SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

VARAŽDIN

Tajana Šokec

# MODELIRANJE KONTEKSTUALNO SVJESNOG AGENTA ZA RAZGOVOR NA HRVATSKOM JEZIKU UZ POMOĆ KONAČNOG AUTOMATA I STROJNOG UČENJA

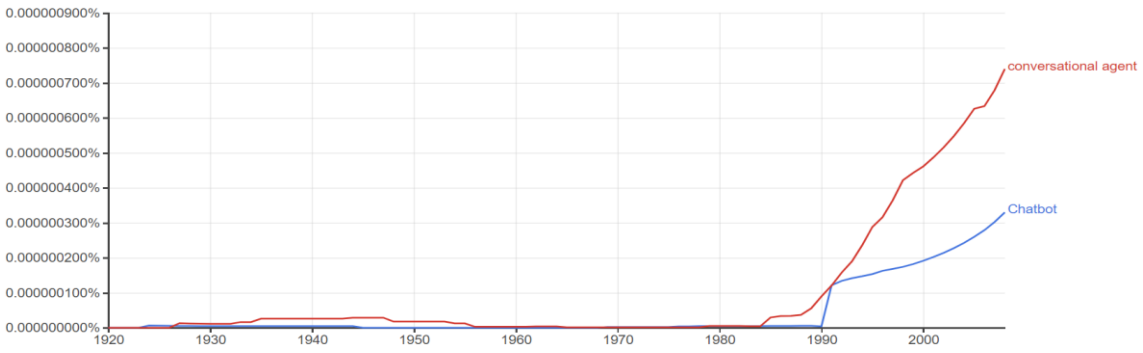
Varaždin, 2019.

*Ovaj rad izrađen je u Laboratoriju za umjetnu inteligenciju na Fakultetu organizacije i informatike u Varaždinu pod vodstvom Izv. prof. dr. sc. Markusa Schattena i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2018./2019.*

# Sadržaj rada

# 1. Uvod

Slika 1 prikazuje trend pojavljivanja fraza „conversational agent“ (agent za razgovor) i „Chatbot“[1] u zbirci knjiga s Google Knjiga [2] napisanih na Engleskom jeziku od 1920. do 2018. godine. Možemo vidjeti da, iako ti pojmovi imaju daleku povijest, njihovo značajnije korištenje počelo je tek nedavno. Jednim od prvih Chatbota smatra se program Eliza kojeg je 1966. godine izradio Joseph Weizenbaum. Program Eliza vodi razgovor s korisnikom tako da u korisnikovom unosu pronalazi ključne riječi na temelju kojih daje odgovore, a ako ključne riječi ne mogu biti pronađene daje generičke odgovore temeljene na prijašnjoj komunikaciji. [1] Eliza je ujedno i prvi program koji je prošao test umjetne inteligencije pod nazivom Turingov test. U Turingovom testu čovjek s ulogom suca razgovara sa stvarnim čovjekom i Chatbotom bez da zna koji je od njih dvoje čovjek, a koje računalni program. Eliza je računalni program koji je prvi uspio natjerati neke suce u Turingovom testu da ne mogu sa sigurnošću reći da li razgovaraju s čovjekom ili Chatbotom.




*Slika 1: Trend pojavljivanja fraza napravljen u Google Books Ngram Viewer*[3]

Ono što Chatbota čini sličnijim čovjeku je dobro razvijena umjetna inteligencija ugrađena u njega. Što je ugrađena umjetna inteligencija bolja, to Chatbot ima veće opće znanje ili dublje specifično znanje, bolju sposobnost učenja, veći opseg riječi ili fraza koje može primiti i kvalitetno obraditi kao ulaz ili dati kao odgovor. S tako razvijenim Chatbotom je ugodnije i

[1] Računalni program namijenjen za razgovor s korisnicima. U nedostatku primjerenog hrvatskog naziva i zbog vrlo učestalog korištenja riječi Chatbot u svijetu tehnologije danas, u nastavku rada bit će korištena ta riječ engleskog podrijetla koja dolazi od riječi „Chat“ što znači čavrljanje i „Bot“ što je skraćeno od „Robot“, a označava računalni program koji se samostalno izvršava.
[2] https://books.google.hr/
[3] https://books.google.com/ngrams/

prirodnije razgovarati pa nije ni čudo da je danas, za vrijeme velike popularnosti Chatbotova, vrlo često pitanje kako u Chatbot ugraditi što bolju umjetnu inteligenciju. Osim ugrađene umjetne inteligencije razvoj softvera za razgovor može otići i korak dalje. Ako u Chatbot ugradimo osobine kao što su samostalno odlučivanje, davanje savjeta, donošenje odluka, pronalaženje podataka na Internetu ili u bazama podataka, takvog Chatbota možemo nazvati agentom za razgovor, a modeliranjem agenata za razgovor ćemo se baviti u ovom radu.

