Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра информационной безопасности

Направление подготовки **09.03.01 Информатика и вычислительная техника**

Направленность(профиль) **Автоматизированные системы обработки информации**

Факультет **Физики, математики, информатики**

Дисциплина **Системы искусственного интеллекта**

Форма обучения **очная**

**ОТЧЕТ**

по лабораторной работе №3

Выполнил: студент 2 курса Красников Я.И.  
Проверил: ассистент кафедры ИБ Юрьев И. А.

Курск – 2025

**Лабораторная работа № 3**

1. Объясните, что делает код в приложении. Прокомментируйте, что делает каждая описанная функция
2. filter(text):
   * Приводит текст к нижнему регистру
   * Оставляет только русские буквы, пробелы и дефисы
   * Удаляет все остальные символы
   * Это подготовка текста для дальнейшего анализа
3. match(text, example):
   * Сравнивает введённый текст с примером из базы знаний
   * Использует расстояние Левенштейна (редакционное расстояние) для определения схожести строк
   * Если нормализованное расстояние меньше 0.4, считает, что строки совпадают
4. get\_intent(text):
   * Определяет намерение пользователя
   * Перебирает все намерения из BOT\_CONFIG и их примеры
   * Если находит совпадение через функцию match, возвращает соответствующее намерение
5. get\_answer\_by\_intent(intent):
   * Получает случайный ответ из списка ответов для данного намерения
6. bot(text):
   * Основная функция бота
   * Пытается определить намерение
   * Если намерение найдено, возвращает соответствующий ответ
   * Иначе возвращает случайную "заглушку" из failure\_phrases
7. Для чего в коде используется библилтека nltk?

Библиотека NLTK (Natural Language Toolkit) используется для вычисления редакционного расстояния между строками через функцию edit\_distance(). Это помогает определять, насколько похож ввод пользователя на примеры фраз в конфигурации бота.

1. Что такое intents в BOT\_CONFIG?  Добавьте ещё один intents (по 4 примера и 4 ответа)

Intents (намерения) в BOT\_CONFIG - это категории запросов/фраз пользователя, которые бот может распознать. Для каждого намерения задаются:

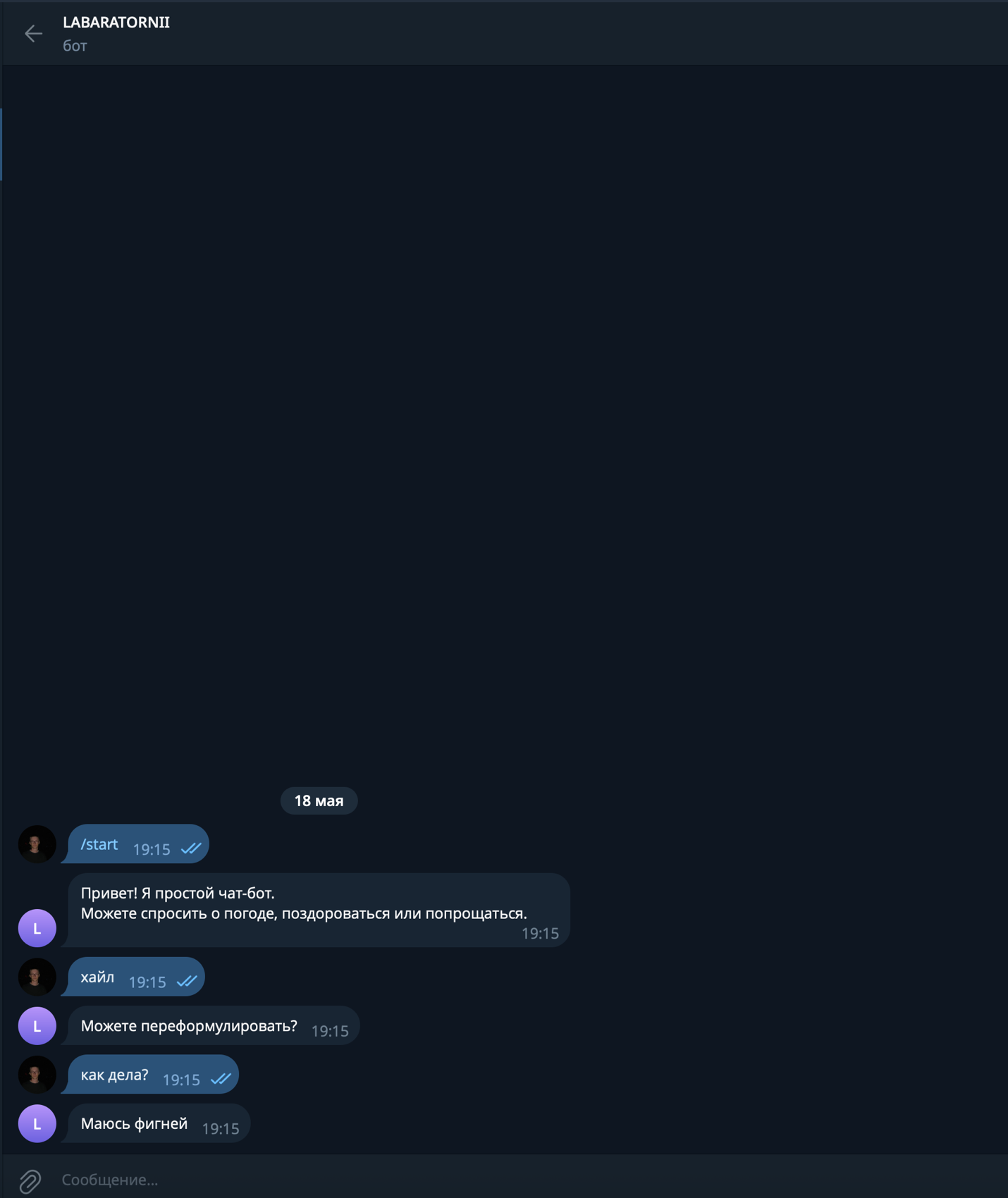
* examples - примеры фраз, которые могут означать это намерение
* responses - возможные ответы бота на это намерение

