Miskolci Egyetem nyári szakmai gyakorlat

Gyakorlatot végző tanuló: Czikó Tivadar

Neptun kód: O2IXLB

Email cím: tivi180@gmai.com

Vállalat (intézmény) neve: Miskolci Egyetem

Konstruktor: Piller Imre

Email cím: matip@gold.uni-miskolc.hu

Beosztás: Egyetemi tanársegéd

Intézmény rövid bemutatása

A nyári szakmai gyakorlatomat a Miskolci egyetemen töltöttem. Ez az intézményben 8 kar található, amelynek a fő profilját jelenleg a Gépészmérnöki és informatikai kar (GEIK) határozza meg. Jelenleg a legnagyobb hallgatói lélekszámmal rendelkező kar.

Nyári szakmai gyakorlat témája: Pythonban való chatbot megvalósítása.

Rövid bemutatása: Olyan programot próbáltam létrehozni, amivel lehet chattelni, ez részben sikerült, de sajnos nem sikerült az eredeti célját elérnem, ami az lett volna, hogy egy Miskolc Egyetemi „Térképészt" hozzak létre, aki meg tudja mondani az összes termet, helységet, hogy hol található és küld egy fotót, ami meg mutatja a térképen, hogy hol található. Azt a nevet adtam neki, hogy M.E mapper. Eleinte social media-s platformokra szerettem volna megvalósítani, de ez sajnos nem jött össze, ezért egy kis programként valósítottam meg a számítógépemen, ahol egy chat felületen lehet vele beszélni.

Megvalósítás:

Előkészületek:

Mivel még ezelőtt nem nagyon használtam python nyelvet, emiatt azzal kezdtem, hogy Tanár Úr által ajánlott irodalmakat elkezdtem olvas és értelmezni, majd elkezdtem magamtól megtanulni és hozzá szokni a Python szintaktikájához, mert más programozási nyelvekhez képest kicsit különlegesebb a szintaktikája, nem a megszokott. Például gyakori hiba a ' ; ' (pontos vesszők) sor végéhez való írása, ami ebben a nyelvben nem szükséges mivel itt maga a sor tagolása határozza meg egy kódnak a végét. (Erről a hibáról nagyon sok mém is készült).

Programozási nyelv:

Pythonban próbáltam megvalósítani mert ezt találtam a legalkalmasabbnak egy mesterséges intelligencia létrehozásához. Számos előnnyel jár, ha Python nyelvet használunk egy AI (Artifacial Intelligence) létrehozásához. A Python fájlok mellet szükség volt még egy JSON fájlra, ami az adatok tárolása miatt kellett.

Python nyelv bemutatása:

Python egy magasszintű programozási nyelv, amit a 90-es évek végén találtak fel. A python a gyors programozás, az átláthatóság elvét részesíti előnyben és nem a program minél gyorsabban való futását. Ez egy interpreteres nyelv, ami jelenti, hogy nincsen ketté választva tárgyikód a forráskódtól. Alkalmas objektum orientált programozásra is. Napjainkban ez a legnépszerű (egyik legtöbbet használt programozási nyelv). Gondolom azért az elmúlt 10 évben lett nagyon felkapott mert a Mesterséges intelligencia egyre nagyobb teret hódít a tudományban, kutatásokban és arra ez a legalkalmasabb programozási nyelv. A következő tulajdonságai miatt használható jobban az Természetes nyelvek feldolgozására (Natural Language Proccessing). A Python egy nagy előnye a többi nyelvhez képest a hatalmas meg írt algoritmus áll rendelkezésre mindenki számára, aki AI (Mesterséges Intelligencia) szeretne

létrehozni. A Pythonban könnyen lehet könyvtárakat beépíteni (Pl.: Jupiter (Anaconda 3) program segítségével könnyen tudok hozzá adni plusz könyvtárakat). A nyelv egyik szépsége abban rejlik, hogy könnyen lehet építeni programot már működő részekből és ezek a részel összeadódnak. Más nyelvek fejlesztői környezetet használnak a projekt íráshoz. Az egyik különleges könyvtár, amit használtam az a NumPy, ami a mesterséges intelligencia miatt kellet használnom, mert nagyon sok értékes könyvtár működik a NumPy-jal, például statisztikai elemzések, könyvtár vizualizációk…stb előnyeit lehet élvezni. A következő az amit szerettem volna tovább vinni, de sajnos addig még nem jutottam el a Tensorflow. Ez a mély tanulásért és a neurális hálózattokért felelős könyvtár, kész osztályokból, neuronról szóló információkból áll, hatalmas adatmennyiséggel dolgozik. A Pythonnal tudunk iOS-re, Androidra, asztali számítógépekre is tudunk készíteni programokat mivel ez egy univerzális intelligencia-szoftver fejlesztési nyelv, tehát nem függ a platformoktól (természetesen néhány módosítás szükséges végrehajtani). Konklúzió a Python a legjobb programozási nyelv, ha gépi tanulásról an szó. Mivel nagyon felhasználó barát egyre többen szeretik használni és AI szoftverek fejlesztésének a népszerűsége évről évre emiatt csak egyre népszerűbb és ezáltal egyre több, használhatóbb könyvtárak jelennek meg.

