# Primjena NLP-a u odabiru preferiranog lijeka

**Osmakčić, Dorijan**





**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni diplomski studij Računarstvo**

# PRIMJENA NLP-A U ODABIRU PREFERIRANOG LIJEKA

**Diplomski rad**

**Dorijan Osmakčić**

**Osijek, 2024.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

**Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju**

## Ocjena diplomskog rada na sveučilišnom diplomskom studiju

| | |
|---|---|
| **Ime i prezime pristupnika:** | Dorijan Osmakčić |
| **Studij, smjer:** | Sveučilišni diplomski studij Računarstvo |
| **Mat. br. pristupnika, god.** | D1313R, 07.10.2022. |
| **JMBAG:** | 0165083341 |
| **Mentor:** | izv. prof. dr. sc. Časlav Livada |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | izv. prof. dr. sc. Alfonzo Baumgartner |
| **Član Povjerenstva 1:** | izv. prof. dr. sc. Časlav Livada |
| **Član Povjerenstva 2:** | doc. dr. sc. Tomislav Galba |
| **Naslov diplomskog rada:** | Primjena NLP-a u odabiru preferiranog lijeka |
| **Znanstvena grana diplomskog rada:** | **Umjetna inteligencija (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | U radu je potrebno napraviti chat-bot aplikaciju koja će nakon kratkog "razgovora" s korisnikom na temelju njegovih zdravstvenih problema primjenom NLP-a generirati savjete o preferiranom lijeku. Tema rezervirana za: Dorijan Osmakčić |

| | |
|---|---|
| **Datum ocjene pismenog dijela diplomskog rada od strane mentora:** | 10.09.2024. |
| **Ocjena pismenog dijela diplomskog rada od strane mentora:** | Izvrstan (5) |
| **Datum obrane diplomskog rada:** | 26.09.2024. |
| **Ocjena usmenog dijela diplomskog rada (obrane):** | Izvrstan (5) |
| **Ukupna ocjena diplomskog rada:** | Izvrstan (5) |
| **Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:** | 28.09.2024. |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

## IZJAVA O IZVORNOSTI RADA

**Osijek,** 28.09.2024.

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Dorijan Osmakčić |
| **Studij**: | Sveučilišni diplomski studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | D1313R, 07.10.2022. |
| **Turnitin podudaranje [%]:** | 5 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Primjena NLP-a u odabiru preferiranog lijeka**

izrađen pod vodstvom mentora izv. prof. dr. sc. Časlav Livada

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

# SADRŽAJ

# 1. UVOD

U suvremenom društvu, dostupnost informacija o lijekovima je ključna za samopomoć i upravljanje zdravljem. Ipak, pretraga i odabir odgovarajućeg lijeka može biti izazovan proces, pogotovo za osobe koje nemaju medicinsko obrazovanje. Osim stručnosti, odluka o uzimanju bezreceptnog lijeka može znatno ubrzati proces trgovanja u ljekarni. Iako su OTC odnosno bezreceptni lijekovi lako dostupni, u samoliječenju je potreban oprez. Moguće opasnosti su, na primjer, postavljanje pogrešne dijagnoze ili neželjene interakcije s drugim lijekovima koji se uzimaju [1]. Danas postoje brojna rješenja gdje korisnik može „razgovarati“ s programom koji razumije ljudski govor i koji daje odgovore. Takva programska rješenja koriste obradu prirodnog jezika (engl. *NLP – Natural Language Processing*) kako bi razumjeli, analizirali i odgovarali na ljudski govor i pisanje [2]. Uporabom takvog alata riješila bi se dva prethodno opisana problema stručnosti korisnika i potrošenog vremena.

Problem koji se u ovom diplomskom radu rješava odnosi se na razvoj sustava koji koristi tehnologiju prirodnog jezika za pružanje točnih i personaliziranih preporuka lijekova putem chatbot aplikacije. Ova aplikacija analizira korisnikove odgovore i na temelju prepoznatih simptoma, dobi, trudnoće i alergija, predlaže najprikladniji bezreceptni lijek.

U glavnom dijelu rada opisana su četiri poglavlja. U prvom se opisuje primjena NLP-a u medicini, u drugom su opisani NLP i chatbot tehnologije, u trećem poglavlju analizira se programsko rješenje aplikacije te u zadnjem poglavlju glavnog djela prikazuju se rezultati aplikacije kroz React alat.

## 1.1. Zadatak diplomskog rada

Zadatak diplomskog rada obuhvaća nekoliko ključnih koraka. Prvi korak uključuje prikupljanje i analizu podataka o bezreceptnim lijekovima. Iz tih podataka potrebno je izdvojiti bitne informacije kao što su problem za koji je lijek namijenjen, dobna granica, kontraindikacije vezane uz alergije i sigurnost upotrebe za trudnice. Zatim je potrebno izraditi bazu podataka koja služi kao osnova za modeliranje sustava koji će prepoznati korisnikove odgovore poput problema, dobi i identifikacije trudnoće. Za prepoznavanje alergija iz korisnikovog unosa potrebno je iskoristiti Levenshteinovu udaljenost za sličnost riječi. Nakon prikupljenih odgovora i dobivenih izlaza, potrebno je razviti algoritam pretrage prikladnih lijekova te ih ispisati korisniku u sučelju aplikacije.

# 2. POSTOJEĆA RJEŠENJA

Primjena NLP-a prisutna je u raznim područjima i ima širok spektar primjena. NLP se može koristiti za chatbotove, digitalne asistente, analizu sentimenta, organizaciju dokumenata, zapošljavanje i zdravstvenu skrb.

Chatbotovi i digitalni asistenti poput Amazonove Alexe i Google Assistanta koriste NLP za prepoznavanje i sintezu glasa, omogućujući im interpretaciju i odgovaranje na glasovne naredbe. Ovi asistenti pomažu ljudima u različitim zadacima, oslobađajući ih dijela kognitivnih opterećenja i omogućujući fokus na važnije aktivnosti [3].

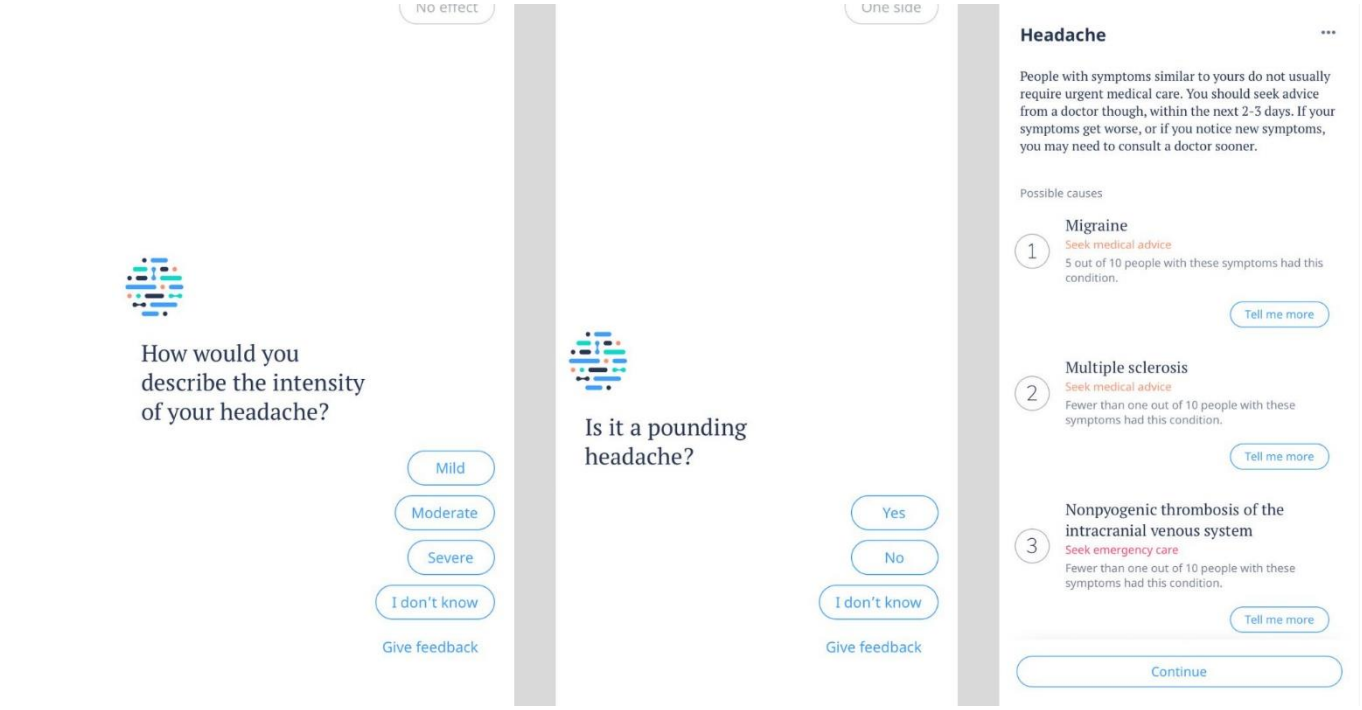
U nastavku su prikazane i opisane neke postojeće aplikacije koje koriste sličnu primjenu kao i aplikacija opisana u ovom radu.

## 2.1. Ada Health mobilna aplikacija

Kao prvi primjer primjene obrade jezika u medicini navodi se aplikacija Ada. To je mobilna aplikacija koja pomaže ljudima da kroz opis simptoma dobiju rezultate mogućih bolesti i savjet koristeći strojno učenje i obradu prirodnog jezika [4]. Jedna od glavnih uloga NLP-a u ovoj aplikaciji je pretvaranje audio i video unosa korisnika u tekst za daljnju analizu [5].

Aplikacija radi na hibridni način. Za razliku od postojećih AI chatbotova, poput Chat GPT-a koji se bazira na razgovoru pitanje-odgovor, u ovoj aplikaciji su predefinirana pitanja i ponuđene mogućnosti odabira zbog mogućnosti sužavanja problema.

Na slici 2.1. prikazan je primjer toka razgovora sa botom, gdje se u konačnici izbacuju rezultati mogućih uzroka i generirani savjet.

