УДК 004

Разработка чат-бота по диагностики неисправностей компьютера

Семченко Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассмотрена разработка чат бота позволяющего смотреть за нагрузкой системы. Разработка будет происходить на языке Java. Конечным результатом является рабочее десктопное приложение чат бот.
Ключевые слова: Чат-бот, Java, chat, bot

Development of a chat bot for diagnosing computer malfunctions

Semchenko Regina Viktorovna
Sholom-Aleichem Priamursky State University
student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the development of a chat bot that allows you to monitor the system load. Development will take place in Java. The end result is a working chatbot desktop application
Keywords: Chatbot, Java, chat, bot

Первый чат-бот появился в 1966 году, и называлась она Eliza. Элиза была, своеобразным психотерапевтом в нее были запрограммированы определенный набор данных ответов на обычные для психотерапевтов вопросы, если она не знала ответ, то отвечала «ясно» и меняла тему своего разговора. Сами чат-боты созданы для повторяемой и однообразной работы, для достижения максимальной скорости выполнения и решения задач.

Чат-боты - это программные приложения, которые используют искусственный интеллект и обработку естественного языка, чтобы понять, чего хочет человек, и направляют их к желаемому результату с минимально возможной работой для конечного пользователя. Это своего рода виртуальный помощник в общении с человеком.

Цель исследования – разработка чат-бота для диагностики неисправностей компьютера.

В статье А.И. Провотар и К.А. Клочко осуществили анализ особенностей имитации речевого поведения человека в процессе общения, разработали модели общения, определили основные функции и принципи работы чат-бота [1]. А.Д. Иванов в своей работе исследовал процесс трансформации каналов распространения информации от СМИ к пользователю. Был произведен анализ влияния новых тенденций, связанных с развитием технологической составляющей журналистики, на формат медиапотребления аудитории. Сделано предположение о дальнейших экспериментах СМИ, определяемых развитием технологий [2]. Д.А. Потапов провел обзор технологий созданий чат-ботов. Рассмотрел причины возникновения роста интереса к чат-ботам, а так же произвели обзор инструментов реализации [3]. В своей статье Д.А. Новиков и Е.М. Спиридонова рассмотрели технологию чат-бота и область ее применения в бизнесе. Разделили чат-ботов по классификации и их назначению. Так же провели краткий процесс разработки [4]. Д.Р. Филонов и В.И. Тупикин рассмотрели в своей работе результаты разработки чат-бота для сервиса Telegram. Представили его внутреннюю структурау и особенности реализации. В рамках проекта была применена модель векторного пространства для обработки текстовых документов, полученных в результате обработки данных социальных сетей [5]. В статье А.А. Кухтичев, А.В. Поповкин, И.Б. Юров провели реализацию создания специализированного чат-бота для мобильного приложения, специально для мгновенных сообщений медицинской информационно-аналитической системы [6]. В статье Н.С. Тарасова и Н.Ю. Сергеева провели исследование чат-бота в повседневной жизни. Боту задается вопрос и из созданной базы данных получает ответ на интересующую тематику [7].

Для начала создадим файл с расширение .java, и импортируем стандартные библиотеки «Swing» для построения окна программы и зададим все необходимые параметры для окна:

- Высота
- Ширина
- CheckBox умного бота
- Начальные координаты
- Название
- Кнопку ввода

Так же добавим некоторые переменные, для использования в будущем (рис.1).

```
import bot.SimpleBot;

import javax.swing.*;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

class SistemChatBot extends JFrame implements ActionListener {

    final String TITLE_OF_PROGRAM = "Chatter: simple chatbot";
    final int START_LOCATION = 200;
    final int WINDOW_WIDTH = 350;
    final int WINDOW_HEIGHT = 450;
    final String CHB_AI = "AI";
    final String BTN_ENTER = "Enter";

    JTextPane dialogue;
    JCheckBox ai;
    JTextField message;
    SimpleBot sbot;
    SimpleAttributeSet botStyle;
```

