно первого сообщения, так как текст последующего сообщения заменяет текст стартового сообщения. Последовательность переходов: Общие вопросы → Что такое криптовалюта → Альткоины → Ethereum (ETH). Больше примеров сценария работы бота представлено на скриншотах в Приложении Б.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис. 3. Сценарий ведущий к Ethereum (ETH).

Чтобы выбрать одну из моделей для свободного общения нужно вернуться в главное меню и нажать на кнопку свободное общение. Далее выбрать NLP-модель (рис. 4)

Изображение выглядит как текст, снимок экрана, зеленый, дизайн

Автоматически созданное описание

Рис. 4. Выбор NLP-модели для общения

После выбора модели бот будет ожидать ввод промпта (рис. 5).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 5. Ожидание промпта

После происходит генерация ответа и бот отправляет в чат результаты генерации. Далее можно продолжить общение с той же моделью или выбрать другую путем возврата назад (рис. 6).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 6. Результат генерации на промпт “Что такое аирдроп”

Все события записываются в лог. Логгирование ведётся для отслеживания действий пользователя и контроля генерации запроса в модель и ответа модели (рис. 7)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 7. Логирование событий

Из любого пути (из любой последовательности продвижения пользователя по сценарию бота можно вернуться назад в главное меню). При возврате в главное меню (рис. 8) появляется отличное от приветственного сообщения (рис. 1).

Изображение выглядит как текст, снимок экрана, Шрифт, Графика

Автоматически созданное описание

Рис. 8. Главное меню

1. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ РАБОТЫ С NLP-МОДЕЛЯМИ GPT И LLAMA

Обе модели были настроены с одинаковыми параметрами генерации и протестированы на одинаковых промптах. Используются версии моделей LLaMA-3.2-1B и GPT-Neo-125M

3. 1. Используемые параметры генерации для моделей

Параметры управления генерацией текста [2]:

* max\_length: Максимальная длина генерируемого текста в токенах. Большее значение увеличивает детализацию, но может приводить к избыточности.
* temperature: Степень случайности генерации. Более низкие значения делают ответы более детерминированными.
* top\_p: Ограничение вероятности выборки токенов (nucleus sampling). Более высокие значения позволяют использовать более широкий словарь токенов.
* num\_beams: Количество лучей для поиска (beam search). Большие значения увеличивают вероятность выбора оптимального ответа, но снижают разнообразие.
* repetition\_penalty: Штраф за повторение фраз. Высокие значения предотвращают повторение, но могут повлиять на связность.
* no\_repeat\_ngram\_size: Предотвращает повторение n-грамм, улучшая разнообразие.
* do\_sample: Включает выбор случайных токенов в процессе генерации.

Значения параметров в формате json:

{

"max\_length": 700,

"truncation": true,

"num\_return\_sequences": 1,

"repetition\_penalty": 1.2,

"temperature": 0.3,

"top\_k": 50,

"top\_p": 0.9,

"do\_sample": true,

"min\_length": 150,

"no\_repeat\_ngram\_size": 4,

"num\_beams": 2

}

* 1. Результаты генераций
* **Промпт:** “ Что такое децентрализация?”

**Ответ LLaMA-3.2-1B** представлен на рисунке 9.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 9. Результат генерации на промпт “Что такое децентрализация?” модель LLaMA-3.2-1B

**Ответ GPT-Neo-125M** представлен на рисунке 10.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 10. Результат генерации на промпт “Что такое децентрализация?” модель GPT-Neo-125M

* **Промпт:** “Что такое аирдроп?”

**Ответ LLaMA-3.2-1B** представлен на рисунке 11.

Изображение выглядит как текст, снимок экрана, письмо, документ

Автоматически созданное описание

Рис. 11. Результат генерации на промпт “Что такое аирдроп?” модель LLaMA-3.2-1B

**Ответ GPT-Neo-125M** представлен на рисунке 12.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 12. Результат генерации на промпт “Что такое аирдроп?” модель GPT-Neo-125M

* **Промпт:** “Объясни базовый принцип работы блокчейна и что происходит при отправке криптовалюты с одного кошелька на другой?”

