Н И Т У « М И С и С »

Кафедра инженерной кибернетики

ОТЧЕТ

ПО

ЛАБОРАТОРНОЙ РАБОТЕ №1

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

учебная дисциплина «Методы искусственного интеллекта»

Группа: БПМ-21-3

Учащийся: Прокуденко К. И.

Преподаватель: Хонер П.Д.

Оценка: 5

Дата защиты: 02.10.24

2024 г.

1. Введение

В данной лабораторной работе целью является обрести навыки взаимодействия на уровне программного кода между собственным проектом и функционалом существующих ИИ-сервисов нового поколения при решении конкретной задачи интеллектуальной обработки информации.

Также необходимо сравнить несколько сервисов по выполнению задачи одного типа и выявить лучший из использованных ИИ-сервисов для её решения.

2. Краткое описание задачи

В качестве задачи возьмём процесс перевода текста. Для оценки качества перевода будем переводить текст с русского на другие языки после чего переведём текст обратно на русский и оценим насколько сохранился первоначальный смысл. В качестве языковой последовательности для теста возьмём: русский – английский – французский – немецкий – английский – испанский – казахский – английский – русский.

3. Выбранные средства для разработки ПО

Изначально для выполнения работы планировалось взять с Rapid api 2 популярных сервиса для перевода: Microsoft Translator и Yandex Translate, но в процессе тестирование сервис от Яндекс зависал на этапе запроса, при этом код не падал и не выдавал ошибки. Поэтому после нескольких попыток было принято решение сменить его на Google Translate. Также сервисы выбирались с учётом того, чтобы у них было достаточное доступное количество запросов для тестирования.

Программа разрабатывалась в VS Code на языке Python. Для работы с запросами использовались библиотеки http.client и json. Для настройки правильной кодировки для всех языков понадобилась библиотека sys. Для визуального интерфейса была взята библиотека tkinter.

4. Исходные данные

В качестве исходных данных возьмём несколько видов текстов на русском языке.

- **Текст из книги:** *«Пусть болтает. Мне нужен отдых, долгий отдых, я же тебе говорил. Бессрочный отдых: едва ли я сюда когда-нибудь вернусь. Да и незачем, все устроено… Постарел я, Гэндальф. Так-то вроде не очень, а кости ноют. Нечего сказать: «Хорошо сохранился!» – Он фыркнул. – Ты понимаешь, я тонкий-претонкий, как масло на хлебе у скупердяя. Скверно это. Надо как-то переиначивать жизнь. Гэндальф не сводил с него пристального, озабоченного взгляда.»*.

- **Техническая документация:** *«Для создания графического окна применяется конструктор Tk(), который определен в модуле tkinter. Создаваемое окно присваивается переменной root, и через эту переменную мы можем управлять атрибутами окна. В частности, с помощью метода title() можно установить заголовок окна.»*.
- **Произвольный набор слов:** «солнце дерево река гора море небо ветер цветок дождь снег лес друзья семья книга музыка фильм город дом улица свет тень птица животное работа путешествие еда вода время счастье печаль улыбка мечта грустный радость любовь вдохновение искусство сказка звезда ночь день песня светлый темнота праздник встреча забота тепло осень зима весна лето».

5. Результат

Сначала введём функции для перевода текста с одного языка на другой через Microsoft Translator и Google Translate.

```python
# Функция для теста Microsoft Translator
def testMicrosoftAPI(textToTranslate:str, fromLanguage:str, toLanguage:str)->str:
    conn = http.client.HTTPSConnection("microsoft-translator-text.p.rapidapi.com")

    headers = {
        'content-type': "application/json",
        'x-rapidapi-key': "e3b15f62f3mshcdfd2a7a42b9a8ep102506jsn6f3828de672f",
        'x-rapidapi-host': "microsoft-translator-text.p.rapidapi.com"
    }

    body = json.dumps([{"Text": textToTranslate}])
    params = f"/translate?api-version=3.0&from={fromLanguage}&to={toLanguage}"

    # Выполняем запрос
    conn.request("POST", params, body, headers)
    res = conn.getresponse()
    data = res.read()
    translated_text = json.loads(data.decode("utf-8"))

    result = translated_text[0]['translations'][0]['text']
    return result
```