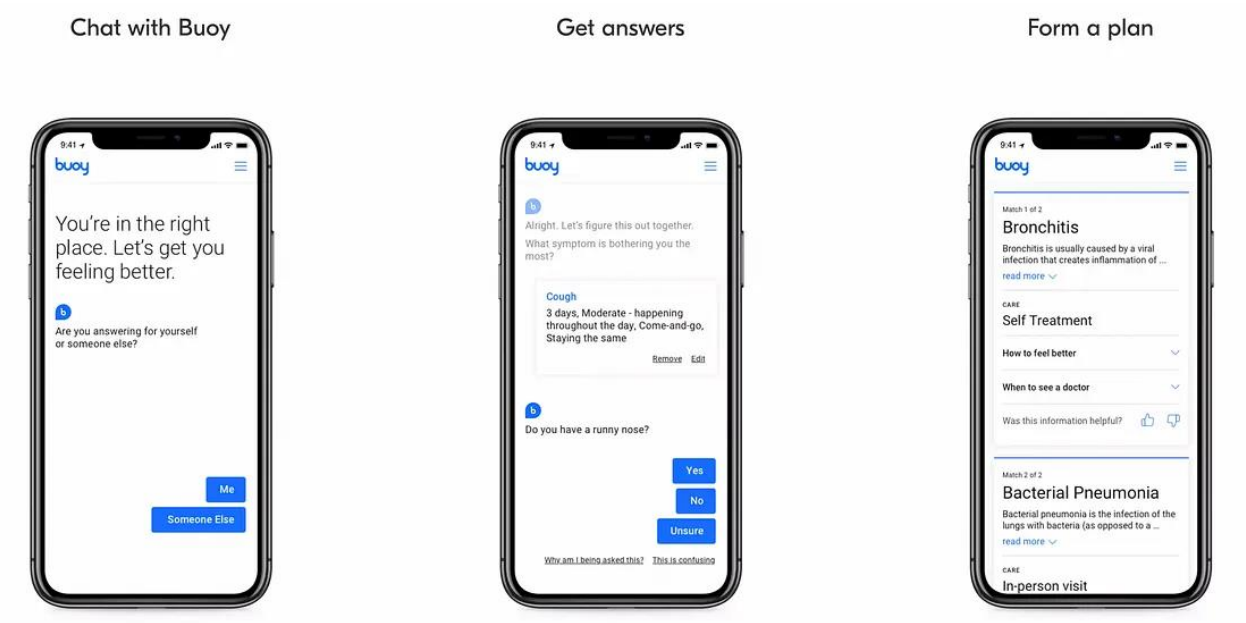



Slika 2.1. Ada Health aplikacija [6]

## 2.2. Buoy Health mobilna aplikacija

Buoy Health je mobilna aplikacija koja koristi umjetnu inteligenciju i obradu prirodnog jezika za pomoć korisnicima u razumijevanju njihovih simptoma. Aplikacija postavlja korisnicima niz pitanja o simptomima, a na temelju njihovih odgovora pruža popis mogućih dijagnoza. Također usmjerava korisnike prema odgovarajućim zdravstvenim ustanovama i nudi opcije liječenja.

Buoy Health je korisno sredstvo za pacijente koji žele razumjeti svoje simptome bez potrebe za posjetom liječniku. NLP sposobnosti aplikacije osiguravaju točnost informacija, omogućujući pacijentima donošenje informiranih odluka o svom zdravlju. Na slici 2.2. je prikazan primjer razgovora kroz aplikaciju.




Slika 2.2. Buoy Health mobilna aplikacija [7]

# 3. OPIS KORIŠTENIH ALATA I TEHNOLOGIJA

Uz već u uvodu ukratko objašnjenu ulogu NLP-a odnosno obrade prirodnog jezika, u ovom dijelu opisuje se detaljnije njegova svrha i određeni alati, navode se i opisuju neki chatbot pristupi uz ulogu i vrste chatbot-ova koji se koriste u ovom radu te se opisuje React biblioteka kroz koju je napravljeno sučelje.

## 3.1. Opis i metode obrade prirodnog jezika

Obrada prirodnog jezika predstavlja jedan od najsloženijih zadataka u računalnim znanostima. Različiti jezici donose širok spektar izazova koji se razlikuju od jednog jezika do drugog. Strukturiranje ili izvlačenje smislenih informacija iz slobodnog teksta može biti izuzetno korisno ako se provede na odgovarajući način. U prošlosti su računalni znanstvenici koristili složene algoritme za analizu jezika, razbijajući ga na njegove gramatičke komponente, poput vrsta riječi, fraza i slično. Cilj je bio razumjeti jezik na dubljoj razini kako bi se iz teksta moglo izvući što više informacija [8].

Postoje brojni procesi koji se koriste u NLP-u, poput klasifikacije teksta, analize sentimenta, sažimanja teksta, prevođenje jezika i razne drugi procesi kojim se dobivaju korisne informacije [9].

U nastavku je opisan proces klasifikacije teksta, CountVectorizer alat i Levenshteinova metoda. Navedeni alati i metode ključni su dio za izvlačenje bitnih informacija koje su korisne za daljnju pretragu lijeka u ovom radu.

### 3.1.1. Klasifikacija teksta primjenom logističke regresije

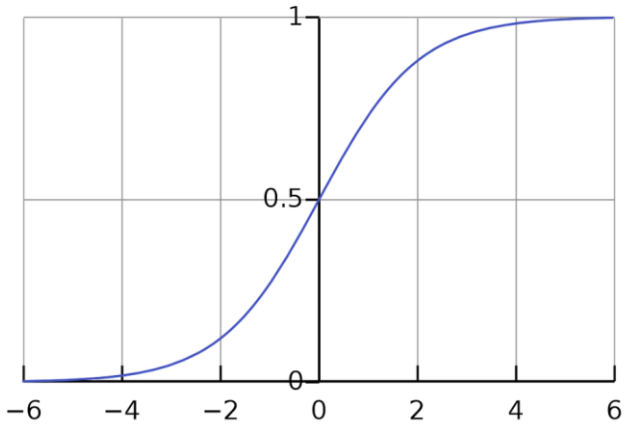
Klasifikacija teksta je jedan od procesa obrade prirodnog jezika, dok je logistička regresija samo jedna od alata odnosno algoritama koji se koriste za klasifikaciju teksta.

Klasifikacija teksta je proces automatskog dodjeljivanja oznaka ili kategorija dijelovima teksta. Ona ima brojne primjene, kao što je sortiranje e-mailova na neželjenu poštu ili redovnu poštu, određivanje je li recenzija proizvoda pozitivna ili negativna, ili čak prepoznavanje teme novinskog članka.

Logistička regresija je statistička metoda koja se koristi za binarne klasifikacijske probleme, a može se proširiti i na klasifikaciju s više klasa. Kada se primjenjuje na klasifikaciju teksta, cilj je predvidjeti kategoriju ili klasu zadanog tekstualnog dokumenta na temelju njegovih značajki.

Za predviđanje kategorije, logistička regresija koristi sigmoid funkciju koja mapira bilo koji realan broj u raspon od 0 do 1. Vrijednost funkcije može se predstaviti kao vjerojatnost, što je u ovom slučaju vjerojatnost pripadanja određenoj klasi [10].

Na slici 3.1. prikazana je sigmoid funkcija koja je prethodno opisana.

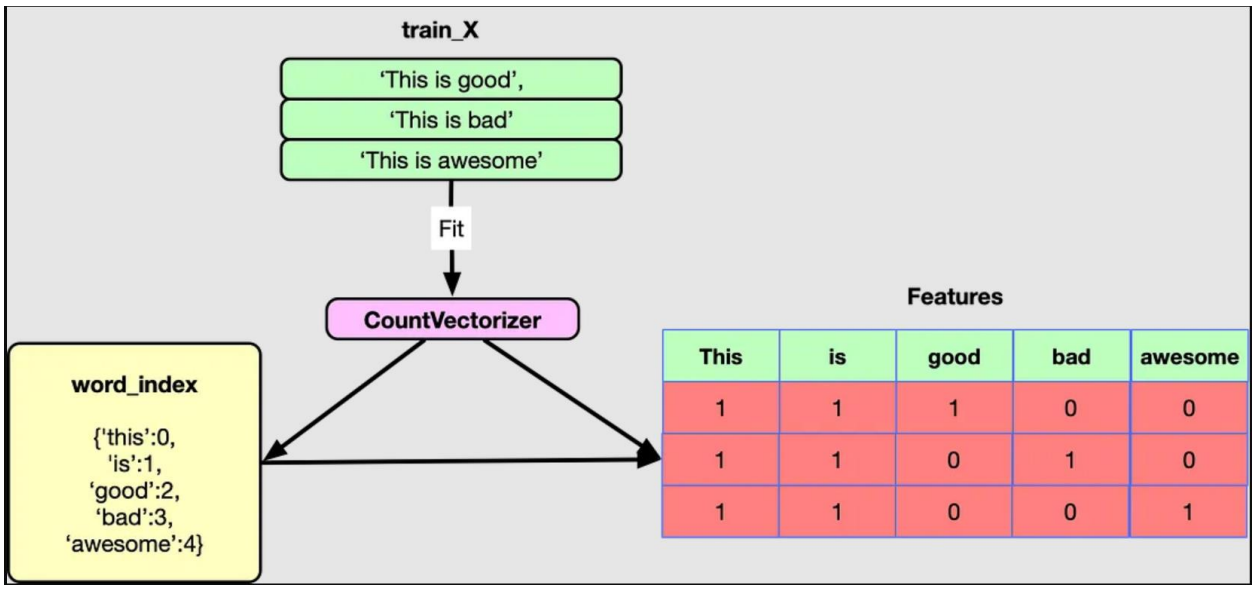



Slika 3.1. Sigmoid funkcija [11]

### 3.1.2. CountVectorizer

Kako bi se pravilno iskoristila logistička regresija, prvo je potrebno pretvoriti podatke u numerički oblik koji služi modelu logističke regresije za daljnju obradu. Alat koji se koristi u ovom radu za tu ulogu je CountVectorizer koji nudi biblioteka scikit-learn unutar programskog jezika Python. Koristi se za transformaciju zadanog teksta u vektor na temelju učestalosti (broja) svake riječi koja se pojavljuje u cijelom tekstu. CountVectorizer stvara matricu u kojoj je svaka jedinstvena riječ predstavljena stupcem matrice, a svaki uzorak teksta iz dokumenta je redak u matrici. Vrijednost svake ćelije nije ništa drugo nego broj riječi u tom određenom uzorku teksta [12].

Na slici 3.2. prikazan je način rada CountVectorizer-a gdje se svakoj riječi iz liste rečenica dodjeli indeks i za svaku riječ se broji njeno pojavljivanje za svaku rečenicu koje se upisuje unutar matrice. Oblikovana riječ se čita po stupcu.