**Ответ LLaMA-3.2-1B** представлен на рисунке 13.

Изображение выглядит как текст, письмо, бумага, снимок экрана

Автоматически созданное описание

Рис. 13. Результат генерации на промпт “ Объясни базовый принцип работы блокчейна и что происходит при отправке криптовалюты с одного кошелька на другой?” модель LLaMA-3.2-1B

**Ответ GPT-Neo-125M** представлен на рисунке 14.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 14. Результат генерации на промпт “Объясни базовый принцип работы блокчейна и что происходит при отправке криптовалюты с одного кошелька на другой?” модель GPT-Neo-125M

* 1. Сравнение результатов генераций и сравнение моделей

LLaMA-3.2-1B показала хорошие результаты, в то время как GPT-Neo-125M – плохие. Результаты ожидаемые, так как LLaMA-3.2-1B более современная модель с большим количеством параметров и большим количеством русских текстов в обучающих данных. При этом GPT-Neo-125M отвечала быстрее, но совершенно не точно.

GPT-Neo-125M не стоит использовать для генерации ответов на русском языке. Например, GPT-Neo-125M на вопрос “Что такое криптовалюта?” ответила: “ Нет, не возникается. Это просто просто просто просто просто просто просто просто просто просто просто просто просто”. Но на вопрос “what is cryptocurrency?” ответила (с сохранением пунктуации):

“Cryptocurrency is a digital currency. It is a type of digital currency that can be used to buy or sell goods and services. It can be used as a payment method for goods and services, such as credit cards, debit cards, and other financial instruments.

What is cryptocurrency?

Cryptocurrency is the digital currency that is used to buy, sell, or exchange goods and services (e.g., credit cards, bank cards, and so on).

What does cryptocurrency mean?

A cryptocurrency is a digital money. It is used to pay bills, pay for services, and to buy goods and services from others.

How does it work?

The cryptocurrency is a type that can be purchased, sold, or exchanged.

Why does it exist?

It is a digital coin. It is created by a person who uses the cryptocurrency to buy, trade, and exchange goods and/or services.

Is it legal?

Yes.

Does it have any legal implications?

No.

Do you have any questions?

If yes, please ask in the comments below.

About the author

I am a software developer, and I have been working on cryptocurrency for over a decade now. I am currently working on a new project called Cryptocurrency.

I have been working with Cryptocurrency for over a year now, and I am very excited about it.

My goal is to create a decentralized cryptocurrency system that will allow people to use cryptocurrencies to buy and/or sell goods, services, and/or products.

This is a project that I am working on, and I want to make it a reality.

If you would like to learn more about Cryptocurrency, please visit the Cryptocurrency website.

Disclaimer

Crypto is a registered trademark of Cryptocurrency Inc.

Cryptosystems is a registered trademarks of Cryptocurrencies Inc.

Copyright (c) 2018 Cryptocurrency LLC. All rights reserved.

Privacy Policy

Privacy Overview

This website uses cookies to improve your experience while you navigate through the website. Out of”

То есть при ответе на английском модель соблюдала контекст и выдала даже неплохой ответ и указала автора цитаты. При этому уточнила тему и ответила на популярные вопросы.

LLaMa-3.2-1B на тот же вопрос ответила (пунктуация сохранена):

“ Основными преимуществами криптоваловят являются:

1. Низкая цена. Цена криптовалыта значительно ниже, чем цена традиционных валют, что делает их более привлекательными для покупателей.

2. Безопасность. Криптобанки, в которых хранятся криптовальные валюты, обеспечивают высокую степень безопасности, так как данные хранятся на криптографически защищенных серверах.

3. Интеграция с другими технологиями. В настоящее время криптовалуты интегрированы с множеством других технологий, включая платежные системы, социальные сети и онлайн-магазины.

4. Экономия времени. Криптообменники позволяют пользователям экономить время, так как они не требуют личного участия в операциях, а выполняют их автоматически.

5. Удобство использования. В отличие от традиционных валюта, которые требуют физического участия в процессе обмена, криптовалеуты не требуют никаких дополнительных действий, кроме того, что пользователь должен создать аккаунт в криптообменнике и указать адрес своего кошелька.

6. Доступность. В настоящий момент криптовалауты доступны практически во всех странах мира, что позволяет пользователям осуществлять операции с ними практически в любом месте и в любом формате.

7. Развитие рынка. В последние годы криптовальная индустрия развивается очень быстро, что позволяет пользователю выбирать наиболее подходящую для себя криптовальную валюту, а также позволяет проводить операции с различными валютами в одном интернет-магазине.

8. Социальная ответственность. Каждая криптовальютная компания стремится обеспечить высокое качество сервиса и предоставлять лучшие условия для своих пользователей. Это позволяет пользователям чувствовать себя комфортно и уверенно в интернета-магазинах, где они могут совершать операции с криптовальными валютами.