Рисунок 1 – Добавление переменных

Далее добавим библиотеку, которая работает с AIML. AIML – язык разметки для ИИ с помощью которого создаются виртуальные собеседники. Создадим самый основной класс, без которого проект даже не запустится, это класс «main». В нем «нарисуем» окно программы, поставим все необходимые параметры, такие как цвет, ответ бота, закрытие по кнопке, ответ по кнопке и т.д. (рис.2).

```
    public static void main(String[] args) { new SistemChatBot(); }

    SistemChatBot() {
        setTitle(TITLE_OF_PROGRAM);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setBounds(START_LOCATION, START_LOCATION, WINDOW_WIDTH, WINDOW_HEIGHT);
        dialogue = new JTextPane();
        dialogue.setEditable(false);
        dialogue.setContentType("text/html");
        JScrollPane scrollBar = new JScrollPane(dialogue);
        botStyle = new SimpleAttributeSet();
        StyleConstants.setItalic(botStyle, b: true);
        StyleConstants.setForeground(botStyle, Color.blue);
        JPanel bp = new JPanel();
        bp.setLayout(new BoxLayout(bp, BoxLayout.X_AXIS));
        ai = new JCheckBox(CHB_AI);
        ai.doClick();
        message = new JTextField();
        message.addActionListener( l: this);
        JButton enter = new JButton(BTN_ENTER);
        enter.addActionListener( l: this);
        bp.add(ai);
        bp.add(message);
        bp.add(enter);
        add(BorderLayout.CENTER, scrollBar);
        add(BorderLayout.SOUTH, bp);
        setVisible(true);
    }
```

Рисунок 2 – Класс main

Следующим делом необходимо реализовать функцию ответа бота на сообщения. То есть, если боту поступает сообщение, он переходит на следующую строчку и сделаем, если включена проверка на умного бота, то будет браться другой ряд ответов пользователю, вместо, более простых (рис.3).

```

    @Override
    public void actionPerformed(ActionEvent event) {
        if (message.getText().trim().length() > 0) {
            try {
                StyledDocument doc = dialogue.getStyledDocument();
                doc.insertString(doc.getLength(), str: message.getText() + "\n",
                        new SimpleAttributeSet());
                doc.insertString(doc.getLength(),
                        str: TITLE_OF_PROGRAM.substring(0, 9) +
                                sbot.sayInReturn(message.getText(), ai.isSelected()) + "\n",
                        botStyle);
            } catch(Exception ignored) { }
        }
        message.setText("");
        message.requestFocusInWindow();
    }
}
```

Рисунок 3 – Функция ответа

Так как класс «Action» уже заполнен и данный проект не имеет многопоточность, то класс становится перегруженным, и для обозначения этого перегруза класс помечается как «@Override».

Первый файл готов. Здесь заданы основные функции и настройки окна, следующим этапом будет создание файла с расширение .java и добавление в него всех форм ответа на заданные вопросы.

Следующим шагом создадим файл с названием «SistemBot.java». Импортируем все необходимые библиотеки для написания кода (рис.4).

```
1   package bot;
2
3   import com.sun.management.OperatingSystemMXBean;
4
5   import java.io.File;
6   import java.lang.management.ManagementFactory;
7   import java.text.DecimalFormat;
8   import java.util.Date;
9   import java.util.HashMap;
10  import java.util.Map;
11  import java.util.Random;
12  import java.util.regex.Pattern;
13
```

Рисунок 4 – Импорт библиотек

Далее создадим класс со случайными ответами в случае не знания ответа на вопрос. Ответы будут даны если пользователь напишет не понятный программе вопрос или предложение (рис.5).

```
14
15  public class SistemBot {
16      final String[] COMMON_PHRASES = {
17              "Нет ничего ценнее слов, сказанных к месту и ко времени.",
18              "Порой молчание может сказать больше, нежели уйма слов.",
19              "Перед тем как писать/говорить всегда лучше подумать.",
20              "Вежливая и грамотная речь говорит о величии души.",
21              "Приятно когда текст без орфографических ошибок.",
22              "Многословие есть признак неупорядоченного ума.",
23              "Слова могут ранить, но могут и исцелять.",
24              "Записывая слова, мы удваиваем их силу.",
25              "Кто ясно мыслит, тот ясно излагает.",
26              "Боюсь Вы что-то не договариваете."};
27      final String[] ELUSIVE_ANSWERS = {
28              "Вопрос непростой, прошу тайм-аут на раздумья.",
29              "Не уверен, что располагаю такой информацией.",
30              "Может лучше поговорим о чём-то другом?",
31              "Простите, но это очень личный вопрос.",
32              "Не уверен, что Вам понравится ответ.",
33              "Поверьте, я сам хотел бы это знать.",
34              "Вы действительно хотите это знать?",
35              "Уверен, Вы уже догадались сами.",
36              "Зачем Вам такая информация?",
37              "Давайте сохраним интригу?"};
```