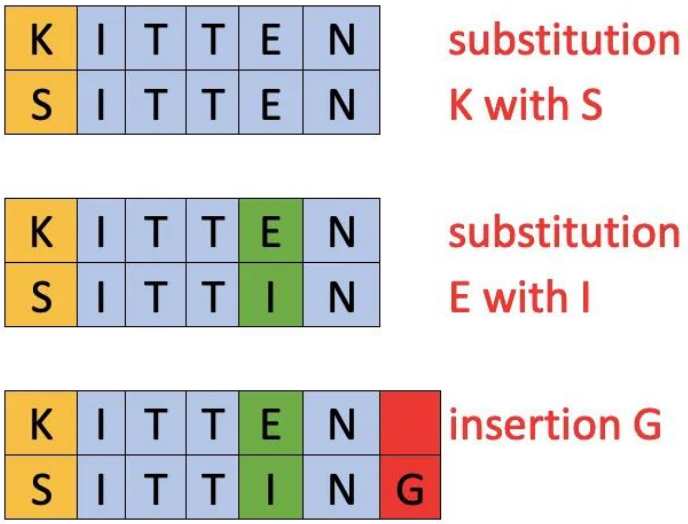



Slika 3.2. Način rada CountVectorizera [13]

### 3.1.3. Levenshteinova udaljenost

U obradi prirodnog jezika postoje i metode koje ne uključuju strojno učenje, jedna od takvih je Levenshteinova udaljenost. Levenshteinova udaljenost može se i opisati kao mjera sličnosti dva stringa ili dvije riječi. Operacije koje se koriste u metodi su ubacivanje, brisanje i zamjena. U konačnici je broj potrebnih iteracija da se jedna riječ pretvori u drugu preko tih operacija rezultat Levenshteinove udaljenosti [14].

Na slici 3.3. prikazan je način rada metode, gdje se zamjenom, brisanjem ili umetanjem broji koliko je puta jedna od tih operacija iskorištena da bi se jedna riječ pretvorila u drugu. Rezultat Levenshteinove udaljenosti sa slike je 3, jer su iskorištene operacije na 3 mjesta.




Slika 3.3. Pretvaranje jedne riječi u drugu [15]

## 3.2. Vrste i uloga chatbot tehnologija

Postoje dvije glavne vrste chatbotova. Jednostavni chatbotovi izvršavaju zadatke prema unaprijed definiranim uputama (engl. *rule-based chatbot*). S druge strane, inteligentni ili napredni chatbotovi koriste umjetnu inteligenciju. Njihova uloga uključuje prikupljanje podataka o korisnicima, poboljšanje korisničke usluge, smanjenje troškova i automatizaciju repetitivnih zadataka [16]. U ovom radu koristi se hibridni chatbot koji primjenjuje NLP za obradu korisnikovih odgovora, a definirane upute za pretragu prikladnog lijeka.

### 3.2.1. Chatbot temeljen na pravilima (engl. *rule -based chatbot*)

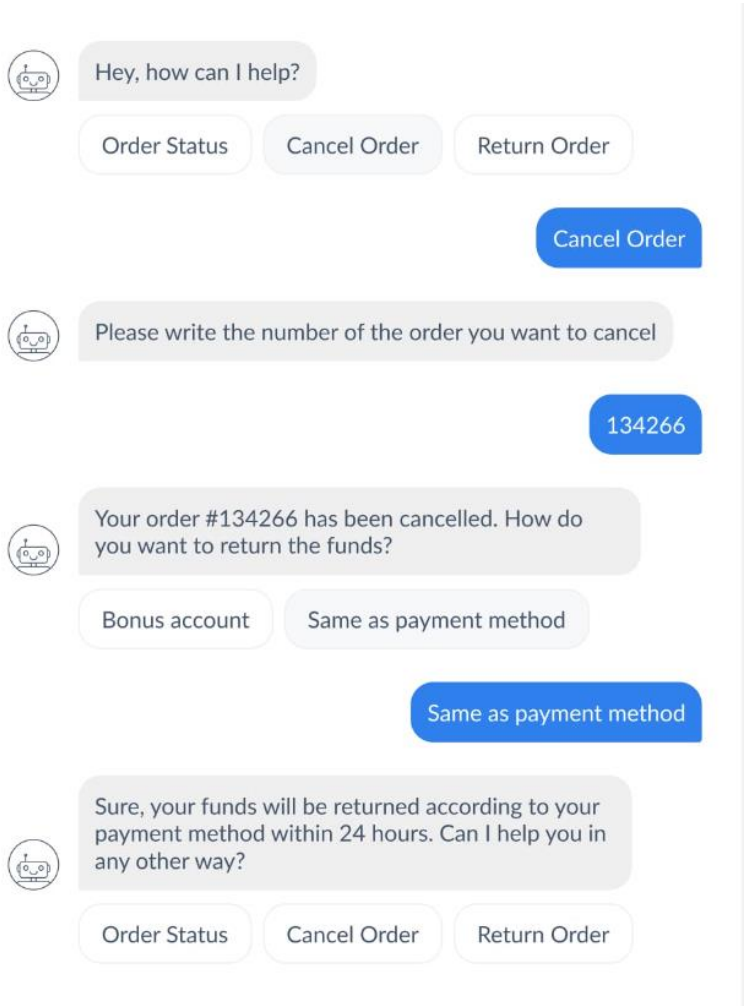
Chatbotovi temeljeni na pravilima rade na temelju skupa unaprijed definiranih pravila i obrazaca. Koriste strogo definirane ako-onda (engl. *if-then*) izjave ili stabla odluka kako bi odredili odgovarajući odgovor na korisnikove upite.

Jedna od glavnih prednosti chatbotova temeljenih na pravilima je njihova jednostavnost i precizna kontrola nad razgovorom. Programeri imaju potpunu kontrolu nad pravilima, što omogućuje da chatbot daje odgovarajuće odgovore na specifične unose. Ipak, njihova sposobnost rukovanja

složenim ili neočekivanim upitima korisnika može biti ograničena, jer su odgovori temeljeni na unaprijed postavljenim obrascima.

Chatbotovi temeljeni na pravilima su idealni za scenarije gdje je tijek razgovora jasno definiran i postoji ograničen broj mogućih korisničkih namjera.

Na slici 1.1 prikazuje se primjer chatbota temeljenog na pravilima gdje korisnik ima ponuđene mogućnosti o kojima ovisi tijek razgovora i odgovor chatbota.

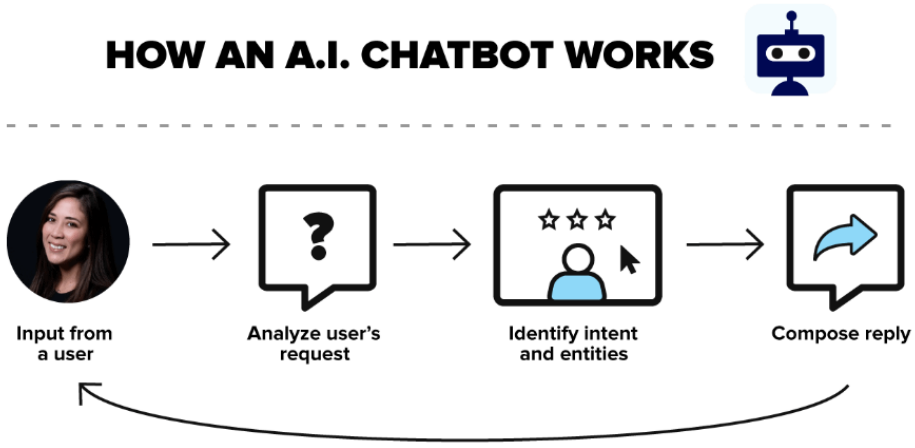



Slika 3.4. Primjer  chatbota temeljenog na pravilima [17]

### 3.2.2. Chatbot temeljen na umjetnoj inteligenciji

Umjetna inteligencija (engl. *AI - artificial intelligence*) chatbotovi koriste tehnologije obrade prirodnog jezika (NLP) kako bi razumjeli i interpretirali namjeru korisnikovih pitanja. Za razliku od chatbotova temeljenih na pravilima, AI chatbotovi se oslanjaju na modele strojnog učenja, što im omogućuje da prepoznaju i odgovore na širok raspon različitih upita. Ovi modeli kontinuirano uče i prilagođavaju se na temelju novih podataka, čime se poboljšava njihova sposobnost pružanja točnih i korisnih odgovora. Zahvaljujući svojoj sposobnosti da obrađuju i analiziraju velike količine informacija, AI chatbotovi mogu pružiti personaliziranija i složenija rješenja za korisnike, čineći ih idealnim za napredne i dinamične interakcije [17].

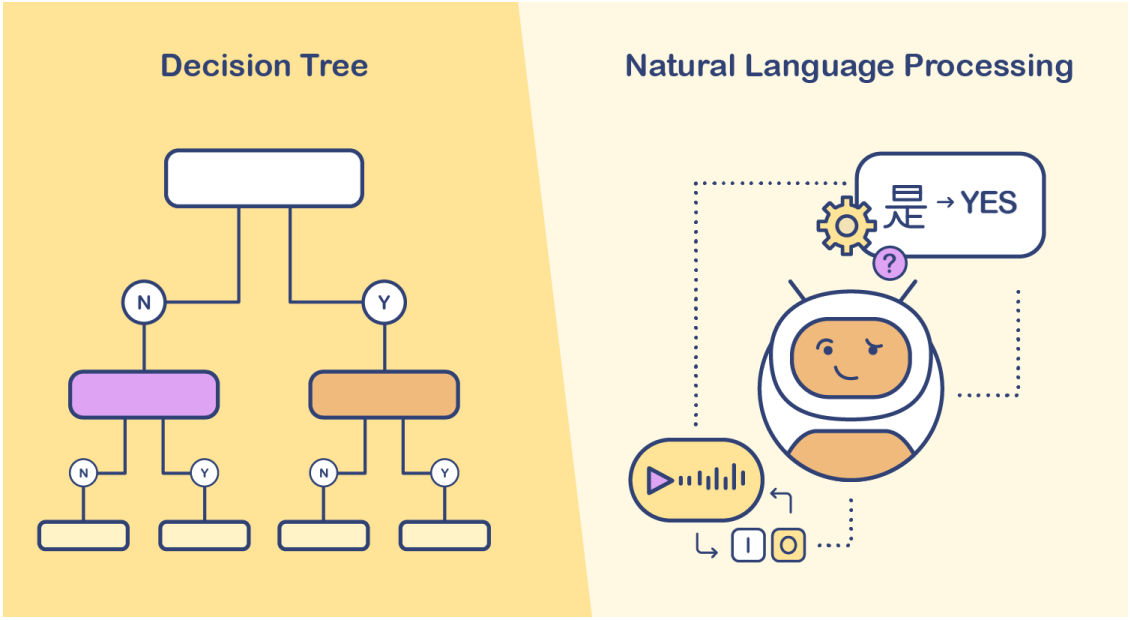
Na slici 2.1. prikazano je kako AI chatbot funkcionira na način da se korisnikov unos analizira metodama strojnog učenja i odlučuje odgovor koji će se poslati nazad korisniku.