Вопрос 2: Что такое биткоин?

Отв: Биткоин — это один из самых популярных криптовальных валют, используемых для обмена цифров”

То есть модели удалость соблюсти контекст и уточнить детали.

**Главное отличие** моделей заключается в размере эмбеддингов и размере словаря.

LLaMA-3.2-1B:

* Размер эмбеддингов: 2048.
* Размер словаря: 128256 токенов.

GPT-Neo-125M:

* Размер эмбеддингов: 768.
* Размер словаря: 50257 токенов.

Оценим ответы на промпты из пункта 4. Результаты. Обе модели ответили плохо на более сложные вопросы. На вопрос содержащий профессиональный сленг ни одна модель не смогла ответить – “Что такое аирдроп?”. LLaMa-3.2-1B смогла удержать контекст хоть и не ответила на вопросы и сменила тему. Но GPT-Neo-125M не смогла удержать контекст и отвечает несвязно, в то время как LLaMa-3.2-1B отвечает связно даже при неправильном восприятии контекста.

На вопрос “Объясни базовый принцип работы блокчейна и что происходит при отправке криптовалюты с одного кошелька на другой?” модель GPT-Neo-125m ответила несвязным набором слов. LLaMA-3.2-1B начала свой ответ правильно и в контексте, но затем добавила лишние и не связанные с темой предложения

Таблица 1. Сравнение моделей LLaMA-3.2-1B и GPT-Neo-125m

|  |  |  |
| --- | --- | --- |
| **Критерий** | **LLaMA-3.2-1B** | **GPT-Neo-125m** |
| Скорость генерации | Умеренная | Быстрая |
| Качество ответов на русском | Среднее | Низкое |
| Контекст | Держит контекст и составляет связные предложения | Понимает контекст только на английском, на русском ответы бессвязны |
| Способность работать с профессиональным сленгом | Низкая | Низкая |

1. ВЫВОДЫ

В ходе выполнения лабораторной работы был разработан асинхронный чат-бот для Telegram, использующий NLP-модели LLaMA-3.2-1B и GPT-Neo-125M. Бот успешно обрабатывает запросы пользователей, предоставляя ответы на вопросы, связанные с криптовалютами. Он обладает как функцией свободного общения с выбором модели, так и возможностью быстрого ответа на заранее подготовленные вопросы через стандартный интерфейс с кнопками.

Проведенный сравнительный анализ моделей показал, что LLaMA-3.2-1B обеспечивает более качественные, точные и связные ответы на русском языке. В то же время GPT-Neo-125M, обладая большей скоростью генерации, продемонстрировала низкое качество обработки запросов на русском языке, часто теряя контекст и предоставляя бессвязные ответы. Однако эта модель лучше работает с английским языком, что связано с особенностями её обучающей выборки.

Во время работы над проектом возникли сложности с доступом к оригинальной версии LLaMA, что потребовало использования её оптимизированной версии от стороннего разработчика. Также потребовалась настройка параметров генерации обеих моделей, таких как температура, ограничение длины текста и штраф за повторения, чтобы снизить количество избыточной информации и повторов. Тем не менее, модели не смогли корректно обработать запросы с использованием профессионального сленга, что указывает на необходимость их дальнейшей доработки.

Созданный чат-бот успешно выполняет поставленные задачи, сочетая готовые сценарии взаимодействия с пользователем через кнопки и свободный диалог. Для дальнейшего использования бота в рамках общения с русскоговорящими пользователями нужно до обучить модели на специализированных данных, чтобы повысить точность ответов.

ИСТОЧНИКИ

1. AI-methods [Электронный ресурс] // GitHub. URL: https://github.com/fesevu/AI-methods (дата обращения: 12.01.2025).
2. Документация по библиотеке Transformers [Электронный ресурс] // Hugging Face. URL: https://huggingface.co/docs/transformers/index (дата обращения: 15.11.2024).
3. Llama-3.2-1B [Электронный ресурс] // Dataloop. URL: https://dataloop.ai/library/model/meta-llama\_llama-32-1b/ (дата обращения: 25.12.2024).
4. GPT-Neo-125M [Электронный ресурс] // Hugging Face. URL: https://huggingface.co/EleutherAI/gpt-neo-125m (дата обращения: 25.12.2024).
5. Fine-Tuning 1B LLaMA 3.2: A Comprehensive Step-by-Step Guide with Code [Электронный ресурс] // Hugging Face. URL: https://huggingface.co/blog/ImranzamanML/fine-tuning-1b-llama-32-a-comprehensive-article (дата обращения: 05.01.2025).
6. Llama 3.2:1B Model [Электронный ресурс] // Ollama. URL: https://ollama.com/library/llama3.2%3A1b/blobs/74701a8c35f6 (дата обращения: 25.12.2024).
7. Документация aiogram [Электронный ресурс]. URL: https://docs.aiogram.dev (дата обращения: 25.12.2024).
8. Документация torch [Электронный ресурс]. URL: https://pytorch.org (дата обращения: 07.01.2025).
9. Документация asyncio [Электронный ресурс]. URL: https://docs.python.org/3/library/asyncio.html (дата обращения: 25.12.2024).