Ovaj rad možemo podijeliti u četiri glavna dijela. (1) Uvod – Kratki uvod u temu s definiranim općim i specifičnim ciljevima. (2) Agent za razgovor – U ovom dijelu rada ćemo proučiti što su agenti za razgovor, detaljnije opisati problem istraživanja i tehnologije koje će biti korištene za istraživanje. (3) Opis istraživanja – Pregled povezane literature i razvoja sustava B.A.R.I.C.A asistentica (engl. *Beautiful ARtificial Intelligence Chat Agent*), agenta za razgovor koji je namijenjen da studentima Fakulteta organizacije i informatike u Varaždinu (u nastavku FOI) pruži informacije o fakultetu, profesorima i rasporedu. (4) Rasprava – Rasprava o istraživanju. (5) Zaključak – Zaključak dobiven istraživanjem.

## 1.1. Opći i specifični ciljevi

Iako umjetna inteligencija teži stvaranju tehnologije što sličnije čovjeku, za sada ne postoji jedinstveni recept za izradu savršenog, multifunkcionalnog Chatbota ili agenta za razgovor već se oni izgrađuju i modeliraju ovisno o tome za što će se koristiti i tko će ih koristiti. Osim toga, izrada agenata za razgovor može ovisiti i o jeziku na kojem agent treba razgovarati. Procesuiranje prirodnog jezika (engl. Natural Language Processing – NLP) daleko se brže razvija za engleski jezik nego, primjerice, za hrvatski.

Jedan od izazova koji se može javiti prilikom izrade Chatbota ili agenta za razgovor je postavljanje korisnikovog unosa u kontekst razgovora. Ako Chatbot ne može prepoznati kontekst u kojem je korisnik nešto rekao, ne može voditi povezan i adekvatan razgovor. Moguće rješenje tog problema je korištenje matematičkog modela konačnog automata (engl. *Finite State Machine* – FSM) za vođenje razgovora. Cilj ovog rada je proučiti modeliranje agenata za razgovor uz pomoć FSM-a radi postavljanja korisnikovog unosa u kontekst razgovora na hrvatskom jeziku. Pokušat ćemo pronaći prednosti i nedostatke modeliranja agenata za razgovor na ovaj način i raspraviti o mogućnostima praktične primjene takvih agenata za razgovor.

# 2. Agent za razgovor

U ovom poglavlju definirat ćemo pojmove i opisati tehnologije važne za nastavak te detaljnije opisati zašto je prilikom izrade agenta za razgovor teško postaviti korisnikov unos u kontekst razgovora.

## 2.1. Opći pojmovi

Za daljnji rad potrebno je definirati neke opće pojmove, povezati međusobno njihova značenja i povezanost ih s radom.

### 2.1.1. Chatbot, umjetna inteligencija i strojno učenje

Kao što je rečeno i u uvodu, **Chatbot** je softver kojem je namjena razgovor s čovjekom bilo preko tipkovnice ili usmeno. Ideja razvoja softvera za razgovor koji daje privid ljudske konverzacije javila se 50-ih godina prošlog stoljeća, a danas se Chatbotovi koriste u različite svrhe kao što su korisnička podrška, osobni asistenti, osobni savjetnici i to na različitim područjima kao što su edukacija, medicina, ekonomija ili zabava. [2]

**Umjetna inteligencija** (engl. *Artificial Intelligence* - AI) je sposobnost tehnologije da poprimi neke ljudske osobine kao što su razumijevanje govora, prepoznavanje objekata na slikama ili video zapisima, mogućnost izražavanja govorom, mogućnost učenja, donošenja odluka i sl. U velikom broju slučaja cilj je da Chatbot poprimi osobine čovjeka kako bi s njim bilo što ugodnije razgovarati pa nije čudo da je pojam umjetne inteligencije usko povezan s razvojem Chatbotova.