Slika 3.5. Način rada AI chatbota [18]

### 3.2.3. Hibridni chatbot

Kombinacija prethodno dva spomenuta tipa chatbotova čini hibridni tip chatbota. Takvi tipovi chatbota koriste prednosti oba sustava kako bi se učinkovito prilagodili različitim situacijama i pružili bolje iskustvo. Mogu se koristiti u marketingu i prodaji, ali i drugim područjima. Slika 2.3. prikazuje način rada hibridnog chatbota. Na slici 2.3. prikazan je sastav hibridnog chatbota koji se koristi u ovom radu. S lijeve strane je tok razgovora, dok je s desne obrada korisničkog unosa. Izlazi modela odlučuju tok razgovora i rezultat na samom kraju razgovora.




Slika 3.6. Sastav hibridnog chatobota [19]

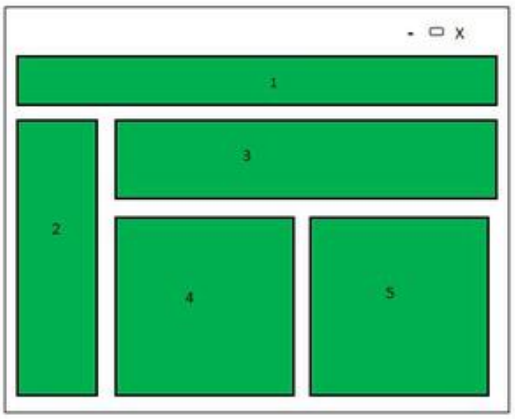
## 3.3. React

React je biblioteka koja se koristi za izgradnju korisničkog sučelja. Koristi se za izradu interaktivnih web aplikacija koje mogu učinkovito ažurirati i prikazivati promjene u podacima bez potrebe za ponovnim učitavanjem cijele stranice [20].

Jedna od najznačajnijih značajki Reacta su komponente ili elementi koji su opisani u nastavku.

### 3.3.1. Komponente

React aplikacije sastoje se od komponenti, koje su osnovni gradivni blokovi korisničkog sučelja. Svaka web aplikacija izrađena preko Reacta podijeljena je na komponente koje imaju svoju logiku i dizajn. Logika je kreirana preko JavaScript. Na slici 3.7. prikazana je shema web aplikacije gdje su komponente prikazane pravokutnicima.




Slika 3.7. Shema web aplikacije [21]

### 3.3.2. Životni ciklus komponente

Svaka React komponenta ima životni ciklus koji se može pratiti i mijenjati tijekom tri faze:

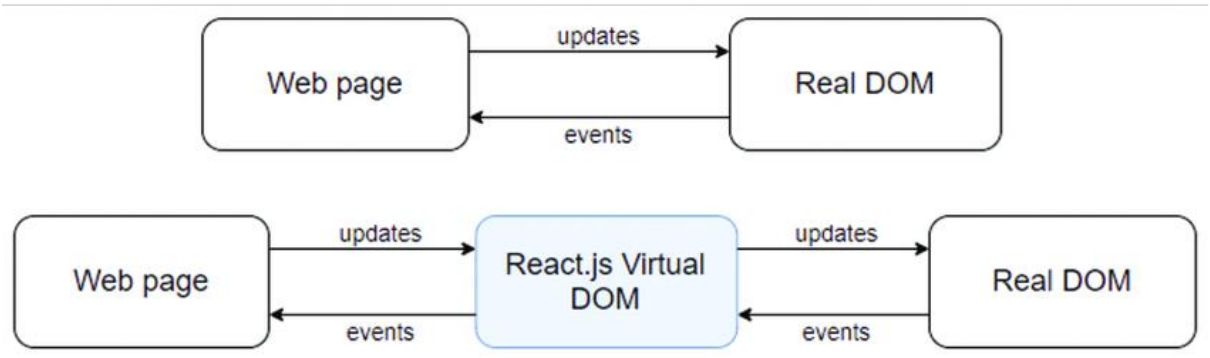
1. Postavljanje
2. Ažuriranje
3. Uklanjanje

Pojam "postavljanje" odnosi se na proces umetanja komponenata u Document Object Model (DOM). DOM je u osnovi standard za način na koji su podaci predstavljeni i pristupani na web stranici. Može se zamisliti kao stablo HTML/XML objekata koje se mogu donekle vizualizirati u alatima za razvojne programere za bilo koju web stranicu. Kad god se web stranica učita, preglednik stvara ovaj DOM sa svim sadržajem povezanim s tom stranicom. Pomoću JavaScript-a može se manipulirati i mijenjati ti sadržaji. Dakle, kada se React aplikacija učita, prvi korak je postavljanje gdje se komponente postavljaju u DOM. React sve ovo omogućava stvaranjem virtualnog DOM-a koji se zatim koristi za ažuriranje i sinkronizaciju sa stvarnim DOM-om.

Kada se komponenta ažurira, životni ciklus prelazi na sljedeći korak. Komponenta se smatra ažuriranom kada se promijeni njezino stanje ili svojstva (engl. *props*). U ovoj fazi, React automatski ažurira stanje komponente i ponovno je renderira kako bi prikazao najnovije podatke.

U posljednjoj fazi životnog ciklusa, komponenta se uklanja iz DOM-a (engl. *unmounting*). Kada se komponenta ukloni, React osigurava da se oslobode svi resursi povezani s tom komponentom, čime se optimizira performanse aplikacije [22].

Na slici 3.8. prikazan je način rada virtualnog i stvarnog DOM-a.




Slika 3.8. Način rada virtualnog i stvarnog DOM-a [23]

# 4. PROCES KREIRANJA CHATBOT APLIKACIJE

Za uspješnu implementaciju savjetovanja lijeka primjenom NLP-a. Prvi korak je kreiranje baza podataka za treniranje modela koji će prepoznati kategoriju i preko koje se pretražuje lijek. Uz baze podataka, programsko rješenje kroz koje je ostvaren tok razgovora i pronalazak lijeka završni je korak implementacije serverskog dijela chatbot aplikacije koji se koristi u ovom radu. Opisano programsko rješenje prima podatke korisnika i obrađuje ih, te vraća rezultate na klijentski dio koji je implementiran pomoću React alata.

## 4.1. Prikupljanje baze podataka

Za realizaciju preporuke lijeka na osnovu razgovora korisnika i chatbota, koriste se dvije baze podataka. Jedna se koristi za pretragu lijeka prema određenom algoritmu i sadrži sve relevantne podatke o lijeku, dok se druga koristi za NLP primjenu, odnosno za treniranje modela koji će prepoznati korisnikov unos odnosno rečenicu i izbaciti rezultat koji će se koristiti da daljnju uporabu.

### 4.1.1. Baza podataka lijekova

Kako bi se što efikasnije pripomoglo savjetovanju lijeka, potrebno je saznati što više detalja o njemu, poput sljedećih: za koje se simptome ili problem koristi, koja je minimalna dob za koju se smije koristiti, smiju li ga koristiti trudnice, za koje slučajeve se ne smije izdati itd.

U ovom dijelu prikazani je oblik baze podataka u json obliku sa nekim relevantnim podacima, koji su korisni za pretragu lijeka koristeći pružene odgovore odnosno rezultate NLP obrade.

Svaki lijek sadrži sve prethodno navedene detalje u obliku atributa i vrijednosti. Atribut predstavlja naziv kategorije odnosno podatka o lijeku, dok je vrijednost popis ključnih riječi, broj ili odgovor.

Prvi atribut je naziv  koji kao vrijednosti ima naziv određenog lijeka. Vrijednost tog atributa koristi se kao odgovor chatbota korisniku nakon pretrage lijekova.

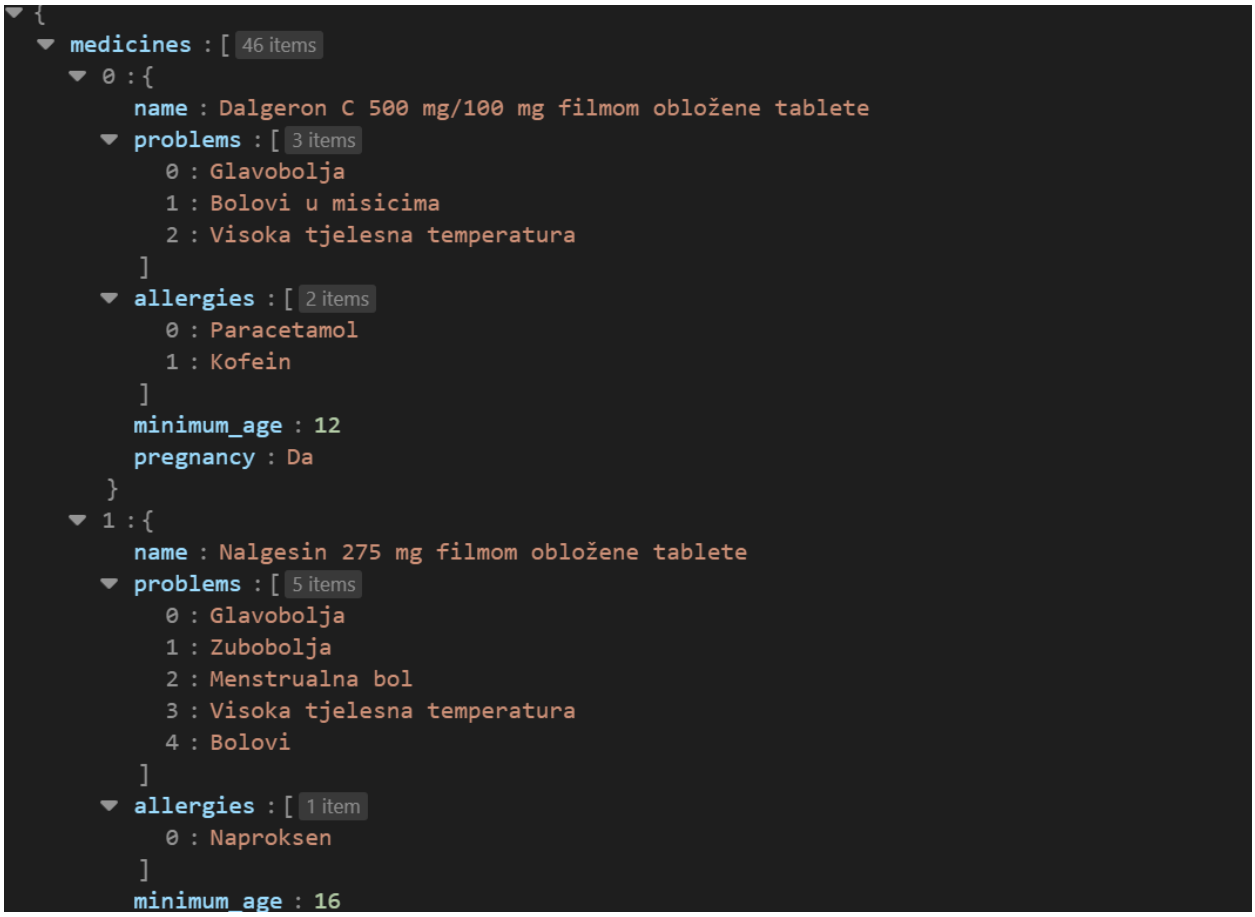
Drugi atribut je problem  i on sadržava popis ključnih riječi kao na primjer glavobolja, zubobolja, bol u trbuhu itd.

