# Komunikacija između čovjeka i računala temeljena na tehnikama dubokog učenja

**Benc, Lucija**

**Undergraduate thesis / Završni rad**

**2022**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:235:161431



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

**Lucija Benc**

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

Mentori:

doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Lucija Benc

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru doc. dr. sc Tomislavu Stipančiću na pruženoj pomoći, uloženom vremenu, trudu i povjerenju tijekom izrade ovog rada.

Posebno se zahvaljujem svojoj obitelji na bezuvjetnoj podršci i razumijevanju, te prijateljima koji su mi dosadašnje studentske dane učinili ljepšim i lakšim.

Lucija Benc

SVEUČILIŠTE U ZAGREBU
**FAKULTET STROJARSTVA I BRODOGRADNJE**
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove.
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

| Sveučilište u Zagrebu<br>Fakultet strojarstva i brodogradnje | |
|---|---|
| Datum | Prilog |
| Klasa: **602 – 04 / 22 – 6 / 1** | |
| Ur.broj: **15 - 1703 - 22 -** | |

# ZAVRŠNI ZADATAK

Student: **Lucija Benc** JMBAG: **0035215933**

Naslov rada na hrvatskom jeziku: **Komunikacija između čovjeka i računala temeljena na tehnikama dubokog učenja**

Naslov rada na engleskom jeziku: **Human-computer communication based on deep learning techniques**

Opis zadatka:

Temeljem napretka umjetne inteligencije ljudima je omogućeno razvijati tehnologije koje mogu imitirati ljudski interakciju koristeći tehnike za prepoznavanje govora i teksta.

U radu je potrebno razviti računalni program za razgovor (eng. *chatbot*) koristeći tehnike za obradu prirodnog jezika (eng. NLP - *Natural Language Processing*). Razvoj programske podrške mora uključivati sljedeće korake:

- kreirati dokument koji sadržava predefinirane komunikacijske upite i odgovore za komunikaciju između korisnika i računalnog programa,
- odrediti odgovarajuće programske biblioteke potrebne za izvršavanje zadatka (NLP biblioteke, biblioteke za duboko učenje, biblioteke za rad s podacima),
- razviti programsku podršku koristeći *Python* programsko okruženje,
- definirati odgovarajuće podatke za trening te trenirati model dubokog učenja.

Razvijenu programsku podršku potrebno je eksperimentalno verificirati u laboratorijskim uvjetima te analizirati njenu pouzdanost.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2021.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
**1. rok:** 24. 2. 2022.
**2. rok (izvanredni):** 6. 7. 2022.
**3. rok:** 22. 9. 2022.

Predviđeni datumi obrane:
**1. rok:** 28. 2. – 4. 3. 2022.
**2. rok (izvanredni):** 8. 7. 2022.
**3. rok:** 26. 9. – 30. 9. 2022.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

# SADRŽAJ

POPIS SLIKA ..... II
POPIS OZNAKA ..... III
SAŽETAK IV
SUMMARY ..... V

1. UVOD ..... 1
2. TEORIJSKA OSNOVA RADA ..... 2
   2.1 Neuronske mreže ..... 2
      2.1.1 Model neurona ..... 2
      2.1.2 Model umjetne neuronske mreže ..... 3
      2.1.3 Učenje neuronske mreže ..... 3
   2.2 Strojno učenje ..... 4
      2.2.1 Podjela strojnog učenja ..... 4
   2.3 DUBOKO UČENJE ..... 5
      2.3.1 Usporedba strojnog učenja i dubokog učenja ..... 5
   2.4 Obrada prirodnog jezika ..... 6
3. IZVEDBA ZADATKA ..... 8
   3.1 Chatbot ..... 8
   3.2 Programski jezik Python ..... 8
      3.2.1 NumPy ..... 8
      3.2.2 Keras ..... 9
      3.2.3 TensorFlow ..... 9
      3.2.4 NLTK ..... 9
      3.2.5 WordNetLemmatizer ..... 10
   3.3 Učitavanje biblioteka ..... 10
   3.4 Priprema baze podataka ..... 11
   3.5 Priprema listi za treniranje ..... 13
   3.6 Stvaranje modela neuronske mreže ..... 14
   3.7 Stvaranje Chatbot aplikacije koja koristi model ..... 16
4. ANALIZA PROGRAMSKE PODRŠKE ..... 18
5. ZAKLJUČAK ..... 20

LITERATURA ..... 21
PRILOZI ..... 22

# POPIS SLIKA

Slika 1. Građa umjetnog neurona [2] ..... 2
Slika 2. Model neuronske mreže [2] ..... 3
Slika 3. Razlika strojnog i dubokog učenja [7] ..... 6
Slika 4. Virtualni asistenti [8] ..... 7
Slika 5. Mogućnosti NLTK biblioteke [15] ..... 10
Slika 6. Učitavanje potrebnih biblioteka ..... 10
Slika 7. Oznaka, uzorak i odgovor iz koda ..... 12
Slika 8. Stvaranje klasa u kodu ..... 12
Slika 9. Dokument lista ..... 13
Slika 10. Pretvaranje riječi u numeričke vrijednosti ..... 13
Slika 11. Model neuronske mreže ..... 14
Slika 12. Pretvaranje riječi u numeričke vrijednosti ..... 15
Slika 13. Trenirana neuronska mreža ..... 16
Slika 14. Funkcije algoritma ..... 17
Slika 15. Graf točnosti ..... 18
Slika 16. Graf gubitaka ..... 18
Slika 18. Model odziva ..... 19
Slika 17. Model preciznosti ..... 19

# POPIS OZNAKA

| Oznaka | Jedinica | Opis |
|---|---|---|
| ω | / | težinski faktor |
| reLU | / | ispravljena linearna jedinica (rectified linear unit) |
| x | / | ulazni signal |

# SAŽETAK

Tema ovog rada je razvitak programske podrške za komunikaciju čovjeka i računala. Razvijena programska podrška temelji se na korištenju programskog jezika Python. U programskom jeziku izrađen je model neuronske mreže i povezan s tehnikom obrade prirodnog jezika (engl. NLP – *Natural Language Processing)* pomoću biblioteka. U prvom dijelu rada objašnjena je teorijska osnova za razumijevanje razvoja programa, a u drugom dijelu primjenom strojnog učenja i neuronskih mreža razvijen računalni program za razgovor čovjeka i računala. Na kraju rada prikazana je analiza rada u laboratorijskim uvjetima.

Ključne riječi: umjetna neuronska mreža, NLP, duboko učenje, Python, razgovor računala i čovjeka

# SUMMARY

The topic of this paper is the development of the software for human-computer communication. The developed software is based on using programming language Python. Neural network is connected to Natural Language Processing by using programming language and libraries. The theory behind software development is explained in the first part of this paper and computer programm for human-computer communication based on application of machine learning and neural networks is explained in the second part of this paper. Finally, results analysis is shown in laboratory conditions.