1. Самостоятельно создайте бота телеграм и подключите этот код, укажите ссылку на вашего бота для проверки. Добавьте к отчёту скрины тестов общения с ботом.

Код бота :

import logging  
from telegram import Update  
from telegram.ext import (  
 ApplicationBuilder,  
 CommandHandler,  
 MessageHandler,  
 ContextTypes,  
 filters  
)  
import nltk  
import random  
  
# Конфигурация бота  
BOT\_CONFIG = {  
 "intents": {  
 "hello": {  
 "examples": ["Привет", "Добрый день", "Шалом", "Здравствуйте"],  
 "responses": ["Привет, человек!", "Доброго времени суток!", "Здравствуйте!", "Приветствую!"]  
 },  
 "bye": {  
 "examples": ["Пока", "Досвидос", "Прощай", "До свидания"],  
 "responses": ["Счастливо!", "До свидания!", "Если что, возвращайтесь!", "Пока-пока!"]  
 },  
 "howdoyoudo": {  
 "examples": ["Как дела", "Что делаешь", "Какие дела", "Как жизнь"],  
 "responses": ["Отлично, как у вас?", "Маюсь фигней", "Отвечаю на вопросы", "Веду вебинары"]  
 },  
 "weather": {  
 "examples": ["Какая погода", "Что по погоде", "Сколько градусов", "Дождь будет"],  
 "responses": ["Сегодня солнечно!", "Ожидается дождь", "Температура +20", "Погода отличная!"]  
 },  
 },  
 "failure\_phrases": [  
 "Я не понял ваш вопрос",  
 "Можете переформулировать?",  
 "Я всего лишь бот, спросите по-другому",  
 "Не совсем понял, о чем вы"  
 ]  
}  
  
  
# Функция фильтрации текста  
def filter\_text(text):  
 text = text.lower()  
 allowed\_chars = 'абвгджзеёийклмнопрстуфхцчшщьыъэюя -'  
 return ''.join(c for c in text if c in allowed\_chars)  
  
  
# Функция сравнения текста  
def match\_texts(user\_text, example\_text):  
 user\_text = filter\_text(user\_text)  
 example\_text = filter\_text(example\_text)  
 distance = nltk.edit\_distance(user\_text, example\_text) / len(example\_text)  
 return distance < 0.4  
  
  
# Определение намерения  
def get\_intent(user\_text):  
 for intent, data in BOT\_CONFIG['intents'].items():  
 for example in data['examples']:  
 if match\_texts(user\_text, example):  
 return intent  
 return None  
  
  
# Получение ответа по намерению  
def get\_response(intent):  
 responses = BOT\_CONFIG['intents'][intent]['responses']  
 return random.choice(responses)  
  
  
# Основная логика бота  
async def handle\_message(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 user\_text = update.message.text  
 intent = get\_intent(user\_text)  
  
 if intent:  
 response = get\_response(intent)  
 else:  
 response = random.choice(BOT\_CONFIG['failure\_phrases'])  
  
 await update.message.reply\_text(response)  
  
  
# Команда /start  
async def start(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 await update.message.reply\_text(  
 "Привет! Я простой чат-бот.\n"  
 "Можете спросить о погоде, поздороваться или попрощаться."  
 )  
  
  
# Команда /help  
async def help\_command(update: Update, context: ContextTypes.DEFAULT\_TYPE):  
 await update.message.reply\_text(  
 "Просто напишите мне сообщение, и я постараюсь ответить!\n"  
 "Я понимаю приветствия, вопросы о погоде и прощания."  
 )  
  
  
# Настройка и запуск бота  
def main():  
 # Настройка логгирования  
 logging.basicConfig(  
 format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',  
 level=logging.INFO  
 )  
  
 # Создание приложения бота  
 application = ApplicationBuilder().token("7114302148:AAENx29yntQlYdm6oiIAdP\_37wt\_8U\_pRY8").build()  
  
 # Регистрация обработчиков  
 application.add\_handler(CommandHandler("start", start))  
 application.add\_handler(CommandHandler("help", help\_command))  
 application.add\_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle\_message))  
  
 # Запуск бота  
 application.run\_polling()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Скриншот переписки:



Ссылка на github: