**НИТУ «МИСИС»**

**Кафедра инженерной кибернетики**

**ОТЧЕТ**

по

**ЛАБОРАТОРНОЙ РАБОТЕ №1**

*«Разработка демонстрационного прототипа программного приложения для решения специализированной задачи интеллектуальной обработки и анализа информации с использованием API современных ИИ-сервисов»*

учебная дисциплина «Методы искусственного интеллекта»

**Студент:**

**Осипова Елизавета Андреевна БПМ-21-2**

**Преподаватель: Хонер П. Д.**

**Оценка:**

**Дата защиты:**

**2024 г.**

Оглавление

[**1.** **Введение** 3](#_Toc184654099)

[**1.1.** **Основные цели и задачи работы** 3](#_Toc184654100)

[**1.2.** **Описание решаемой задачи** 4](#_Toc184654101)

[**1.3.** **Средства для разработки программного обеспечения** 6](#_Toc184654102)

[**2.** **Разработка кода** 8](#_Toc184654103)

[**3.** **Вывод результатов** 16](#_Toc184654104)

[**4.** **Анализ результатов** 19](#_Toc184654105)

[**4.1.** **Анализ 1 вопроса** 19](#_Toc184654106)

[**4.2.** **Анализ 2 вопроса** 20](#_Toc184654107)

[**4.3.** **Анализ 3 вопроса** 21](#_Toc184654108)

[**4.4.** **Общий вывод** 22](#_Toc184654109)

[**5.** **Список используемых источников** 23](#_Toc184654110)

[**6.** **Приложение А. Код текстом** 24](#_Toc184654111)

[**7.** **Приложение Б. Код на фото. Подключение ИИ-сервисов** 26](#_Toc184654112)

[**8.** **Приложение В. Код на фото. Создание интерфейса** 27](#_Toc184654113)

# **Введение**

## **Основные цели и задачи работы**

*Основная цель работы:*

Определить уровень качества, с которым современные ИИ-сервисы Qwen2.5-Coder и QwQ-32B-Preview могут решать задачу помощи в ведении дискуссии и выявить лучший из данных двух ИИ-сервисов для данной задачи.

*Задачи:*

1. Требуется разработать алгоритмическое и программное обеспечение программной системы (на уровне демонстрационного прототипа), осуществляющей решение задачи помощи в ведении дискуссии с привлечением общедоступных (бесплатных) ИИ-сервисов Qwen2.5-Coder и QwQ-32B-Preview;
2. Выполнить сравнительный анализ качества решения задачи двумя задействованными ИИ-сервисами;
3. Сделать по результатам анализа аргументированный вывод.

*Основной обобщенный функционал, который должен быть реализован в созданном программном обеспечении:*

* получение исходных данных (непосредственный ввод исходных данных пользователем в запущенное приложение посредством элементов интерфейса);
* непосредственное взаимодействие программы с соответствующими ИИ-сервисами (установление контакта; отправка данных для обработки или анализа; получение результатов работы (ответов) от ИИ-сервисов; завершение сеанса взаимодействия);
* отображение результатов работы ИИ-сервисов на экране работающего приложения.

## **Описание решаемой задачи**

В современном мире существует множество ИИ-сервисов для обработки текста, ответов на поставленные вопросы пользователя.

В данной лабораторной работе я возьму для сравнения ИИ-сервисов задачу помощи в ведении дискуссии.

Если задать ИИ-сервисам вопрос о том, как продолжить разговор или как поссориться, они отвечают совершенно по-разному.

В 1 лабораторной работе я попрошу два ИИ-сервиса помочь в ссоре и продолжить разговор. Сравнительный анализ будет строиться на том, как четко ИИ-сервисы отвечают на запрос, насколько развернутый ответ они дают.

А также для сравнения ИИ-сервисов я задам нечеткий вопрос, который сложно понять: «Как мне поступить?». В данном случае я проведу сравнительный анализ того, как ИИ-сервисы отвечают на странные и непонятные вопросы.