Key words: neural network, NLP, deep learning, Python, human-computer communication

# 1. UVOD

Ubrzan napredak u razvoju umjetne inteligencije omogućuje inovacije u tehnologijama u raznim granama industrije, medicine i ekonomije… Omogućeno nam je razvijati tehnologije koje imitiraju ljudsku interakciju, a sve zahvaljujući razvoju neurologije, odnosno boljem razumijevanju rada ljudskog mozga. Tako su po uzoru na ljudski živčani sustav uređene umjetne neuronske mreže, često korištena metoda umjetne inteligencije.

Jedna od primjena umjetne inteligencije je komunikacija robota i čovjeka u korisničkoj podršci koja će biti detaljnije opisana u ovom radu. Potrebno je prikupiti skup podataka koji se potencijalno javljaju u određenoj interakciji robota i čovjeka. Prije prelaska na trening, potrebno je kreirati vokabular interakcije i kreirati listu klasa, odnosno oznaka skupa podataka. Cilj treninga je da algoritam što točnije i preciznije prepozna upit korisnika iz skupa podataka i ponudi odgovor iz odgovarajuće liste klasa.

# 2. TEORIJSKA OSNOVA RADA

## 2.1 Neuronske mreže

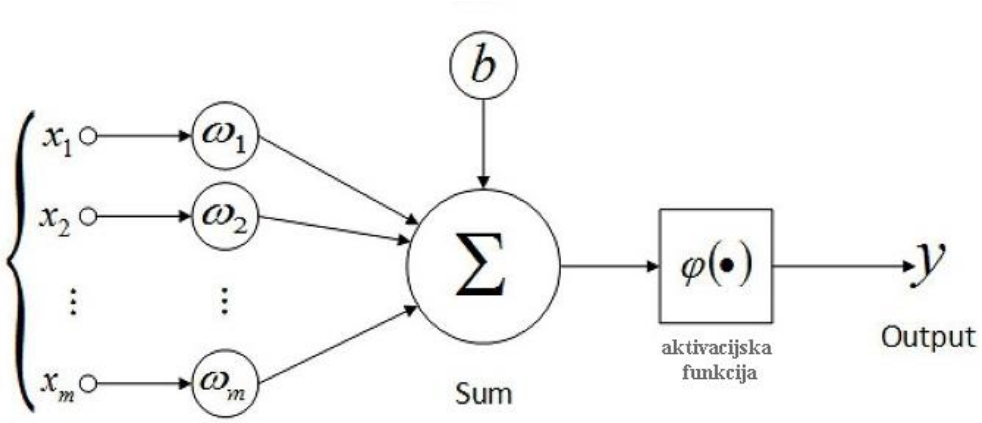
Umjetne neuronske mreže nastoje oponašati mrežu neurona ljudskog mozga kako bi računala imala mogućnost razumijevanja i donošenja odluka poput ljudi. Dizajnirane su programiranjem računala da se ponašaju kao spojene moždane stanice. Naš se mozak ponaša kao paralelni procesor koji ima mogućnost povlačenja informacija iz različitih dijelova memorije, a to je ideja umjetnih neuronskih mreža.

Neke od prednosti umjetnih neuronskih mreža: [1]

- rad s većim brojem varijabli
- mogućnost rada bez obzira na odsutnost dijela podataka
- mogućnost stvaranja odnosa između podataka koji nisu eksplicitno zadani
- prilagodljivost okolini
- dobra procjena nelineranih odnosa uzoraka

### 2.1.1 Model neurona

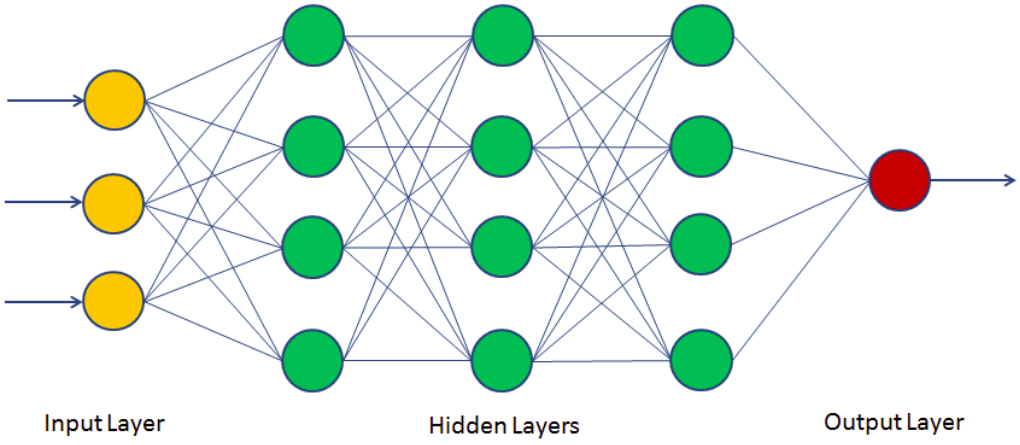
Signal je opisan numeričkom vrijednošću. Na ulazu u neuron signale ($x_1, x_2, x_m$) je potrebno pomnožiti težinskim faktorom ($\omega_1, \omega_2, \omega_m$) koji daje informaciju o jakosti sinapse, spoju dva neurona. Pomnožene signale potrebno je sumirati. Na tu se sumu primjenjuje aktivacijska funkcija koja daje izlazni signal. Prijenos signala prikazan je na slici [Slika 1].




**Slika 1. Građa umjetnog neurona [2]**

### 2.1.2 Model umjetne neuronske mreže

Neuronska mreža je višeslojna mreža koja ne sadrži povratne veze. Model umjetne neuronske mreže prikazan je na slici [Slika 2]. Sastoji se od ulaznog sloja (engl. *input layer*), izlaznog sloja (engl. *output layer*) i jednog ili više skrivenih slojeva (engl. *hidden layer*). Svi neuroni, osim onih u ulaznom sloju, imaju vezu s neuronima prethodnog sloja. Učenje mreže temelji se na promatranju podataka, stvaranju predviđanja i stvaranju poboljšanja na temelju krivih predviđanja. Taj se proces ponavlja više puta, a mreža se poboljšava. [2]




**Slika 2. Model neuronske mreže [2]**

### 2.1.3 Učenje neuronske mreže

Kad govorimo o učenju neuronske mreže, ideja je prilagoditi promjene težinskih faktora kako bi mreža dala što bolje rezultate i dala što manje greške. Težinski faktori ažuriraju se po nekom od pravila učenja. To su perceptron trening (eng. perceptron training), linearno programiranje, delta pravilo i algoritam unazadne propagacije izalzne pogreške (eng. back propagation). Informacija prolazi neuronskom mrežom i generira vrijednost koju uspoređujemo sa stvarnom vrijednošću. Težinski faktori ispravljaju se na temelju razlike stvarne i izračunate vrijednosti težinskog faktora. U čitavom procesu mreža uči predviđati stvarne vrijednosti i samim tim smanjuje razlike u vrijednostima izlaznih veličina. [3]

## 2.2 Strojno učenje

Strojno učenje podrazumijeva primjenu umjetne inteligencije u kojoj strojevi imaju sposobnost učenja i poboljšavanja vlastitih izvedbi bez izričitog programiranja. Učenje algoritma strojnog učenja možemo podijeliti u 4 faze: [4]

- prikupljanje i priprema (organizacija) ulaznih podataka koje algoritam obrađuje, njihova integracija i selekcija
- odabir i trening modela učenja
- interpretacija rezultata, evaluacija (ako zadan kriterij nije zadovoljen, povratak na drugi korak)
- uporaba znanja, stvaranje petlje

### 2.2.1 Podjela strojnog učenja

Strojno učenje možemo podijeliti u tri skupine: nadgledano, nenadgledano i pojačano [5].

