STREDNÁ ODBORNÁ ŠKOLA HANDLOVÁ

VIRTUÁLNY PORADCA PRE DUŠEVNÚ PODPORU

ADAM BESEDA

2025

VIRTUÁLNY PORADCA PRE DUŠEVNÚ PODPORU

ROČNÍKOVÝ PROJEKT

ADAM BESEDA

STREDNÁ ODBORNÁ ŠKOLA HANDLOVÁ

Študijný odbor: 2561 M Informačné a sieťové technológie
Vedúci ročníkového projektu: Ing. Alena Michalovičová

Dátum odovzdania práce: 6. 3. 2025

HANDLOVÁ 2025

Prihlaska

ČESTNÉ PREHLÁSENIE

Týmto čestne prehlasujem, že tento ročníkový projekt som vypracoval samostatne a iba za použitia zdrojov uvedených v bibliografii. Pri tvorbe projektu som dodržiaval etické normy a autorský zákon.

V Handlovej dňa: _____

podpis

Obsah

Zoznam skratiek

AI – Artificial Intelligence (Umelá inteligencia)

API – Application Programming Interface (Aplikačné programové rozhranie)

BoW – Bag of Words (Vrece slov)

CBT – Cognitive Behavioral Therapy (Kognitívno-behaviorálna terapia)

CPU – Central Processing Unit (Centrálny procesor)

CRM – Customer Relationship Management (Systém riadenia vzťahov so zákazníkmi)

CUDA – Compute Unified Device Architecture (Paralelná výpočtová platforma od NVIDIA)

FAIR – Facebook AI Research (Výskumné laboratórium umelej inteligencie Facebooku)

GPT-4 – Generative Pre-trained Transformer 4 (Štvrtá generácia veľkého jazykového modelu od OpenAI)

GPU – Graphics Processing Unit (Grafická procesorová jednotka)

JSON – JavaScript Object Notation (Štruktúrovaný formát na výmenu údajov)

ML – Machine Learning (Strojové učenie)

NLP – Natural Language Processing (Spracovanie prirodzeného jazyka)

NLTK – Natural Language Toolkit (Nástroj na spracovanie prirodzeného jazyka)

NumPy – Numerical Python (Knižnica na numerické výpočty v Pythone)

OS – Operating System (Operačný systém)

PIL – Python Imaging Library (Knižnica na spracovanie obrázkov v Pythone)

ReLU – Rectified Linear Unit (Rektifikovaná lineárna jednotka)

TF-IDF – Term Frequency - Inverse Document Frequency (Frekvencia výskytu termínu – Inverzná frekvencia dokumentu)

Zoznam obrázkov

Úvod

V súčasnej digitálnej dobe je stále väčší dôraz kladený na starostlivosť o duševné zdravie, pričom technologické riešenia a digitálni asistenti čoraz viac prispievajú k podpore psychickej pohody jednotlivcov. So vzrastajúcou dostupnosťou technológií a obľubou aplikácií zameraných na duševnú podporu vzniká dopyt po chatbotových riešeniach, ktoré by dokázali ponúknuť užívateľom potrebnú pomoc a poradenstvo. Tento ročníkový projekt sa zaoberá vývojom takéhoto chatbota zameraného na duševnú podporu s využitím programovacieho jazyka Python.

Chatboti predstavujú interaktívne softvérové nástroje, ktoré sú schopné viesť rozhovory a reagovať na potreby používateľov prostredníctvom simulácie prirodzenej komunikácie. Cieľom tejto práce je preto vytvoriť prístupný a užívateľsky prívetivý chatbot, ktorý dokáže efektívne komunikovať a poskytovať užívateľom základnú emocionálnu oporu.

Programovací jazyk Python bol zvolený pre tento projekt najmä pre svoju jednoduchosť, čitateľnosť a širokú podporu knižníc na spracovanie prirodzeného jazyka, tiež preto že som s týmto jazykov strávil posledné dva roky a myslím že som sa v ňom dobre zorientoval. Tak isto chcem vyskúšať aké to je pracovať na plnohodnotnom projekte. Verím že to zvládnem a že ma táto ročníková práca posunie na vyššiu úroveň.

Hlavným cieľom tejto projektovej práce je vytvoriť chatbota zameraného na duševnú podporu, ktorý bude implementovaný v jazyku Python s dôrazom na jeho komunikačné schopnosti, prístupnosť a spoľahlivosť. Projekt zahŕňa integráciu techník spracovania prirodzeného jazyka, implementáciu základných princípov psychologickej podpory a poskytovanie následných rád, ktoré chatbot užívateľovi ponúkne.

1 Teoretická časť

1.1.1 Úvod do chatbotov

Chatboty sú softvérové aplikácie, ktoré simulujú ľudskú konverzáciu. Využívajú technológie ako spracovanie prirodzeného jazyka (NLP) a strojové učenie na porozumenie a generovanie odpovedí. Tieto technológie umožňujú chatbotom komunikovať s používateľmi prirodzeným spôsobom, čím zlepšujú ich interakcie a poskytujú rýchlejšie odpovede na otázky.

Popularita chatbotov rýchlo stúpa vďaka ich schopnostiam poskytovať okamžitú podporu a automatizovať úlohy, čo šetrí čas a náklady. Chatboty sú dnes používané v mnohých odvetviach vrátane zákazníckej podpory, marketingu, zdravotníctva, vzdelávania a dokonca aj v oblasti mentálneho zdravia. Ich schopnosť poskytovať nepretržitú podporu ich robí neoddeliteľnou súčasťou moderného sveta. Okrem toho, chatboty prispievajú k digitalizácii a inovácii mnohých procesov, čím umožňujú firmám a organizáciám držať krok s rýchlo meniacim sa technologickým prostredím. Vďaka ich flexibilite a prispôsobivosti sú často prvou voľbou pre firmy, ktoré chcú zlepšiť svoje služby a zvýšiť spokojnosť zákazníkov.

1.1.2 Typy chatbotov

Pravidlové chatboty používajú preddefinované pravidlá a logiku na odpovedanie na otázky. Sú vhodné pre jednoduché a priamočiare úlohy, no ich schopnosti sú obmedzené na preddefinované scenáre. Tento typ chatbotov je efektívny, keď je potrebné riešiť opakujúce sa otázky, ako napríklad otázky týkajúce sa pracovnej doby alebo jednoduchých informácií o produktoch. Pravidlové chatboty fungujú na základe skriptov a sú schopné poskytovať rýchle a konzistentné odpovede.

Inteligentné chatboty môžu analyzovať veľké množstvo dát a prispôsobovať svoje odpovede individuálnym potrebám používateľov, čo z nich robí veľmi hodnotný nástroj v rôznych oblastiach. Tento druh chatbotov je schopný riešiť komplexné problémy a poskytovať personalizovanú podporu na základe kontextu a histórie interakcií s používateľom. Inteligentné chatboty sú tiež schopné integrovať sa s inými systémami a platformami, čím umožňujú zlepšenú spoluprácu a koordináciu medzi rôznymi nástrojmi a technológiami.