Рисунок 5 – Случайные ответы

Для лучшей работы, была подключена библиотека AIML, в которой хранится уже заложенное количество вопросов, они берутся от туда при условии, что есть похожее слово из заданных (рис. 6-7).

```
<aiml>
<category>
    <pattern>Привет! Как дела?</pattern>
        <template>
        <random>
            <li>Привет.</li>
            <li>Привет. Отлично. Как у тебя?</li>
        </random>
    </template>
</category>

<category>
    <pattern>Чем занимаешься?</pattern>
        <pattern>Что делаешь?</pattern>
        <template>
        <random>
            <li>Помогаю тебе :)</li>
            <li>Тружусь в поте лица</li>
        </random>
    </template>
</category>
```

Рисунок 6 – Ответы

```
<category>
    <pattern>Покажи данные системы</pattern>
        <template>
        <random>
            <li>Конечно, сейчас вывожу:</li>
            <li>Пожалуйста:</li>
        </random>
    </template>
</category>

<category>
    <pattern>Что ты можешь посоветовать?</pattern>
        <template>
        <if sys > 50>
            <li>Пожалуй тебе необходимо закрыть ненужные службы</li>
        </if>
        <if sys < 50>
            <li>У тебя все отлично. Работай спокойно :)</li>
        </if>
    </template>
</category>
```

Рисунок 7 – Ответы

Одним из форматов разметки базы знаний является стандарт языка разметки AIML (Artificial Intelligence Markup Language). Ключевыми словами в языке являются «category», «pattern» и «template». Тег «category» является родительским к тегам «pattern» и «template», хранящими шаблоны вопроса и ответов. Тег «random» позволяет указать несколько ответов к вопросу, выбираемые интерпретатором случайным образом. Несколько тегов «pattern» позволяют описать различные варианты вопросов, соответствующие данной категории, на которые должны последовать одни и те же варианты ответов. Интерпретатор языка разметки должен позволять находить наиболее релевантные вопросы следующими дополняющими друг друга способами:

1. Поиск по всей фразе вопроса на основе регулярного выражения.
2. Поиск по количеству совпадающих слов в вопросе и паттернах.
3. Поиск по совпадению текущей темы и темах категорий.
4. Поиск по количеству совпадающих слов в текущей предыстории разговора и предысториях категорий.

Теперь напишем переменные, которые будут отвечать за некоторые ответы, где необходимо будет взять данные о чтении диска, памяти, процессора (рис.8).

```
final Map<String, String> ANSWERS_BY_PATTERNS = new HashMap<>() {{
    double n = 0.0000010 / 1024 ;
    double b = 0.5;
    final String VOLUME = "c:";
    final File file = new File(VOLUME);
    DecimalFormat df = new DecimalFormat( pattern: "###.###");

    OperatingSystemMXBean osBean = ManagementFactory.getPlatformMXBean(OperatingSystemMXBean.class);

    String DiskFree = df.format( number: file.getFreeSpace() * n);
    String DiskTotal = df.format( number: file.getTotalSpace() * n);

    String MemoryFree = df.format( number: osBean.getFreePhysicalMemorySize() * n - b);
    String MemoryTotal = df.format( number: osBean.getTotalPhysicalMemorySize() * n - b);

    double p = ManagementFactory.getOperatingSystemMXBean().getSystemLoadAverage();
    double o = (( osBean.getTotalPhysicalMemorySize() * n - b) - ( osBean.getFreePhysicalMemorySize() * n - b))
            * 100 / ( osBean.getTotalPhysicalMemorySize() * n - b);
    double z = ((file.getTotalSpace() * n) - (file.getFreeSpace() * n)) * 100;
    double q = (z / (file.getTotalSpace() * n));
```

Рисунок 8 – Создание переменных

Функция «file.getTotalSpace» возвращает данные о полной памяти жесткого диска в байтах, далее идет перемножение для получения данных в ГБ и приводим к формату 2 знака после запятой с помощью функции «df.format». Для нахождения оперативной памяти используется та же библиотека и те же методы, за исключение, что ищется она не в созданной директории «file», а в «osBean».

На этом создание бота практически готово, осталось добавить функцию, где бот будет находить дату и время и выводить ее (рис.9-10).

```
169        Pattern pattern;
170        Random random;
171        Date date;
172
173        public SimpleBot() {
174            random = new Random();
175            date = new Date();
176        }
177
178
```

Рисунок 9 – Переменные

```
183 @      public String sayInReturn(String msg, boolean ai) {
184            String say = (msg.trim().endsWith("?"))?
185                    ELUSIVE_ANSWERS[random.nextInt(ELUSIVE_ANSWERS.length)]:
186                    COMMON_PHRASES[random.nextInt(COMMON_PHRASES.length)];
187            if (ai) {
188                String message =
189                        String.join( delimiter: " ", msg.toLowerCase().split( regex: "[ {,|.}?]+"));
190                for (Map.Entry<String, String> o : PATTERNS_FOR_ANALYSIS.entrySet()) {
191                    pattern = Pattern.compile(o.getKey());
192                    if (pattern.matcher(message).find())
193                        if (o.getValue().equals("whattime")) return date.toString();
194                        else return ANSWERS_BY_PATTERNS.get(o.getValue());
195                }
196            }
197            return say;
198        }
199
```

Рисунок 10 – функция вывода времени

На этом написание бота готово. Все функции в полностью рабочем состоянии, при компиляции не выходит никаких ошибок, но имеется 4 предупреждения об использовании не современных функций оперделения оперативной памяти. Но данные предупреждения не критичны и функции вполне работоспособны

Осталось протестировать бота в действии. Запустим программу и увидим, что создается окно, которое можно растянуть, свернуть и закрыть. При завершении код останавливается с ответом «0», что означает, что никаких ошибок во время работы не выявлено (рис.11).

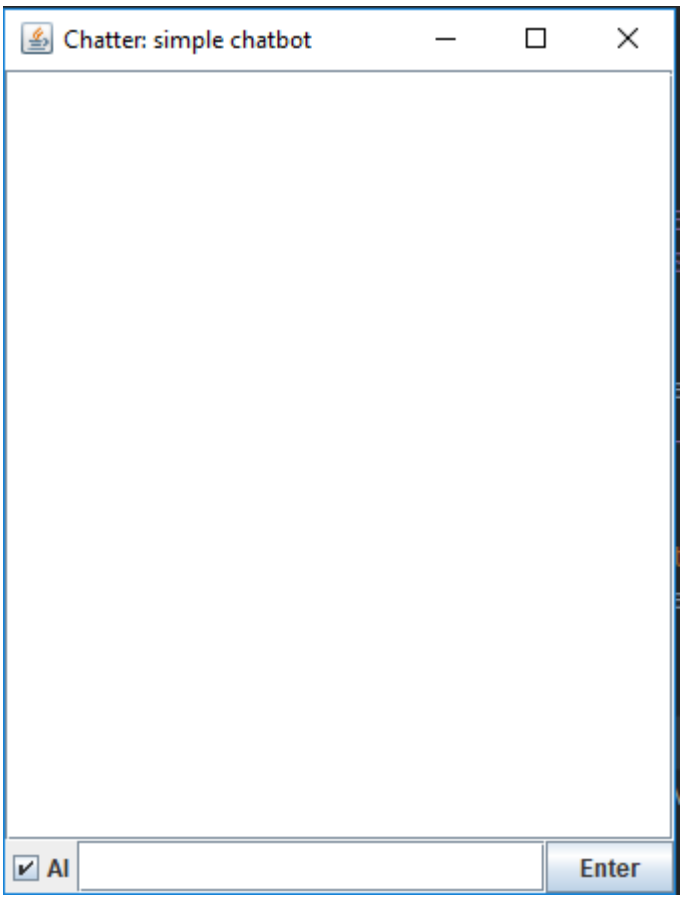

Рисунок 11 – окно программы

Теперь зададим ему все интересующие нас вопросы (рис.12).

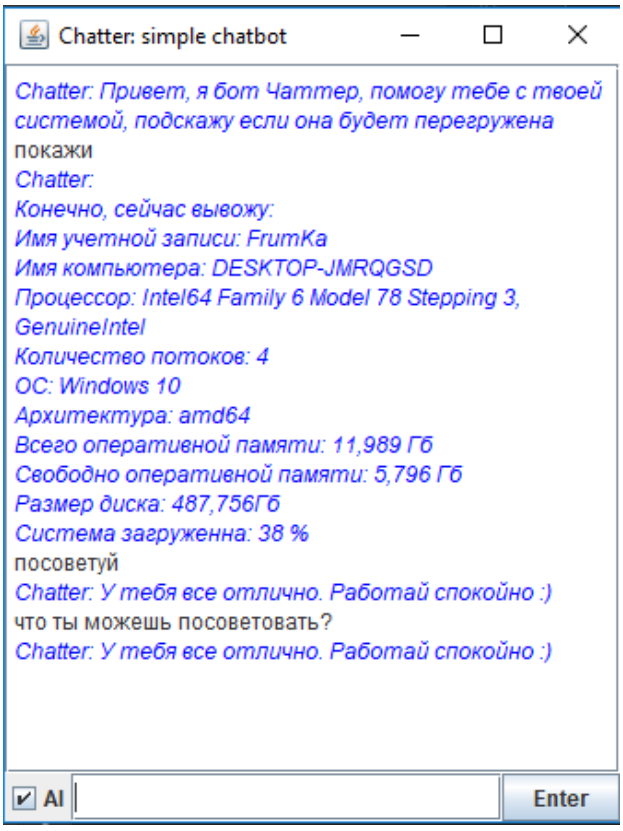

Рисунок 12 – задаем вопросы

Основные вопросы для понятия загружены ли система или нет это: «Диск», «Память», «Процессор», «Вся система». Память – обозначение оперативной памяти. Вся система – выводит абсолютно все показатели и пишет общую загрузку системы. Если система загружена менее, чем на 50 – 70 процентов, то никаких вмешательств она не требует, если же вдруг больше, то необходимо устранять неполадки, а с чем именно связанны неполадки можно узнать из определенных запросов.

В данной статье была рассмотрена разработка чат бота, по отслеживанию загрузки системы, где ответ поступает об определенном элементе системы, при запросе пользователя с использованием библиотеки для создания ИИ виртуальному собеседнику.

Библиографический список

1. Провотар А.И. , Клочко К.А. Особенности и проблемы виртуального общения с помощью чат-ботов // Научные труды Винницкого национального технологического университета. 2013. №3. С. 2.
2. Иванов А.Д. Чат-бот в telegram и вконтакте как новый канал распространения новостей // Вестник волжского университета им. в.н. татищева. 2016. №3. С. 126-132.
3. Потапов Д.А. Обзор технологий создания чат-ботов // Современные проблемы науки и образования. 2017. №4. С. 5-8.
4. Новиков Д.А., Спиридонова Е.М. Чат-боты как инструмент интернет-бизнеса // Заметки по информатике и математике. 2017. №3. С. 115-120.
5. Филонов Д.Р., Тупикин В.И. Чат-бот для telegram для помощи абитуриентам // Заметки по информатике и математике. 2017. С.152-156.
6. Кухтичев А.А., Поповкин А.В., Юров И.Б. Специализированные чат-боты в приложениях мгновенных сообщений медицинской информационно-аналитической системы "ЦИФРОМЕД" // Материалы XX юбилейной международной конференции по вычислительной механике и современным прикладным системам, 2017. С. 150-151.
7. Тарасова Н.С., Сергеева Н.Ю. Использование чат-ботов в повседневной жизни // Вестник современных исследований, 2017. С. 195-197.