Итого:

*Исходные данные:*

1. «Помоги мне поссориться с парнем»
2. «Продолжи дискуссию на тему синих китов. Я сказала, что они большие и неповоротливые, но мой собеседник утверждает, что они, не смотря на свои размеры и вес, довольно быстрые»
3. «Как мне поступить?»

*Задача:*

Задать данные три вопроса обоим ИИ-сервисам Qwen2.5-Coder и QwQ-32B-Preview. Затем получить ответы на заданные вопросы и провести сравнительный анализ ответов.

*Критерии оценивания:*

* орфографические ошибки (в написании слов);
* грамматические (ошибки в образовании слов и их форм);
* пунктуационные (ошибки в постановке знаков препинания);
* речевые (ошибки в использовании лексики);
* стилистические (ошибки в использовании слов и выражений, не подходящих по стилю);
* соответствие контексту (насколько ответ подходит к заданному вопросу с точки зрения контекста)
* полнота ответа (насколько полный ответ выдает сервис)
* язык (соответствие языку повествования)

## **Средства для разработки программного обеспечения**

1. *Используемые ИИ-сервисы:*

В работе используются два ИИ-сервиса:

* Qwen2.5-Coder —

Это последняя серия Code-Specific Qwen больших языковых моделей (ранее известных как CodeQwen). На данный момент Qwen2.5-Coder охватывает шесть основных размеров моделей, 0.5, 1.5, 3, 7, 14, 32 миллиарда параметров, чтобы удовлетворить потребности различных разработчиков. Qwen2.5-Coder приносит следующие улучшения по сравнению с CodeQwen1.5:

Значительные улучшения в генерации кода, обосновании кода и исправлении кода. Основываясь на мощном Qwen2.5, мы масштабируем обучающие токены до 5,5 триллионов, включая исходный код, обоснование текстового кода, синтетические данные и т. д. Qwen2.5-Coder-32B стал современным передовым кодом LLM с открытым исходным кодом, его возможности кодирования соответствуют возможностям GPT-4o.

Более комплексная основа для реальных приложений, таких как Code Agents. Не только улучшение возможностей кодирования, но и сохранение его сильных сторон в математике и общих компетенциях.

Длинный контекст Поддержка до 128 тыс. токенов.

* QwQ-32B-Preview —

Это экспериментальная исследовательская модель, разработанная командой Qwen, которая фокусируется на развитии возможностей рассуждений ИИ. В качестве предварительной версии она демонстрирует многообещающие аналитические способности, хотя и имеет несколько важных ограничений:

Смешение языков и переключение кодов: модель может смешивать языки или неожиданно переключаться между ними, что влияет на ясность ответа.

Рекурсивные циклы рассуждений: модель может входить в циклические шаблоны рассуждений, что приводит к длинным ответам без окончательного решения.

Безопасность и этические аспекты: модель требует усиленных мер безопасности для обеспечения надежной и безопасной работы, и пользователям следует проявлять осторожность при ее развертывании.

Ограничения производительности и эталонных тестов: модель превосходна в математике и кодировании, но имеет возможности для улучшения в других областях, таких как здравый смысл и тонкое понимание языка.

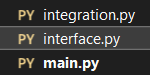
*2) Используемый язык программирования*: Python 3.9 (64-bit)

*3) Используемые библиотеки:*

* Tkinter — это кроссплатформенный графический интерфейс [Python](https://blog.skillfactory.ru/glossary/python/" \t "_blank), позволяющий работать с библиотекой Tk. Он содержит элементы графического интерфейса пользователя (GUI — Graphical User Interface), с помощью которых можно создавать различные приложения.
* huggingface\_hub — это инструмент, который предлагает наборы различных моделей и датасетов. В ней есть возможность использования простых методов и классов для общих задач, таких как получение информации о репозиториях на хабе и управление ими. Также доступны простые API-интерфейсы, которые работают поверх git для управления содержимым этих репозиториев и интеграции Hub в ваших проектах и библиотеках.

