***Wojskowa Akademia Techniczna***

***im. Jarosława Dąbrowskiego***



**Wydział Cybernetyki, kierunek informatyka - inżynieria systemów**

Sprawozdanie z laboratorium z przedmiotu:

*Standarty w projektowaniu systemów dialogowych*

Temat laboratoriów:

***Projektowanie ChatBota przy pomocy biblioteki Pythona „Flask”***

**Opracował:** Radosław Relidzyński, **Grupa:** WCY23IX3S4

Spis treści

[Wstęp 3](#_Toc169098387)

[Tworzenie warstwy wizualnej chatbota 3](#_Toc169098388)

[Strona html 3](#_Toc169098389)

[Styl css 4](#_Toc169098390)

[Tworzenie aplikacji dla chatbota 6](#_Toc169098391)

[Tworzenie skryptu trenującego 7](#_Toc169098392)

[Tworzenie narzędzi pomocniczych dla aplikacji i skryptu trenującego 9](#_Toc169098393)

[Przygotowanie danych 9](#_Toc169098394)

[Struktura projektu 13](#_Toc169098395)

[Trenowanie chatbota 13](#_Toc169098396)

[Uruchomienie chatbota 14](#_Toc169098397)

[Podsumowanie: 14](#_Toc169098398)

# Wstęp

Zadaniem jest stworzenie chatbota, który oferuje pomoc z w zakresie zagadnień sportowych i około tego.

# Tworzenie warstwy wizualnej chatbota

### Strona html

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Chatbot</title>  
 <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}"/>  
</head>  
<body>  
<div id="chat-container">  
 <div id="chat-header">ChatBot</div>  
 <div id="chat-box">  
 <div id="chat-output">  
 <!-- User and bot messages will be added here -->  
 </div>  
 </div>  
 <div id="chat-input">  
 <input type="text" id="user-input" placeholder="Type a message..."/>  
 <button id="send-button">Send</button>  
 </div>  
</div>  
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
<script>  
 function scrollToBottom() {  
 var chatOutput = *document*.getElementById("chat-output");  
 chatOutput.scrollTop = chatOutput.scrollHeight;  
 }  
  
 $(*document*).ready(function () {  
 $("#send-button").on("click", sendMessage);  
 $("#user-input").on("keydown", function (event) {  
 if (event.which == 13) {  
 event.preventDefault();  
 sendMessage();  
 }  
 });  
  
 function sendMessage() {  
 var user\_msg = $("#user-input").val();  
 $("#chat-output").append("<p>You: " + user\_msg + "</p>");  
 $("#user-input").val('');  
 $.ajax({  
 url: "/get\_response",  
 type: "POST",  
 data: {user\_msg: user\_msg},  
 success: function (response) {  
 var bot\_response = response.response;  
 $("#chat-output").append("<p>Bot: " + bot\_response + "</p>");  
 scrollToBottom();  
 }  
 });  
 }  
 });  
</script>  
</body>  
</html>

### Styl css

body {  
 font-family: 'Poppins', sans-serif;  
 background: linear-gradient(135deg, #9b59b6, #8e44ad);  
 color: #eee;  
 margin: 0;  
 padding: 0;  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 min-height: 100vh;  
}  
  
#chat-container {  
 align-items: center;  
 border-radius: 20px;  
 background: linear-gradient(145deg, #8e44ad, #9b59b6);  
 box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);  
 color: white;  
 overflow: hidden;  
 border: 2px solid #fff;  
 display: flex;  
 flex-direction: column;  
 height: 95vh;  
 width: 95%;  
 border-top: 5px solid #4a235a;  
 animation: chatEntry 0.5s ease;  
}  
  
#chat-header {  
 padding: 20px;  
 text-align: center;  
 font-size: 2.5em;  
 font-weight: bold;  
 color: #ecf0f1;  
 animation: headerEntry 1s ease;  
 background: linear-gradient(145deg, #8e44ad, #9b59b6);  
 width: 100%;  
}  
  