U nadgledanom učenju (engl. *supervised learning*) podaci za trening su označeni, a uključuju izlazne i ulazne podatke. Podaci se koriste za predviđanje budućih ishoda i klasifikaciju podataka. Nadgledano učenje uključuje probleme regresije i klasifikacije. Neki od algoritma koje koristi nadgledano učenje su drvo odluke, Bayesov zaključak (engl. *Bayesian learning*) i već spomenute neuronske mreže. Jedan od mnogih primjera korištenja nadgledanog učenja je predviđanja cijena dionica. Na temelju poznatih ulaznih podataka, predviđa se pad ili rast cijena.

Kod nenadgledanog učenja (engl. *unsupervised learning*) koriste se algoritmi za donošenje zaključaka o neoznačenim podacima. Algoritmi koriste tehnike grupiranja (engl. clustering), analizu i povezivanje neoznačenih podataka. Mogućnost pronalaska sličnosti i razlika u podacima čine nadgledano učenje idealnim za primjenu u analizi podataka, primjerice istraživanje tržišta, prepoznavanje anomalija, prepoznavanje fotografija…

Pojačano učenje (engl. *reinforcement learning*) je model strojnog učenja koji se temelji na donošenju odluka. Model podrazumijeva učenje optimalnog ponašanja koje bi postiglo

najveću nagradu. Za dobre odluke postoje pozitivne povratne informacije, a za loše negativne povratne informacije. Za razliku od nadgledanog učenja, ne postoji baza označenih podataka, već su izvor podataka povratne informacije. Na taj način ulazna i izlazna informacija ne ovise jedna o drugoj, već je izlazna informacija posljedica trenutnog ulaza, a sljedeća ulazna informacija ovisi o prethodnoj izlaznoj. Najbolji primjer pojačanog učenja su računalne igrice poput labirinta u kojima je potrebno pronaći najkraći put do izlaza, pritom izbjegavajući prepreke.

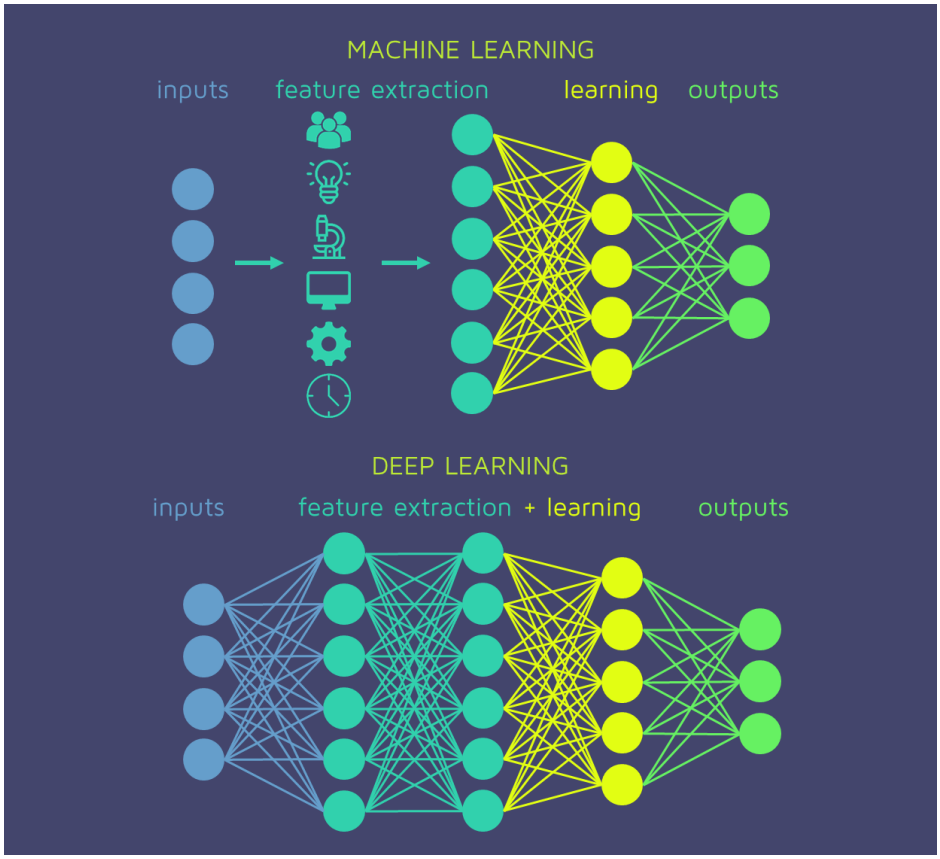
## 2.3 DUBOKO UČENJE

Duboko učenje (engl. *deep learning*) je jedna od podvrsta strojnog učenja koje se bavi algoritmima povezanim s umjetnim neuronskim mrežama. Ova vrsta učenja koristi različite slojeve neurona za analizu podataka. Modeli dubokog učenja najčešće se temelje na različitim modelima neuronskih mreža. To su primjerice duboke neuronske mreže (eng. *deep neural networks*), konvolucijske neuronske mreže (eng. *convolution neural networks*) i duboke mreže vjerovanja (eng. *deep belief network*). Primjena dubokog učenja je raznovrsna. Zahvaljujući razvitku umjetnih neuronskih mreža danas svjedočimo napretku u biologiji, ponajviše genomici, prepoznavanje govora, strojni prijevod, autonomni automobili, stvaranje glazbe [6]…

### 2.3.1 Usporedba strojnog učenja i dubokog učenja

Nakon što smo definirali strojno i duboko učenje, možemo zaključiti da je duboko učenje zapravo napredak strojnog učenja. S obzirom na napredak, razlikuju se u mogućnostima, a glavna prednost dubokog učenja je odsutnost ljudskog faktora. Naime, kad algoritam strojnog učenja da netočan rezultat ili predviđanje, čovjek mora odlučiti i shvatiti kako je do toga došlo, te kako to popraviti. Kod dubokog učenja „ljudski dio posla“ obavljaju neuronske mreže, odnosno one donose odluke. Najbolji primjer za razlikovanje dubokog od strojnog učenja je ručna svjetiljka. U smislu strojnog učenja može biti programirana da se uključi kad čuje zvučni signal, primjerice riječ „mrak“. Kad se radi o dubokom učenju, svjetiljka iz konteksta poput „Svjetlo je slabo“ shvaća da se treba uključiti. [6]

Osnovne razlike prikazne su na slici. [Slika 2]