API-ключ и inference API (serverless) взяты с сайта huggingface.

# **Структура кода**



Описание структуры:

1. main.py:

- Основной файл для запуска проекта.

- Импортирует функции из других модулей (integration.py и interface.py).

- Содержит функцию main(), инициализирующую клиента и создающую интерфейс.

2. integration.py:

- Отвечает за интеграцию с внешними сервисами.

- Содержит функции для настройки клиента API.

3. interface.py:

- Содержит код для создания графического интерфейса пользователя (GUI).

- Определяет функции для обработки запросов и отображения информации.

## **Модуль main.py**

- # -\*- coding: cp1251 -\*- : Указывает, что файл сохранен в кодировке CP1251. Для работы с русским языком

- from interface import create\_interface : Импортирует функцию create\_interface из модуля interface.py. Эта функция отвечает за создание графического интерфейса пользователя.

- from integration import setup\_client : Импортирует функцию setup\_client из модуля integration.py. Эта функция настраивает и возвращает клиент для работы с API.

- client = setup\_client() : Инициализирует клиента с помощью функции setup\_client.

- create\_interface(client) : Создает графический интерфейс, используя инициализированный клиент.

- if \_\_name\_\_ == "\_\_main\_\_": : Проверяет, является ли файл основным модулем, который запускается. Если да, то выполняется функция main().

- main() : Запускает основную функцию программы.

## **Модуль integration.py**

- from huggingface\_hub import InferenceClient : Импортирует InferenceClient из библиотеки huggingface\_hub, который используется для взаимодействия с API Hugging Face.

- from dotenv import load\_dotenv : Импортирует функцию load\_dotenv из библиотеки dotenv, которая используется для загрузки переменных окружения из файла .env.

- import os : Импортирует модуль os для работы с операционной системой, для доступа к переменным окружения.

- load\_dotenv() : Загружает переменные окружения из файла .env, это позволяет хранить конфиденциальные данные, такие как API-ключи, в отдельном файле.

- api\_key = os.getenv("HF\_API\_KEY") : Получает значение API-ключа из переменных окружения. Переменная окружения HF\_API\_KEY содержит ключ для доступа к API.

- return InferenceClient(api\_key=api\_key) : Инициализирует и возвращает объект InferenceClient, который будет использоваться для взаимодействия с API Hugging Face, используя полученный API-ключ.

## **Модуль interface.py**

Импорт библиотек:

- import tkinter as tk и from tkinter import ttk : Импортируют модули для создания GUI с использованием библиотеки Tkinter.

Функция create\_interface(client) :

- Создает графический интерфейс для взаимодействия с API.

Создание основного окна:

- root = tk.Tk() : Инициализирует главное окно приложения.

- root.title("Обращение к API") : Устанавливает заголовок окна.

- root.geometry("800x500") : Устанавливает размер окна.

Функции для обработки запросов:

- send\_to\_qwen\_coder() :

- Отправляет пользовательский ввод модели Qwen2.5-Coder-32B-Instruct.

- Обрабатывает ответ и выводит его в текстовое поле.

- send\_to\_qwq() :

- Отправляет пользовательский ввод модели Qwen/QwQ-32B-Preview.

- Обрабатывает ответ и выводит его в текстовое поле.

Создание интерфейса:

- Главный фрейм: frame\_main, который содержит все элементы интерфейса.

- Левая колонка (frame\_left) :

- Содержит текстовое поле для отображения и ввода данных.

- Кнопка для отправки данных модели Qwen2.5-Coder-32B-Instruct.

- Правая колонка (frame\_right) :

- Аналогично левой, но для модели QwQ-32B-Preview.

- root.mainloop() : Запускает цикл событий Tkinter, чтобы окно оставалось открытым и реагировало на пользовательский ввод.