#chat-header h1 {  
 margin: 0;  
 font-size: 1.5em;  
 font-weight: bold;  
 background-color: transparent;  
}  
  
#chat-box {  
 background: #f8f9f9;  
 color: #2c3e50;  
 padding: 20px;  
 text-align: left;  
 font-size: 1.2em;  
 overflow-y: auto;  
 flex-grow: 1;  
 border-radius: 20px;  
 margin: 20px;  
 width: 60%;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);  
 animation: boxEntry 1s ease;  
}  
  
#chat-output {  
 padding: 10px;  
 word-wrap: break-word;  
 background-color: transparent;  
 font-family: 'Poppins', sans-serif;  
}  
  
#chat-input {  
 display: flex;  
 align-items: center;  
 background-color: #f8f9f9;  
 padding: 20px;  
 border-radius: 20px;  
 margin: 20px;  
 width: 60%;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);  
 animation: inputEntry 1s ease;  
}  
  
input[type="text"] {  
 flex: 1;  
 padding: 15px;  
 border: none;  
 border-radius: 25px;  
 margin-right: 10px;  
 outline: none;  
 font-size: 20px;  
 font-family: 'Poppins', sans-serif;  
 background: #ecf0f1;  
 color: #2c3e50;  
}  
  
#send-button {  
 background-color: #e74c3c;  
 color: white;  
 border: none;  
 border-radius: 25px;  
 padding: 15px 30px;  
 cursor: pointer;  
 outline: none;  
 transition: background 0.3s ease;  
}  
  
#send-button:hover {  
 background-color: #c0392b;  
}  
  
@keyframes chatEntry {  
 from {  
 transform: translateY(-100%);  
 opacity: 0;  
 }  
 to {  
 transform: translateY(0);  
 opacity: 1;  
 }  
}  
  
@keyframes headerEntry {  
 from {  
 transform: translateY(-100%);  
 opacity: 0;  
 }  
 to {  
 transform: translateY(0);  
 opacity: 1;  
 }  
}  
  
@keyframes boxEntry {  
 from {  
 transform: scale(0.8);  
 opacity: 0;  
 }  
 to {  
 transform: scale(1);  
 opacity: 1;  
 }  
}  
  
@keyframes inputEntry {  
 from {  
 transform: translateY(100%);  
 opacity: 0;  
 }  
 to {  
 transform: translateY(0);  
 opacity: 1;  
 }  
}

# Tworzenie aplikacji dla chatbota

import json  
import random  
import torch  
from flask import Flask, render\_template, request, jsonify  
  
from model import NeuralNet  
from utils.nltk\_utils import bag\_of\_words, tokenize  
  
app = Flask(\_\_name\_\_)  
  
# Load intents and the chatbot model  
with open('source/intents.json', 'r') as json\_data:  
 intents = json.load(json\_data)  
  
FILE = "source/data.pth"  
data = torch.load(FILE)  
  
input\_size = data["input\_size"]  
hidden\_size = data["hidden\_size"]  
output\_size = data["output\_size"]  
all\_words = data["all\_words"]  
tags = data["tags"]  
model\_state = data["model\_state"]  
  
model = NeuralNet(input\_size, hidden\_size, output\_size)  
model.load\_state\_dict(model\_state)  
model.eval()  
  
  
# Function to get chatbot responses  
def get\_response(msg):  
 sentence = tokenize(msg)  
 X = bag\_of\_words(sentence, all\_words)  
 X = X.reshape(1, X.shape[0])  
 X = torch.from\_numpy(X)  
 output = model(X)  
 \_, predicted = torch.max(output, dim=1)  
  
 tag = tags[predicted.item()]  
  
 probs = torch.softmax(output, dim=1)  
 prob = probs[0][predicted.item()]  
  
 if prob.item() > 0.75:  
 for intent in intents['intents']:  
 if tag == intent["tag"]:  
 return random.choice(intent['responses'])  
  
 return "I do not understand..."  
  