Dio umjetne inteligencije koji se odnosi na sposobnost učenja je **strojno učenje** (engl. *Machine learning*). U Chatbotovima se strojno učenje temelji na podacima za treniranje (engl. Training dana). To su podaci o razgovoru iz kojih se matematičkim algoritmima odlučuje što Chatbot treba odgovoriti u određenom trenutku. Ti podaci se za vrijeme korištenja programa mogu nadopunjavati tako da softver gleda kako korisnik odgovara na njegove unose i sprema to u podatke za treniranje. Na ovaj način Chatbot nakon nekog vremena može znati adekvatno odgovoriti na nešto što mu u početku nije bilo eksplicitno definirano nego je to naučio od korisnika.

### 2.1.2. Agent, agent za razgovor

**Agent** je računalni sustav koji ima sposobnost samostalnog djelovanja. To znači da u skladu s definiranim ciljevima samostalno odlučuje kako će postići te ciljeve. [3] Primjer takvog agenta je računalni sustav koji samostalno na korisnikov upit pretražuje Internet ili neke druge baze podataka kako bi korisniku dostavio potrebne informacije ili čak obavio neku radnju na Internetu umjesto njega.

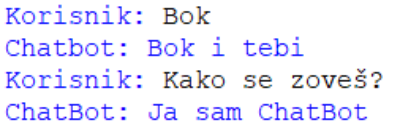
**Agent za razgovor** je Chatbot koji prilikom razgovora oponaša ljudske geste. Agent za razgovor može biti i utjelovljen što znači da je utjelovljen u neki oblik avatara ili humanoidnog robota. [4] Sustav B.A.R.I.C.A asistentica, o kojem će kasnije biti riječi, je agent za razgovor utjelovljen u humanoidnog robota koji na korisnikov upit ima sposobnost dohvaćanja informacija koje je korisnik tražio s Interneta ili vlastitih izvora unutar sustava.

## 2.2. Opis problema postavljanja korisnikovog unosa u kontekst razgovora

U programskom kodu 2.2.1. prikazan je vrlo jednostavan primjer programa za razgovor u svega nekoliko linije koda u Python programskom jeziku. U programu se čeka na korisnikov unos, a zatim mu se odgovara prema dva jednostavna pravila koja govore što treba odgovoriti na određeni korisnikov unos. Primjer korištenja na slici 2 pokazuje da se ovim programom može voditi vrlo kratka ali smislena konverzacija između programa i korisnika, no značajnijim povećanjem broja definiranih pravila kvaliteta razgovora između korisnika i računalnog programa mogla bi se također značajnije povećati. U nekim slučajevima možda bi se na ovaj način mogao izraditi i zadovoljavajući Chatbot.

### *2.2.1. Programski kod: Primjer naivnog programa za razgovor*

```
1.  pravila = {"Bok": "Chatbot: Bok i tebi",
2.             "Kako se zoveš?":
3.                 "ChatBot: Ja sam ChatBot"}
4.
5.  while True:
6.      ulaz = input("Korisnik: ")
7.      print(pravila[ulaz])
```




*Slika 2: Primjer izvođenja programskog koda pod brojem 2.2.1.*

Za ovaj primjer programa ne možemo reći sa sadrži umjetnu inteligenciju. Ne samo da program nema nikakvu sposobnost strojnog učenja, već ako bi korisnik u unosu pogriješio u samo jednom znaku, program ne bi znao što odgovoriti. Ovakav problem se u Python programskom jeziku može riješiti uz pomoć Python biblioteka za izgradnju Chatbotova koje najčešće pružaju jednostavniji način ugradnje strojnog učenja. Korištenjem takvih biblioteka mogu se kreirati ChatBotovi u kojima korisnikov unos može u određenoj mjeri odudarati od definiranih pravila, a uz to još mogu imati sposobnost učenja prilikom razgovora pa nakon nekog vremena mogu znati pravilno razgovarati o nečemu što mu prilikom izgradnje programa nije bilo eksplicitno definirano.