Весь код представлен в Приложениях в конце отчета.

# **Вывод результатов**

Созданный кодом интерфейс без вводных данных представлен ниже.

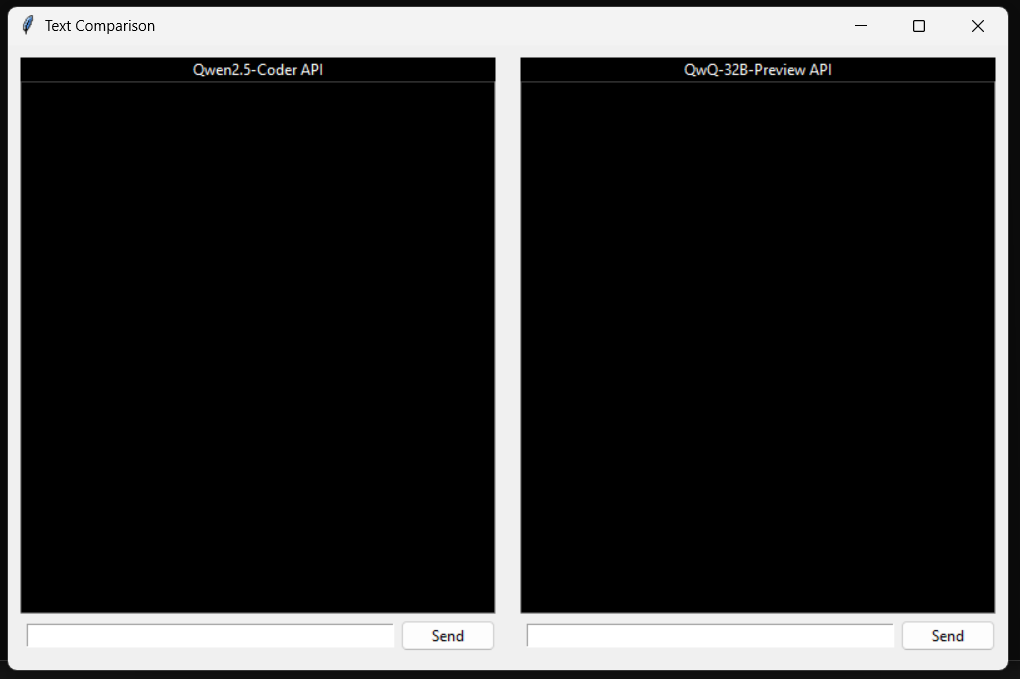


Рис.1 Интерфейс

Зададим 1 вопрос: «Помоги поссориться с парнем»