  
# Define the home page route  
@app.route('/')  
def home():  
 return render\_template('index.html')  
  
  
# Define a route for handling chat interactions  
@app.route('/get\_response', methods=['POST'])  
def chat():  
 user\_msg = request.form['user\_msg']  
 bot\_response = get\_response(user\_msg)  
 return jsonify({'response': bot\_response})  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug=True)

# Tworzenie skryptu trenującego

import json  
  
import torch  
import torch.nn as nn  
from torch.utils.data import Dataset, DataLoader  
  
from model import NeuralNet  
from utils.nltk\_utils import tokenize, stem, bag\_of\_words  
  
with open('source/intents.json', 'r') as f:  
 intents = json.load(f)  
  
all\_words = []  
tags = []  
xy = []  
  
for intent in intents['intents']:  
 tag = intent['tag']  
 tags.append(tag)  
 for pattern in intent['patterns']:  
 w = tokenize(pattern)  
 all\_words.extend(w)  
 xy.append((w, tag))  
  
ignore\_words = ['?', '!', '.', ',']  
all\_words = [stem(w) for w in all\_words if w not in ignore\_words]  
all\_words = sorted(set(all\_words))  
tags = sorted(set(tags))  
  
X\_train = []  
y\_train = []  
  
for (pattern\_sentence, tag) in xy:  
 bag = bag\_of\_words(pattern\_sentence, all\_words)  
 X\_train.append(bag)  
  
 label = tags.index(tag)  
 y\_train.append(label)  
  
X\_train = torch.tensor(X\_train, dtype=torch.float32)  
y\_train = torch.tensor(y\_train, dtype=torch.int64)  
  
  
class ChatDataset(Dataset):  
 def \_\_len\_\_(self):  
 return len(X\_train)  
  
 def \_\_getitem\_\_(self, index):  
 return X\_train[index], y\_train[index]  
  
  
dataset = ChatDataset()  
batch\_size = 8  
dataloader = DataLoader(dataset=dataset, batch\_size=batch\_size, shuffle=True)  
  
input\_size = len(X\_train[0])  
hidden\_size = 8  
output\_size = len(tags)  
learning\_rate = 0.001  
num\_epochs = 1000  
  
model = NeuralNet(input\_size, hidden\_size, output\_size)  
  
criterion = nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)  
  
loss = None  
  
  
for epoch in range(num\_epochs):  
 for (words, labels) in dataloader:  
 outputs = model(words)  
 loss = criterion(outputs, labels)  
  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
  
print(f'Final loss: {loss.item():.4f}')  
  
data = {  
 "model\_state": model.state\_dict(),  
 "input\_size": input\_size,  
 "hidden\_size": hidden\_size,  
 "output\_size": output\_size,  
 "all\_words": all\_words,  
 "tags": tags  
}  
  
FILE = "source/data.pth"  
torch.save(data, FILE)  
  
print(f'Training complete. File saved to {FILE}')

# Tworzenie narzędzi pomocniczych dla aplikacji i skryptu trenującego

import nltk  
import numpy as np  
from nltk.stem.porter import PorterStemmer  
  
stemmer = PorterStemmer()  
  
  
def tokenize(sentence):  
 return nltk.word\_tokenize(sentence)  
  
  
def stem(word):  
 return stemmer.stem(word.lower())  
  
  
def bag\_of\_words(tokenized\_sentence, all\_words):  
 tokenized\_sentence = [stem(w) for w in tokenized\_sentence]  
 bag = np.zeros(len(all\_words), dtype=np.float32)  
 for idx, w in enumerate(all\_words):  
 if w in tokenized\_sentence:  
 bag[idx] = 1.0  
 return bag