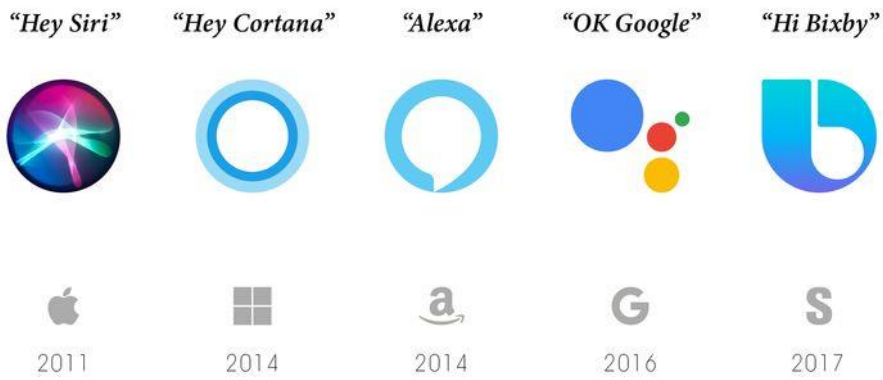



**Slika 3. Razlika strojnog i dubokog učenja [7]**

## 2.4 Obrada prirodnog jezika

Obrada prirodnog jezika (engl. *Natural language processing, NLP*) je grana umjetne inteligencije koja se bavi mogućnostima računala da razumiju tekst i odgovoraju na njega na isti način kao i ljudi. Ova grana kombinira jezične zakonitosti ljudskog govora s modelima strojnog i dubokog učenja. Sve češće se možemo susresti s jezičnim asistentima poput Siri (Apple), Bixby (Samsung), Alexa (Amazon)… Ovi asistenti pružaju sve više mogućnosti poput puštanja glazbe, pronalaska različitih lokacija na karti, daju nam pregled obaveza u kalendaru i mnogo više, ali sve uz jezičnu pravilnu naredbu. Ljudski jezik ima mnoge nepravilnosti koje uvjetuju, odnosno otežavaju, brz razvoj algoritma koji točno određuje namjeru i podatak teksta ili zvuka. To je primjerice korištenje sarkazma i metafora, jezična

raznolikost koja nam omogućuje promjenu struktura rečenica, homofoni i homonimi, naglasci ili primjerice izuzetno brz govor. [8]




**Slika 4. Virtualni asistenti [8]**

# 3. IZVEDBA ZADATKA

## 3.1 Chatbot

Chatbot je naziv za računalni program kojem je cilj simulacija razgovora nalik na interakciju govorom dvoje (ili više) ljudi. Glavni mu je zadatak pomoći korisniku odgovarajući na nasumična pitanja za koja je treniran. Sve se češće primjenjuje u korisničkoj podršci, prodaji, ali može se pohvaliti i implementacijom u medicini gdje mu je cilj na osnovu simptoma, povijesti bolesti i sl. otkriti od čega pacijent boluje. [9]

## 3.2 Programski jezik Python

Python je interpretacijski, objektno orijentirani programski jezik kojim se postiže visoka razina programiranja zahvaljujući velikom broju biblioteka. Ovaj programski jezik može se pohvaliti jednostavnošću, intuitivnošću, širokom primjenom za početnike i napredne i mogućnošću brzog učenja sintaksi. Python je besplatan programski jezik koji nudi module i pakete koji potiču modularnost i ponovnu iskoristivost koda. Svoju popularnost stekao je zbog svestranosti, pa je tako u Pythonu moguće razvijati web stranice, analizirati podatke pomoću strojnog učenja ili automatizirati dijelove procesa. Činjenica je da je Python jedan od najpoznatijih programskih jezika, a to dokazuju brojna poznata poduzeća koja ga koriste. To su: Google, Facebook, Dropbox, Spotify… Također, važno je da se Python može instalirati na različite Unix verzije: Linux, Windows, macOS. [10]
U ovom radu korišten je Python 3.7 jer sadrži biblioteke koje su potrebne za razvoj komunikacije čovjeka i robota, a što je cilj rada. Biblioteke koje nisu implementirane u ovoj verziji Pythona, jednostavno sam instalirala upisujući „pip install [biblioteka]“ u terminal. Korištene biblioteke bit će objašnjene u nastavku.

### 3.2.1 NumPy

NumPy je biblioteka programskog jezika Python koja podržava višedimenzijska polja (engl. *array*) i matrice, te matematičke funkcije visoke razine koje se obavljaju na tim poljima. Ova biblioteka nudi brza rješenja za operacije na poljima, uključujući matematičke i logičke operacije, Fourierove transformacije, operacije osnovne algebre, ali i osnovne operacije

korištene u statistici. NumPy, brz i svestran, podržava objektno orijentirani pristup, te je zbog toga postao jezik višedimenzijske razmjene podataka korištene u Pythonu. [11]

### 3.2.2 Keras

Keras je biblioteka otvorenog koda koja pruža sučelje za umjetne neuronske mreže. Djeluje kao sučelje za TensorFlow biblioteku. Odlikuju se svojim motom koji naglašava važnost ostvarivanja cilja na najkraći način, odnosno smanjenje trajanja puta između ideje i cilja. Ova je biblioteka popularna zbog svoje dostupnosti. Moguće je korištenje u preglednicima mobilnih uređajima, ali i na najvećim grafičkim procesorima. Keras je jednostavan, fleksibilan i moćan u rješavanju problema. Korišten je u projektima NASA-e, YouTube-a, što dokazuje moć ove biblioteke. [12]

### 3.2.3 TensorFlow

TensorFlow je biblioteka otvorenog koda za strojno učenje. Kao ideja je razvijena u Google Brain timu kako bi doprinijela istraživanjima i razvitku. Odlikuje se opsežnim i fleksibilnim alatima i bibliotekama korištenim u strojnom učenju. Pogodna je za kreiranje neuronskih mreža s više slojeva. Najčešće se koristi za klasifikaciju, razumijevanje, percipiranje i predviđanje. TensorFlow se može koristiti u mnogim programskim jezicima, a ta fleksibilnost mu omogućuje razvoj i primjenu u različitim sektorima. [13]

### 3.2.4 NLTK

NLTK (engl. Natural Language Toolkit) je skup biblioteka, programa i izvora obrazovanja za izrađivanje programa obrade prirodnog jezika pomoću programskog jezika Python. Sadrži biblioteke za pretvaranje govora u tekst, prepoznavanje entiteta (imena), antonima (riječi suprotnog značenja) i sinonima (riječi istog značenja), identifikaciju riječi koje se odnose na isto (primjerice „ona“ i „Ana“), analizu emocija, razvijanje rečenica po komponentama, odvajanje riječi, redukcija riječi na originalno značenje... [14]
Neke od mogućnosti NLTK biblioteke prikazane su na slici [Slika 3].