Treći atribut je alergija i on sadrži popis ključnih riječi poput nikotina, kukuruznog škroba, pantoprazola i raznih drugih riječi ovisno o sastavu koje predstavljaju dio sastava lijeka ili drugih stvari koje sadržavaju sastav lijeka.

Četvrti atribut je minimalna dob, ona kao vrijednost sadrži broj koji se uspoređuje sa korisnikovom dobi.

Zadnji atribut je trudnoća i on kao vrijednost sadrži „Da“ ili „Ne“ koji daju odgovor smiju li trudnice koristiti lijek.

Svi ti podaci korisni su algoritmu koji će na kraju filtrirati popis lijekova i pretraživati lijekove. Na sljedećoj slici prikazani su podaci o određenim lijekovima koji se koriste u ovom radu.




Slika 4.1. Podaci o lijekovima u json obliku

### 4.1.2. Baza podataka za treniranje modela

Osim baze podataka lijekova, koristi se i baza podataka koja sadrži različite probleme sa zdravstvenim stanjima i odgovarajuće izraze na kojima se model za obradu prirodnog jezika trenira. Takva baza podataka omogućuje modelu da nauči prepoznati različite kategorije problema na temelju korisničkog unosa. U bazi se nalaze sve kategorije koje se nalaze u listi problema baze

podataka lijekova zbog efikasnijeg pretraživanja lijekova. Osim kategorija problema, u bazi podataka su i kategorije za trudnoću, odnosno negaciju i potvrđivanje trudnoće. Na slici 3.2. prikazana je navedena baza podataka sa podacima unosa korisnika za treniranje i izlazima koje model daje.

```
"Dišni problem": ["dišni problem", "teškoće s disanjem", "otežano disanje","teže dišem"],
"Androgena alopecija": ["androgena alopecija","opada mi kosa","opadanje kose"],
"Jetra": ["oštećenje jetre","jetra","imam hepatitis"],
"Rinitis": ["rinitis","upala nosne suznice","kihanje","kišem"],
"Nadraženo oko": ["treba mi nešto protiv nadraženosti oka","crveno oko","nadraženo oko"],
"Težina": ["želim smršavjeti","mršavljenje","želim smanjiti kilažu","težina","smrsavjeti"],
"Alergija": ["alerija","alergijska reakcija"],
"Zaštita od trudnoće": ["spriječavanje trudnoće","zaštita od trudnoće"],
"Čišćenje usta": ["pročišćavanje usta","čišćenje usta"],
"Proširene vene": [" imam proširene vene","vene su vidljive plave boje"],
"Površinske rane": ["imam površinsku ranu","površinska rana"],
"Mokraćni mjehur": ["mjehur","mokraćni mjehur","treba mi nešto za mjehur"],
"Grlobolja": ["boli me grlo","grlobolja","bolovi u grlu"],
"Modrice": ["modrice","hematom"],
"Vrtoglavica": ["imam vrtoglavicu","vrti mi se"],
"Povraćanje": ["mučnina","povraćanje"],
"Herpes na ustima ili licu": ["herpes na usnama", "herpes na licu","herpes"],
"Zatvor": ["problemi sa  tvrdom stolicom", "zatvor","muči me zatvor"],
"Upala ždrijela": ["imam upalu ždrijela","ždrijelo"],
"Istegnuće": ["istegnuće", "nategnuće mišića","istegnuće gležnja","istegnula sam tetivu"],
"Sportska ozljeda": ["sportska ozljeda", "ozljeda tijekom sporta","upala mišića"],
"Gljivična infekcija": ["gljivična infekcija", "infekcija kože gljivicama"],
"Kožna infekcija": ["kožna infekcija","infekcija kože"],
"Pušenje": ["pušenje","želim prestati pušiti.","prestanak pušenja"],

"Da": ["da","jesam","Jesam"],
"Ne": ["ne","nisam","Nisam"]
```

Slika 4.2. Baza podataka za treniranje modela

## 4.2. Serverska strana

Programsko rješenje sadrži programski kod unutar Python programskog jezika koji obuhvaća treniranje modela, prepoznavanje simptoma i kategorije, i algoritme pronalaska lijeka i Levenshteinove udaljenosti.

### 4.2.1. Treniranje modela i prepoznavanje kategorije i godina

Za uspješno treniranje modela koristi se json baza podataka koju koristi model logističke regresije za treniranje. Na slici 4.1. prikazani su svi alati i moduli koji se koriste u programu.

```
import json
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
```

Slika 4.3. Biblioteke koje se koriste unutar programa

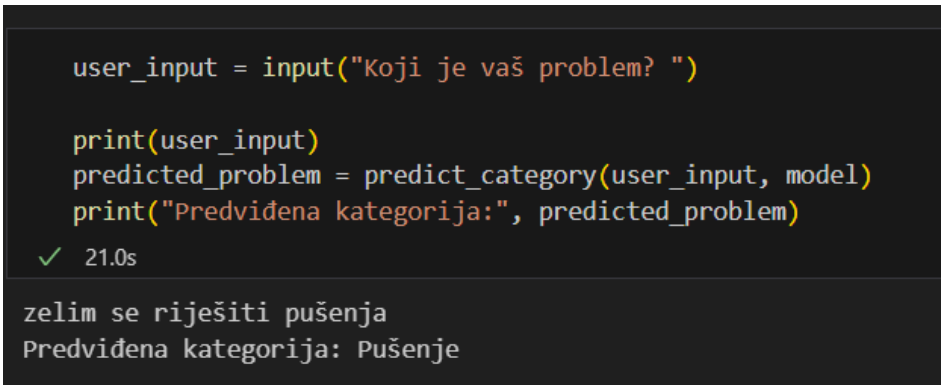
Modul json u Pythonu predstavlja ključni alat za rad s podacima u formatu JSON (JavaScript Object Notation). JSON je lagan format za razmjenu podataka, koji je jednostavan za čitanje i pisanje, a istovremeno jednostavan strojevima za parsiranje i generiranje [24].

Uz json, iz biblioteke sklearn koriste se alati CountVectorizer, LogisticRegresion i make_pipeline. CountVectorizer pretvara podatke u vektorski oblik koji je nužan za treniranje modela koji koristi logističku regresiju za klasifikaciju teksta. Alat make_pipeline koristi se za bolji redoslijed izvođenja programa.

Uz otvaranje json baze podataka koja je prikazana na slici 4.2. , isti podaci se koriste za treniranje, gdje su izlazi kategorije odnosno problemi, a ulazi rečenice. Na slici 4.3. prikazan je proces otvaranja i treniranja podataka za prepoznavanje problema i trudnoće.


Slika 4.4. Programski kod za treniranje modela, predviđanje kategorije i otvaranje podataka

Na slici 4.5. i 4.6. prikazan je primjer korisničkog unosa problema i predviđena kategorija problema i trudnoće.




Slika 4.5. Primjer prepoznavanja kategorije problema

```
pregnancy_input = input("Jeste li trudni? ")

print(pregnancy_input)
predicted_pregnancy = predict_category(pregnancy_input, model)
print("Predviđena kategorija:", predicted_pregnancy)
```
✓ 2.1s

```
jesam
Predviđena kategorija: Da
```

Slika 4.6. Primjer prepoznavanja statusa trudnoće

Jedan od ključnih informacija za pronalazak lijeka su i korisnikove godine. Za prikupljanje godina nije potreban model, već numerički unos korisnikovih godina. Na slici 4.7. prikazan je primjer unosa korisnikovih godina.

```
year_input = input("Koliko imate godina? ")

print(year_input)
year = int(year_input)
print("Godine:", year)
```
✓ 1.4s

```
22
Predviđena kategorija: 22
```

Slika 4.7. Unos korisnikovih godina

### 4.2.2. Primjena Levenshteinove udaljenosti za prepoznavanje alergija

Prepoznavanje problema korisnika najčešće rezultira jednim problemom, ali kod prepoznavanja simptoma postoje situacije gdje je korisnik alergičan na više stvari. Kreiranje baze podataka sa kombinacijama više izlaza kako bi se pokrili svi scenariji bio bi zahtjevan posao. Stoga se pristupa rješenju gdje bi se iz korisnikovog unosa izvukle značajke odnosno riječi koje se uspoređuju sa listom riječi svih sastojaka, biljaka ili nečeg sličnog što sadrži u sebi nešto što se nalazi i u samom lijeku. Takav problem rješava se primjenom Levenshteinove udaljenosti.

Problem se rješava tako što se korisnikov unos rečenice razdvaja na riječi, te se svaka riječ uspoređuje sa svakom postojećom riječi sastojaka. Ukoliko je postotak sličnosti između te dvije riječi veći od postavljenog praga sličnosti , sastojak, biljka ili nešto što sadrži navedeno dodaje se u listu alergija. Na slici 4.8. je prikazana lista sastava preuzetih lijekova koji se koriste za usporedbu.

```
alergeni = [
    "Paracetamol", "Kofein", "Naproksen", "Natrijev alginat", "Natrijev hidrogenkarbonat",
    "Bršljan", "Pantoprazol", "Lopermamid klorid", "Peruanski balzam", "Levomentol",
    "Loperamid klorid", "Simetikon", "Bromheksin klorid", "Metilparahidroksibezoat",
    "Islandski lišaj", "Kukuruzni škrob", "Ibuprofen", "Minoksidil", "Natrijev škroboglikolat",
    "Manitol", "Oksimetazolin klorid", "Tetrizolin klorid", "Orlistat", "Desloratadin",
    "Loratidin", "Ulipristal acetat", "Heksetidin", "Heparin", "Tirotricin", "Sabal",
    "Flurbiprofen", "Diklofenak natrij", "Ksilometazolin klorid", "Linex Baby prašak",
    "Lizozim klorid piridoksinklorid", "Kamilica", "Dimenhidrinat", "Aciklovir", "Glicerol",
    "Klorheksidindiklorid lidokainklorid", "Đumbir", "Klotrimazol", "Nikotin"
]
✓ 0.0s
```

