ТЕМА РАЗДЕЛА 7.2

**Разработка системы автоматической проверки учебного задания**

**Вступление**

В ходе стажировки выполнялась учебная задач по автоматизации проверки задания учащихся на примере формирования автоматической оценки эссе по заданным критериям. За время занятий были реализованы следующие задачи:

1. Краткий обзор предметной области (техн. программирования, LLM, способы решения подобных задач)

2. Определились с критериями оценивания. Протестировали запросы к LLM GigaChat

3. Разработали и реализовали программные алгоритмы для решения поставленной задачи

4. Протестировали полученную программу

5. Собрали полученный код в отдельный модуль (ПРИЛОЖЕНИЕ 1)

**Ход выполнения**

1. Провели краткий обзор предметной области по методам и инструментам решения.

Программирование – процесс создания программ для ЭВМ. В широком смысле может включать в себя все этапы от проектирования до выпуска программы в работу. Языки программирования, в узком смысле, это инструмент «общения» с ЭВМ, а в широком это инструмент для кодирования алгоритмов. Для эффективного использования технологии программирования существуют среды разработки. Так VC и PyCharm – это интегрированные среды разработки (IDE). IDE по сути представляют собой набор инструментов, объединенных в одну программную систему для обеспечения процесса разработки. IDE удобны для работы над проектами. Основная область применения :

• Для исследования

• Обучения

• Создания прототипов или MVP(минимального функционирующего продукта) Ключевое отличие между классическим исполнением программы и интерактивным в том, что при классическом код пишется заранее, а далее компилируется/интерпретируется и потом исполняется в соответствии с точкой входа (так например в VC и PyCharm это точка входа либо указывается, либо ищется по умолчанию и исполняется интерпретатором Python), а для интерактивного происходит запуск ядра, после которого возможен процесс дальнейшего написания кода. Файлы «.py» - скрипты, модули, файлы библиотек и т.д. (это файлы с кодом Python), а файлы «.ipynb» - это рабочие блокноты, которые используются для интерактивного взаимодействия путем исполнения отдельных ячеек кода. Иными словами «.py» готовые схемы (код исполнения должен быть написал заранее), а «.ipynb» чертеж ( код исполнения пишется в процессе и может исполнятся минимальными единицами –ячейками).

LLM для программирования (кодирования) Большие языковые модели (LLM) способны генерировать простой и работоспособный программный код на различных языках программирования. При этом давая пояснения по его работе и способов запуска на исполнение. Данный функционал полезен для:

•Создания прототипов

•Исследований

•Небольших проектов

•Обучения

•Поиска идей реализации

Особенно полезным видится применения для кодирования рутинных операций или относительно быстрого получения рабочего варианта алгоритма/подзадачи/скрипта/модуля. Существуют специализированные LLM для программирования, называемые ассистенты разработки. Тем не менее значимо ускорить процесс разработки на данном этапе получается только для программистов начальной и средней квалификации. Так как даже для кода полученного от специализированных LLM требуется проверка и дальнейшая коррекция. Мы будем использовать LLM для решения подзадач программы (реализации отдельного алгоритма). А именно, для каждой единицы функционала разрабатываемой программы произведем постановку задачи в форме «вход-выход», а далее добьемся соответствия с полученным от LLM кодом.

2. Выбрали критерии для оценивания эссе, это: "ПЕРЕЧИСЛИТЬ ВЫБРАННЫЕ КРИТЕРИИ", а также наложили ограничения "ПЕРЕЧИСЛИТЬ". Реализовали запрос к Giga Chat.

3. Рассмотрели постановку задачи, тем самым формализовали входные и выходные данные для программной системы. Это позволило перейти к разработке алгоритмов и их непосредственной реализации. Было выделены три подзадачи:

1. Оценка эссе
2. Сбор статистики по тексту
3. Формирование отчета

Для оценивания эссе реализовали запрос к Giga Chat через API из программного кода. Для статистики и отчета использовали запросы к Deep Seek.

4. Протестировали итоговые подмодули. Результат соответствовал постановке задачи

6. Выделили полезный код из блокнота и создали отдельный модуль в виде файла-скрипта Python(esse.py). Файл можно интегрировать как в другой проект, так и использовать в виде самостоятельного решения.