**Slika 5. Mogućnosti NLTK biblioteke [15]**

### 3.2.5 WordNetLemmatizer

Lematizacija je postupak grupiranja izoliranih oblika riječi tako da se svaka analizira zasebno kao posebna riječ. Pojedinačnim riječima se lematizacijom daje kontekst. WordNetLemmatizer je opsežna, besplatna i javno dostupna leksička baza engleskog jezika kojoj je cilj uspostaviti semantičke veze između riječi. Nudi mogućnosti lematizacije i jedna je od najčešće korištenih baza. [16]

## 3.3 Učitavanje biblioteka

Sve navedene biblioteke učitavamo u Python skriptu, što je prikazano na slici [Slika 4.].

```
import random
import json
import pickle
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD
```

**Slika 6. Učitavanje potrebnih biblioteka**

## 3.4 Priprema baze podataka

Prije samog treniranja neuronskih mreža, potrebno je izraditi JSON datoteku namjera (engl. *intents*) koja definira moguće namjere korisnika, odnosno kontekst razgovora robota i čovjeka. Da bismo mogli razviti kontekst, potrebno je kreirati set oznaka (engl. *tags*). Primjerice, ako korisnik želi znati ime sugovornika, kreira se oznaka „ime“, (engl. *name*). U zadatku sam kreirala 8 oznaka koje uključuju opće informacije, ali i informacije koje bi ubuduće mogle poslužiti internetskoj trgovini (engl. web-shop). Oznake su:

- početni pozdrav (engl. *greetings*)
- ime (engl. *name*)
- dob (engl. *age*)
- datum (engl. *date*)
- adresa (engl. *address*)
- radno vrijeme (engl. *working hours*)
- kupnja (engl. *shop*)
- krajnji pozdrav (engl. *goodbye*)

Za svaku od izrađenih oznaka, moramo definirati uzorke (engl. *patterns*). Uzorci opisuju različite načine na koje se korisnik može obratiti robotu. Primjerice, pod oznakom *adresa,* korisnik može zatražiti informacije o lokaciji na više načina – „Koja je tvoja adresa?“ (engl. *What is your address?*), „Gdje živiš?“ (engl. *Where do you live?*), „Gdje mogu pronaći tvoje proizvode?“ (engl. *Where can I find your products?*), „Koja je tvoja lokacija?“ (engl. *What is your location?*). Chatbot uzima uzorke i koristi ih kao podatke za trening kako bi odredio načine ispitivanja korisnika za adresu, tj. lokaciju i kako bi se prilagodio različitim načinima ispitivanja o adresi. Cilj je da korisnici ne moraju koristiti potpuno jednake ulaze koje je chatbot naučio. Primjerice, kad se radi o adresi, pitanje korisnika može glasiti „U kojem se gradu te mogu pronaći?“ (engl. *In what town can I find you?*) i chatbot će znati da korisnik traži podatke o lokaciji, te primjereno tome daje odgovor. U JSON datoteci se uz oznake i uzorke nalaze i statički odgovori kojima chatbot odgovara na postavljene upite. Primjer iz JSON datoteke s oznakom adrese i pripadajućim uzorcima i odgovorima prikazan je na slici [Slika 5.].

```
{
  "tag": "address",
  "patterns": [
    "what is your address?",
    "where do you live?",
    "where can I find your products?",
    "What is your location?"
  ],
  "responses": [
    "I am located in Zagreb, address is available on our webpage",
    "Our address data is available on our webpage",
    "Our products are available in Zagreb, details are on our webpage",
    "We are located in Zagreb, the capital of Croatia"
  ]
},
```

**Slika 7. Oznaka, uzorak i odgovor iz koda**

Kako bi kreirali ostale potrebne podatke za treniranje (engl. *training data*), potrebno je kreirati listu klasa (engl. *list of classes*) koje predstavljaju oznake, listu vokabularu, listu svih uzoraka unutar JSON datoteke i listu svih povezanih oznaka koje dolaze s uzorkom u JSON datoteci. Također, izrađena je lista znakova koje chatbot može ignorirati, odnosno listu znakova koji ne ovise o upitu korisnike i ne utječu na odgovore, a to su interpunkcijski znakovi. Kod prikazuje dio gdje chatbot prolazi namjerama (engl. *intents*) i prepoznaje oznake i uzorke). Kad  chatbot naiđe na oznaku koja još nije u klasi, dodaje je u istu. Postupak je prikazan na slici [Slika 6.].

```
words = []
classes = []
documents = []
ignore_letters = ['?', '!', ',', '.']

for intent in intents['intents']:
    for pattern in intent['patterns']:
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)
        documents.append((word_list, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

**Slika 8. Stvaranje klasa u kodu**

Kao rezultat dobivamo uvid kako izgleda lista, prikazana na slici, Slika [7].

```
print(documents)

[(['Hello'], 'greetings'), (['hey'], 'greetings'), (['greetings'], 'greetings'),
 (['Hi'], 'greetings'), (['good', 'day'], 'greetings'), (['How', 'is', 'it', 'going', '?'], 'greetings'),
 (['What', "'s", 'up', '?'], 'greetings'), (['how', 'old', 'are', 'you', '?'], 'age'),
(['when', 'is', 'your', 'birthday', '?'], 'age'), (['when', 'was', 'you', 'born', '?'], 'age'),
(['what', 'is', 'your', 'age', '?'], 'age'), (['what', 'is', 'your', 'address', '?'], 'address'),
(['where', 'do', 'you', 'live', '?'], 'address'), (['where', 'can', 'I', 'find', 'your', 'products', '?'], 'address'),
 (['What', 'is', 'your', 'location', '?'], 'address'), (['what', 'are', 'you', 'doing', 'this', 'weekend', '?'], 'date'),
 (['do', 'you', 'want', 'to', 'hang', 'out', 'some', 'time', '?'], 'date'),
(['What', 'are', 'your', 'plans', 'for', 'this', 'weekend', '?'], 'date'),
(['I', "'d", 'like', 'to', 'buy', 'something'], 'shop'),
(['What', 'are', 'your', 'products'], 'shop'), (['What', 'do', 'you', 'recommend', '?'], 'shop'),
 (['Can', 'you', 'tell', 'me', 'more', 'about', 'your', 'products', '?'], 'shop'),
 (['When', 'are', 'you', 'guys', 'open', '?'], 'hours'),
 (['What', 'are', 'your', 'hours', '?'], 'hours'), (['When', 'are', 'you', 'working', '?'], 'hours'),
 (['Can', 'you', 'tell', 'me', 'more', 'about', 'your', 'working', 'hours', '?'], 'hours'),
(['what', "'s", 'your', 'name', '?'], 'name'), (['what', 'are', 'you', 'called', '?'], 'name'),
(['who', 'are', 'you', '?'], 'name'), (['bye'], 'goodbye'), (['g2g'], 'goodbye'), (['see', 'ya'], 'goodbye'),
 (['adios'], 'goodbye'), (['cya'], 'goodbye')]
```

**Slika 9. Dokument lista**

## 3.5 Priprema listi za treniranje

Sada kada smo podijelili sve riječi i liste, možemo krenuti na treniranje algoritma. Umjetne neuronske mreže koriste numeričke vrijednosti, što znači da je riječi potrebno pretvoriti u numeričke vrijednosti. Kako bi pretvorili riječi u numeričke vrijednosti koristimo tehniku vreća riječi (eng. *bag of words*). Postavljamo pojedinačne riječi na vrijednost 0 ili 1, ovisno javlja li se riječ u uzorku, odnosno klasi. Taj dio koda prikazan je na slici [Slika 8.].