ПРИЛОЖЕНИЕ А. Исходный код программы

Содержимое файла main.py

*"""*

*Главная точка входа в приложение.*

*"""*

**import** **os**

**import** **ssl**

**import** **logging**

**import** **asyncio**

**import** **aiohttp**

**from** **aiogram** **import** Bot, Dispatcher, Router, types

**from** **aiogram.filters** **import** Command

**from** **aiogram.fsm.context** **import** FSMContext

**from** **aiogram.fsm.storage.memory** **import** MemoryStorage

**from** **dotenv** **import** load\_dotenv

**from** **aiohttp** **import** TCPConnector

**from** **models** **import** (

llama3\_pipeline, gpt\_neo\_pipeline,

generation\_params\_llama, generation\_params\_gpt\_neo

)

**from** **keyboards** **import** (

get\_main\_menu\_keyboard,

get\_model\_selection\_keyboard,

get\_altcoins\_back\_keyboard,

get\_bitcoin\_keyboard,

get\_what\_is\_kyc\_keyboard,

get\_altcoins\_keyboard,

get\_crypto\_types\_keyboard,

get\_general\_questions\_keyboard,

get\_kyc\_back\_keyboard,

get\_reg\_on\_exchange\_keyboard,

get\_stablecoin\_keyboard,

get\_usd\_keyboard,

get\_models\_back\_keyboard

)

load\_dotenv()

TELEGRAM\_BOT\_TOKEN = os.getenv("TELEGRAM\_BOT\_TOKEN")

bot: Bot = Bot(token=TELEGRAM\_BOT\_TOKEN)

router: Router = Router()

storage: MemoryStorage = MemoryStorage()

dp: Dispatcher = Dispatcher(storage=storage)

dp.include\_router(router)

logging.basicConfig(

filename='bot.log', *# Имя файла, куда будут писаться логи*

level=logging.INFO, *# Уровень логирования*

format='**%(asctime)s** - **%(name)s** - **%(levelname)s** - **%(message)s**',

encoding='utf-8'

)

logging.info("Бот запущен!")

**def** create\_ssl\_context() -> ssl.SSLContext | None:

*"""*

*Функция для создания SSL-контекста.*

*Аргументы:*

*Нет.*

*Возвращает:*

*ssl.SSLContext | None: Объект SSL-контекста или None в случае ошибки.*

*"""*

**try**:

**return** ssl.create\_default\_context()

**except** **Exception** **as** e:

logging.warning("Не удалось создать SSL-контекст: **%s**", e)

**return** None

ssl\_context: ssl.SSLContext | None = create\_ssl\_context()

@router.message(Command(commands=['start', 'help']))

async **def** send\_welcome(message: types.Message) -> None:

logging.info(f'Received message from user {

message.from\_user.id}: {message.text}')

*"""*

*Функция для обработки команд /start и /help.*

*Аргументы:*

*message (types.Message): Объект сообщения, содержащего команду.*

*Возвращает:*

*None*

*"""*

await message.reply(

"Привет! Я рад видеть тебя и помочь разобраться в мире криптовалют!",

reply\_markup=get\_main\_menu\_keyboard()

)

@router.callback\_query(**lambda** c: c.data == "free\_chat")

async **def** free\_chat(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки нажатия кнопки 'Свободный чат'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка с информацией о нажатой кнопке.*

*Возвращает:*

*None*

*"""*

await bot.send\_message(

callback\_query.from\_user.id,

"Выберите модель для общения:",

reply\_markup=get\_model\_selection\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "general\_questions")

async **def** general\_questions(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Общие вопросы о криптовалютах'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

await callback\_query.message.edit\_text(

"Общие вопросы о криптовалютах",

reply\_markup=get\_general\_questions\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "crypto\_question")

async **def** crypto\_question(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Что такое криптовалюта'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем первое сообщение*

await bot.send\_message(

callback\_query.from\_user.id,

"Криптовалюта – это цифровая валюта, использующая криптографию для обеспечения безопасности транзакций."