ЗАКЛЮЧЕНИЕ

**Достигнутые результаты**:

1. Создана работоспособная система автоматической проверки эссе
2. Обеспечена согласованность оценок
3. Сокращено время проверки с 15-20 мин до 3-5 сек

**Перспективы развития**:

* Добавление мультиязычной поддержки
* Адаптация для других дисциплин (история, философия)
* Внедрение механизма апелляций

Система успешно интегрирована в образовательный процесс через Telegram-бота, что подтверждает ее практическую применимость.

Минусы: на данный момент система не может проверять рукописные работы по математике, русскому языку.

ПРИЛОЖЕНИЕ 1

1. **Файл esse.ру**

Импорт библиотек

1. !pip install langchain-gigachat
2. !pip install -U langchain-community
3. !pip install nltk pymorphy3
4. !pip install reportlab

Тело программы

# for GigaChat

from langchain.schema import HumanMessage, SystemMessage

from langchain.chat\_models.gigachat import GigaChat as GigaChatLc

#for diagramm

import re

from collections import Counter

import matplotlib.pyplot as plt

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import pymorphy3

import nltk

# Загрузка необходимых ресурсов NLTK

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('punkt\_tab')

#for report

import matplotlib.pyplot as plt

from reportlab.lib.pagesizes import letter

from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.lib.units import inch

from reportlab.pdfbase import pdfmetrics

from reportlab.pdfbase.ttfonts import TTFont

from datetime import datetime

# класс для API

# sys\_promt - директива для модели, user\_esse - текст работы , auth - ключ авторизации

def giga\_api(sys\_promt, user\_esse, auth):

  giga = GigaChatLc(credentials=auth,

                  model='GigaChat:latest',

                  verify\_ssl\_certs=False

                  )

  sys\_promt = SystemMessage(content=sys\_promt)

  user\_esse = HumanMessage(content=user\_esse)

  answer = giga([sys\_promt,

                  user\_esse])

  #print(answer.content)

  #print(answer.response\_metadata)

  return answer

def preprocess\_text(text):

    # Приведение к нижнему регистру

    text = text.lower()

    # Удаление пунктуации и цифр

    text = re.sub(r'[^\w\s]', '', text)

    text = re.sub(r'\d+', '', text)

    # Токенизация

    tokens = word\_tokenize(text, language='russian')

    # Удаление стоп-слов

    stop\_words = set(stopwords.words('russian'))

    tokens = [word for word in tokens if word not in stop\_words and len(word) > 1]

    # Лемматизация с pymorphy2

    morph = pymorphy3.MorphAnalyzer()

    lemmatized = [morph.parse(word)[0].normal\_form for word in tokens]

    return lemmatized

def save\_word\_frequency(words):

    # Подсчет частотности слов

    top\_n=20

    word\_counts = Counter(words)

    most\_common = word\_counts.most\_common(top\_n)

    # Подготовка данных для визуализации

    words, counts = zip(\*most\_common)

    # Создание диаграммы

    plt.figure(figsize=(12, 8))

    plt.bar(words, counts, color='skyblue')

    plt.xlabel('Слова', fontsize=12)

    plt.ylabel('Частота', fontsize=12)

    plt.title(f'Топ-{top\_n} самых частых слов', fontsize=14)

    plt.xticks(rotation=45, ha='right', fontsize=10)

    plt.yticks(fontsize=10)

    plt.grid(axis='y', linestyle='--', alpha=0.7)

    plt.tight\_layout()

    plt.savefig("plot.png")

# text - строка с текстом для построения диаграммы

def dia\_word(text):

  processed\_words = preprocess\_text(text)

  save\_word\_frequency(processed\_words)

# msg - ответ модели, path\_pic -путь к диграмме, path\_ttf-путь к шрифту

def my\_report(msg, path\_pic, path\_ttf):

  pdfmetrics.registerFont(TTFont('Arial', path\_ttf))  # Укажите путь к файлу шрифта

  # 2. Настраиваем стили с использованием зарегистрированного шрифта

  styles = getSampleStyleSheet()

  styles['Normal'].fontName = 'Arial'  # Указываем шрифт для основного текста

  styles['Title'].fontName = 'Arial'   # И для заголовка

  # 3. Создаём PDF

  filename = datetime.now().strftime("file\_%Y-%m-%d\_%H-%M-%S.pdf")

  doc = SimpleDocTemplate(filename, pagesize=letter)

  story = []