```
40  training = []
41  output_empty = [0] * len(classes)
42
43  for document in documents:
44      bag = []
45      word_patterns = document[0]
46      word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns]
47      for word in words:
48          bag.append(1) if word in word_patterns else bag.append(0)
49
50      output_row = list(output_empty)
51      output_row[classes.index(document[1])] =1
52      training.append([bag, output_row])
53
54  random.shuffle(training)
55  training = np.array(training)
56
57  train_x = list(training[:,0])
58  train_y = list(training[:,1])
```

**Slika 10. Pretvaranje riječi u numeričke vrijednosti**

## 3.6 Stvaranje modela neuronske mreže

Nakon što su podatci pretvoreni u numeričke vrijednosti, možemo početi graditi model neuronske mreže u kojeg pohranjujemo podatke za trening. Ideja je da model predviđa oznaku povezanu s upitom i izabere prikladan odgovor pod odgovarajućom oznakom. Počinjemo definiranjem prethodno istreniranog modela Sequential i dodajemo dodatne slojeve (engl. layers). Izrađivanjem ostatka modela koristimo aktivacijske funkcije dubokog učenja, softmax i reLU. ReLU, (engl. *rectified linear unit*) je funkcija koja vraća 0 kad primi negativni ulaz, a za svaki pozitivni ulaz vraća isti taj ulaz. Softmax nam omogućuje da predvidimo raspodjelu vjerojatnosti pojedinog rezultata. Slojevi koji su korišteni su Dense i Dropout. Dense predstavlja pravilnu povezanost sloja koji opskrbljuje sve izlaze iz prethodnog sloja svojim neuronima, pri čemu svaki neuron daje jedan izlaz sljedećem sloju. Dropout sloj nasumično postavlja ulazne jedinice na 0, čime onemogućuje pretreniranje (engl. *overfitting data*). Pretreniranje je pogreška koja se javlja kad je funkcija predefinirana limitiranim setom podataka. Kod modela prikazan je na slici [Slika 9].

```
60  model = Sequential()
61  model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
62  model.add(Dropout(0.5))
63  model.add(Dense(64, activation='relu'))
64  model.add(Dropout(0.5))
65  model.add(Dense(len(train_y[0]), activation='softmax'))
66
67  sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
68  model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
69
70  hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
71  model.save('chatbotmodel.h5', hist)
72  print("Done")
```

**Slika 11. Model neuronske mreže**

Dio trenirane neuronske mreže prikazan je na slici [Slika 10.]. Napravljeno je 200 epoha (engl. *epoch*). Epohe definiraju broj ponavljanja kojima algoritam prolazi kroz čitavi set podataka za treniranje.

```
Epoch 2/200
7/7 [==============================] - 0s 3ms/step - loss: 2.0954 - accuracy: 0.1471
Epoch 3/200
7/7 [==============================] - 0s 5ms/step - loss: 2.0664 - accuracy: 0.1471
Epoch 4/200
7/7 [==============================] - 0s 3ms/step - loss: 1.9783 - accuracy: 0.2941
Epoch 5/200
7/7 [==============================] - 0s 3ms/step - loss: 1.9379 - accuracy: 0.1765
Epoch 6/200
7/7 [==============================] - 0s 3ms/step - loss: 1.8897 - accuracy: 0.3529
Epoch 7/200
7/7 [==============================] - 0s 5ms/step - loss: 1.8417 - accuracy: 0.4118
Epoch 8/200
7/7 [==============================] - 0s 5ms/step - loss: 1.7216 - accuracy: 0.5882
Epoch 9/200
7/7 [==============================] - 0s 5ms/step - loss: 1.6721 - accuracy: 0.4412
Epoch 10/200
7/7 [==============================] - 0s 6ms/step - loss: 1.6406 - accuracy: 0.5588
Epoch 11/200
7/7 [==============================] - 0s 5ms/step - loss: 1.5586 - accuracy: 0.5882
Epoch 12/200
7/7 [==============================] - 0s 5ms/step - loss: 1.5337 - accuracy: 0.5294
Epoch 13/200
7/7 [==============================] - 0s 3ms/step - loss: 1.3857 - accuracy: 0.5294
Epoch 14/200
7/7 [==============================] - 0s 4ms/step - loss: 1.2796 - accuracy: 0.6471
Epoch 15/200
7/7 [==============================] - 0s 5ms/step - loss: 1.2623 - accuracy: 0.5000
Epoch 16/200
7/7 [==============================] - 0s 3ms/step - loss: 1.0837 - accuracy: 0.6765
Epoch 17/200
7/7 [==============================] - 0s 3ms/step - loss: 1.1489 - accuracy: 0.6176
Epoch 18/200
7/7 [==============================] - 0s 3ms/step - loss: 1.1042 - accuracy: 0.6471
```

**Slika 12. Pretvaranje riječi u numeričke vrijednosti**

## 3.7 Stvaranje Chatbot aplikacije koja koristi model

Potrebno je učitati model neuronske mreže napravljen u prethodnom koraku i učitati biblioteke potrebne za stvaranje aplikacije. Taj dio koda prikazan je na slici [Slika 11].

```
import random
import json
import pickle
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import load_model

lemmatizer = WordNetLemmatizer()
intents = json.loads(open('intents.json').read())

words = pickle.load(open('words.pkl', 'rb'))
classes = pickle.load(open('classes.pkl', 'rb'))
model = load_model('chatbotmodel.h5')
```

**Slika 13. Trenirana neuronska mreža**

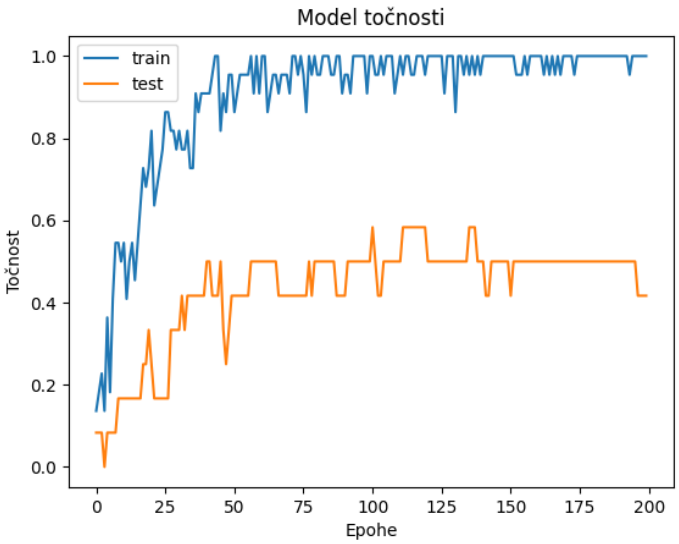
S obzirom da smo na početku sve tekstne podatke pretvorili u numeričke vrijednosti, sada je potrebno numeričke vrijednosti vratiti u tekst. Za to ćemo koristiti funkcije: funkciju koja analizira rečenice (engl. *clean up sentence*) i vraća pojedinačne riječi, funkciju koja otvara vreću riječi (engl. *bag of words*), funkciju koja predviđa klasu na temelju rečenice (engl. *predict class*) i funkciju koja vraća odgovore (engl. *get response*), odnosno odgovara na pitanja. Dio koda u kojem su definirane spomenute funkcije prikazan je na slici [Slika 12].