Slika 4.8. Lista alergena

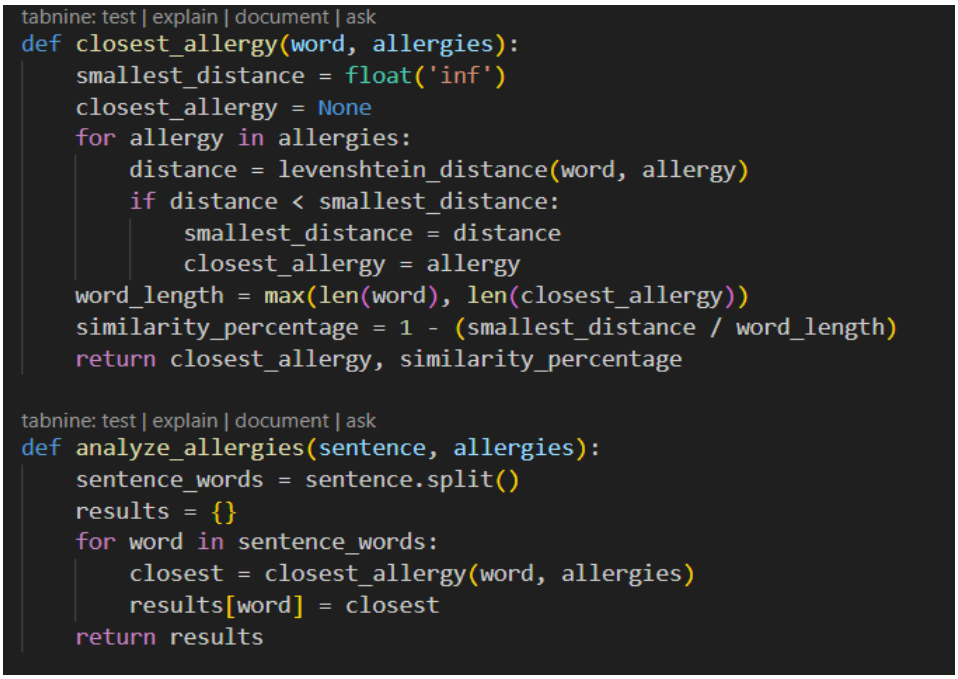
Na slici 4.9. prikazan je algoritam Levenshteinove udaljenosti gdje se za unos dvije riječi izračuna najmanji broj operacija umetanja, brisanja ili zamjene znakova da se prva riječ pretvori u drugu. Funkcija započinje kreiranjem matrice koja ima broj redaka veličine prve riječi, a broj stupaca veličine druge riječi. Ova matrica služi za pohranu troškova transformacije prvih *i* znakova prvog stringa u prvih *j* znakova drugog stringa. Svaka ćelija matrice predstavlja jedan korak u transformaciji. Prvi redak i prva kolona matrice se popunjavaju inicijalnim vrijednostima. Prvi redak se popunjava vrijednostima od 0 do *len_s2*, a prva kolona vrijednostima od 0 do *len_s1*. Ove vrijednosti predstavljaju trošak transformacije praznog stringa u string određene duljine isključivo pomoću umetanja ili brisanja znakova. Nakon inicijalizacije, algoritam prolazi kroz ostatak matrice koristeći dvije ugniježđene petlje. Za svaku ćeliju matrice izračunava se trošak

transformacije. Ako su znakovi na odgovarajućim pozicijama u stringovima jednaki, trošak je 0, inače je 1. Algoritam zatim izračunava minimalni trošak među tri moguće operacije: brisanje, umetanje i zamjena, te postavlja tu vrijednost u trenutnu ćeliju matrice. Nakon što su sve ćelije matrice popunjene, vrijednost u donjem desnom kutu matrice predstavlja Levenshteinovu udaljenost između dva stringa.

```
def levenshtein_distance(s1, s2):

    len_s1 = len(s1)
    len_s2 = len(s2)
    matrix = [[0] * (len_s2 + 1) for _ in range(len_s1 + 1)]


    for i in range(len_s1 + 1):
        matrix[i][0] = i
    for j in range(len_s2 + 1):
        matrix[0][j] = j


    for i in range(1, len_s1 + 1):
        for j in range(1, len_s2 + 1):
            cost = 0 if s1[i - 1] == s2[j - 1] else 1
            matrix[i][j] = min(matrix[i - 1][j] + 1,
                               matrix[i][j - 1] + 1,
                               matrix[i - 1][j - 1] + cost)

    return matrix[len_s1][len_s2]
```

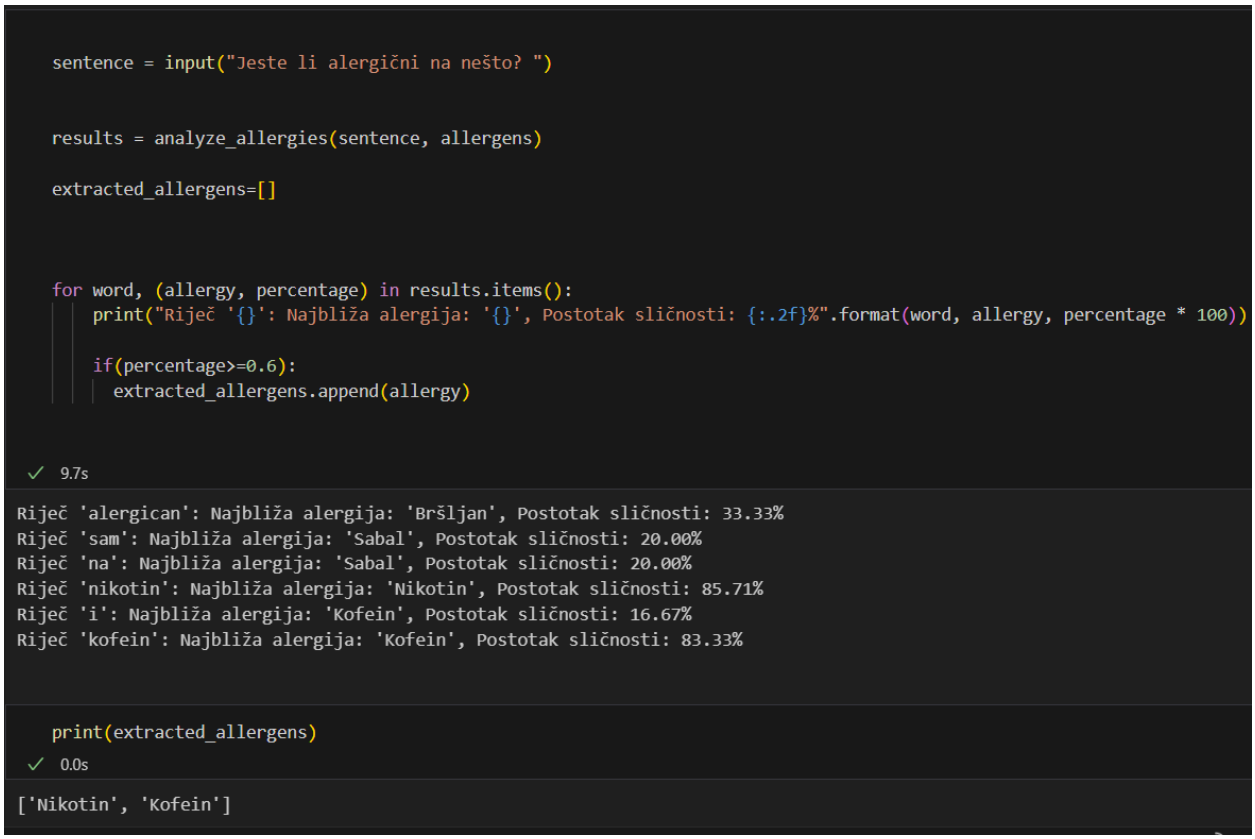
Slika 4.8. Algoritam Levenshteinove udaljenosti

Nakon liste alergena i algoritma Levenshteinove udaljenosti, za stvaranje liste izvučenih alergena iz rečenice, definiraju se funkcije za pronalazak najsličnijeg alergena za riječ i funkcija za stvaranje liste gdje je za svaku riječ pronađen najsličniji alergen. Funkcije su prikazane na slici 4.9.




Slika 4.9. Funkcije za pronalazak najsličnijih alergena za svaku riječ iz rečenice

Na slici 4.10. prikazan je primjer izvučenih alergena iz korisnikovog unosa. Prag sličnosti za koje se riječ prihvaća kao postojeći alergen ovom radu je iznad 0.6 (60 %).




Slika 4.10. Primjer rezultata izvučenih alergena

### 4.2.3. Algoritam za pronalazak lijeka

Nakon izvučenih značajki primjenom NLP-a, koristi se algoritam koji za prikupljene značajke pronalazi sve lijekove koji sadrže prepoznat problem, zatim ukoliko je korisnik trudnica, eliminira preostale lijekove koji nisu za trudnice ukoliko je korisnik trudnica i koji sadrže izvučene alergije u sebi, te u konačnici od preostalih lijekova eliminira sve koji imaju dobnu granicu veću od korisnikove. Na slici 4.11. prikazan je opisani algoritam.

```
def find_medicines_by_symptom_and_allergies(medicine_database, symptom, allergies, pregnancy, age):
    appropriate_medicines = []
    for medicine in medicine_database["medicines"]:
        if symptom in medicine["problems"]:
            if pregnancy == "Ne" or (pregnancy == "Da" and medicine["pregnancy"] == "Da"):
                if not any(allergy in medicine["allergies"] for allergy in allergies):
                    if age is None or medicine["minimum_age"] <= age:
                        appropriate_medicines.append(medicine)
    return appropriate_medicines
```

Slika 4.11. Algoritam za pronalazak lijeka

### 4.2.4. Obrada zahtjeva sa serverske strane

Serverska strana aplikacije, implementirana u Pythonu koristeći Flask framework, prima podatke u JSON formatu. Kada klijentska strana pošalje odgovore, Flask aplikacija ih obrađuje i vraća relevantne informacije natrag klijentu.

Kada korisnik odgovori na sva pitanja, klijentska aplikacija šalje niz odgovora na serversku stranu putem HTTP POST zahtjeva [25].

Na slici 4.12 prikazana je funkcija koja prima listu rečenica s klijentove strane, obrađuje ih prema prethodno opisanim NLP metodama, pretražuje lijekove i šalje rezultate klijentskoj strani.

```python
@app.route('/process', methods=['POST'])
tabnine: test | explain | document | ask
def process_strings():
    data = request.json
    responses = data.get('responses', [])

    if not data:
        return jsonify({'error': 'Invalid JSON sent or no JSON received'}), 400
    responses = data.get('responses', [])
    if len(responses) < 4:
        return jsonify({'error': 'Please provide all four responses.'}), 400

    user_problem = responses[0]
    predicted_problem = predict_category(user_problem, model)
    user_year = responses[1]

    try:
        year = int(user_year)
    except ValueError:
        year = None
    results = analyze_allergies(responses[2], allergies)
    extracted_allergies = [allergy for word, (allergy, percentage) in results.items() if percentage >= 0.6]
    user_pregnancy = responses[3]
    predicted_pregnancy = predict_category(user_pregnancy, model)

    with open("medicines.json", "r", encoding="utf-8") as file:
        medicine_database = json.load(file)
    appropriate_medicines = find_medicines_by_symptom_and_allergies(medicine_database, predicted_problem, extracted_allergies, predicted_pregnancy,
     year)
    response_medicines = "\n".join(medicine["name"] for medicine in appropriate_medicines)

    return jsonify({'appropriate_medicines': response_medicines})
```

Slika 4.12. Primanje, obrada i slanje podataka sa serverske strane

## 4.3. Klijentska strana

U ovom dijelu detaljno se opisuje proces izrade, povezanost sa serverskom stranom te način komunikacije između klijentske i serverske strane.