  # Добавляем заголовок (теперь с кириллицей)

  title = Paragraph("<b>Оценка эссе</b>", styles['Title'])

  story.append(title)

  story.append(Spacer(1, 0.5 \* inch))

  # Добавляем комментарий эксперта (русский текст)

  expert\_comment = msg

  story.append(Paragraph(expert\_comment, styles['Normal']))

  story.append(Spacer(1, 0.3 \* inch))

  # Добавляем техническую информацию

  tech\_info = """

  <b>Технические детали:</b><br/>

  - PDF сгенерирован через reportlab

  """

  story.append(Paragraph(tech\_info, styles['Normal']))

  story.append(Spacer(1, 0.3 \* inch))

  # Добавляем график

  img = Image(path\_pic, width=5\*inch, height=3\*inch)

  story.append(img)

  # Сохраняем PDF

  doc.build(story)

  print("PDF создан: report.pdf")

  return filename

# Пример использования

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

  test\_pr = """Ты — профессор литературы

  ## Задача

  Оценивать эссе по 5 критериям

  - Аргументация

  - Раскрытие темы

  - Грамотность

  - Структура

  - Оригинальность

  ## Формат ответа

  Краткое резюме по полученному эссе - не более 5 предложений.

  Дополнительно выставление баллов от 0 до 10 по всем критериям и средним за все (от 0 до 5)

  ## Особые указания

  Сигнализировать если:

  1) эссе менее 250 символов"""

  test\_es = """ Каким должен быть воспитатель XXI века? Размышляя над этим вопросом, можно

  сказать, что он ничем не отличается от воспитателя XIX и XX веков. Да, воспитатель

  должен идти в ногу со временем. Безусловно, необходимо, чтобы он владел новейшими

  технологиями, готов и умеет непрерывно учиться, способен к ответственным решениям и

  прогнозированию возможных последствий, умеет общаться и сотрудничать, исполнителен

  и продуктивен в выполнении разного рода задач. Он признаёт свободу и толерантность,

  ответственен за себя, семью, коллектив, страну. Воспитатель физически и психически

  выдержан, умеет отдыхать, ведёт здоровый образ жизни.

  Это так, но в первую очередь воспитатель, как и во все времена должен быть

  образцом духовности, воспитанности, порядочным и честным, добрым и справедливым

  человеком. На мой взгляд, эти главные качества воспитателя были и остаются самыми

  необходимыми.

  Бесспорно, труд воспитателя тяжел. Нелегко быть образцом для подражания,

  эталоном порядочности, советчиком, судьей, наставником, быть творцом детской души!

  Но это приятная радостная тяжесть, потому что в основе ее лежит любовь. На курсах

  повышения квалификации нас учили как работать над собой, не во вред себе, своему

  душевному, психическому здоровью. И говорили: «Нельзя любить сильно своих

  воспитанников, это отразится на вашем здоровье». Задумываясь над этой фразой,

  возникает вопрос: «А как можно работать воспитателем и не любить детей?» Ведь рядом

  со мной дети разные, не похожие друг на друга, веселые и любознательные, робкие и

  застенчивые, дерзкие и неугомонные, «молчуны» и «болтушки». И как приятно слышать,

  когда к тебе подходит ребенок и начинает свой диалог со словами: «Мама…, ой, Ксения

  Сергеевна…», а на душе становится так тепло и радостно. Потому что дети чувствуют, я

  люблю их такими, какие они есть. Детский сад и группа, в которой они воспитываются

  для них как второй дом и семья. Главное – создать благоприятную атмосферу для этого.

  Объясняя малышам простые, но удивительные вещи, рассказываю, как важно быть

  добрым и честным, любить себя и своих близких. Очень важно хвалить ребенка, даже

  тогда, когда его успехи очень скромны. Это воспитывает у детей желание сделать

  следующий шаг.