No, iako Chatbotovi na ovaj način imaju ugrađenu umjetnu inteligenciju ponekad ni ovo nije dovoljno da bi Chatbot mogao na zadovoljavajući način razgovarati s korisnikom što će biti slučaj i u sustavu B.A.R.I.C.A asistentica. Pretpostavimo da Chatbot korištenjem neke Python biblioteke umjetnu inteligenciju temelji na pravilima što odgovoriti korisniku na njegov unos slično pravilima iz programskog koda 2.2.1. Ta pravila, kako smo prije spomenuli, nazivaju se podaci za treniranje. Iako takvih podataka može biti puno i dalje se postavljaju pitanja kako navesti Chatbota da vodi smisleni povezani razgovor, da u jednom trenutku zna nešto što mu je korisnik već prije rekao ili da bude svjestan konteksta i tijeka njihovog razgovora. Koko navesti Chatbota da zna da u nekom trenutku korisnikov unos znači jedno, a u nekom drugo trenutku da isti takav korisnikov unos znači nešto sasvim drugo i treba mu odgovoriti na potpuno drugačiji način. Primjer takvog problema možemo vidjeti u programskom kodu 2.2.2. gdje program ne vidi razliku između drugog i trećeg korisnikovog unosa što možemo vidjeti na primjeru korištenja na slici 3.

### *2.2.2. Programski kod: Primjer problema stavljanja korisnikovog unosa u kontekst*

```
1.  pravila = {"Bok": "Chatbot: Bok i tebi",
2.             "Kako se zoveš?":
3.                 "ChatBot: Ja sam ChatBot\nŠto prvo pitaš nekoga koga si tek upoznao?"
    }
4.
5.  while True:
```

```
6.      ulaz = input("Korisnik: ")
7.      print(pravila[ulaz])
```

```
Korisnik: Bok
Chatbot: Bok i tebi
Korisnik: Kako se zoveš?
ChatBot: Ja sam ChatBot
Što prvo pitaš nekoga koga si tek upoznao?
Korisnik: Kako se zoveš?
ChatBot: Ja sam ChatBot
Što prvo pitaš nekoga koga si tek upoznao?
```

*Slika 3: Primjer izvođenja programskog koda 2.2.2.*

U ovom radu pozabavit ćemo se rješavanjem problema postavljanja korisnikovog unosa u kontekst razgovora tako da se u računalni program za razgovor ugradi FSM. Ugradnjom FSM-a Chatbot u svakom trenutku zna u kojem stanju razgovora se nalazi pa odgovara korisniku ovisno o tome. Pravila za davanje odgovorna na određeni korisnikov unos možemo dodatno grupirati pa, ovisno o tome u kojem se stanju nalazi Chatbot, koristiti pravilo iz određene grupe pravila.

## 2.3. Konačni automat

Konačni automat (engl. *Finite State Machine* - FSM) je diskretni matematički model koji se definira kao šestorka $(\Sigma, \Gamma, S, s_0, \delta, \omega)$ pri čemu je:

- $\Sigma$ – ulazna abeceda
- $\Gamma$ – izlazna abeceda
- S – konačni neprazni skup stanja
- $s_0$ – početno stanje
- $\delta: S \times \Sigma \rightarrow S$ – funkcija prijelaza
- $\omega: S \times \Sigma \rightarrow \Gamma$ – izlazna funkcija.

FSM se može koristi u različite svrhe kao što su automatizirano testiranje protokola [5], automatizirano upravljanje događajima [6], prepoznavanje i modeliranje ljudskih gesti [7] [8]. U ovom radu bit će pokazano kako se može koristiti za upravljanje i modeliranje razgovora u agentima za razgovor.

Možemo dati primjer FSM-a koji bi, ako upravlja komunikacijom, pomogao riješiti problem sa slike 3. Definirajmo prvo sve elemente FSM-a za taj primjer.

- $\Sigma = \{unosBok, unosKakoSeZoveš\}$ – skup mogućih korisnikovih unosa

*Tablica 1: Opis ulaza za primjer FSM-a*

| Ulaz | Opis ulaza |
|---|---|
| $unosBok$ | Korisnik je unio riječ „Bok“. |
| $unosKakoSeZoveš$ | Korisnik je unio pitanje „Kako se zoveš?“. |

- $\Gamma = \{\text{BokITebi}, \text{ImeIPitaj}, \text{DobroPitanje}\}$ – skup mogućih agentovih odgovora