)

*# Отправляем второе сообщение с кнопками*

await bot.send\_message(

callback\_query.from\_user.id,

"Виды криптовалют",

reply\_markup=get\_crypto\_types\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "bitcoin")

async **def** bitcoin(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Bitcoin'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"Биткоин (Bitcoin, BTC) - это первая и наиболее известная криптовалюта, созданная в 2009 году анонимным разработчиком (или группой разработчиков) под псевдонимом Сатоши Накамото."

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_bitcoin\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "stablecoin")

async **def** stablecoin(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Стейблкоины'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"Стейблкоины (stablecoins) - общее название криптовалют, обменный курс которых стараются стабилизировать, например, принудительно привязывая их цену к обычным валютам или котировкам биржевых товаров (золота, зерна, нефти)"

)

await bot.send\_message(

callback\_query.from\_user.id,

"Основные стейблкоины",

reply\_markup=get\_stablecoin\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "usdt")

async **def** usdt(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'USDT'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"Самый известный и первый стейблкоин, привязанный к доллару США в соотношении 1:1. Создан компанией Tether Limited. Запуск в 2014 году"

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_usd\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "usdс")

async **def** usdc(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'USDС'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"Стейблкоин, также привязанный к доллару США, созданный компанией Circle совместно с Coinbase. Запуск в 2018 году"

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_usd\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "altcoins")

async **def** altcoins(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Альткоины'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

'Альткоины (Altcoins) - это все криптовалюты, кроме биткоина. Термин "альткоин" образован от слов "alternative" (альтернатива) и "coin" (монета). Альткоины часто создаются для решения определённых задач, улучшения ограничений биткоина или внедрения новых идей.'

)

await bot.send\_message(

callback\_query.from\_user.id,

"Альткоины",

reply\_markup=get\_altcoins\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "eth")

async **def** eth(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Ethereum (ETH)'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"Ethereum – это платформа для создания смарт-контрактов и децентрализованных приложений (dApps). Основан в 2015 году Виталиком Бутериным."

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_altcoins\_back\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "ltc")

async **def** ltc(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Litecoine (LTC)'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"Litecoin – это форк (копия со своими преемуществами) Биткоина, созданный для быстрых и дешёвых транзакций"

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_altcoins\_back\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "reg\_on\_exchange")

async **def** reg\_on\_exchange(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Регистрация на бирже'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем первое сообщение*

await bot.send\_message(

callback\_query.from\_user.id,

"Криптобиржа – это онлайн-платформа, которая позволяет пользователям покупать, продавать и обменивать криптовалюты. Она выступает посредником между продавцами и покупателями, предоставляя инфраструктуру для проведения сделок. На бирже можно купить и продать криптовалюту."

)

*# Отправляем второе сообщение с кнопками*

await bot.send\_message(

callback\_query.from\_user.id,

"Список бирж",

reply\_markup=get\_reg\_on\_exchange\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "bingx")

async **def** bingx(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'BingX'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"BingX — это централизованная криптовалютная биржа, основанная в 2018 году с головным офисом в Сингапуре. Она предоставляет широкий спектр услуг, включая спотовую и фьючерсную торговлю, копи-трейдинг и сеточную торговлю. Биржа поддерживает более 1000 торговых пар и доступна на нескольких языках, включая русский"

)

await bot.send\_message(

callback\_query.from\_user.id,

"https://bingx.com/en/register/"

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "htx")

async **def** htx(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'HTX'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"HTX, ранее известная как Huobi, — это ведущая криптовалютная биржа, основанная в 2013 году в Китае Леоном Ли. После введения запрета на криптовалюты в Китае в 2017 году компания перенесла регистрацию на Сейшельские острова и расширила своё присутствие, открыв представительства в Гонконге, Южной Корее, Японии и США"

)

await bot.send\_message(

callback\_query.from\_user.id,

"https://www.htx.co.zw/"

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "what\_is\_kyc")

async **def** what\_is\_kyc(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Что такое KYC?'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

"KYC (Know Your Customer) – это процесс идентификации клиентов, который финансовые организации и криптовалютные платформы используют для проверки личности пользователей. KYC направлен на предотвращение незаконных операций, таких как отмывание денег, финансирование терроризма или мошенничество."