JSON nyelv bemutatása:

Ez a nyelv ismerős volt a tanulmányai során is, mivel Javat kellet tanulnunk Objektum Orientált Programozásból és Web-technológiákon is kellett használni. A JSON egy egyszerű ember által is olvasható szöveg alapú szabvány számítógépek közötti adatcserére. A gyakorlat során én is adatok tárolására használtam. A JSON a Java Scriptből alakult ki. Nyelv független a legtöbb nyelvhez van értelmezője.

Megvalósítás mérföld kövei:

1. **Legelső Python programom:** Itt a pyautogui és a time könyvtárakat használtam, ez egy nagyon primitív „chatbot” egyetlen egy python fájlból áll. Ebben a fájlban is csak egy integer és egy while ciklus található if else szerkezettel. A pyautogui az üzenet küldéséhez volt szükséges, a time könyvtár pedig arra, hogy várjon pár másod percet hogy a következő mondatot küldje tovább.

   A kód:

```
import pyautogui
import time

#print('Hello Mate!\nHow can I help you?')

S = 0

while True:
    if S < 5:
        S = S + 2
        time.sleep(S)
```

```
            pyautogui.typewrite('Hello Mate!')
            S = S + 3
            time.sleep(S)
            pyautogui.typewrite('\nHow Can I help you?')
            pyautogui.press('enter')

        else:
            break
```

2. **Kezdetleges chatbot**: (chatbot 0.2)
   Ennél a projektnél másabb könyvtárakat használtam és már megjelent egy JSON fájl. Itt már viszonylag egy egész jó chatbotot tudtam létre hozni. Az alábbi módon készítettem el. Legelőször importáltam a JSON, RE könyvtárakat és random_responses.py (saját python fájlt). A JSON könyvtár a JSON fájl miatt volt szükséges, ahol is a kérdésekre a válaszokat tároltam. A random_responses-re azért volt szükség, hogy a chatbot valamit nem ért akkor tudjun vissza válaszolni, hogy nem érti amit mondunk neki. A main.py fájlban meg a működése található.

   A main.py fájlban lettek a szükséges könyvtárak segéd fájlok importálva. Majd csináltam egy def függvény ami a JSON fájl betöltésért felel azután a load_json-nel betöltöttem és meg adtam a fájlnak a helyét, hogy honnan töltsön be. Utána egy következő def függvénnyel amiben több for ciklus és if volt ágyazva azért volt szükség hogy a JSONban megírt kis programot tudja értelmezni, hogy mely válaszra mit kell válaszolni. Itt is megjelenik a while ciklus egy True-val, ami azért fele, hogy mutassa nekünk a konzolban, hogy ki mit ír. Például, ahol az alábbi képen is lehet látni.

```
PS D:\Development\PillerImreGrafika\O2IXLB_Szakmaigyakorlat\Chatbot\Chatbot-2.0\PythonFiles>
ython main.py
Loaded '../DataDirectories/bot.json' successfully!
You: hi
Bot: Hey there!
You: how are you
Bot: I'm great! Thanks for asking.
You: tell me something
Bot: I'm terribly sorry, I didn't quite catch that.
You: bye
Bot: See you later!
You:
```

   main.py kód:

```
mport json
import re
import random_responses

def load_json(file):
    with open(file) as bot_responses:
        print(f"Loaded '{file}' successfully!")
        return json.load(bot_responses)
```

```
response_data = load_json("../DataDirectories/bot.json")


def get_response(input_string):
    split_message = re.split(r'\s+|[,;?!.-]\s*', input_string.lower())
    score_list = []

    for response in response_data:
        response_score = 0
        required_score = 0
        required_words = response["required_words"]

        if required_words:
            for word in split_message:
                if word in required_words:
                    required_score += 1

        if required_score == len(required_words):
            for word in split_message:
                if word in response["user_input"]:
                    response_score += 1
        score_list.append(response_score)

    best_response = max(score_list)
    response_index = score_list.index(best_response)

    if input_string == "":
        return "Please type something so we can chat :("

    if best_response != 0:
        return response_data[response_index]["bot_response"]

    return random_responses.random_string()


while True:
    user_input = input("You: ")
    print("Bot:", get_response(user_input))
```