# Przygotowanie danych

{  
 "intents": [  
 {  
 "tag": "greeting",  
 "patterns": [  
 "Hello",  
 "Hi there",  
 "Good day",  
 "Hey",  
 "Hi",  
 "Good morning",  
 "Good evening",  
 "Greetings",  
 "Howdy",  
 "Hi everyone",  
 "Hey there"  
 ],  
 "responses": [  
 "Hello, welcome to the Sports Advisory Forum!",  
 "Hi there! How can we help you with your sports-related questions today?",  
 "Good day! Need advice on sports training or equipment?",  
 "Hey! How can we assist you with your sports needs today?",  
 "Greetings! What sports issues can we help you with?"  
 ]  
 },  
 {  
 "tag": "goodbye",  
 "patterns": [  
 "Bye",  
 "Goodbye",  
 "See you later",  
 "Thanks for the help",  
 "Catch you later",  
 "Take care",  
 "See you next time",  
 "I'm off",  
 "Farewell"  
 ],  
 "responses": [  
 "Goodbye! Stay active and healthy!",  
 "See you next time, enjoy your sports activities!",  
 "Thank you for visiting the Sports Advisory Forum!",  
 "Bye! Stay fit and have fun!",  
 "Catch you later! Have a great day!"  
 ]  
 },  
 {  
 "tag": "thanks",  
 "patterns": [  
 "Thanks",  
 "Thank you",  
 "Much appreciated",  
 "Thanks for your assistance",  
 "Thanks a lot",  
 "I appreciate it",  
 "Thanks so much",  
 "Many thanks",  
 "Thank you very much"  
 ],  
 "responses": [  
 "You're welcome! Happy to help!",  
 "Any time! We're here for all your sports needs.",  
 "Glad to assist! Enjoy your training!",  
 "No problem! We're always here to help.",  
 "You're welcome! Stay active and take care!"  
 ]  
 },  
 {  
 "tag": "training\_advice",  
 "patterns": [  
 "How do I improve my running speed?",  
 "What exercises help build muscle?",  
 "Tips for increasing endurance?",  
 "How can I get better at soccer?",  
 "What training is best for basketball?"  
 ],  
 "responses": [  
 "To improve running speed, focus on interval training and strength exercises for your legs.",  
 "Building muscle requires a combination of strength training exercises like weightlifting and a protein-rich diet.",  
 "For endurance, incorporate long-distance running or cycling and mix in interval training.",  
 "Improving at soccer involves practicing ball control, shooting drills, and playing regularly with others.",  
 "Basketball training should include dribbling drills, shooting practice, and agility exercises."  
 ]  
 },  
 {  
 "tag": "nutrition",  
 "patterns": [  
 "What should I eat before a workout?",  
 "Best foods for muscle recovery?",  
 "How to stay hydrated during exercise?",  
 "What diet is best for athletes?",  
 "Post-workout meal suggestions?"  
 ],  
 "responses": [  
 "Before a workout, eat a balanced meal with carbohydrates and protein about 2-3 hours prior.",  
 "For muscle recovery, consume foods rich in protein like chicken, fish, and legumes, along with carbohydrates.",  
 "Stay hydrated during exercise by drinking water regularly, and consider sports drinks for long-duration activities.",  
 "Athletes benefit from a diet rich in lean proteins, whole grains, fruits, and vegetables.",  
 "A good post-workout meal includes protein and carbohydrates, such as a smoothie with protein powder and fruits."  
 ]  
 },  
 {  
 "tag": "equipment\_advice",  
 "patterns": [  
 "Best running shoes?",  
 "What type of bike is good for beginners?",  
 "How to choose a tennis racket?",  
 "What equipment do I need for home workouts?",  
 "Recommended brands for sports gear?"  
 ],  
 "responses": [  
 "The best running shoes depend on your foot type and running style; visit a specialty store for a fitting.",  
 "For beginners, a hybrid bike is a good choice as it combines features of road and mountain bikes.",  
 "When choosing a tennis racket, consider your skill level and playing style; consult with a specialist for advice.",  
 "For home workouts, basic equipment includes dumbbells, resistance bands, a yoga mat, and a stability ball.",  
 "Popular and reliable brands for sports gear include Nike, Adidas, Under Armour, and Reebok."  
 ]  
 },  
 {  
 "tag": "injury\_prevention",  
 "patterns": [  
 "How to prevent running injuries?",  