*Tablica 2: Opis izlaza za primjer FSM-a*

| Odgovor | Opis odgovora |
|---|---|
| $BokITebi$ | Agent odgovara „Bok i tebi“. |
| $ImeIPitaj$ | Agent odgovara „Ja sam Chatbot. Što prvo pitaš nekoga koga si tek upoznao?“. |
| $DobroPitanje$ | Agent odgovara „Dobro pitanje“.[4] |

- $S = \{\text{Slušam}, \text{Pozdrav}, \text{KakoSeZoveš}, \text{DobroPitanje}\}$ - skup stanja u kojima se agent može nalaziti
- $s_0 = Slušam$ – početno stanje u kojem se nalazi agent

Funkciju prijelaza i izlaznu funkciju možemo definirati uz pomoć tablica ili dijagramom stanja. Mi ćemo koristiti UML (engl. *Unified Modeling Language*) dijagram stanja koji se nalazi na slici 4. Na njemu vidimo pravila po kojima se prelazi iz stanja u stanje ovisno o korisnikovom unosu te što agent radi prilikom ulaska u neko stanje. Možemo vidjeti da će agent, bez obzira na isti korisnikov unos, izvesti dvije različite akcije. Na ovaj način FSM upravlja komunikacijom tako da agent zna u kojem stanju se nalazi, što treba očekivati kao unos, što unos znači, i što treba napraviti u određenom stanju na određeni korisnikov unos. Odnosno, korisnikov unos stavlja u **kontekst** razgovora što je i cilj ovog rada.

[4] Ovaj odgovor nije moguće dati u naivnom scenariju iz programskog koda 2.2.2. jer je tamo nije moguće definirati dva pravila za isti korisnikov unos.


*Slika 4: UML dijagram stanja za primjer FSM-a*

## 2.4. Opis tehnologije

Sustav B.A.R.I.C.A asistentica napravljen je u Python [9] i JavaScript [10] programskim jezicima. U ovom poglavlju opisat ćemo korištene Python biblioteke otvorenog koda (engl. *Open source*) koje značajno pojednostavljuju specifične izazove prilikom izrade programa i ostale tehnologije koje su korištene u sustavu B.A.R.I.C.A asistentica, a važne su za razumijevanje istraživanja.

### 2.4.1. Python biblioteka ChatterBot

Najprije ćemo opisati Python biblioteku ChatterBot [11] koja je korištena za automatizirano davanje odgovora korisnicima. Ova biblioteka se temelji na tzv. podacima za treniranje. Ti podaci, slično pravilima iz naivnih primjera u programskim kodovima 2.2.1. i 2.2.2., su pravila koja definiraju što program treba dati kao odgovor na korisnikov unos. Programski kod 2.4.1.1. prikazuje primjer korištenja biblioteke ChatterBot s podacima za treniranje od linije 8. do 16. U primjeru korištenja tog programskog koda na slici 5 možemo vidjeti da korisnikov unos više ne mora biti u potpunosti istovjetan onome u podacima za treniranje (u podacima za treniranje stoji „Kako se zoveš?“, a u primjeru korištenja „Kako se zoves“). Prilikom pokretanja ovog programa Chatbot se trenira tako što se svi podaci za treniranje spremaju u bazu podataka, a kasnije se naredbom kao u 20. liniji koda dohvaća odgovor iz baze podataka koji najbolje odgovara korisnikovom unosu.

#### *2.4.1.1. Programski kod: Primjer korištenja biblioteke ChatterBot*

```
1.  from chatterbot import ChatBot
2.
3.  chatbot = ChatBot(
4.      'Chatbot',
5.      trainer='chatterbot.trainers.ListTrainer'
6.  )
7.
8.  chatbot.train([
9.      "Bok",
```

```
    "Bok i tebi"
])

chatbot.train([
    "Kako se zoveš?",
    "Ja sam ChatBot"
])

while True:
    ulaz = input("Korisnik: ")
    print(chatbot.get_response(ulaz)
```

```
Korisnik: Bok
Bok i tebi
Korisnik: Kako se zoves
Ja sam ChatBot
```

*Slika 5: Primjer izvođenja programskog koda 2.4.1.1.*

Za dohvaćanje odgovora iz baze podataka biblioteka ChatterBot koristi strojno učenje. Koju metodu strojnog učenja koristi, ovisi o postavkama definiranog Chatbota u programu jer je moguće u programu dodatno definirati logički adapter koji opisuje logiku odabira odgovora. U dokumentaciji je navedeno da, između ostalog, koristi (1) algoritme traženja i (2) klasifikacije.