1.1.3 Technológie za chatbotmi

Spracovanie prirodzeného jazyka (NLP) umožňuje chatbotom porozumieť a generovať ľudský jazyk. Medzi hlavné techniky NLP patrí tokenizácia, stemming, lemmatizácia a analýza sentimentu. Tokenizácia rozdeľuje text na jednotlivé slová alebo frázy, čím uľahčuje jeho spracovanie. Stemming a lemmatizácia redukujú slová na ich základné tvary, čím sa zlepšuje presnosť analýzy. Analýza sentimentu umožňuje chatbotom rozpoznať emocionálny tón textu, čo vedie k empatickejším odpovediam.

Strojové učenie (ML) zohráva kľúčovú úlohu v zlepšovaní chatbotov. Algoritmy, ako sú neurónové siete a rozhodovacie stromy, umožňujú chatbotom učiť sa z historických interakcií a lepšie predikovať odpovede. Neustály vývoj v oblastiach NLP a ML prispieva k zvyšovaniu presnosti, personalizácie a celkovej efektivity chatbotov, čím sa zlepšuje aj ich využiteľnosť v rôznych oblastiach.

1.1.4 Chatboty v oblasti mentálneho zdravia

Chatboty zohrávajú čoraz významnejšiu úlohu v oblasti mentálneho zdravia, keďže poskytujú nepretržitú podporu, pomáhajú zvládať stres a úzkosť a ponúkajú odporúčania na odbornú pomoc. Príkladom je chatbot Woebot, ktorý využíva kognitívno-behaviorálnu terapiu (CBT) na podporu používateľov so psychickými problémami.

Hlavnou výhodou chatbotov je ich dostupnosť 24/7, čo je dôležité pre ľudí, ktorí potrebujú okamžitú pomoc alebo nemajú prístup k tradičnej terapii. Navyše, anonymita týchto nástrojov môže motivovať používateľov k otvorenejšiemu zdieľaniu svojich problémov. Chatboty sú schopné personalizovať odporúčania na základe individuálnych potrieb, čím zvyšujú efektivitu poskytovanej podpory.

Okrem poskytovania psychologickej podpory môžu chatboty pomáhať pri identifikácii symptómov a navrhovať vhodné stratégie zvládania. V prípade krízových situácií môžu zabezpečiť okamžitý prístup k informáciám a usmerneniam. Taktiež prispievajú k odbúravaniu stigmy spájanej s mentálnym zdravím tým, že ponúkajú vzdelávacie materiály a anonymné konzultácie.

Využívanie chatbotov v tejto oblasti predstavuje inovatívny prístup k duševnej pohode, ktorý umožňuje širšiemu spektru ľudí získať potrebnú podporu a informácie v reálnom čase.

1.1.5 Výhody a nevýhody chatbotov

Medzi hlavné výhody chatbotov patrí ich dostupnosť, škálovateľnosť a úspora nákladov. Chatboty môžu byť k dispozícii kedykoľvek a kdekoľvek a môžu zvládať veľké množstvo interakcií naraz. Týmto spôsobom znižujú potrebu ľudských pracovníkov pre jednoduché úlohy a umožňujú firmám a organizáciám zamerať sa na komplexnejšie úlohy.

Na druhej strane, existujú aj nevýhody, ako sú obmedzené porozumenie zložitého kontextu, závislosť od dát a etické otázky. Chatboty môžu mať problémy s pochopením zložitého kontextu a kvalita ich odpovedí závisí od množstva a kvality tréningových dát. Zabezpečenie súkromia a bezpečnosti používateľských dát je kľúčové a predstavuje významnú výzvu.

Pri navrhovaní a implementácii chatbotov pre mentálne zdravie je dôležité zohľadniť tieto faktory, aby boli bezpečné a účinné. Etické otázky týkajúce sa použitia chatbotov zahŕňajú ochranu súkromia, dôvernosť údajov a zodpovedné používanie technológií. Chatboty musia byť navrhnuté tak, aby dodržiavali prísne štandardy ochrany údajov a poskytovali transparentné informácie o tom, ako sú údaje používateľov spracovávané a ukladané. Okrem toho je dôležité zabezpečiť, aby chatboty nešírili nepravdivé alebo zavádzajúce informácie, čo by mohlo mať negatívny vplyv na používateľov.

Napriek týmto nevýhodám je potenciál chatbotov v oblasti mentálneho zdravia obrovský. Môžu poskytovať okamžitú podporu, pomáhať pri zvládaní stresu a úzkosti a poskytovať odporúčania na odbornú pomoc. Tieto chatboty musia byť navrhnuté s ohľadom na citlivé a etické otázky, aby boli bezpečné a účinné.

Vývoj technológií ako NLP a ML naďalej zlepšuje schopnosti chatbotov a otvára nové možnosti pre ich využitie v rôznych oblastiach. V oblasti mentálneho zdravia môžu byť chatboty obzvlášť užitočné, pretože umožňujú poskytovať podporu a pomoc v reálnom čase a môžu byť k dispozícii kedykoľvek. V kombinácii s pokročilými technikami strojového učenia a spracovania prirodzeného jazyka môžu chatboty ponúkať stále sofistikovanejšie a efektívnejšie riešenia pre rôzne problémy súvisiace s mentálnym zdravím. Chatboty môžu tiež pomôcť pri identifikácii symptómov a poskytovať personalizované odporúčania. Napríklad, ak používateľ uvádza príznaky úzkosti, chatbot môže poskytnúť techniky na zvládanie úzkosti a odporučiť ďalšie kroky, ako je napríklad kontaktovanie odborníka.

1.1.6 Užívateľská skúsenosť a inkluzívny dizajn

Okrem technických a etických aspektov je dôležité zohľadniť aj užívateľskú skúsenosť. Chatboty by mali byť navrhnuté tak, aby boli užívateľsky prívetivé a intuitívne na používanie. Používatelia by mali mať možnosť ľahko komunikovať s chatbotmi a získavať rýchle a relevantné odpovede. Dizajn chatbotov by mal tiež zohľadňovať rôzne kultúrne a jazykové prostredia, aby boli prístupné pre široké spektrum používateľov. Je dôležité, aby chatboty boli schopné komunikovať v rôznych jazykoch a prispôsobiť sa rôznym kultúrnym kontextom. Týmto spôsobom môžu byť chatboty efektívnym nástrojom na celom svete, poskytujúc podporu a pomoc rôznym skupinám ľudí.

Vývoj chatbotov je dynamický a neustále sa vyvíja. Nové technológie a metódy, ako je hlboké učenie a pokročilé NLP techniky, neustále zlepšujú schopnosti chatbotov. Tieto pokroky umožňujú chatbotom poskytovať stále presnejšie a relevantnejšie odpovede a zlepšovať celkovú používateľskú skúsenosť. Vývojári a výskumníci neustále hľadajú nové spôsoby, ako zlepšiť efektivitu a schopnosti chatbotov, čím rozširujú ich potenciál v rôznych oblastiach. Chatboty majú tiež potenciál na integráciu s inými technológiami, ako sú nositeľné zariadenia a internet vecí (IoT). Tieto integrácie môžu umožniť chatbotom zhromažďovať a analyzovať údaje z rôznych zdrojov, čím poskytujú ešte presnejšie a personalizované odpovede. Napríklad chatbot integrovaný s nositeľným zariadením môže monitorovať fyzické príznaky stresu alebo úzkosti a poskytnúť používateľovi okamžitú podporu a odporúčania.