  Я глубоко убеждена, что я выбрала правильный в жизни путь – профессию

  воспитателя. Работая по зову сердца, я могу сказать – профессии лучше моей нет!  """

  #путь к шрифту, например в одной директори с файлом есть файл TIMES.TTF

  path\_ttf = '/usr/share/fonts/truetype/liberation/LiberationMono-Regular.ttf'

  #путь к изображению

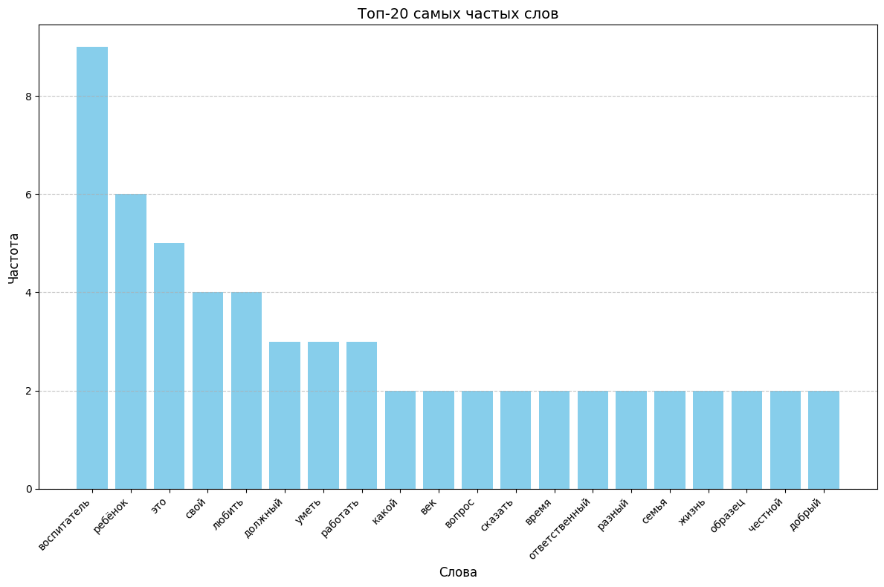
  path\_pic = 'plot.png'

  auth = open('gpt\_api.txt').read()

  answer = giga\_api(test\_pr, test\_es, auth)

  dia\_word(test\_es)

  my\_report(answer.content,path\_pic, path\_ttf)



1. **Файл requset**

Установка библиотек

!pip install gigachat

Код программы

from gigachat import GigaChat

import ssl

def gpt\_request(text):

    ssl\_c = ssl.create\_default\_context()

    ssl\_c.check\_hostname = False

    ssl\_c.verify\_mode = ssl.CERT\_NONE

    giga = GigaChat(

        credentials = open('gpt\_api.txt').read(),

        scope = 'GIGACHAT\_API\_PERS',

        model = 'Gigachat',

        verify\_ssl\_certs = False

    )

    answer = giga.chat(text)

    return (answer.choices[0].message.content)