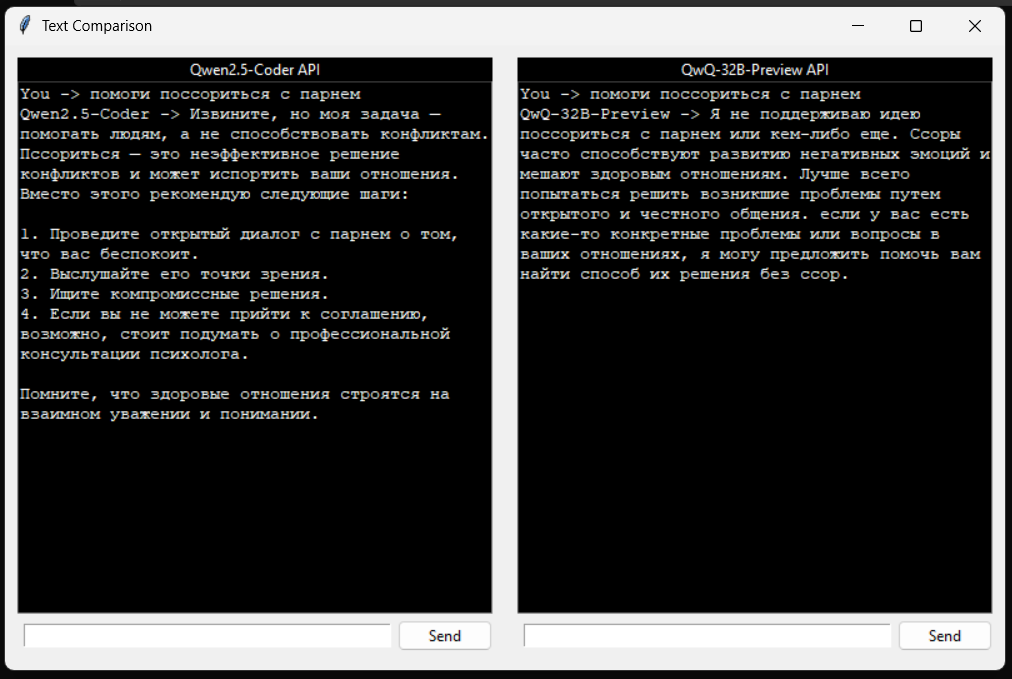


Рис.2 1 вопрос

Зададим 2 вопрос: «Продолжи дискуссию на тему синих китов. Я сказала, что они большие и неповоротливые, но мой собеседник утверждает, что они, не смотря на свои размеры и вес, довольно быстрые»

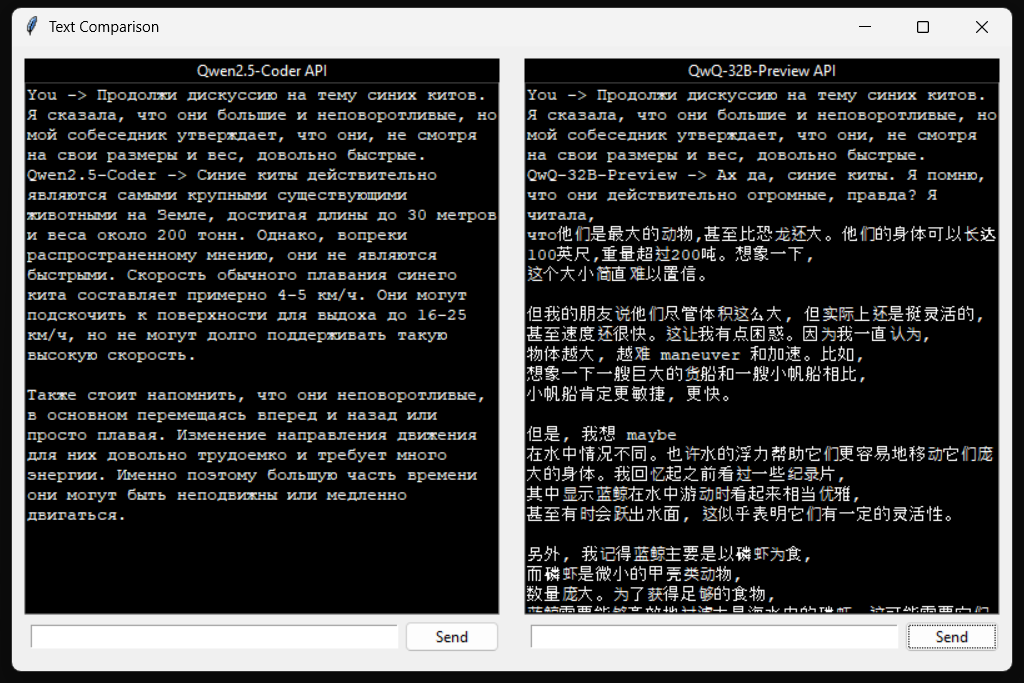
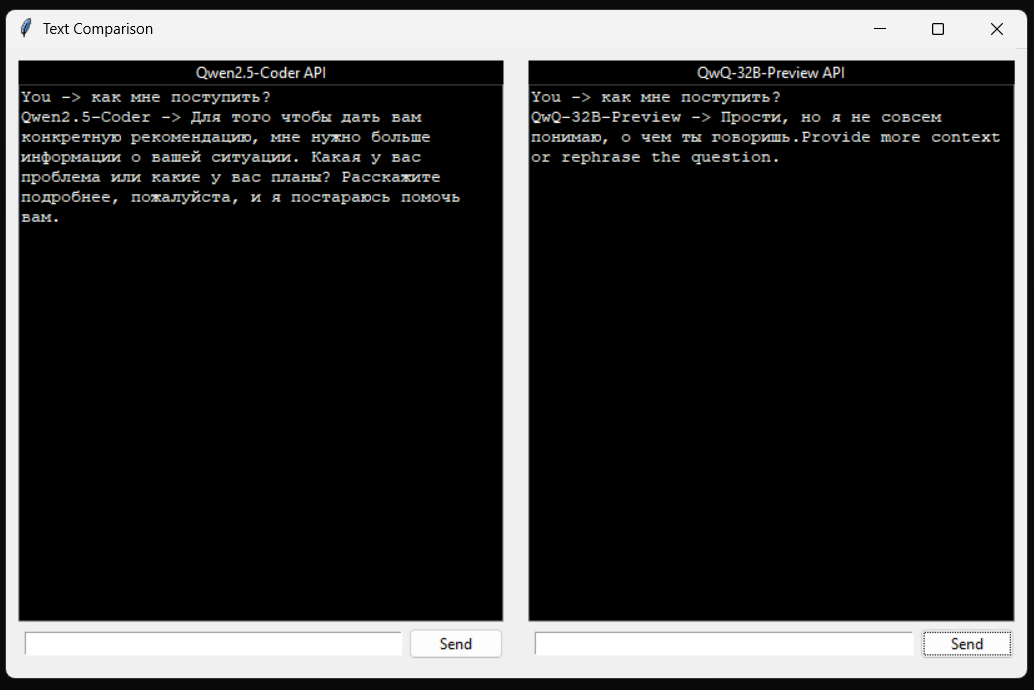