1.1.7 Budúcnosť chatbotov

Budúcnosť chatbotov v oblasti mentálneho zdravia je sľubná. So zlepšujúcimi sa technológiami a rastúcim povedomím o dôležitosti mentálneho zdravia majú chatboty potenciál poskytovať ešte efektívnejšiu a prístupnejšiu podporu. Pri správnom navrhovaní a implementácii môžu chatboty zohrávať významnú úlohu v zlepšovaní mentálneho zdravia a celkového blaha ľudí po celom svete. Ich schopnosti sa neustále zlepšujú vďaka pokrokom v technológiách, ako sú NLP a strojové učenie. Napriek určitým výzvam a obmedzeniam, potenciál chatbotov je obrovský a môže priniesť významné prínosy pre spoločnosť.

1.1.8 Dôveryhodnosť a transparentnosť

Pri návrhu a implementácii chatbotov pre mentálne zdravie je dôležité zvážiť aj aspekty ako dôveryhodnosť, spoľahlivosť a transparentnosť. Používatelia musia veriť, že informácie a rady, ktoré dostávajú od chatbotov, sú presné a dôveryhodné. Transparentnosť v komunikácii o tom, ako chatboty fungujú, ako spracúvajú údaje a aké sú ich obmedzenia, je kľúčová pre budovanie dôvery medzi používateľmi a chatbotmi. Spoľahlivosť chatbotov znamená, že by mali byť schopné poskytovať konzistentné a kvalitné odpovede v rôznych situáciách. Ďalším aspektom, ktorý treba zvážiť, je inkluzívny dizajn chatbotov. Chatboty by mali byť navrhnuté tak, aby boli prístupné pre rôzne skupiny používateľov, vrátane tých s rôznymi schopnosťami a potrebami. To zahŕňa zabezpečenie, že chatboty sú kompatibilné s asistenčnými technológiami, ako sú čítačky obrazovky, a že používajú jednoduchý a zrozumiteľný jazyk. Inkluzívny dizajn umožňuje, aby chatboty boli užitočné a prístupné pre čo najširšie spektrum používateľov, čím zvyšuje ich efektivitu a prínos.

1.1.9 Aktualizácie a zlepšovania

V neposlednom rade je dôležité, aby chatboty v oblasti mentálneho zdravia boli neustále aktualizované na základe najnovších výskumov a poznatkov v oblasti psychológie a psychiatrie. Týmto spôsobom môžu poskytovať najaktuálnejšie a najefektívnejšie rady a podporu. Aktualizácie a zlepšovania by mali byť pravidelné, aby chatboty mohli reagovať na nové výzvy a meniace sa potreby používateľov. Napríklad nové výskumy môžu priniesť nové techniky a prístupy na zvládanie stresu alebo úzkosti, ktoré by mali byť integrované do chatbotov čo najskôr.

1.1.10 Záver teoretickej časti

Chatboty predstavujú významný pokrok v technológii a majú potenciál priniesť významné výhody pre spoločnosť. Ich využitie v oblasti mentálneho zdravia je obzvlášť prínosné, pretože umožňujú poskytovať podporu a pomoc širokému spektru používateľov. S neustálym vývojom a zdokonaľovaním technológií môžu chatboty naďalej zlepšovať kvalitu života a poskytovať inovatívne riešenia pre rôzne problémy. Budúcnosť chatbotov je svetlá a plná možností, ktoré môžu zlepšiť životy ľudí po celom svete. Tieto inovácie a pokroky umožňujú chatbotom riešiť stále komplexnejšie problémy a poskytovať personalizovanú podporu a pomoc používateľom v reálnom čase. Je dôležité, aby vývoj chatbotov pokračoval v súlade s etickými zásadami a zameriaval sa na zlepšovanie kvality života používateľov.

Pri návrhu a implementácii chatbotov pre mentálne zdravie je dôležité zvážiť aj aspekty ako dôveryhodnosť, spoľahlivosť a transparentnosť. Používatelia musia veriť, že informácie a rady, ktoré dostávajú od chatbotov, sú presné a dôveryhodné. Transparentnosť v komunikácii o tom, ako chatboty fungujú, ako spracúvajú údaje a aké sú ich obmedzenia, je kľúčová pre budovanie dôvery medzi používateľmi a chatbotmi. Spoľahlivosť chatbotov znamená, že by mali byť schopné poskytovať konzistentné a kvalitné odpovede v rôznych situáciách. Ďalším aspektom, ktorý treba zvážiť, je inkluzívny dizajn chatbotov. Chatboty by mali byť navrhnuté tak, aby boli prístupné pre rôzne skupiny používateľov, vrátane tých s rôznymi schopnosťami a potrebami. To zahŕňa zabezpečenie, že chatboty sú kompatibilné s asistenčnými technológiami, ako sú čítačky obrazovky, a že používajú jednoduchý a zrozumiteľný jazyk. Inkluzívny dizajn umožňuje, aby chatboty boli užitočné a prístupné pre čo najširšie spektrum používateľov, čím zvyšuje ich efektivitu a prínos.

V neposlednom rade je dôležité, aby chatboty v oblasti mentálneho zdravia boli neustále aktualizované na základe najnovších výskumov a poznatkov v oblasti psychológie a psychiatrie. Týmto spôsobom môžu poskytovať najaktuálnejšie a najefektívnejšie rady a podporu. Aktualizácie a zlepšovania by mali byť pravidelné, aby chatboty mohli reagovať na nové výzvy a meniace sa potreby používateľov. Napríklad nové výskumy môžu priniesť nové techniky a prístupy na zvládanie stresu alebo úzkosti, ktoré by mali byť integrované do chatbotov čo najskôr.

2 Praktická časť

2.1 Spracovanie prirodzeného jazyka

Pri tvorbe ročníkového projektu sme začali spracovaním prirodzeného jazyka (NLP) pomocou knižnice NLTK. Z dôvodu nefunkčnosti slovenskej jazykovej podpory v knižnici NLTK je možné komunikovať s chatbotom iba v anglickom jazyku.

2.1.1 Vysvetlenie spracovania prirodzeného jazyka v mojom programe

- Import knižníc – Použili sme knižnicu NLTK ktorá spracováva prirodzený jazyk a knižnicu numpy, ktorá slúži na efektívnu prácu s poľami a umožňuje s nimi vykonávať matematické operácie oveľa rýchlejšie.
- Tokenizácia – Vytvorili sme funkciu tokenize do ktorej vkladáme reťazec sentence (veta) a rozdeľujeme ju na samotné slová, pomocou funkcie word_tokenize z knižnice NLTK.
- Stemming – Tu sme vytvorili funkciu stem do ktorej vkladáme už samotné slová a odstraňujeme z nich prípony alebo predpony. Zostáva len slovný základ. Definovali sme objekt stemmer ktorý uchováva inštanciu triedy PorterStemmer z knižnice NLTK. Vo funkcii už len okrem odstránenia prípon aj zmeníme všetky písmena na malé.

```
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())
```

Obrázok 1: Spracovanie prirodzeného jazyka

2.1.2 NLTK

NLTK je popredná platforma na vytváranie programov v jazyku Python na prácu s ľudským jazykom. Ponúka ľahko použiteľné rozhrania pre viac ako 50 korpusov a lexikálnych zdrojov, ako je napríklad WordNet. Obsahuje sadu knižníc na spracovanie textu, tokenizáciu, stemming, tagovanie alebo parsovanie.