Рис. 1 – Функция для теста Microsoft Translator.

```python
# Функция для теста Google Translate
def testGoogleAPI(textToTranslate:str, fromLanguage:str, toLanguage:str)->str:
    conn = http.client.HTTPSConnection("google-translator9.p.rapidapi.com")

    payload = json.dumps({
        "q": textToTranslate,
        "source": fromLanguage,
        "target": toLanguage
    })

    headers = {
        'x-rapidapi-key': "e3b15f62f3mshcdfd2a7a42b9a8ep102506jsn6f3828de672f",
        'x-rapidapi-host': "google-translator9.p.rapidapi.com",
        'Content-Type': "application/json"
    }

    # Выполняем запрос
    conn.request("POST", "/v2", payload, headers)
    res = conn.getresponse()
    data = res.read()
    translated_text = json.loads(data.decode("utf-8"))

    result = translated_text['data']['translations'][0]['translatedText']
    return result
```

Рис. 2 – Функция для теста Google Translate.

Далее введём функцию которая прогонят переданный текст через последовательность заданных языков используя переданную функцию.

```python
# Функция для тестирования
def testBySequence(textToTranslate:str, sequence:list, translateFunc: Callable[[str, str, str], str]) -> str:
    tmpTextToTranslate = textToTranslate

    for lanIndex in range(1, len(sequence)):
        tmpTextToTranslate = translateFunc(tmpTextToTranslate, sequence[lanIndex - 1], sequence[lanIndex])
        print(f"{sequence[lanIndex - 1]} -> {sequence[lanIndex]}: {tmpTextToTranslate}")
    return tmpTextToTranslate
```

Рис. 3 – Функция для тестирование различных переводчиков.

Для запуска сравнения переводчиков введём функцию обработки введённого текста, которая будет вызываться при нажатии на кнопку.

```python
def runTests() -> None:
    input_text = text_input.get("1.0", tk.END).strip()
    if not input_text:
        messagebox.showwarning("Предупреждение", "Введите текст в первое окно.")
        return

    languageSequence = ["ru", "en", "fr", "de", "en", "es", "kk", "en", "ru"]

    # Получение результатов тестов
    microsoft_result = testBySequence(input_text, languageSequence, testMicrosoftAPI)
    google_result = testBySequence(input_text, languageSequence, testGoogleAPI)

    # Вывод результатов в соответствующие окна
    result_microsoft.config(state=tk.NORMAL)
    result_microsoft.delete("1.0", tk.END)
    result_microsoft.insert(tk.END, microsoft_result)

    result_google.config(state=tk.NORMAL)
    result_google.delete("1.0", tk.END)
    result_google.insert(tk.END, google_result)
```

Рис. 4 – Функция обработки пользовательского ввода.

Для тестирования программы на предварительно подготовленных данных введём функцию которая будет запускать 1 случайный тест из 3 текстов.

```python
def runPreTests() -> None:
    bookText = " Пусть болтает. Мне нужен отдых, долгий отдых, я же тебе говорил. \
    Бессрочный отдых: едва ли я сюда когда-нибудь вернусь. Да и незачем, все устроено… \
    Постарел я, Гэндальф. Так-то вроде не очень, а кости ноют. \
    Нечего сказать: «Хорошо сохранился!» Он фыркнул. Ты понимаешь, я тонкий-претонкий, как масло на хлебе у скупердяя. 
    Скверно это. Надо как-то переиначивать жизнь. \
    Гэндальф не сводил с него пристального, озабоченного взгляда."
    docText = "Для создания графического окна применяется конструктор Tk(), который определен в модуле tkinter.\
    Создаваемое окно присваивается переменной root, и через эту переменную мы можем управлять атрибутами окна. \
    В частности, с помощью метода title() можно установить заголовок окна."
    randomText = "солнце дерево река гора море небо ветер\
    цветок дождь снег лес друзья семья книга музыка фильм город дом\
    улица свет тень птица животное работа путешествие еда вода\
    время счастье печаль улыбка мечта грустный радость любовь\
    вдохновение искусство сказка звезда ночь день песня светлый темнота\
    праздник встреча забота тепло осень зима весна лето"
    num = random.randint(1, 3)
    textSamples = {
        1: bookText,
        2: docText,
        3: randomText
    }

    text_input.delete("1.0", tk.END)
    text_input.insert(tk.END, textSamples[num])
    runTests()
```

Рис. 5 – Функция запуска подготовленных тестов.