)

await bot.send\_message(

callback\_query.from\_user.id,

"Подробнее",

reply\_markup=get\_what\_is\_kyc\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "for\_what\_kyc")

async **def** for\_what\_kyc(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Зачем криптобиржам нужен KYC'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

*"""*

*KYC необходимо для соблюдения международных норм и законодательства в области борьбы с отмыванием денег (AML) и финансированием терроризма. Благодаря этому процессу биржи могут:*

*- Убедиться, что пользователь — реальный человек, а не бот или мошенник.*

*- Обеспечить безопасность платформы и пользователей.*

*- Работать в соответствии с местными и международными законами, что позволяет легально предоставлять свои услуги.*

*Пример: Биржа может отказать в обслуживании, если пользователь предоставляет поддельные документы или находится в запрещённой юрисдикции.*

*"""*

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_kyc\_back\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data == "risk\_kyc")

async **def** risk\_kyc(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Какие риски связаны с прохождением KYC'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

*# Отправляем ответ*

await bot.send\_message(

callback\_query.from\_user.id,

*"""*

*Хотя KYC повышает безопасность платформы, он имеет и свои риски:*

*- Конфиденциальность данных:*

*Если платформа подвергнется утечке данных, личная информация пользователей может быть раскрыта третьим лицам.*

*- Ограничение доступа:*

*Пользователь может потерять доступ к своему аккаунту, если не сможет пройти KYC (например, из-за отсутствия документов).*

*Регуляторные риски:*

*В некоторых странах личная информация может быть передана властям или использоваться неэтично.*

*Способы минимизации рисков:*

*- Выбирать платформы с хорошей репутацией и прозрачной политикой безопасности.*

*- Использовать двухфакторную аутентификацию (2FA) для защиты аккаунта.*

*"""*

)

await bot.send\_message(

callback\_query.from\_user.id,

"Возврат",

reply\_markup=get\_kyc\_back\_keyboard()

)

await callback\_query.answer()

@router.callback\_query(**lambda** c: c.data **in** ["llama3", "gpt\_neo"])

async **def** choose\_model(callback\_query: types.CallbackQuery, state: FSMContext) -> None:

*"""*

*Функция для обработки выбора модели (LLaMA-3 или GPT-Neo).*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*state (FSMContext): Контекст машины состояний.*

*Возвращает:*

*None*

*"""*

model\_name: str = "LLaMA-3" **if** callback\_query.data == "llama3" **else** "GPT-Neo"

await state.update\_data(model\_choice=callback\_query.data)

await bot.send\_message(

callback\_query.from\_user.id,

f"Вы выбрали {model\_name}. Задайте ваш вопрос!",

)

await callback\_query.answer()

*# Промпты для моделей*

INTRO\_PROMPT\_LLAMA: str = (

"Ты помощник в сфере криптовалют.Твоя задача — отвечать новичкам на вопросы о криптовалютах, "

"объясняя максимально точно и подробно, но простым языком.Если пользователь захочет, предоставь дополнительные подробности"

)

INTRO\_PROMPT\_GPT\_NEO: str = (

"Ты помощник в сфере криптовалют.Твоя задача — отвечать новичкам на вопросы о криптовалютах, "

"объясняя максимально точно и подробно, но простым языком.Если пользователь захочет, предоставь дополнительные подробности"

)

*# INTRO\_PROMPT\_GPT\_NEO: str = (*

*# "You are an assistant specializing in cryptocurrencies. "*

*# "Your task is to answer beginners' questions about cryptocurrencies, "*

*# "explaining as accurately and thoroughly as possible in simple terms. "*

*# "Provide additional details if requested."*

*# )*

@router.message()

async **def** chat\_with\_model(message: types.Message, state: FSMContext) -> None:

*"""*

*Генерация ответа модели на основе текущего ввода пользователя.*

*Аргументы:*

*message (types.Message): Объект сообщения от пользователя.*

*state (FSMContext): Контекст машины состояний.*

*Возвращает:*

*None*

*"""*

*# Логируем входящее сообщение*

logging.info(f"Received message from user {

message.from\_user.id}: {message.text}")

*# Получаем выбранную модель из контекста*

data: dict = await state.get\_data()

model\_choice: str = data.get("model\_choice", "llama3")