Vďaka praktickej príručke, ktorá predstavuje základy programovania spolu s témami z počítačovej lingvistiky, a komplexnej dokumentácii API je NLTK vhodný pre lingvistov, inžinierov, študentov, pedagógov a výskumníkov.

NLTK je k dispozícii pre systémy Windows, Mac OS X a Linux. Najlepšie zo všetkého je, že NLTK je bezplatný projekt s otvoreným zdrojovým kódom, riadený komunitou.
(www.nltk.org)

2.1.3 NumPy

NumPy je knižnica programovacieho jazyka Python, ktorá poskytuje infraštruktúru na prácu s vektormi, maticami a viacrozmernými poľami vo všeobecnosti.

Okrem potrebných dátových štruktúr poskytuje aj množstvo užitočných matematických funkcií, ktoré s týmito štruktúrami pracujú, napríklad základné metódy lineárnej algebry alebo diskrétnu Fourierovu transformáciu.

Podporu ďalších funkcií pre vedecké výpočty nad dátovými štruktúrami z knižnice NumPy poskytuje knižnica SciPy, podľa ktorej je projekt pomenovaný a ktorá zastrešuje obe tieto a mnohé ďalšie knižnice. (https://cs.wikipedia.org/wiki/NumPy)

2.2 Tvorba tréningových dát

Pre správnu funkciu chatbota je potrebné vytvoriť tréningový model a naučiť ho rozpoznať rôzne typy vstupov a následne ich priradiť k správnym kategóriám. Model sa takto naučí rozpoznávať vstupy, na konci dáta uloží a uchová na ďalšie použitie. Z tohto dôvodu sa po vykonaných zmenách spúšťa tréning na novo.

Táto časť kódu spočíva v tom že si importujeme modul json, ktorý budeme potrebovať na čítanie dát z nášho json súboru. Následne importujeme funkcie tokenizácie a stemmingu z predošlého kódu. Pre vytváranie a trénovanie neurónových sietí použijeme knižnicu PyTorch.

Teraz potrebujeme v programe čítať súbor intents.json. V tomto súbore mám spísané rôzne frázy a slová, ktoré majú určitý zámer a podľa toho bude chatbot odpovedať na otázky. Pomocou cyklu pre každý zámer vygenerujeme tag a uložíme do zoznamu tags. Následne voláme naše funkcie a vkladáme do nich vety, ktoré sú uložené v našom json súbore. Tokenizované slová uložíme do vytvoreného zoznamu all_words. Do zoznamu xy uložíme slová spoločne s ich tagom. V ďalšom kroku program prechádza všetky slová v zozname all_words pomocou funkcie stem. Slová v respektíve znaky, ktoré sú v zozname ignore_words vynechá. Následne program vynechá duplikované slová a zoradí ich podľa abecedy. To isté aplikuje aj na zoznam tags.

```
import json
from nltk_utils import tokenize, stem, bag_of_words
import numpy as np

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

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))
```

Obrázok 2: Tréningové dáta

2.3 Konverzia textu do numerického formátu (BoW)

Na konverziu textu do numerického formátu využijeme jednoduchú a efektívnu metódu bag of words (BoW). Princíp tejto metódy v našom kóde spočíva v tom že vytvoríme vektor dĺžky len(all_words) čiže počtu slov. Tento vektor bude obsahovať iba nuly. Potom pomocou cyklu prejdeme každé slovo v json súbore a ak sa dané slovo nachádza aj v zozname all_words na príslušnom indexe zmení hodnotu na číslo 1.

Výhody metódy:

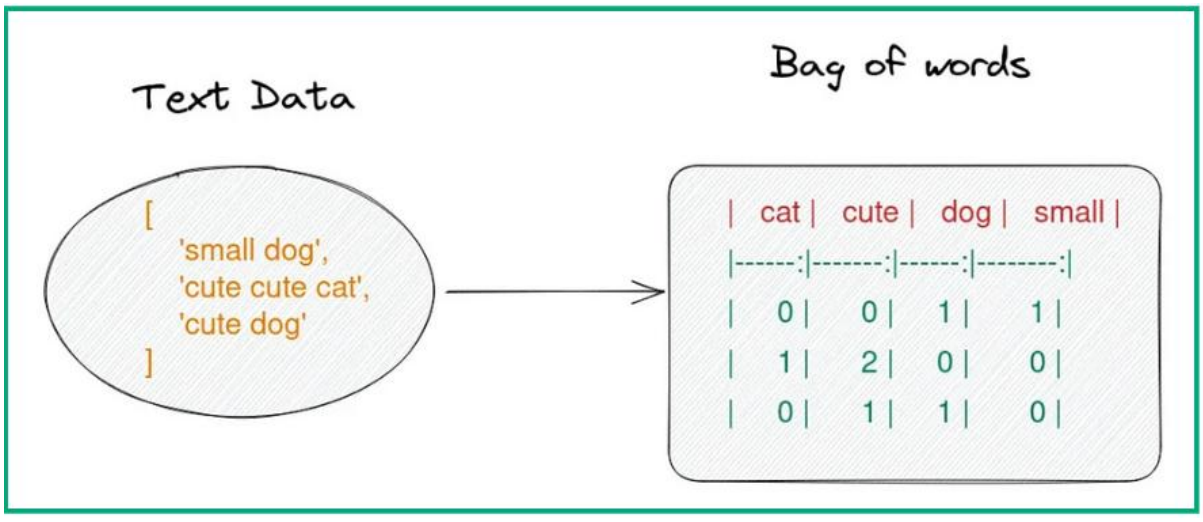
- Jednoduchosť a efektívnosť
- Vhodné pre základné NLP projekty ako napríklad ChatBot

Nevýhody metódy:

- Nepozná synonymá (slová goodbye a bye sú pre neho rôzne slová)
- Nevhodný pre projekty s veľkým obsahom slov

Alternatívy tejto metódy sú TF-IDF alebo Word Embeddings. Pre môj projekt je ale najlepšia a najefektívnejšia práve metóda BoW.

Na nasledujúcom obrázku si môžeme všimnúť ako táto metóda funguje.



Obrázok 3: Vysvetlenie BoW, https://datascientyst.com/create-a-bag-of-words-pandas-python/

2.3.1 Implementácia BoW

V našom programe BoW implementujeme do časti nltk_utils, kde máme definované základné funkcie na spracovanie prirodzeného jazyka. Ako prvé sme si definovali funkciu bag_of_words do ktorej vkladáme hodnoty tokenized_sentence a all_words.

V cykle sa prechádza každé slovo v zadanej vete a je stemované. Následne vytvoríme nulový vektor, pričom každá nula predstavuje jedno slovo. Nakoniec prechádzame každé slovo a ak sa slovo nachádza v zozname tokenized_sentence index sa zmení z nuly na jednotku, teraz už máme definované všetky funkcie na spracovanie prirodzeného jazyka.

```
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
```

Obrázok 4: Implementácia BoW

2.3.2 Tréningové dáta pre BoW

Inicializujeme prázdne zoznamy X_train a y_train.