 "Tips for avoiding sports injuries?",  
 "What are the best stretches before a workout?",  
 "How to avoid overtraining?",  
 "Preventing muscle strains?"  
 ],  
 "responses": [  
 "To prevent running injuries, ensure proper warm-up, use the right shoes, and gradually increase mileage.",  
 "Avoid sports injuries by warming up properly, using appropriate gear, and following proper techniques.",  
 "Effective pre-workout stretches include dynamic stretches like leg swings, arm circles, and walking lunges.",  
 "Avoid overtraining by allowing adequate rest, listening to your body, and varying your training routine.",  
 "Prevent muscle strains by warming up, incorporating flexibility exercises, and not overexerting yourself."  
 ]  
 },  
 {  
 "tag": "recovery",  
 "patterns": [  
 "How to recover after a marathon?",  
 "Best recovery methods for sore muscles?",  
 "Tips for post-workout recovery?",  
 "How to reduce muscle soreness?",  
 "Effective ways to recover from a workout?"  
 ],  
 "responses": [  
 "After a marathon, focus on hydration, gentle stretching, and rest to allow your body to recover.",  
 "For sore muscles, try foam rolling, gentle stretching, and applying ice or heat as needed.",  
 "Post-workout recovery tips include rehydrating, consuming a balanced meal with protein and carbs, and resting.",  
 "Reduce muscle soreness by staying hydrated, incorporating light activities like walking, and using a foam roller.",  
 "Effective recovery methods include proper hydration, nutrition, stretching, and getting adequate sleep."  
 ]  
 },  
 {  
 "tag": "sports\_psychology",  
 "patterns": [  
 "How to stay motivated in sports?",  
 "Tips for overcoming performance anxiety?",  
 "How to set effective sports goals?",  
 "Dealing with pressure in competitions?",  
 "Improving mental toughness?"  
 ],  
 "responses": [  
 "Stay motivated by setting achievable goals, tracking your progress, and finding a workout buddy or coach.",  
 "Overcome performance anxiety by practicing relaxation techniques, visualization, and positive self-talk.",  
 "Set effective sports goals by making them specific, measurable, attainable, relevant, and time-bound (SMART).",  
 "Deal with competition pressure by focusing on your own performance, practicing mindfulness, and staying positive.",  
 "Improve mental toughness by challenging yourself regularly, staying focused on your goals, and building resilience."  
 ]  
 },  
 {  
 "tag": "sports\_rules",  
 "patterns": [  
 "What are the basic rules of soccer?",  
 "How is basketball scored?",  
 "What are the rules of tennis?",  
 "How do you play volleyball?",  
 "Explain the offside rule in soccer."  
 ],  
 "responses": [  
 "The basic rules of soccer involve scoring by getting the ball into the opponent's goal and preventing the opposition from scoring.",  
 "Basketball is scored with two points for regular field goals, three points for shots beyond the arc, and one point for free throws.",  
 "Tennis rules include serving diagonally, scoring points through rallies, and winning sets by reaching six games with at least a two-game lead.",  
 "Volleyball is played by hitting a ball over a net with the aim to land it in the opponent's court, and teams can hit the ball up to three times before returning it.",  
 "The offside rule in soccer states that a player is offside if they are nearer to the opponent's goal line than both the ball and the second last opponent when the ball is played to them."  
 ]  
 }  
 ]  
}

# Struktura projektu

A screenshot of a computer

Description automatically generated

# Trenowanie chatbota

A computer screen with text

Description automatically generated

# Uruchomienie chatbota

A screenshot of a chatbot

Description automatically generated

# Podsumowanie:

W ramach ćwiczeniu udało się stworzyć chatbota doradzającego w zakresie sportu, dającego wskazówki jak się przygotować i jak radzić sobie z kwestiami okołosportowymi.

Chat został wytrenowany przy pomocy bibliotek torch, nltk oraz numpy. Aplikacja została postawiona jako lokalny serwer http przy pomocy frameworka Flask.