```
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word) for word in sentence_words]
    return sentence_words

def bag_of_words(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for w in sentence_words:
        for i, word in enumerate(words):
            if word ==w:
                bag[i] = 1
    return np.array(bag)

def predict_class(sentence):
    bow = bag_of_words(sentence)
    res = model.predict(np.array([bow]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]

    results.sort(key= lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({'intent': classes[r[0]], 'probability': str(r[1])})
    return return_list

def get_response(intents_list, intents_json):
    tag = intents_list[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if i['tag'] == tag:
            result = random.choice(i['responses'])
            break
    return result
```

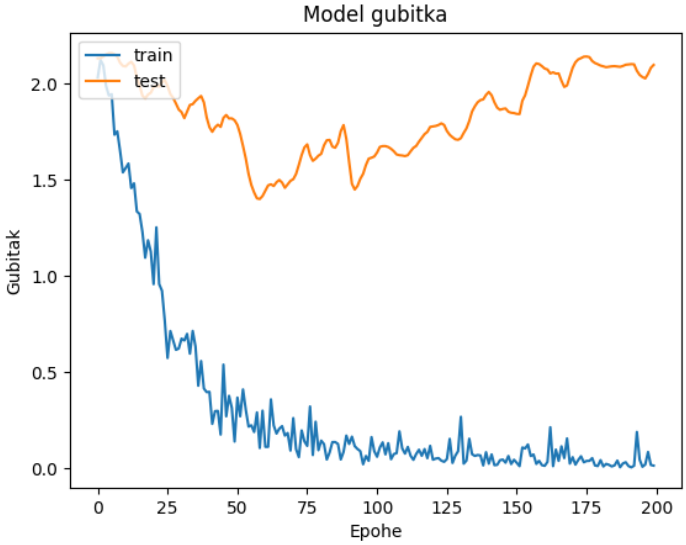
**Slika 14. Funkcije algoritma**

# 4. ANALIZA PROGRAMSKE PODRŠKE

Slike [Slika 15., Slika 16] prikazuju grafove točnosti i gubitka modela. Graf točnosti [Slika 15.] prikazuje točnost predviđanja klasa kroz epohe. Plava linija prikazuje točnost modela za podatke koji čine trening, a narančasta točnost modela mreže za testne riječi. Uočljiv je rast kroz epohe, a izraženiju točnost postiže skup podataka za trening. U zadnjoj epohi dobivaju se rezultati točnosti koji se mogu isčitati na osi „Točnost“. Pogledom na graf modela gubitka mreže [Slika 16.], uočljivo je da se greška gubitka kod testnog skupa podataka čak i povećava, ukazuje na činjenicu da je mreža pretrenirana, odnosno potrebno joj je poboljšanje.

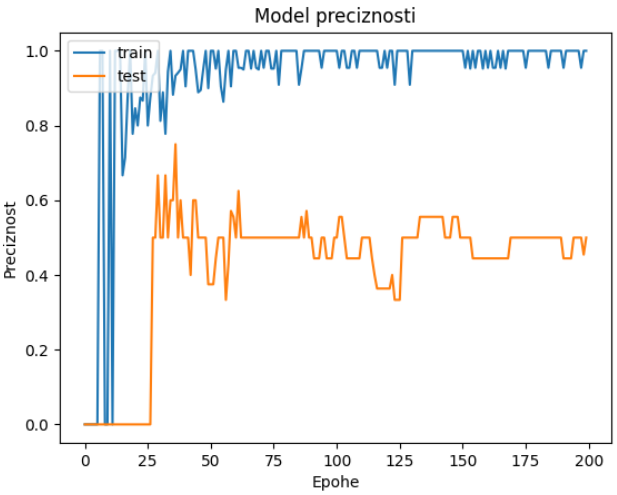



**Slika 15. Graf točnosti**

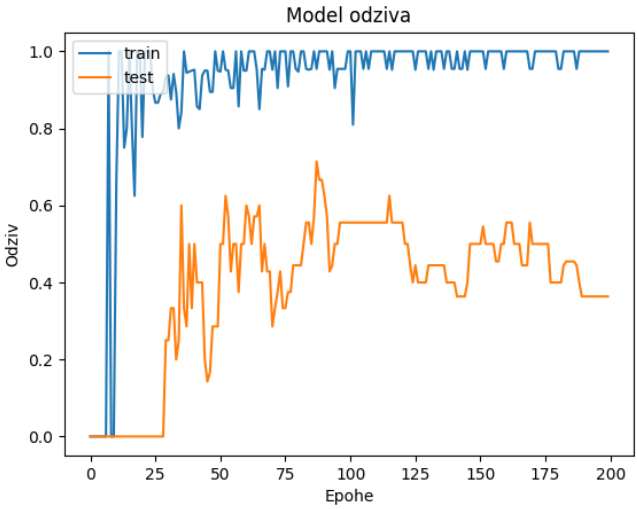



**Slika 16. Graf gubitaka**

Slike [Slika 17., Slika 18.] prikazuju grafove preciznosti i odziva. Preciznost (engl. precision) izražava omjer broja točno klasificiranih uzoraka kao pozitivnih i ukupnog broj klasificiranih uzorka. Odziv se računa kao omjer broja točno klasificiranih uzoraka kao pozitivnih i ukupnog broja pozitivnih.




**Slika 18. Model preciznosti**




**Slika 17. Model odziva**

# 5. ZAKLJUČAK

U radu je prikazan razvoj programske podrške za komunikaciju čovjeka i računala. Teorijska osnova rada objašnjena je u početku, te je nužna za razumijevanje razvijenog programa. Samo testiranje modela izvedeno je u programskom jeziku Python. Testiranje je pokazalo da su za opću upotrebu ove programske podrške potrebna poboljšanja, ali je dobra osnova za razvoj „chatbot“ aplikacije. Razvojem biblioteka i tehnologije koja će prevladati poteškoće koje se javljaju u razumijevanju ljudskog govora poput sarkazma, ironije i sl., bit će moguća šira upotreba ovakve i slične programske podrške, a koja nam uvelike može pomoći u savladavanju zadataka.

# LITERATURA

[1] https://www.researchgate.net/post/What-are-the-advantages-of-using-Artificial-Neural-Network-compared-to-other-approaches dostupno dana 03.02.2022.

[2] https://www.mdpi.com/2079-9292/9/5/731/htm dostupno dana 03.02.2022.

[3] Ujević Andrijić, Ž.: OSVJEŽIMO ZNANJE: Umjetne neuronske mreže, Zagreb, 2003.

[4] https://towardsdatascience.com/machine-learning/home dostupno dana 04.02.2022.

[5] https://towardsdatascience.com/what-is-machine-learning-a-short-note-on-supervised-unsupervised-semi-supervised-and-aed1573ae9bb dostupno dana 05.02.2022.

[6] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5938574/ dostupno dana 06.02.2022.

[7] https://quantdare.com/what-is-the-difference-between-deep-learning-and-machine-learning/deep_learning/ dostupno dana 06.02.2022.

[8] https://www.ibm.com/cloud/learn/natural-language-processing dostupno dana 06.02.2022.

[9] https://www.chatbot.com/ dostupno dana 11.02.2022.

[10] https://www.python.org/ dostupno dana 11.02.2022.

[11] https://numpy.org/ dostupno dana 11.02.2022.

[12] https://keras.io/about/ dostupno dana 11.02.2022.

[13] https://www.tensorflow.org/ dostupno dana 12.02.2022.

[14] https://www.nltk.org/ dostupno dana 12.02.2022.

[15] https://data-flair.training/blogs/nltk-python-tutorial/ dostupno dana 12.02.2022.

[16] https://www.kite.com/python/docs/nltk.WordNetLemmatizer dostupno dana 13.02.2022.

# PRILOZI

I. JSON datoteka

II. Python kod

# I. JSON DATOTEKA