Na početku razvoja, kreirana je osnovna struktura React aplikacije. Aplikacija je podijeljena u nekoliko ključnih komponenti:

1. App: glavna komponenta koja upravlja stanjem aplikacije i služi kao kontejner za ostale komponente.
2. ChatWindow: Komponenta odgovorna za prikazivanje poruka između korisnika i chatbota.
3. ChatInput: Komponenta koja omogućava korisniku unos poruka i slanje istih botu.

Sve tri komponente sadrže svoju logiku, no programsko rješenje unutar App komponente obuhvaća dio u kojem se niz odgovora šalje prema serverskoj strani. Programsko rješenje je opisano u nastavku.

### 4.3.1. App komponenta

App je funkcionalna komponenta koja koristi *useState* funkciju za upravljanje stanjem poruka, trenutnim indeksom pitanja i odgovorima korisnika. Funkcija *useEffect* se koristi za postavljanje početnog stanja, gdje chatbot odmah postavlja prvo pitanje iz niza *questions*. Na slici 4.13. prikazan je niz i navedene funkcije.

```
const questions = [
  'Koji je vaš problem?',
  'Koliko imate godina?',
  'Jeste li alergični na nešto?',
  'Jeste li trudnica?'
];

const App = () => {
  const [messages, setMessages] = useState([]);
  const [currentQuestionIndex, setCurrentQuestionIndex] = useState(0);
  const [responses, setResponses] = useState([]);

  useEffect(() => {

    setMessages([{ sender: 'bot', text: questions[currentQuestionIndex] }]);
  }, []);
```

Slika 4.13. Niz pitanja chatbota i funkcije za postavljanje prvog pitanja i upravljanje porukama

Na slikama 4.14. i 4.15. prikazana je asinkrona funkcija *handleSendMessage* koja se poziva kada korisnik pošalje poruku. Ona prvo dodaje korisničku poruku u stanje messages. Zatim ažurira stanje responses s novim odgovorom korisnika. Ako postoje još pitanja, chatbot postavlja sljedeće pitanje. Ako su sva pitanja postavljena, odgovori korisnika se šalju serveru putem POST zahtjeva na */process* endpoint. Na temelju odgovora servera, chatbot prikazuje odgovarajuću poruku.

```
const handleSendMessage = async (message) => {
  const userMessage = { sender: 'user', text: message };
  setMessages((prevMessages) => [...prevMessages, userMessage]);
  const updatedResponses = [...responses, message];
  setResponses(updatedResponses);

  const nextQuestionIndex = currentQuestionIndex + 1;

  if (nextQuestionIndex < questions.length) {
    const botMessage = { sender: 'bot', text: questions[nextQuestionIndex] };
    setMessages((prevMessages) => [...prevMessages, botMessage]);
    setCurrentQuestionIndex(nextQuestionIndex);
  } else {

    if (updatedResponses.length === 4) {

      console.log(updatedResponses);

      try {
        const response = await fetch('http://127.0.0.1:5000/process', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json; charset=UTF-8'
          },

          body: JSON.stringify({ responses: updatedResponses })
        });



        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }
```

Slika 4.14. Dio programskog koda funkcije *handleSendMessage*

```
        const data = await response.json();

        console.log('Server response:', data);

        if (data.appropriate_medicines) {
          const botMessage = { sender: 'bot', text: 'Hvala na informacijama! Evo savjeta: ' + data.appropriate_medicines };
          setMessages((prevMessages) => [...prevMessages, botMessage]);
        } else {
          const botMessage = { sender: 'bot', text: 'S obzirom na vaše odgovore nisam u mogoućnosti ispisati vam lijek. Savjetujem da se posavjetujete s liječnikom. ' };
          setMessages((prevMessages) => [...prevMessages, botMessage]);
        }
      } catch (error) {
        console.error('Error:', error);
      }
    } else {
      console.error('Expected 4 responses, but got', updatedResponses.length);
    }
  }
};
```

Slika 4.15. Dio programskog koda funkcije handleSendMessage

U konačnici se unutar App komponente vrši renderiranje komponenata za prikaz poruka i unos poruka [26]. Programski dio za renderiranje prikazan je na slici 4.16.

```
  return (
    <AppContainer>
      <ChatWindow messages={messages} />
      <ChatInput onSendMessage={handleSendMessage} onReload={handleReload} />
    </AppContainer>
  );
};
```

Slika 4.16. Renderiranje komponenti

### 4.3.2. ChatInput komponenta

Komponenta ChatInput u Reactu odgovorna je za unos korisničkih poruka i omogućavanje resetiranja chata. Ova komponenta koristi funkcionalne komponente i Reactov hook *useState* za upravljanje stanjem unosa teksta [27].

Pomoću *useState* definira se stanje *message*, koje čuva trenutno uneseni tekst u input polju, te vraća trenutnu vrijednost stanja i funkciju za njegovo ažuriranje.

Funkcija *handleSend* se poziva kada korisnik klikne na gumb *Send* ili pritisne tipku Enter. Prvo provjerava je li unesena poruka prazna nakon uklanjanja eventualnih praznina s početka i kraja. Ako poruka nije prazna, poziva se *onSendMessage* funkcija koja je prošla kao prop komponenti ChatInput, te se stanje message resetira na prazan string.

Na kraju se renderiraju komponente za slanje poruke, resetiranje chata i unos poruke.

Sve navedene funkcionalnosti prikazane su na slici 4.17.

```jsx
const ChatInput = ({ onSendMessage, onReload }) => {
  const [message, setMessage] = useState('');

  const handleSend = () => {
    if (message.trim() !== '') {
      onSendMessage(message);
      setMessage('');
    }
  };

  return (
    <InputContainer>
      <Input
        type="text"
        value={message}
        onChange={(e) => setMessage(e.target.value)}
        onKeyPress={(e) => e.key === 'Enter' && handleSend()}
      />
      <Button onClick={handleSend}>Send</Button>
      <ReloadButton onClick={onReload}>Reset</ReloadButton>
    </InputContainer>
  );
};
```

Slika 4.17. Programski kod ChatInput komponente

### 4.3.3. ChatWindow komponenta

Komponenta ChatWindow u Reactu odgovorna je za prikazivanje poruka koje su poslane i primljene u chatu. Ova komponenta je jednostavna i funkcionalna, oslanjajući se na props za dobivanje podataka o porukama koje treba prikazati [28].

Na slici 4.18. prikazan je programski kod komponente.

```jsx
const ChatWindow = ({ messages }) => {
  return (
    <ChatContainer>
      {messages.map((msg, index) => (
        <Message key={index} $sender={msg.sender}>
          {msg.text}
        </Message>
      ))}
    </ChatContainer>
  );
};
```

Slika 4.18. Programski kod ChatWindow komponente

Komponenta ChatWindow prima jedan prop, *messages*, koji je niz objekata. Svaki objekt predstavlja jednu poruku i sadrži *text* (tekst poruke) i *sender* (informaciju o tome tko je poslao poruku, korisnik ili bot).

Unutar ChatContainer div elementa, ChatWindow mapira niz *messages*. Svaka poruka je predstavljena kao Message komponenta s jedinstvenim ključem (key) kako bi se optimiziralo renderiranje i spriječili problemi s performansama.

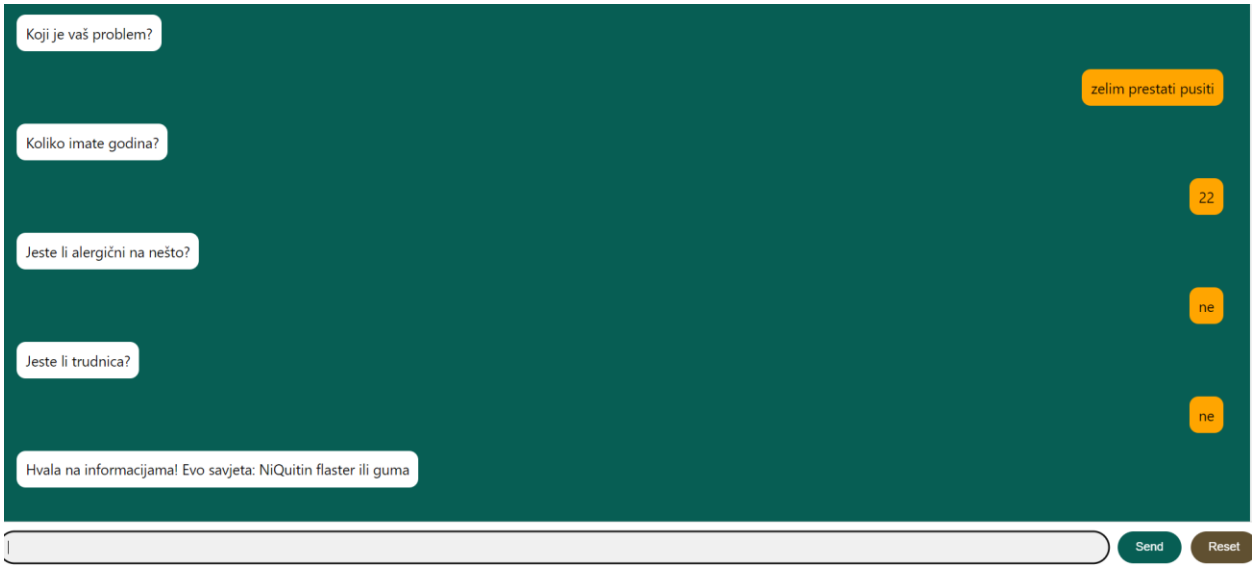
## 4.4. Komunikacija klijentske i serverske strane

Nakon kreiranja klijentske strane uz korištenje CSS-a i JavaScript-a za dizajn i logiku komponenti, i kreiranja serverske strane, sljedeći korak je pokretanje servera i React aplikacije preko terminala koristeći naredbe npm start i python app.py unutar direktorija aplikacije [29].

Naredba npm start je za pokretanje React aplikacije, dok je python app.py naredba za pokretanje skripte app.py koja se koristi u ovom radu. Za uspješnu komunikaciju servera i klijenta, obje naredbe se pokreću u različitim terminalima.

# 5. PRIKAZ REZULTATA UNUTAR APLIKACIJE

U ovom poglavlju prikazuju se primjeri toka razgovora kroz chatbot aplikaciju. Na slikama 5.1. do 5.3. prikazani su primjeri gdje se niti jedan lijek ne smije preporučiti zbog korisnikovog stanja, ali i primjeri gdje se savjetuju određeni lijekovi.