A JSON fájlban legelőször szét szettem típusokra, de igatából a típusok azt csak a programozót embert segíti. Utána elkezdtem írni a felhasználói inputot szavakra taglalva, alatta meg egy bot válaszok található, ahol egy kész mondatott írtam és ez alatt egy ajánlott szavak, ami azt teszi lehetővé, ha valami rövidét vagy véletlenül elírtt egy szót vagy kihagyott akkor is tudjon válaszolni.

bot.json kód:

```
[
```

```
  {
    "response_type": "greeting",
    "user_input": ["hello", "hi", "hey"],
    "bot_response": "Hey there!",
    "required_words": []
  },

  {
    "response_type": "introducing",
    "user_input": ["what", "is", "your", "name"],
    "bot_response": "My name is M.E mapper.",
    "required_words": ["your", "name"]
  },
  {
    "response_type": "question",
    "user_input": ["where", "is", "the", "restaurant"],
    "bot_response": "You can find on the middle of the university.",
    "required_words": ["where", "restaurant"]
  },
  {
    "required_words": ["thank", "you"]
  },
```

A random_responses.py-ban egy véletlenszerű válasz generáló van, aminek a megírásához szükségem volt a random könyvtár importálásához, amivel, véletlenszerű válaszokat tud adni egy témában a bot.
random_responses.py kód:

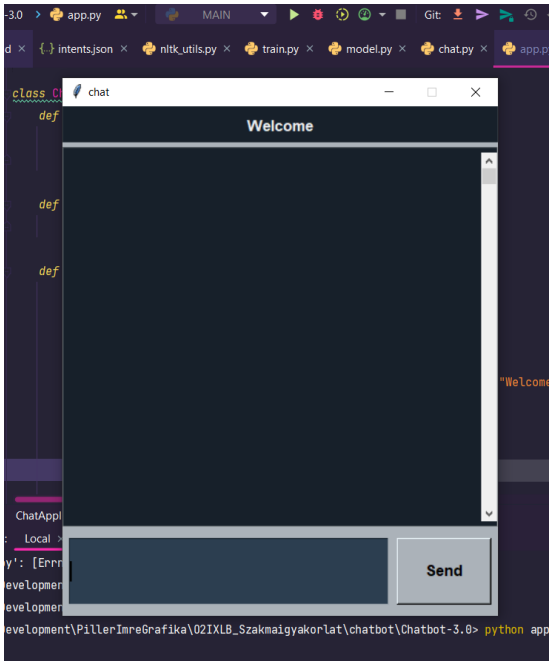
```
import random


def random_string():
    random_list = [
        "Please try writing something more descriptive.",
        "Oh! It appears you wrote something I don't understand yet",
        "Do you mind trying to rephrase that?",
        "I'm terribly sorry, I didn't quite catch that.",
        "I can't answer that yet, please try asking something else."
    ]

    list_count = len(random_list)
    random_item = random.randrange(list_count)

    return random_list[random_item]
```

3. **Chatbot program:** Mivel nem tudtam meg csinálni ezt hogy egy messengeren működjen a chatboton ezért ezt az ablakos megoldást választottam.



Ehhez a chatbothoz 5 darab Python fájlhoz és 1 darab JSON-hoz volt szükség.
A fájlok az alábbi neveken vannak: intents.json, nltk_utils.py, train.py, model.py chat.py, app.py. Mindegyik más-más funkciót lát el. Itt az alábbi könyvtárakat használtam nltk, numpy as np, PortesStemmer, json, random, torch, nn, NeuralNet, tokenize, bag_of_worlds.

A JSON olyan, mint az előzőekben csak itt kicsit egyszerűbben oldottam meg. Itt is az adatok tárolására használom.

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": [
        "Hi",
        "Hey",
        "Hello",
        "Good day"
      ],
      "responses": [
        "Hey :-) My name is M.E mapper",
        "Hi there, what can I do for you?",
        "Hi there, how can I help?"
      ]
    }
  ]
}
```

Az nltk_utils.py fájl azt csinálja, meghatározza a mondatokat, értelmezi a felhasználó mondatát és itt határozza meg, hogy mely mondattal kellene vissza válaszolnia a chatbotnak.

```
import nltk
import numpy as np
#nltk.download('punkt')
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_word):
    """
    sentence = ["hello", "how", "are", "you"]
    words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]
    bog = [0, 1, 0, 1, 0, 0, 0]
    """
    tokenized_sentence = [stem(w) for w in tokenized_sentence]

    bag = np.zeros(len(all_word), dtype=np.float32)
    for idx, w in enumerate(all_word):
        if w in tokenized_sentence:
            bag[idx] = 1.0
    return bag
```

A train.py-ban meg a chatbotunkat tanítjuk, be hogy hogyan válaszoljon a felhasználónak és értelmezi a JSON fájlunkat, hogy ott mely mondatra milyen mondattal válaszoljon.