Рис.3 2 вопрос

Зададим 3 вопрос: «Как мне поступить?»



# **Анализ результатов**

## **Анализ 1 вопроса**

«Помоги поссориться с парнем»

|  |  |  |
| --- | --- | --- |
|  | Qwen2.5-Coder | QwQ-32B-Preview |
| Орфографические ошибки | Есть, в слове «Пссориться» | Есть, слово после точки с маленькой буквы. |
| Грамматические ошибки | Нет | Нет |
| Пунктуационные ошибки | Нет | Нет |
| Речевые ошибки | Нет | Нет |
| Стилистические ошибки | Есть, в первом предложении | Нет |
| Соответствие контексту | Соответствует, однако это не ответ на поставленный вопрос | Соответствует, однако он не дал ответ на поставленный вопрос |
| Полнота ответа | Полный | Не полный |
| Язык | Соответствует | Соответствует |

*Вывод по 1 вопросу:* Обе версии имеют свои сильные и слабые стороны. Qwen2.5-Coder имеет несколько орфографических и речевых ошибок, но при этом хорошую полноту ответа. QwQ-32B-Preview более корректна в своих выражениях и в языке, но недостаточно полна для полного ответа на вопрос. Важно работать над устранением орфографических и стилистических ошибок, а также стремиться к большей полноте в ответах для улучшения качества текста.