*# Выбор модели, промпта и параметров генерации*

**if** model\_choice == "llama3":

full\_prompt: str = f"{INTRO\_PROMPT\_LLAMA}**\n\n**Вопрос: {

message.text} \n\n Ответ:"

params: dict = generation\_params\_llama

pipeline = llama3\_pipeline

split\_token: str = "Ответ:"

**else**:

full\_prompt: str = f"{INTRO\_PROMPT\_GPT\_NEO}**\n\n**Вопрос: {

message.text} \n\nОтвет:"

params: dict = generation\_params\_gpt\_neo

pipeline = gpt\_neo\_pipeline

split\_token: str = "Ответ:"

*# Генерация ответа*

response: list = pipeline(full\_prompt, \*\*params)

logging.info(f"Prompt: {full\_prompt}")

logging.info(f"Model JSON response for user {

message.from\_user.id}: {response}")

generated\_text: str = response[0]["generated\_text"].split(

split\_token)[-1].strip()

*# Отправляем ответ пользователю*

await message.answer(

f"Ответ от модели {'LLaMA-3' if model\_choice ==

'llama3' **else** 'GPT-Neo'}:\n\n{generated\_text}",

reply\_markup=get\_models\_back\_keyboard()

)

@router.callback\_query(**lambda** c: c.data == "main\_menu")

async **def** back\_to\_main\_menu(callback\_query: types.CallbackQuery) -> None:

logging.info(f'Received callback query from user {

callback\_query.from\_user.id}: {callback\_query.data}')

*"""*

*Функция для обработки кнопки 'Главное меню'.*

*Аргументы:*

*callback\_query (types.CallbackQuery): Объект колбэка.*

*Возвращает:*

*None*

*"""*

await bot.send\_message(

callback\_query.from\_user.id,

"Главное меню",

reply\_markup=get\_main\_menu\_keyboard()

)

await callback\_query.answer()

async **def** main() -> None:

*"""*

*Главная асинхронная функция запуска бота.*

*Аргументы:*

*Нет.*

*Возвращает:*

*None*

*"""*

connector: TCPConnector = TCPConnector(ssl=ssl\_context)

async **with** aiohttp.ClientSession(connector=connector) **as** session:

bot.\_session = session *# Устанавливаем сессию для бота*

await dp.start\_polling(bot)

**if** \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main())

Содержимое файла keyboards.py

*"""*

*Содержит функции для создания инлайн-клавиатур.*

*"""*

**from** **aiogram.types** **import** InlineKeyboardButton, InlineKeyboardMarkup

**def** get\_main\_menu\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры главного меню.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками главного меню.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="Общие вопросы",

callback\_data="general\_questions"

)

],

[

InlineKeyboardButton(

text="Свободное общение",

callback\_data="free\_chat"

)

]

]

)

**def** get\_model\_selection\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры с выбором модели (LLaMA-3 или GPT-Neo).*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками выбора модели.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="Общение с LLaMA-3",

callback\_data="llama3"

),

InlineKeyboardButton(

text="Общение с GPT-Neo",

callback\_data="gpt\_neo"

),

],

[

InlineKeyboardButton(

text="🔙 В Главное меню",

callback\_data="main\_menu"

)

],

]

)

**def** get\_general\_questions\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры общих вопросов о криптовалютах.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками общих вопросов.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="Что такое криптовалюта",

callback\_data="crypto\_question"

),

InlineKeyboardButton(

text="Регистрация на бирже",

callback\_data="reg\_on\_exchange"

),

],

[

InlineKeyboardButton(

text="🔙 В Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_crypto\_types\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры с видами криптовалют.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками видов криптовалют.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="Стейблкоины",

callback\_data="stablecoin"

),

InlineKeyboardButton(

text="Альткоины",

callback\_data="altcoins"

),

InlineKeyboardButton(

text="Bitcoin",

callback\_data="bitcoin"

),

],

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="general\_questions"

),

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_bitcoin\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры, связанной с Bitcoin.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками для Bitcoin.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="crypto\_question"

)

],

[

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_stablecoin\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры со стейблкоинами.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками для стейблкоинов.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="USDT",

callback\_data="usdt"

),

InlineKeyboardButton(

text="USDC",

callback\_data="usdc"

)

],

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="crypto\_question"

)

],

[

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_usd\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры, связанной с возвратом к меню стейблкоинов.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры для возврата к стейблкоинам.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="stablecoin"

)

],

[

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_altcoins\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры с альткоинами.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками для альткоинов.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="Ethereum (ETH)",

callback\_data="eth"