- X_train bude obsahovať číselný vektor
- y_train bude obsahovať tagy reprezentujúce kategórie odpovedí chatbota.

Následne tieto zoznamy s hodnotami pomocou knižnice NumPy prevedieme na tak zvané numpy zoznamy s ktorými sa ľahšie vykonávajú matematické operácie.

```
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 = np.array(X_train)
y_train = np.array(y_train)
```

Obrázok 5: Tréningové dáta

2.4 Vytvorenie Datasetu a DataLoadera pomocou PyTorch

2.4.1 Definícia triedy ChatDataset

Trieda ChatDataset dedí z vlastnosti a metódy z triedy Dataset z knižnice PyTorch. Následne definujme vlastný dataset pre trénovanie modelu. V ďalšom kroku do premennej self.n_samples uložíme počet vzoriek v tréningovom datasete a do premenných self.x_data a self.y_data uložíme X_train a y_train.

Ďalej vytvoríme funkciu ktorá nám umožní dataset indexovať pomocou dataset[i], vrátime dvojicu x_data[index] a y_data[index] pričom x sú číselné vektory a y je tag kategórie. Ako poslednú vytvoríme funkciu v respektíve metódu v datasete, ktorá nám vráti celkový počet viet v datasete.

```
class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train

    #dataset[idx]
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    def __len__(self):
        return self.n_samples
```

Obrázok 6: Trieda ChatDataset

2.4.2 Definovanie Hyperparametrov

Hyperparametre určujú, ako sa bude model správať počas tréningu a ako rýchlo bude spracovávať informácie. S týmito hodnotami je možné experimentovať pre dosiahnutie najlepšieho výkonu. V našom kóde máme nastavené napríklad že model spracuje 8 vzoriek naraz pri každom kroku učenia sa. Medzi ďalšie parametre patria napríklad počet neurónov na skrytej vrstve, počet možných tried alebo kategórii, dĺžka vstupného vektora, rýchlosť učenia sa alebo počet opakovaní.

```
# Hyperparameters
batch_size = 8
hidden_size = 8
output_size = len(tags)
input_size = len(X_train[0])
learning_rate = 0.001
```

Obrázok 7: Hyperparametre

2.4.3 Vytvorenie datasetu a Dataloadera

Vytvoríme inštanciu triedy ChatDataset tak že vytvoríme objekt dataset, tento objekt teraz uchováva dáta, ktoré sme vložili do triedy ChatDataset.

2.4.4 DataLoader

DataLoader je trieda definovaná v knižnici PyTorch a používa sa na načítavanie dát počas tréningu, prijíma dataset a nastavuje mu rôzne parametre. Pre DataLoader definujeme aký dataset chceme použiť, definujeme aj koľko vzoriek má DataLoader naraz spracovávať, túto hodnotu sme si už definovali v Hyperparametroch.

Shuffle=True znamená, že sa dáta náhodne premiešajú pred tým ako sa začnú spracovávať. Pre náš tréningový model je to veľmi dôležité, nakoľko nechceme aby sa model naučil pracovať len na jednom poradí dát. Keby je shuffle nastavené na False stalo by sa že chatbot by bol naučený na interakcie s užívateľom len v jednom poradí napríklad pozdrav, otázka a odpoveď, keby ale používateľ zadá ako prvú otázku model bude mať problém porozumieť.

```
dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset, batch_size=batch_size, shuffle=True)
```

Obrázok 8: DataLoader

2.4.5 PyTorch

PyTorch je open-source knižnica založená na knižnici Torch a určená primárne na účely strojového učenia (machine learning) a hĺbkového učenia (deep learning). Táto knižnica sa najčastejšie používa v kombinácii s jazykom Python, avšak je možné ju použiť aj s jazykom C++. Jej najväčšie využitie je v oblasti Spracovania prirodzeného jazyka (NLP). Je vyvíjaná hlavne výskumným tímom Facebooku pre umelú inteligenciu (FAIR = Facebook's AI Research lab).

Vlastnosti PyTorchu:

- Jednoduchosť na naučenie a používanie.
- Obsahuje rozsiahlu dokumentáciu a množstvo tutoriálov.

2.5 Implementácia neurónovej siete

Pomocou knižnice PyTorch implementujeme jednoduchú umelú neurónovú sieť. Vytvoríme model chatbotu, ktorý bude prijímať vstupné dáta a spracovávať cez lineárne vrstvy.

2.5.1 Neurónová sieť

Neurónové siete sú v súčasnosti mnohými považované za jeden z najlepších algoritmov strojového učenia. Vo vybraných úlohách dokážu priniesť výsledky s veľmi vysokou presnosťou a majú pomerne širokú škálu použitia.

Napríklad pri aplikáciách počítačového videnia či spracovaní zvuku. V závislosti od použitej architektúry ich niekedy možno označiť aj ako hlboké strojové učenie.

Rovnako ako mnohé iné algoritmy strojového učenia aj neurónové siete fungujú tak, že vytvoríme model, ktorý následne natrénujeme na čo najväčšom množstve označených dát. Na základe týchto známych trénovacích dát sa tak model naučí „predpovedať“ výsledky nových neznámych prípadov.(Úvod do neurónových sietí - Umelá Inteligencia.sk)

2.5.2 Trieda NeuralNet

Definujeme triedu NeuralNet ktorá dedí z nn.Module knižnice PyTorch. Pre správnu funkčnosť musia všetky neurónové siete implementované pomocou knižnice PyTorch dediť z nn.Module.

2.5.3 Inicializácia

Definujeme funkciu __init__ ktorá sa používa na inicializáciu parametrov neurónovej siete. Vkladáme do nej parametre input_size čiže počet vstupných prvkov, hidden_size čiže počet neurónov na skrytých vrstvách. Následne zavoláme rodičovskú triedu, aby sa správne inicializovala základná funkcionalita neurónovej siete.

2.5.4 Prvá lineárna vrstva

Prvá lineárna vrstva prijíma vstup o rozmere input_size. Vytvára výstup o rozmere hidden_size.

2.5.5 Druhá lineárna vrstva

Lineárna vrstva číslo dva prijíma vstup o rozmere hidden_size z prvej vrstvy, výstup vytvára rovnako o rozmere hidden_size, zachovanie rovnakého rozmeru výstupov nám zabezpečí flexibilnejšie učenie sa komplexných vzorov.

2.5.6 Tretia lineárna vrstva

Tretia, alebo aj výstupná lineárna vrstva prijíma vstup o rozmere hidden_size z predchádzajúcej vrstvy. Výstup má rozmery num_classes teda počet tried ktoré bude model predikovať.

2.5.7 Aktivačná funkcia ReLU

ReLU je nelineárna aktivačná funkcia definovaná matematickým vzorcom:

$$ReLU(x)=max(0,x)$$

Zjednodušene, funkcia ReLU nám zmení výstup na hodnotu nula ak je pôvodný výstup záporný. Ak je pôvodný výstup kladný, zostane nezmenený.