```
import numpy as np
import random
import json

import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

from nltk_utils import bag_of_words, tokenize, stem
from model import NeuralNet

with open('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))
```

A model.py-ban a AI-nak a modelljét csináltam meg ami a mély tanulásban segít a neurális hálókkal.

```
import torch
import torch.nn as nn


class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation & no softmax
        return out
```

A chat.py-ban magát a csevegés lett megvalósítva meg van benne egy olya rész, hogy nem tudja a választ a bot akkor kiírja, hogy nem tudja.

```
import random
import json
import torch
from model import NeuralNet
from nltk_utils import bag_of_words, tokenize

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

with open('intents.json', 'r') as f:
    intents = json.load(f)

FILE = "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).to(device)
model.load_state_dict(model_state)
model.eval()

bot_name = "M.E mapper"
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).to(device)

    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..."
```

Az app.py-ban itt maga bot kinézetét csináltam meg tehát az ablakot, ahol tudunk beszélni a bottal.

```
from tkinter import *
from chat import get_response, bot_name

BG_GRAY = "#ABB2B9"
BG_COLOR = "#17202A"
TEXT_COLOR = "#EAECEE"
ENTRY_BOX_COLOR = "#2C3E50"

FONT = "Helvetica 14"
FONT_BOLD = "Helvetica 13 bold"

class ChatApplication:
    def __init__(self):
```

```
        self.window = Tk()
        self._setup_main_window()

    def run(self):
        self.window.mainloop()

    def _setup_main_window(self):
        self.window.title("chat")
        self.window.resizable(width=False, height=False)
        self.window.configure(width=470, height=550, bg=BG_COLOR)

        head_label = Label(self.window, bg=BG_COLOR, fg=TEXT_COLOR,
text="Welcome", font=FONT_BOLD, pady=10)
        head_label.place(relwidth=1)

        line = Label(self.window, width=450, bg=BG_GRAY)
        line.place(relwidth=1, rely=0.07, relheight=0.012)

        self.text_widget = Text(self.window, width=20, height=2, bg=BG_COLOR,
fg=TEXT_COLOR, font=FONT, padx=5, pady=5)

        self.text_widget.place(relheight=0.745, relwidth=1, rely=0.08)
        self.text_widget.configure(cursor="arrow", state=DISABLED)

        scrollbar = Scrollbar(self.text_widget)
        scrollbar.place(relheight=1, relx=0.974)
        scrollbar.configure(command=self.text_widget.yview)

        bottom_label = Label(self.window, bg=BG_GRAY, height=80)
        bottom_label.place(relwidth=1, rely=0.825)

        self.msg_entry = Entry(bottom_label, bg=ENTRY_BOX_COLOR,
fg=TEXT_COLOR, font=FONT)
        self.msg_entry.place(relwidth=0.74, relheight=0.06, rely=0.008,
relx=0.011)
        self.msg_entry.focus()
        self.msg_entry.bind("<Return>", self._on_enter_pressed)

        send_button =  Button(bottom_label, text="Send", font=FONT_BOLD,
width=20, bg=BG_GRAY, command=lambda: self._on_enter_pressed(None))
        send_button.place(relx=0.77, rely=0.008, relheight=0.06,
relwidth=0.22)



    def _on_enter_pressed(self, event):
        msg = self.msg_entry.get()
        self._insert_message(msg, "You")
```

```
    def _insert_message(self, msg, sender):
        if not msg:
            return

        self.msg_entry.delete(0, END)
        msg1 = f"{sender}: {msg}\n\n"
        self.text_widget.configure(state=NORMAL)
        self.text_widget.insert(END, msg1)
        self.text_widget.configure(state=DISABLED)

        msg2 = f"{bot_name}: {get_response(msg)}\n\n"
        self.text_widget.configure(state=NORMAL)
        self.text_widget.insert(END, msg2)
        self.text_widget.configure(state=DISABLED)

        self.text_widget.see(END)


if __name__ == "__main__":
    app = ChatApplication()
    app.run()
```

Tehát a szakmai gyakorlat során részlegesen eltudtam sajátítani a Python nyelvnek a Mesterséges Intelligenciával foglalkozó részét, nagyobb betekintést kaptam, hogy milyen egy számomra ismeretlen dologgal foglalkozni, úgy hogy az egyetemen tanult ismeretek és a kutatásaim során felhasznált információkra voltam utalva, milyen aktívan programozással foglalkozni, github rendszeres használataés találkozhattam egy érdekese témával a Természetes nyelvek feldolgozásán belül a chatbotok világával.