```
{
  "intents": [
    {
      "tag": "greetings",
      "patterns": [
        "Hello",
        "hey",
        "greetings",
        "Hi",
        "good day",
        "How is it going?",
        "What's up?"
      ],
      "responses": [
        "Hey",
        "Hello",
        "How are you doing?",
        "Greetings!",
        "What can I do for you?"
      ]
    },
    {
      "tag": "age",
      "patterns": [
        "how old are you?",
        "when is your birthday?",
        "when was you born?",
        "what is your age?"
      ],
      "responses": [
        "I am 24 years old",
        "I was born in 1996",
        "My birthday is July 3rd and I was born in 1996",
        "03/07/1996"
      ]
    },
    {
      "tag": "address",
      "patterns": [
        "what is your address?",
        "where do you live?",
        "where can I find your products?",
        "What is your location?"
      ],
      "responses": [
        "I am located in Zagreb, address is available on our webpage",
        "Our address data is available on our webpage",
        "Our products are available in Zagreb, details are on our webpage",
        "We are located in Zagreb, the capital of Croatia"
      ]
    },
    {
```

```
    "tag": "date",
    "patterns": [
      "what are you doing this weekend?",
      "do you want to hang out some time?",
      "What are your plans for this weekend?"
    ],
    "responses": [
      "I am available all week",
      "I don't have any plans",
      "I am not busy"
    ]
  },
  {
    "tag": "shop",
    "patterns": [
      "I'd like to buy something",
      "What are your products",
      "What do you recommend?",
      "Can you tell me more about your products?"
    ],
    "responses": [
      "You can check our offer online and in stores"
    ]
  },
  {
    "tag": "hours",
    "patterns": [
      "When are you guys open?",
      "What are your hours?",
      "When are you working?",
      "Can you tell me more about your working hours?"
    ],
    "responses": [
      "24/7"
    ]
  },
  {
    "tag": "name",
    "patterns": [
      "what's your name?",
      "what are you called?",
      "who are you?"
    ],
    "responses": [
      "My name is Lucija",
      "I'm Lucija",
      "Lucija"
    ]
  },
  {
    "tag": "goodbye",
    "patterns": [
      "bye",
      "g2g",
      "see ya",
      "adios",
      "cya"
    ],
    "responses": [
      "It was nice speaking to you",
      "See you later",
      "Speak soon!"
    ]
  }
]
}
```

II. Python kod

- kod za manipuliranje podatcima

```
import random
import json
import pickle
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD

lemmatizer = WordNetLemmatizer()

intents = json.loads(open('intents.json').read())

words = []
classes = []
documents = []
ignore_letters = ['?','!',',','.']

for intent in intents['intents']:
    for pattern in intent['patterns']:
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)
        documents.append((word_list, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

print(documents)

words = [lemmatizer.lemmatize(word) for word in words if word not in ignore_letters]
words = sorted(set(words))

classes = sorted(set(classes))

pickle.dump(words, open('words.pkl','wb'))
pickle.dump(classes, open('classes.pkl','wb'))

training = []
output_empty = [0] * len(classes)

```

```
for document in documents:
    bag = []
    word_patterns = document[0]
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns]
    for word in words:
        bag.append(1) if word in word_patterns else bag.append(0)

    output_row = list(output_empty)
    output_row[classes.index(document[1])] =1
    training.append([bag, output_row])

random.shuffle(training)
training = np.array(training)

train_x = list(training[:,0])
train_y = list(training[:,1])

model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('chatbotmodel.h5', hist)
print("Done")
```

- kod za Chatbot aplikaciju

```
import random
import json
import pickle
import numpy as np

import nltk
from nltk.stem import WordNetLemmatizer

from tensorflow.keras.models import load_model

lemmatizer = WordNetLemmatizer()
intents = json.loads(open('intents.json').read())

words = pickle.load(open('words.pkl', 'rb'))
classes = pickle.load(open('classes.pkl', 'rb'))
model = load_model('chatbotmodel.h5')


def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word) for word in sentence_words]
    return sentence_words

def bag_of_words(sentence):
    sentence_words = clean_up_sentence(sentence)
    bag = [0] * len(words)
    for w in sentence_words:
        for i, word in enumerate(words):
            if word ==w:
                bag[i] = 1
    return  np.array(bag)

def predict_class(sentence):
    bow = bag_of_words(sentence)
    res = model.predict(np.array([bow]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]

```

```
39      results.sort(key= lambda x: x[1], reverse=True)
40      return_list = []
41      for r in results:
42          return_list.append({'intent': classes[r[0]], 'probability': str(r[1])})
43      return return_list
44
45  def get_response(intents_list, intents_json):
46      tag = intents_list[0]['intent']
47      list_of_intents = intents_json['intents']
48      for i in list_of_intents:
49          if i['tag'] == tag:
50              result = random.choice(i['responses'])
51              break
52      return result
53
54  print("RUNNING")
55
56  while True:
57      message = input("")
58      ints = predict_class(message)
59      res = get_response(ints, intents)
60      print(res)
61
```

```
39      results.sort(key= lambda x: x[1], reverse=True)
40      return_list = []
41      for r in results:
42          return_list.append({'intent': classes[r[0]], 'probability': str(r[1])})
43      return return_list
44
45  def get_response(intents_list, intents_json):
46      tag = intents_list[0]['intent']
47      list_of_intents = intents_json['intents']
48      for i in list_of_intents:
49          if i['tag'] == tag:
50              result = random.choice(i['responses'])
51              break
52      return result
53
54  print("RUNNING")
55
56  while True:
57      message = input("")
58      ints = predict_class(message)
59      res = get_response(ints, intents)
60      print(res)
61
```