2.5.8 Funkcia forward

Funkcia forward nám definuje ako dáta prechádzajú neurónovou sieťou. Do funkcie vkladáme x ako vstup. X bude prechádzať prvou lineárnou vrstvou a výsledok uložíme do premennej out. Následne sa na výsledok aplikuje ReLU. Potom upravený výstup prechádza druhou vrstvou a na výstup sa opäť aplikuje ReLU. Následne upravený výstup z druhej vrstvy prechádza treťou vrstvou. Tento výstup vrátime.

```python
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)
        return out
```

Obrázok 9: Neurónová sieť

2.5.9 Použitie GPU na zrýchlenie trénovania

GPU dokáže vďaka viacerým paralelným jadrám pracovať rýchlejšie na výpočtoch strojového učenia. V našom kóde budeme zisťovať či zariadenie má GPU a nainštalované ovládače CUDA. To znamená grafické karty Nvidia, ktoré PyTorch podporuje. Ak zariadenie nemá GPU s CUDA ovládačmi, použije sa CPU, ktoré je síce pomalšie, ale má ho každý počítač.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Obrázok 10: Použitie GPU

2.5.10 Inicializácia modelu do tréningu

Vytvoríme inštanciu neurónovej siete NeuralNet s parametrami:

- **input_size**: Počet neurónových vstupov.
- **hidden_size**: Počet neurónov na skrytej vrstve.
- **output_size**: Počet výstupných neurónov.

Funkcia straty

Funkcia straty nám meria ako ďaleko sú skutočné hodnoty od predikovaných hodnôt modelu. V našom kóde si definujeme funkciu CrossEntropyLoss z knižnice Numpy.

2.5.11 Nastavenie optimalizátora

Z knižnice PyTorch použijeme najpopulárnejší optimalizátor pre neurónové siete s menom Adam optimizer. Vložíme do neho všetky trénovateľné parametre a rýchlosť učenia.

2.5.12 Adam optimizer

Adam optimizer alebo aj Adaptive Moment Estimation optimizer je algoritmus pre optimalizačnú technológiu gradientného zostupu. Optimalizátor je veľmi efektívny pri práci s veľkým množstvom dát alebo parametrov.

Spotrebúva málo pamäte a preto je stále najpopulárnejší optimalizátor používaný v neurónových sieťach. Optimalizátor je kombináciou dvoch algoritmov. (www.geeksforgeeks.org/adam-optimizer)

2.5.13 Optimalizácia

Vytvoríme vonkajší cyklus, ktorý sa bude opakovať num_epoch krát. Následne vytvoríme vnútorný cyklus, ktorý bude prechádzať cez samotné dáta. Pomocou premennej train_loader sme rozdelili dáta na batche alebo dávky. Každý batch obsahuje vstupné dáta čiže words a očakávané výstupy labels. Následne presunieme dáta na správne zariadenie CPU alebo GPU.

1. **Forward Pass** – Vstupné dáta posielame do modelu a model predikuje výstupy, pravdepodobnosti pre každú triedu. Následne sa vypočíta strata medzi predikovanými výstupmi a reálnymi hodnotami.
2. **Backward Pass** – Ako prvé musíme vynulovať gradienty z predchádzajúcej iterácie. Následne môžeme vypočítať gradienty pre všetky parametre nášho modelu podľa chyby.

Teraz nastavíme aby sa po 100 epochách vypísala aktuálna epocha a hodnota straty. Po ukončení tréningu sa vypíše posledná hodnota straty.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = NeuralNet(input_size, hidden_size, output_size)

# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(device, dtype=torch.int64)

        # forward
        outputs = model(words)
        loss = criterion(outputs, labels)

        # backward and optimizer step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

if (epoch +1) % 100 == 0:
    print(f'epoch {epoch+1}/{num_epochs},loss={loss.item():.4f}')

print(f'final loss, loss={loss.item():.4f}')
```

Obrázok 11: Optimalizácia

2.5.14 Tvorba dátového slovníka

Sem budeme ukladať dôležité informácie o modeli, ktoré budeme potrebovať opätovne načítať a použiť. Vytvoríme slovník data, do neho budeme vkladať potrebné informácie.

2.5.15 Uloženie dát

Najskôr si definujeme súbor do ktorého budeme tréningové dáta ukladať. Použijeme príponu .pth tento typ súborov sa používa v knižnici PyTorch na ukladanie modelov a tensorov.

Následne pomocou torch.save uložíme tréningové dáta do súboru. Vďaka tomuto nebudeme musieť pred každým spustením chatbotu trénovať model. Následne už len vypíšeme správu o úspešnom tréningu a uložení trénovacích dát.

```
data = {
    "model_state": model.state_dict(),
    "input_size": input_size,
    "output_size": output_size,
    "hidden_size": hidden_size,
    "all_words": all_words,
    "tags": tags
}

FILE = "data.pth"
torch.save(data, FILE)

print(f'training complete. file saved to {FILE}')
```

Obrázok 12: Uloženie dát

2.6 Implementácia chatu

2.6.1 Import knižníc, súborov a načítanie tréningových dát

Ako prvú vec si musíme importovať potrebné knižnice a funkcie ktoré sme si už vytvorili. Následne zisťujeme či bude chatbot fungovať na GPU alebo CPU.

Teraz načítame súbor intents.json kde máme uložené zámery a možné odpovede. Pomocou json.load(f) načítame obsah do slovníka. Ako ďalšiu vec načítame uložené tréningové dáta. Z dát vyberéme informácie ako počet vstupných prvkov, počet neurónov na skrytých vrstvách, počet výstupných tried, zoznam všetkých slov, tagy alebo stav modelu.

2.6.2 Inicializácia modelu

Vytvoríme nový model triedy NeuralNet s parametrami input_size, hidden_size a output_size. Následne presunieme model či už na GPU alebo CPU. V ďalšom kroku načítame parametre trénovaného modelu a prepneme ho do hodnotiaceho režimu, aby sa počas predikcie vypli určité tréningové mechanizmy ako napríklad dropout.

Dropout je technika regulácie, ktorá sa používa na zníženie pretrénovania. Počas tréningovej fázy dropout náhodne vypne niektoré neuróny na skrytých vrstvách, aby zabránil pretrénovaniu. Ak by sa dropout nevypol mohol by vypínať niektoré neuróny a to by viedlo k nepresným výpočtom a predikciám.

```
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 = "HappyMind"
```

Obrázok 13: Model

2.6.3 Inicializácia chatbota

Teraz sa pustíme do tvorby samotného chatbota, ktorého si pomenujeme HappyMind v slovenčine šťastná myseľ. Vypíšeme úvodnú správu pre používateľa, ktorého informujeme že môže začať chatovať a pre ukončenie musí napísať quit. Pomocou cyklu while zabezpečíme že chatbot bude fungovať až pokiaľ používateľ nezadá quit. Ďalej pridáme slovo You: na začiatok riadku do ktorého bude používateľ písať.

2.6.4 Spracovanie vstupnej vety

Pomocou funkcie tokenize vetu rozdelíme na jednotlivé slová, následne prevedieme slová na číselné vektory a upravíme tvar vektora na požadovaný formát to znamená že budú na jednom riadku a bude ich rovnako ako počet slov.

```
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)
```

Obrázok 14: Spracovanie vstupnej vety

2.6.5 Výber odpovede

Aplikujeme funkciu softmax z knižnice PyTorch na výstupy modelu, aby model získal pravdepodobnosti pre každú kategóriu. Ak je pravdepodobnosť vyššia ako 75%, chatbot vyberie odpoveď z preddefinovaných odpovedí pre daný tag. Ak pravdepodobnosť nedosahuje hodnoty 75% chatbot odpovie že nechápe danej požiadavke.