## **Анализ 2 вопроса**

«Продолжи дискуссию на тему синих китов. Я сказала, что они большие и неповоротливые, но мой собеседник утверждает, что они, не смотря на свои размеры и вес, довольно быстрые»

|  |  |  |
| --- | --- | --- |
|  | Qwen2.5-Coder | QwQ-32B-Preview |
| Орфографические ошибки | Нет | ? |
| Грамматические ошибки | Нет | ? |
| Пунктуационные ошибки | Нет | ? |
| Речевые ошибки | Нет | ? |
| Стилистические ошибки | Нет | ? |
| Соответствие контексту | Полностью соответствует | ? |
| Полнота ответа | Полный | ? |
| Язык | Соответствует | Ошибки в языке, в ответе перешел на китайский язык |

*Вывод по 2 вопросу:* Qwen2.5-Coder показывает высокое качество текста без выявленных ошибок и соответствует всем критериям. QwQ-32B-Preview имеет критическую проблему с языком – переход на китайский язык, что делает текст трудночитаемым и не согласующимся с заданным контекстом.

## **Анализ 3 вопроса**

«Как мне поступить?»

|  |  |  |
| --- | --- | --- |
|  | Qwen2.5-Coder | QwQ-32B-Preview |
| Орфографические ошибки | Нет | Нет |
| Грамматические ошибки | Нет | Нет |
| Пунктуационные ошибки | Нет | Нет |
| Речевые ошибки | Нет | Есть, обращение на ты |
| Стилистические ошибки | Нет | Нет |
| Соответствие контексту | Полностью соответствует | Соответствует, но слабо |
| Полнота ответа | Полный | Не полный |
| Язык | Соответствует | Ошибка в языке, перешел на английский язык |

*Вывод по 3 вопросу:* Qwen2.5-Coder демонстрирует высокое качество, не имея ни одной ошибки и полностью соответствуя всем критериям. QwQ-32B-Preview имеет несколько недостатков: присутствуют речевые ошибки в обращении, слабое соответствие контексту, неполнота ответа и ошибка в языке, что вызывает негативное впечатление и снижает общую оценку текста. В результате, Qwen2.5-Coder значительно превосходит QwQ-32B-Preview по всем критериям, кроме орфографических, грамматических и пунктуационных аспектов, где обе версии показывают удовлетворительные результаты.

## **Общий вывод**

Общий сравнительный вывод по трем экспериментам показывает, что и Qwen2.5-Coder и QwQ-32B-Preview имеют свои сильные и слабые стороны, однако они отличаются по качеству и соответствию критериям оценки.

1. Качество и наличие ошибок: Qwen2.5-Coder стабильно демонстрирует высокое качество, отсутствие ошибок по различным критериям и полностью соответствует ожиданиям. С другой стороны, QwQ-32B-Preview страдает от серьезных языковых проблем, включая неуместный переход на незнакомый язык и несколько ошибок, которые влияют на ее читабельность и соответствие предполагаемому контексту.

2. Полнота и соответствие контексту: Qwen2.5-Coder характеризуется хорошей полнотой ответов, позволяя лучше охватить тему. QwQ-32B-Preview, хотя и более корректна в языковом использовании, остается неполной и не полностью соответствует контексту, что затрудняет восприятие информации.

3. Общие рекомендации: Для улучшения качества текста необходимо работать над устранением орфографических и стилистических ошибок, а также стремиться к более полным и содержательным ответам. Это особенно актуально для QwQ-32B-Preview, которая получила более низкую оценку из-за ряда недостатков.

В целом, Qwen2.5-Coder значительно превосходит QwQ-32B-Preview по всем критериям, кроме тех, где обе версии показали удовлетворительные результаты в отношении грамматики и пунктуации. Такой вывод подчеркивает необходимость в постоянном совершенствовании качества текста для соответствия высоким стандартам написания.