Далее рассмотрим примеры работы программы.

Рис. 6 – Результат запуска предварительного теста со случайными словами.

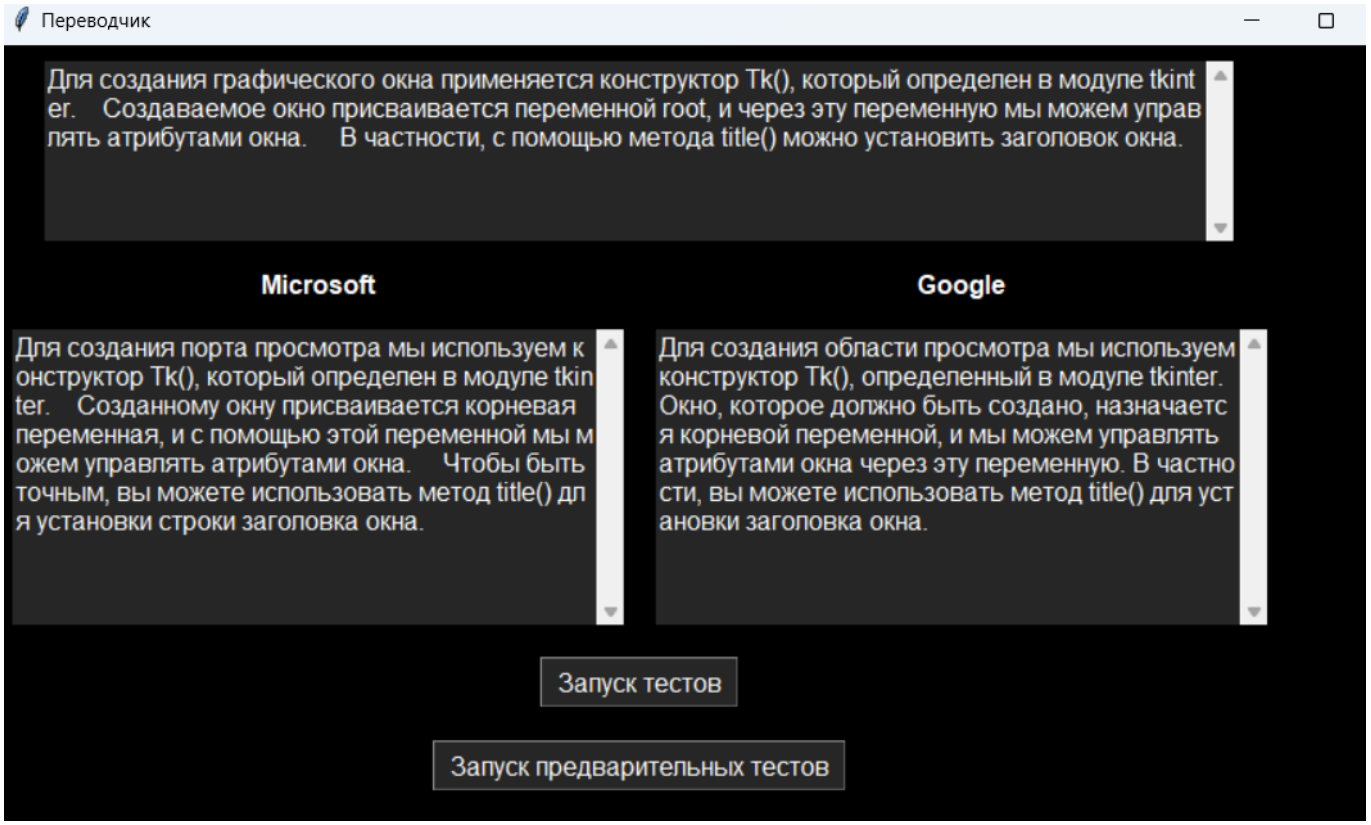

Рис. 7 – Результат запуска предварительного теста с документацией.

.