```
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..."
```

Obrázok 15: Výber odpovede

2.7 JSON databáza

2.7.1 Json

JSON je ľahký formát na výmenu údajov. Ľudia ho môžu ľahko čítať a zapisovať a stroje ho môžu ľahko analyzovať a generovať. Je založený na podmnožine programovacieho jazyka JavaScript. JSON je textový formát úplne nezávislý od jazyka, ale používa konvencie známe programátorom jazykov rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a ďalšie). Vďaka tomu je JSON ideálnym jazykom na výmenu údajov. (www.json.org/json-cz.html)

2.7.2 Naplnenie databázy

Pre správnu funkčnosť chatbota, potrebujeme zabezpečiť čo najväčší objem dát pre tréning. Tieto dáta by mali byť rôznorodé a kvalitné, aby sa mal model na čom učiť. Chatbot má poskytovať rady a informácie o rôznych duševných náladách a stavoch, nie je určený na udržiavanie zdĺhavej komunikácie. Preto ho naplníme informáciami o náladách a duševných stavoch, ako napríklad smútok, depresia, úzkosť a ďalšie. Keď sa užívateľ bude pýtať mimo tému chatbot odpovie že danej otázke nerozumie. Čím viac rôznych interakcií a otázok databáza obsahuje, tým lepšie bude chatbot reagovať na rôzne požiadavky používateľov a poskytovať kvalitné odpovede.

2.7.3 Štruktúra a kategorizácia dát

Pre databázu sme použili formát JSON. Dáta sme rozdelili do rôznych kategórií, tak zvaných tagov, ktoré reprezentujú konkrétne témy interakcie. Každý tag je spojený s množstvom vzorcom otázok, tak zvaných patterns, a odpovedí responses.

```
"intents": [
  {
    "tag": "greeting",
    "patterns": [
      "Hi",
      "Hey",
      "How are you",
      "Is anyone there?",
      "Hello",
      "Good day",
      "What's up?",
      "How's it going?"
    ],
    "responses": [
      "Hello! I'm here to guide you with tips and exercises for your mental well being.",
      "Hi there! Let me help you with some relaxation techniques.",
      "Hey, welcome! How can I assist you in improving your mood today?",
      "Hello! I can share some exercises and tips to help you feel better.",
      "Good day! I'm here with advice and exercises to improve your mental health."
```

Obrázok 16: JSON databáza

2.8 Používateľské rozhranie

2.8.1 Používateľské rozhranie v jazyku Python

Používateľské rozhranie je rozhodujúcim prvkom, ktorý zabezpečuje interakciu medzi používateľom a aplikáciou. Python poskytuje rôzne nástroje a knižnice na tvorbu používateľských rozhraní, ktoré umožňujú vývojárovi vytvoriť aplikácie s grafickým alebo textovým rozhraním. K dispozícii sú knižnice ako Tkinter, ktorá je súčasťou štandardnej knižnice Pythonu, ale aj pokročilejšie nástroje ako PyQt alebo Kivy, ktoré umožňujú vytvárať interaktívne aplikácie s bohatším vizuálnym prostredím. V našom prípade sme sa rozhodli pre knižnicu Tkinter, avšak trošku vylepšenú s názvom CustomTkinter.

2.8.2 Knižnica CustomTkinter

CustomTkinter je knižnica používateľského rozhrania Python založená na knižnici Tkinter, ktorá poskytuje nové, moderné a plne prispôsobiteľné widgety. Sú vytvorené a používané ako bežné widgety Tkinter a možno ich použiť aj v kombinácii s bežnými prvkami Tkinter. CustomTkinter získame konzistentný a moderný vzhľad vo všetkých desktopových platformách. (www.customtkinter.tomschimansky.com)

2.8.3 Výhody knižnice CustomTkinter oproti knižnici Tkinter

CustomTkinter ponúka moderný vzhľad, ktorý je prispôsobený súčasným trendom v dizajne používateľských rozhraní, zatiaľ čo Tkinter má základný a miestami aj zastaralý vzhľad. Ďalšou veľkou výhodou je že knižnica CustomTkinter podporuje tmavý režim a lepšia podpora pre responzívne rozhranie. To znamená že aplikácie lepšie reagujú na rôzne veľkosti obrazoviek a ľahšie sa prispôsobia k zmene.

2.9 Tvorba používateľského rozhrania

2.9.1 Import knižníc

Ako prvé sme si importovali knižnice:

- **CustomTkinter** sa používa na vytvorenie moderného používateľského rozhrania.
- **chat.py** obsahuje funkcie get_response() na generovanie odpovedí a premennú bot_name, ktorá reprezentuje názov chatbota.
- **PIL (Pillow)** je využitá na načítanie obrázka, konkrétne loga aplikácie.

2.9.2 Grafické nastavenia a inicializácia aplikácie

V úvode sme definovali základné a často využívané farby a štýly, ktoré určujú dizajn aplikácie. Následne sme vytvorili triedu App ktorá dedí od ctk.CTk, čo znamená, že obsahuje všetky funkcie potrebné na vytvorenie hlavného okna. Pri inicializácii sa automaticky volá metóda intro(), ktorá nastavuje základné rozhranie.

2.9.3 Úvodná obrazovka aplikácie

Metóda intro() definuje prvé okno aplikácie. Nastavili sme názov a veľkosť okna a zakázali manipuláciu s jeho veľkosťou. Ďalej sme nastavili vzhľad aplikácie na svetlý režim a modrú farebnú tému.

V úvodnej obrazovke sa nachádza:

1. **Hlavička s logom aplikácie**, ktoré je načítané cez knižnicu PIL.
2. **Uvítací text**, ktorý víta používateľa.
3. **Vstupné pole,** pre zadanie mena.
4. **Tlačidlo „Štart“**, ktoré po kliknutí spustí hlavnú obrazovku chatbota.
5. **Sekcia „O projekte“**, ktorá obsahuje text vysvetľujúci účel aplikácie a dôvod použitia anglického jazyka v chatbote.

2.9.4 Hlavná obrazovka chatu

Po zadaní mena používateľa a kliknutí na tlačidlo „Štart“ alebo klávesy enter sa spustí metóda _setup_main_window(), ktorá inicializuje hlavné okno konverzácie. Toto okno obsahuje:

- **Hlavičku**, s logom a textom.
- **Hlavné textové pole pre zobrazovanie správ**, ktoré je deaktivované, aby do neho používateľ nemohol písať a meniť obsah.
- **Vstupné pole**, na zadávanie správ.
- **Tlačidlo „Send“**, ktoré odosiela správu chatbotovi.