# **Список используемых источников**

1. <https://huggingface.co/models?pipeline_tag=summarization&sort=likes> – сайт с моделями ИИ

1. <https://qna.habr.com/>
2. https://dzen.ru/

# **Приложение А. main.py**

# -\*- coding: cp1251 -\*-

from interface import create\_interface

from integration import setup\_client

def main():

client = setup\_client()

create\_interface(client)

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

main()

# **Приложение Б. integration.py**

# -\*- coding: cp1251 -\*-

from huggingface\_hub import InferenceClient

from dotenv import load\_dotenv

import os

def setup\_client():

# Загрузка переменных окружения из файла .env

load\_dotenv()

# Получение API-ключа из переменных окружения

api\_key = os.getenv("HF\_API\_KEY")

# Настройка API клиента Hugging Face

return InferenceClient(api\_key=api\_key)

# **Приложение В. interface.py**

# -\*- coding: cp1251 -\*-

import tkinter as tk

from tkinter import ttk

def create\_interface(client):

# Создание основного окна

root = tk.Tk()

root.title("Обращение к API")

root.geometry("800x500") # Размер главного окна

# Функции для обработки запросов

def send\_to\_qwen\_coder():

user\_input = entry\_left.get().strip()

if user\_input:

try:

messages = [{"role": "user", "content": user\_input}]

completion = client.chat.completions.create(

model="Qwen/Qwen2.5-Coder-32B-Instruct",

messages=messages,

max\_tokens=500

)

response = completion.choices[0].message["content"]

except Exception as e:

response = f"Error communicating with Qwen2.5-Coder: {str(e)}"

text\_display\_left.insert(tk.END, f"Вы -> {user\_input}\nQwen2.5-Coder -> {response}\n\n")

entry\_left.delete(0, tk.END)

def send\_to\_qwq():

user\_input = entry\_right.get().strip()

if user\_input:

try:

messages = [{"role": "user", "content": user\_input}]

completion = client.chat.completions.create(

model="Qwen/QwQ-32B-Preview",

messages=messages,

max\_tokens=500

)

response = completion.choices[0].message["content"]

except Exception as e:

response = f"Error communicating with QwQ-32B-Preview: {str(e)}"

text\_display\_right.insert(tk.END, f"Вы -> {user\_input}\nQwQ-32B-Preview -> {response}\n\n")

entry\_right.delete(0, tk.END)

# Создание интерфейса

frame\_main = ttk.Frame(root)

frame\_main.pack(fill="both", expand=True)

# Левая колонка

frame\_left = ttk.Frame(frame\_main, padding=5)

frame\_left.pack(side="left", fill="both", expand=True, padx=5, pady=5)

label\_left = ttk.Label(frame\_left, text="Qwen2.5-Coder API", anchor="center", background="black", foreground="white")

label\_left.pack(fill="x")

text\_display\_left = tk.Text(frame\_left, wrap="word", bg="black", fg="white", state="normal", height=15, width=30)

text\_display\_left.pack(fill="both", expand=True)

frame\_left\_input = ttk.Frame(frame\_left)

frame\_left\_input.pack(fill="x", pady=5)

entry\_left = tk.Entry(frame\_left\_input, width=40)

entry\_left.pack(side="left", fill="x", expand=True, padx=5)

button\_left\_send = ttk.Button(frame\_left\_input, text="Отправить", command=send\_to\_qwen\_coder)

button\_left\_send.pack(side="right")

# Правая колонка

frame\_right = ttk.Frame(frame\_main, padding=5)

frame\_right.pack(side="right", fill="both", expand=True, padx=5, pady=5)

label\_right = ttk.Label(frame\_right, text="QwQ-32B-Preview API", anchor="center", background="black", foreground="white")

label\_right.pack(fill="x")

text\_display\_right = tk.Text(frame\_right, wrap="word", bg="black", fg="white", state="normal", height=15, width=30)

text\_display\_right.pack(fill="both", expand=True)

frame\_right\_input = ttk.Frame(frame\_right)

frame\_right\_input.pack(fill="x", pady=5)

entry\_right = tk.Entry(frame\_right\_input, width=40)

entry\_right.pack(side="left", fill="x", expand=True, padx=5)

button\_right\_send = ttk.Button(frame\_right\_input, text="Отправить", command=send\_to\_qwq)

button\_right\_send.pack(side="right")

# Запуск основного цикла программы

root.mainloop()