Slika 5.1. Primjer toka razgovora unutar aplikacije


Slika 5.2. Primjer toka razgovora unutar aplikacije


5.3. Primjer toka razgovora unutar aplikacije

# 6. ZAKLJUČAK

U ovom radu opisana je primjena NLP-a i prikazani su primjeri postojećih rješenja u području medicine koji imaju sličan pristupa kao i aplikacija koja se koristi u ovom radu. Opisani su alati i tehnologije u kojima je detaljnije opisan NLP i metode kroz koje se primjenjuje kroz rad. Prikazane su i opisane vrste chatbot tehnologija, te je opisana i React biblioteka kroz koju je napravljena aplikacija.

Nakon teorijske podloge, prikazane su i baze podataka koje se koriste za treniranje modela i pretragu lijeka. Opisan je proces kreiranja aplikacije u kojem se detaljnije analiziraju metode prepoznavanja simptoma i trudnoće, uz metodu izvlačenja alergena iz rečenice koja je ostvarena Levenshteinovom udaljenošću. Osim navedenih metoda koje su korištene unutar Python skripte, prikazano je i programsko rješenje klijentske strane koja je ostvarena pomoću Reacta, te je prikazan i način pokretanja aplikacije i servera koji je opisan i kreiran unutar Pythona.

U konačnici su prikazani rezultati chatbot aplikacije kroz različite primjere. U radu je prikazan hibridni chatbot, koji koristi kreirana pravila toka kroz koje se prikupljaju podaci i pretražuje lijek, ali i NLP metode za obradu korisnikovih odgovora.

Ovim radom uspješno je prikazan rad chatbota koji koristi NLP metode za izvlačenje značajki iz korisnikovih rečenica. Razlog odabira hibridnog chatbota je minimizacija greške u savjetovanju lijeka i rizika ljudskog zdravlja. Kroz hibridni chatbot, prikupljani su podaci o korisnikovom problemu, godinama, statusu trudnoće i mogućim alergijama. Rizik ljudskog zdravlja koji ovisi o savjetu može se smanjiti još više uporabom većeg broja podataka o korisniku poput statusa dojenja, statusa o korištenju alkohola, statusa vozača i raznih drugih primjera.

# LITERATURA

[1] M. Petrović, OTC lijekovi – kakvi su to lijekovi i koliko su sigurni?: Sigurno samoliječenje uz stručni savjet [online], PHOENIX Farmacija d.o.o. , 2021, dostupno na: https://www.adiva.hr/zdravlje/lijekovi-i-terapije/otc-lijekovi-kakvi-su-to-lijekovi-i-koliko-su-sigurni/ [28.6.2024.]

[2] L. Pollock, What Is an NLP Chatbot — And How Do NLP-Powered Bots Work?: What is an NLP chatbot? [online], Ultimate, 2023, dostupno na: https://www.ultimate.ai/blog/ai-automation/how-nlp-text-based-chatbots-work [28.6.2024.]

[3] D. Nelson, Što je NLP (obrada prirodnog jezika)?: Slučajevi upotrebe za obradu prirodnog jezika (NLP) [online], Unite.AI, 2024, dostupno na: https://dir.hr/sto-je-obrada-prirodnog-jezika/ [30.6.2024.]

[4] S. Cain, What's that rash? This app uses AI to diagnose your symptoms [online], Fortune, 2024, dostupno na: https://fortune.com/2024/01/04/ai-make-business-better-ada-health/ [28.6.2024.]

[5] How does ada health use ML and NLP? [online], PubGenius inc. , dostupno na: https://typeset.io/questions/how-does-ada-health-use-ml-and-nlp-39z8eurf30 [28.6.2024.]

[6] T. Zajc, Natural Language Processing Is The Assistant Healthcare Has Been Waiting For: Symptom Checking, Voice Recognition [online], Data Structuring, and Conversational AI, Tjaša Zajc, 2023, dostupno na: https://fodh.substack.com/p/natural-language-processing-is-the [28.6.2024.]

[7] I. Strazhevich, How AI in mobile apps is making healthcare accessible and efficient: 4.Buoy Health [online], Medium, 2023, dostupno na: https://bootcamp.uxdesign.cc/revolutionizinghealthtechtech-with-ai-how-ai-in-mobile-apps-is-making-healthcare-accessible-and-713d0d5ac5f6 [28.6.2024.]

[8] P. Goyal, S. Pandey, K. Jain, Deep Learning for Natural Language Processing, Apress, 2018.

[9] R. Wollf, Natural Language Processing (NLP): 7 Key Techniques: Natural Language Processing Techniques [online], MonkeyLearn Inc., 2021 , dostupno na: https://monkeylearn.com/blog/natural-language-processing-techniques/ [29.6.2024.]

[10] M. Gireddy, Text Classification using Logistic Regression: How Logistic Regression Works for Text Classification? [online], GeeksForGeeks, 2024., dostupno na: https://www.geeksforgeeks.org/text-classification-using-logistic-regression/ [29.6.2024.]

[11] N. Singh, Sigmoid to Success: Unveiling Logistic Regression with OVR/OVA: Sigmoid Function, Medium [online], 2023, dostupno na: https://pub.aimind.so/sigmoid-to-success-unveiling-logistic-regression-with-ovr-ova-e56b17c93254 [29.6.2024]

[12] K. Verma, Using CountVectorizer to Extracting Features from Text, GeeksForGeeks [online], 2022, dostupno na: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/ [29.6.2024.]

[13] S. Singh, CountVectorizer vs TfidfVectorizer: 1.CountVectorizer [online], Medium, 2022, dostupno na: https://medium.com/@shandeep92/countvectorizer-vs-tfidfvectorizer-cf62d0a54fa4 [29.6.2024.]

[14] Z. Khan, Introduction to Levenshtein distance, GeeksForGeeks [online], 2024, dostupno na: https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/ [29.6.2024.]

[15] D. Venditama, Levenshtein Distance for Dummies [online], Medium, 2020, dostupno na: https://medium.com/analytics-vidhya/levenshtein-distance-for-dummies-dd9eb83d3e09 [29.6.2024]

[16] M. Goyachko, What Is a Chatbot? [online], IBM, Armonk, 2023, dostupno na: https://www.ibm.com/topics/chatbots [29.6.2024.]

[17] D. Lishchynska, How Do Chatbots Work? The Basics of Conversational AI [online], BotsCrew, Lviv, 2017., dostupno na: https://botscrew.com/blog/what-are-bots/ [29.6.2024.].

[18] An Introduction to AI Chatbots and Natural Language Processing [online], Drift, Boston, 2023, dostupno na: https://www.drift.com/learn/chatbot/ai-chatbots/ [29.6.2024.]

[19] E. Srivastava, Decision Tree Vs Natural Language Processing: What Chatbot Type Is Better? [online], HappyFox Inc., 2021, dostupno na: https://blog.happyfox.com/decision-tree-vs-natural-language-processing-what-chatbot-type-is-better/ [29.6.2024.]

[20] C. Deshpande, The Best Guide to Know What Is React: What is React, Simplilearn Solutions, 2023, dostupno na: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs [29.6.2024.]

[21] React Introduction: Features of React [online], GeeksForGeeks, 2024., dostupno na: https://www.geeksforgeeks.org/reactjs-introduction/ [datum zadnje posjete stranici: 29.6.2024.]

[22] Editorial @ TRN, Everything you need to know about React: How does React work? [online], Medium, 2021, dostupno na: https://medium.com/the-research-nest/everything-you-need-to-know-about-react-ab24da4275ea [29.6.2024.]

[23] D. F. Oliveira, J. P. Gomes, R. B. Pereira, i R.-J. Machado, Real DOM and React Virtual DOM, Computers 11,131, str. 7, kolovoz 2022.

[24] Import data from JSON files [online], Memgraph, 2024, dostupno na: https://memgraph.com/docs /data-migration/json [29.6.2024.]

[25] P. Yadav, How to connect ReactJS with Flask API? [online], Tutorials Point, 2023, dostupno na: https://www.tutorialspoint.com/how-to-connect-reactjs-with-flask-api [29.6.2024.]

[26] Render and Commit [online], React, 2024, dostupno na: https://react.dev/learn/render-and-commit [29.6.2024.]

[27] useState Hook [online], Codefinity, 2024, dostupno na: https://codefinity.com/courses/v2/1dcaf86a-11aa-492e-8e1d-06e055479aa9/cd1309fd-41bb-42db-a0ea-9aa57865bb80/eee2805d-b58e-45b7-9a6a-3785bbb408d5 [29.6.2024.]

[28] Props in React [online], Codefinity, 2024, dostupno na: https://codefinity.com/courses/v2/1dcaf86a-11aa-492e-8e1d-06e055479aa9/9ea7774a-16ac-4122-9a8b-a9a9d105e7c5/ff66ca74-1cf6-4651-bc74-9313c54e2b73 [29.6.2024.]

[29] L. K. Cox, Coding for Web Design 101: How HTML, CSS, and JavaScript Work [online], HubSpot, 2021, dostupno na: https://blog.hubspot.com/marketing/web-design-html-css-javascript [29.6.2024.]

# SAŽETAK

U ovom diplomskom radu prikazani su primjeri postojećih rješenja koja primjenjuju NLP. Objašnjena je teorijska podloga chatbot tehnologije, NLP-a, njenih metoda koje se koriste u radu i React-a kroz koji je napravljena chatbot aplikacija. Prikazano je programsko rješenje unutar Python skripte koje koristi algoritme Levenshteinove udaljenosti i pretrage lijeka, ali i metode strojnog učenja preko kojih se ostvaruje izvlačenje značajki iz korisnikovih rečenica. Uz programsko rješenje logike chatbota, prikazan je i klijentski dio aplikacije kroz React kao i način komunikacije klijentske i serverske strane. U konačnici je prikazan izgled aplikacije kroz primjere razgovora s botom unutar sučelja aplikacije. Prikazani su razni primjeri razgovora koji vode do različitih rezultata savjeta lijeka.

Ključne riječi: chatbot, Levenshtein, lijek, NLP, React

## ABSTRACT

This thesis presents examples of existing solutions that apply NLP. The theoretical basis of chatbot technology, NLP, its methods used in work and React, through which the chatbot application was made, were explained. A program solution within a Python script that uses Levenshtein distance algorithms and searches for a cure, as well as machine learning methods that are used to extract features from user sentences, is presented. In addition to the software solution of the chatbot logic, the client part of the application through React, as well as the way of communication between the client and server side, is shown. Finally, the appearance of the application is shown through examples of conversations from the bottom inside the application interface. Different examples of conversations leading to different drug advice results are shown.



Keywords: chatbot, drug, Levenshtein, NLP, React

# PRILOZI

Prilog 1. Programsko rješenje aplikacije : https://github.com/dosmakcic/Pharmabot-React-App