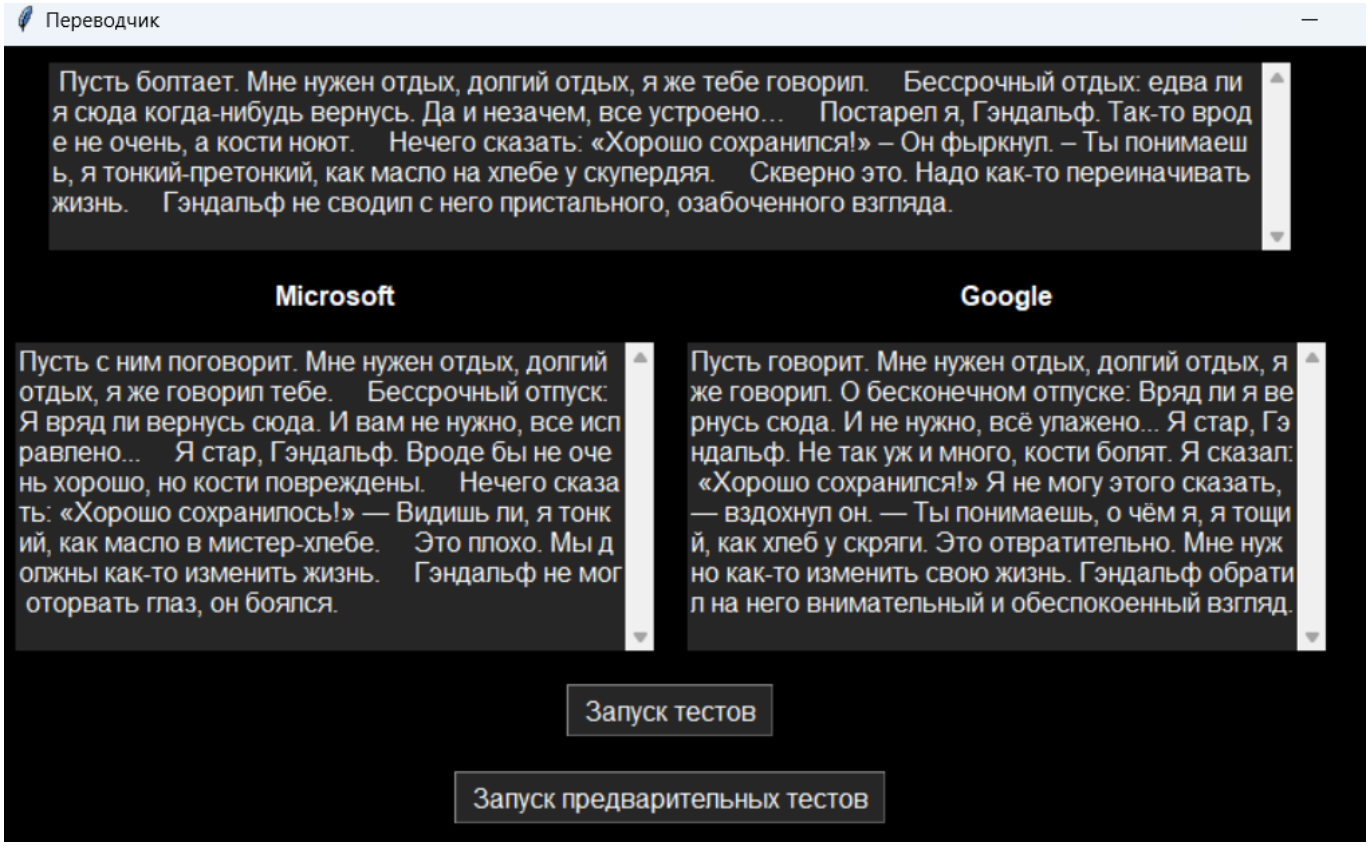

Рис. 8 – Результат запуска предварительного теста с отрывком из книги.

Рис. 9 – Результат работы программы с произвольным текстом.

6. Вывод

В ходе выполнения данной лабораторной работы я научился интегрировать сторонние API в свой программный код и написал программу для сравнения сохранности смысла текста после перевода Microsoft Translator и Google Translate. Как видно из результатов работы программы, Google Translate лучше переводит текст и сохраняет первоначальный смысл на всех тестах. В то время как Microsoft Translator может заметно исказить текст, после чего смысл часто становится отличным от первоначального.

7. Список источников

1) Документация по tkinter для Python // URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 01.10.24).
2) Api Microsoft Trasnlator // URL: https://rapidapi.com/microsoft-azure-org-microsoft-cognitive-services/api/microsoft-translator-text/playground (дата обращения: 01.10.24).
3) Api Google Translate // URL: https://rapidapi.com/googlecloud/api/google-translate1/playground/ (дата обращения: 01.10.24).