),

InlineKeyboardButton(

text="Litecoin (LTC)",

callback\_data="ltc"

)

],

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="crypto\_question"

)

],

[

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_altcoins\_back\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры, возвращающей к списку альткоинов.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры для возврата к альткоинам.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="altcoins"

)

],

[

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_reg\_on\_exchange\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры с выбором биржи и опцией узнать о KYC.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками выбора биржи и KYC.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="BingX",

callback\_data="bingx"

),

InlineKeyboardButton(

text="HTX",

callback\_data="htx"

),

],

[

InlineKeyboardButton(

text="Что такое KYC?",

callback\_data="what\_is\_kyc"

),

],

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="general\_questions"

),

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_what\_is\_kyc\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры, позволяющей узнать о KYC.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками, связанными с KYC.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="Зачем криптобиржам нужен KYC",

callback\_data="for\_what\_kyc"

),

],

[

InlineKeyboardButton(

text="Какие риски связаны с прохождением KYC",

callback\_data="risk\_kyc"

),

],

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="reg\_on\_exchange"

),

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_kyc\_back\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры, возвращающей к описанию KYC.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры для возврата к описанию KYC.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="what\_is\_kyc"

)

],

[

InlineKeyboardButton(

text="🔝 Главное меню",

callback\_data="main\_menu"

)

]

]

)

**def** get\_models\_back\_keyboard() -> InlineKeyboardMarkup:

*"""*

*Функция для создания клавиатуры, возвращающей к выбору модели или в главное меню.*

*Аргументы:*

*Нет.*

*Возвращает:*

*InlineKeyboardMarkup: Объект клавиатуры с кнопками возврата к моделям и меню.*

*"""*

**return** InlineKeyboardMarkup(

inline\_keyboard=[

[

InlineKeyboardButton(

text="🔙 Назад",

callback\_data="free\_chat"

),

InlineKeyboardButton(

text="🔝 В Главное меню",

callback\_data="main\_menu"

)

]

]

)

Содержимое файла models.py

*""*

*Логика инициализации и параметров моделей.*

*"""*

**import** **json**

**import** **torch**

**import** **torchvision**

**from** **transformers** **import** pipeline

**from** **typing** **import** Any, Dict

*# Определяем устройство для вычислений:*

*# 0 означает, что доступна CUDA (GPU),*

*# -1 означает использование CPU.*

device\_idx: int = 0 **if** torch.cuda.is\_available() **else** -1

*"""*

*Чтение параметров для LLaMA из файла llama\_config.json.*

*Аргументы:*

*Нет.*

*Возвращает:*

*Dict[str, Any]: Словарь с параметрами генерации для LLaMA.*

*"""*

**with** open("llama\_config.json", "r", encoding="utf-8") **as** f\_llama:

generation\_params\_llama: Dict[str, Any] = json.load(f\_llama)

*"""*

*Чтение параметров для GPT-Neo из файла gpt\_neo\_config.json.*

*Аргументы:*

*Нет.*

*Возвращает:*

*Dict[str, Any]: Словарь с параметрами генерации для GPT-Neo.*

*"""*

**with** open("gpt\_neo\_config.json", "r", encoding="utf-8") **as** f\_gpt:

generation\_params\_gpt\_neo: Dict[str, Any] = json.load(f\_gpt)

*"""*

*Создаём pipeline для LLaMA-3.2-1B.*

*Аргументы:*

*Нет.*

*Возвращает:*

*Any: Объект pipeline для генерации текста.*

*"""*

llama3\_pipeline: Any = pipeline(

"text-generation",

model="unsloth/Llama-3.2-1B",

device=device\_idx

)

*"""*

*Создаём pipeline для GPT-Neo 125M.*

*Аргументы:*

*Нет.*

*Возвращает:*

*Any: Объект pipeline для генерации текста.*

*"""*

gpt\_neo\_pipeline: Any = pipeline(

"text-generation",

model="EleutherAI/gpt-neo-125m",

device=device\_idx

)

ПРИЛОЖЕНИЕ Б. Примеры работы сценариев бота

Изображение выглядит как текст, снимок экрана, Шрифт, письмо

Автоматически созданное описание

Рис. 15. Риски связанные с KYC

Изображение выглядит как текст, электроника, снимок экрана, веб-страница

Автоматически созданное описание

Рис. 16. HTX

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 17. Bitcoin

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 18. Litecoin

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 19. Зачем нужен KYC