```
def _setup_main_window(self):
    for widget in self.winfo_children():
        widget.destroy()
    self.title("HappyMind")
    self.geometry("600x750")
    self.resizable(False, False)

    ctk.set_appearance_mode("light")
    ctk.set_default_color_theme("blue")

    self.configure(fg_color=BG_COLOR)

    #Header
    self.header_frame = ctk.CTkFrame(self, fg_color=BG_BLUE, corner_radius=15)
    self.header_frame.pack(fill="x", pady=(10, 0), padx=10)

    img = ctk.CTkImage(light_image=Image.open('logo.png'),
    size=(70,70))

    my_label = ctk.CTkLabel(self.header_frame, text="", image=img)
    my_label.pack(pady=7, padx=7, side="left")
```

Obrázok 17: Hlavná obrazovka chatu

2.9.5 Funkcie pre interakciu s chatbotom

Následne sme zadefinovali funkcie pre správne fungovanie chatbota.

- **Funkcia enter()** sa aktivuje po stlačení klávesy Enter alebo kliknutí na tlačidlo. Získa text z vstupného poľa, vymaže ho a následne zavolá funkciu get_response(), ktorá vráti odpoveď od chatbota. Táto odpoveď sa následne zobrazí v textovom poli.
- **Funkcia _insert_message()** zabezpečuje, že sa všetky správy zobrazia v textovom poli.
- **Funkcia clear_placeholder() a restore_placeholder()** sa starajú o odstránenie a obnovenie predvoleného textu v poli na zadávanie správ.

2.9.6 Spustenie aplikácie

Na konci sme potrebovali už len zabezpečiť aby sa aplikácia spustila. Zabezpečili sme to pomocou podmienky if a to tak že ak je tento súbor spustený ako hlavný program, vytvorí sa inštancia triedy App a spustí sa metóda run(), ktorá inicializuje okno aplikácie.

```
    def enter(self, event):
        msg = self.msg_entry.get()
        if msg and msg != "Type your message here...":
            self.msg_entry.delete(0, "end")
            self._insert_message(f"{self.user_name}: {msg}\n\n")
            response = get_response(msg)
            self._insert_message(f"{bot_name}: {response}\n\n", bot=True)

    def _insert_message(self, msg, bot=False):
        self.text_widget.configure(state="normal")
        self.text_widget.insert("end", msg,)
        self.text_widget.configure(state="disabled")
        self.text_widget.see("end")

    def clear_placeholder(self, event):
        if self.msg_entry.get() == "Type your message here...":
            self.msg_entry.delete(0, "end")

    def restore_placeholder(self, event):
        if not self.msg_entry.get():
            self.msg_entry.insert(0, "Type your message here...")

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

Obrázok 18: Používateľské rozhranie

2.10 Testovanie aplikácie

2.10.1 Presnosť a relevantnosť odpovedí chatbota

Po dokončení aplikácie sme sa zamerali na testovanie a simuláciu reálnych otázok od používateľa. Zadávali sme rôzne variácie otázok a sledovali kvalitu odpovedí. Sledovali sme či chatbot správne rozumie otázkam a či poskytuje relevantné odpovede, podľa potreby sme doplnili databázu. Ak sme zaznamenali nepresné odpovede snažili sme sa zistiť príčinu a upraviť alebo doplniť databázu a urobiť tréning modelu na novo. Týmto sa nám podarilo postupne zvýšiť presnosť a relevantnosť odpovedí.

2.10.2 Funkčnosť a používateľská skúsenosť

Používateľské rozhranie sme testovali s cieľom overiť jeho intuitívnosť, prehľadnosť či celkový pocit z používania aplikácie. Skúmali sme, či sú všetky prvky správne rozmiestnené, či aplikácia správne reaguje a či sa v aplikáciu nevyskytujú chyby, či nedochádza k nečakaným vypnutiam.

2.11 Využité predmety

Počas tvorby tohto projektu som využil skúsenosti a vedomosti z viacerých predmetov ktoré som počas štyroch rokov štúdia na škole absolvoval. Hlavným predmetom, ktorý som v tomto projekte využil bolo programovanie, ďalšími boli predmety databázové aplikácie a aplikačný softvér. Najmä skúsenosti z programovania mi veľmi pomohli pri tvorbe projektu a čerpaní informácii z internetu o ďalších knižniciach a funkciách.

Záver

Práca na tomto projekte mi poskytla cenné skúsenosti a vedomosti v oblasti spracovania prirodzeného jazyka, tvorby používateľského prostredia a implementácie neurónových sietí. Počas riešenia projektu som sa utvrdil v tom, že ma programovanie baví, a zároveň som si overil, že s pomocou internetu, kníh či video tutoriálov dokážem efektívne zvládnuť nové technológie, moduly a knižnice v programovacom jazyku Python.

S výsledkom projektu som spokojný, keďže som dokázal splniť všetky plánované ciele. Zároveň si uvedomujem, že tento projekt má veľký potenciál na ďalší rozvoj. Možnosti rozšírenia zahŕňajú napríklad implementáciu rozhrania pre mobilné zariadenia, integráciu veľkých jazykových modelov, ako je GPT-4, či optimalizáciu výkonnosti a presnosti modelu.

Verím, že tento projekt ma posunul vpred nielen po technickej, ale aj po analytickej a tvorivej stránke. Skúsenosti, ktoré som získal, určite využijem pri ďalšom štúdiu na vysokej škole, na ktorú sa hlásim, a neskôr aj v profesionálnej praxi. Tento projekt mi ukázal, aké široké možnosti ponúka oblasť umelej inteligencie, a motivoval ma k ďalšiemu vzdelávaniu a zdokonaľovaniu v tejto oblasti.

Bibliografia

VELEBOVÁ, Bc. Michaela Velebová *Chatbot jako nástroj pro rozvoj informační gramotnosti* [Magisterská práce]. Brno: Masarykova Univerzita Filozofická fakulta Katedra informačních studií a knihovnictví [s. n.], 8 s Dostupné z: https://is.muni.cz/th/sr2if/DP_-_projekt.pdf

What Is a Chatbot? Cloud Infrastructure | Oracle Česká republika [online]. Dostupné z: https://www.oracle.com/cz/chatbots/what-is-a-chatbot/

Analyzing Text with the Natural Language Toolkit | Steven Bird, Ewan Klein, and Edward Loper [online]. Dostupné z: https://www.nltk.org/book/

NumPy | Nauč se Python! [online]. Dostupné z: https://naucse.python.cz/lessons/intro/numpy/

Introduction to NLTK | Geeksforgeeks [online]. Dostupné z: www.geeksforgeeks.org/introduction-to-nltk-tokenization-stemming-lemmatization-pos-tagging/

Creating a Basic hardcoded ChatBot using Python -NLTK| Tamoghna Das [online]. Dostupné z: www.tutorialspoint.com/creating-a-basic-hardcoded-chatbot-using-python-nltk

What is Adam Optimizer? | Geeksforgeeks [online]. Dostupné z: www.geeksforgeeks.org/adam-optimizer

Úvod do JSON | JSON Česká republika [online]. Dostupné z: www.json.org/json-cz.html

Bag of words (BoW) model in NLP| Geeksforgeeks [online]. Dostupné z: https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/

FRYDRYCHOVÁ, Bc. Alena Frydrychová. 2020. *Praktické možnosti analýzy textových dat se zaměřením na analýzu sentimentu a vizualizaci* [Magisterská diplomová práce]. Brno: Masarykova Univerzita Filozofická fakulta Katedra informačních studií a knihovnictví [s. n.], 2020. 117 s. Dostupné z: https://is.muni.cz/th/n38t2/diplomova_prace.pdf