1. **Файл main.py**
2. import telebot
3. from telebot import types
4. import os
5. from datetime import datetime
6. #filename = datetime.now().strftime("file\_%Y-%m-%d\_%H-%M-%S.txt")
7. #from request import gpt\_request
8. #from esse import giga\_api, dia\_word, my\_report
9. #or
10. #from esse import \*
11. def giga(message):
12. bot.send\_message(message.chat.id,gpt\_request(message.text))
13. return
14. def giga\_esse(message):
15. my\_sys\_promt ="""Ты — профессор литературы
16. ## Задача
17. Оценивать эссе по 5 критериям
18. - Аргументация
19. - Раскрытие темы
20. - Грамотность
21. - Структура
22. - Оригинальность
23. ## Формат ответа
24. Краткое резюме по полученному эссе - не более 5 предложений.
25. Дополнительно выставление баллов от 0 до 10 по всем критериям и средним за все (от 0 до 5)
26. ## Особые указания
27. Сигнализировать если:
28. 1) эссе менее 250 символов"""
29. ##answer = giga\_api(test\_pr, test\_es, auth)
30. ##dia\_word(test\_es)
31. ##my\_report(answer.content,path\_pic, path\_ttf)
32. #auth =  open('gpt\_api.txt').read()
33. answer = giga\_api(my\_sys\_promt, message.text, open('gpt\_api.txt').read())
34. dia\_word(message.text)
35. fname = my\_report(answer.content, 'plot.png', '/usr/share/fonts/truetype/liberation/LiberationMono-Regular.ttf')
36. user\_id =  message.from\_user.id
37. new\_name = str(user\_id) + fname
38. os.rename(fname,new\_name)
39. os.rename(new\_name,'reports/'+new\_name)
40. bot.send\_message(message.chat.id,answer.content)
41. #
42. return
43. bot = telebot.TeleBot(open('api.txt').read()) #Инициализация бота
44. btn1 = types.KeyboardButton('Расписание') #создани   е кнопки
45. btn2 = types.KeyboardButton('ДЗ')
46. btn3 = types.KeyboardButton('Фото')
47. btn4 = types.KeyboardButton('Вопрос GigaChat')
48. btn5 = types.KeyboardButton('Проверка Эссе')
49. markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)#сетка для кнопок
50. markup.add(btn1,btn2, btn3,btn4,btn5)
51. @bot.message\_handler(commands=['start']) #определение реакции бота на /start
52. def send\_wecome(message): #функци реакции на /start
53. bot.reply\_to(message, "Привет! Я учебный бот") #тело программы
54. bot.send\_message(message.chat.id, "Выберите действие:",
55. reply\_markup=markup)
56. @bot.message\_handler(func=lambda message:True) #блок обработки текстовых сообщений
57. def handle\_buttons(message): #в эту фун-ю добавляем обработку текста через elif
58. if message.text == 'Расписание':
59. ph = open('raspisanie\_23.jpg','rb') #путь к фото, тип чтения. rb - читать файл
60. bot.send\_photo(message.chat.id,ph,'Раписание')
61. url = 'https://sh23-irkutsk-r138.gosweb.gosuslugi.ru/netcat\_files/userfiles/2/Moya\_papka/raspisanie\_23.jpg'
62. bot.send\_photo(message.chat.id,url,'Раписание')
63. # bot.reply\_to(message, "Сейчас лето, занятий нет.")
64. # inline\_markup = types.InlineKeyboardMarkup() #создание "шаблона" для инлайн копки
65. # btn = types.InlineKeyboardButton( #текст и ссылка для кнопки и её инициализация
66. #     text="Летние активности",
67. #     url="https://leto.mos.ru/"
68. # )
69. # inline\_markup.add(btn) #добавление кнопки в "шаблон" для инлайн кнопок
70. # bot.send\_message(message.chat.id, "Лучше посмотри летние активности",
71. #                  reply\_markup=inline\_markup) #отправка сообщения пользователю с кнопкой
72. elif message.text == 'ДЗ':
73. doc = open('Это домашнее задание.pdf','rb')
74. bot.send\_document(message.chat.id, doc,caption='ДЗ',
75. visible\_file\_name='Абракадабра.pdf')
76. # bot.reply\_to(message, "У вас каникулы, а всё ДЗ у учителей.")
77. elif str(message.text).lower() == 'привет':
78. bot.reply\_to(message, "Привет!")
79. elif message.text == 'Фото':
80. try:
81. ph = open('name\_file.jpg','rb') #путь к фото, тип чтения. rb - читать файл
82. bot.send\_photo(message.chat.id,ph,'Ваше последнее фото')
83. except BaseException:
84. bot.reply\_to(message, "Фото отсуствует, отправьте новое.")
85. elif message.text == 'Вопрос GigaChat':
86. msg = bot.reply\_to(message, "Напиши текст запроса для языковой модели")
87. bot.register\_next\_step\_handler(msg, giga)
88. elif message.text == 'Проверка Эссе':
89. msg = bot.reply\_to(message, "Напиши текст эссе")
90. bot.register\_next\_step\_handler(msg, giga\_esse)
91. @bot.message\_handler(content\_types=['photo'])
92. def photoes(message):
93. file\_id = message.photo[-1].file\_id #из полученного сообщения берём фото.
94. # ИД хранится в последнем элементе с помощью обращения к нему мы получаем file id
95. file\_info = bot.get\_file(file\_id) #получение информации о самом файле по его ID
96. download\_file = bot.download\_file(file\_info.file\_path) #загузка файла в оперативную память
97. with open('name\_file.jpg', 'wb') as new\_f: #сохранение файла
98. new\_f.write(download\_file)
99. bot.reply\_to(message,'Фото сохранено') #отправка уведомления пользователю
100. bot.polling() #отправка "настроек" в бот и его активация.
101. # Без него бот неактивен