(1) **Algoritmi traženja** – Traženje je jedan od najvažnijih dijelova biblioteke kako bi Chatbot bio u mogućnosti brzo dati zadovoljavajući odgovor. Na rezultat traženja mogu utjecati atributi kao što su sličnost korisnikovog unosa i pojedinog podatka iz baze podataka, učestalost korištenja pojedinog podatka iz baze podataka i vjerojatnost da korisnikov unos odgovara pojedinom kategorijama iz baza podataka.

(2) **Algoritmi klasifikacije** – Neki logički adapteri u biblioteci ChatterBot koriste **naivan Bayesov klasifikator** za utvrđivanje zadovoljava li korisnikov unos skup kriterija za određeni logički adapter prema Bayesovoj formuli vjerojatnosti koja glasi:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

$X$ je n-dimenzionalni vektor $X = (x_1, x_2, x_3, \dots, x_n)$, $P(X)$ označava vjerojatnost pojave događaja $X$. $C$ je skup klasa $C = \{c_1, c_2, c_2, \dots, c_m\}$, $P(C_i)$ označava vjerojatnost pojave događaja $C_i$. $P(C_i|X)$ označava uvjetnu vjerojatnost odnosno vjerojatnost pojave događaja $C_i$ uz uvjet da se dogodilo $X$, dok $P(X|C_i)$ označava uvjetnu vjerojatnost odnosno vjerojatnost pojave događaja $X$ uz uvjet da se dogodilo $C_i$. Bayesov klasifikator dodjeljuje svaki $X$ nekoj od

klasa $C_i$ i to tako da će $X$ biti dodijeljen klasi $C_i$ ako i samo ako $P(C_i|X) > P(C_j|X)$ tako da je $1 \leq j \leq m$. [12]

Biblioteka ChatterBot u sustavu B.A.R.I.C.A asistentica će nam pomoći kod upravljanja s podacima za treniranje. Podaci za treniranje bit će smješteni u različite baze podataka tako da ni u jednoj bazi podataka nema spremljenog korisnikovog unosa za koji može biti više izlaza. Bit će napravljena po jedna instanca objekata ChatterBot za svaku od baza podataka, a koristit će se neka od njih ovisno o tome u kojem stanju FSM-a se sustav trenutno nalazi.

### 2.4.2. Python biblioteka Transitions

Python biblioteka Transitions [13] omogućava jednostavniju implementaciju FSM-a u Python programskom jeziku. Korištenje ove biblioteke može nam pomoći pri stavljanju korisnikovog unosa u kontekst razgovora kao što je primjer u programskom kodu 2.4.2.1. Kako je i objašnjeno u prethodnom poglavlju, napravljene su dvije Chatterbot instance (od linije 3. do linije 13.) koje koriste dvije različite baze podataka. Istrenirana su oba modela strojnog učenja (od linije 15. do linije 26.). Nadalje je definirana klasa FSM-a i kreirana instanca te klase uz pomoć Transitions biblioteke prema UML dijagramu sa slike 4 (od linije 30. do linije 44.). U ostatku programskog koda se upravlja komunikacijom na način da se korisnikov unos obrađuje s obzirom na stanje FSM-a.

#### *2.4.2.1. Programski kod: Primjer stavljanja unosa korisnika u kontekst uz pomoć biblioteke Chatterbot*

```
1.  from chatterbot import ChatBot
2.
3.  chatbot1 = ChatBot(
4.      'Chatbot1',
5.      trainer='chatterbot.trainers.ListTrainer',
6.      database='db1.sqlite3'
7.  )
8.
9.  chatbot2 = ChatBot(
10.     'Chatbot2',
11.     trainer='chatterbot.trainers.ListTrainer',
12.     database='db2.sqlite3'
13. )
14.
15. chatbot1.train([
16.     "Bok",
17.     "Bok i tebi"
18. ])
19. chatbot1.train([
20.     "Kako se zoveš?",
```

```
    "Ja sam ChatBot. Što prvo pitaš nekoga koga si tek upoznao?"
])
chatbot2.train([
    "Kako se zoveš?",
    "Dobro pitanje"
])

from transitions import Machine

class FiniteStateMachine:

    states = ['Slusam', 'Pozdrav', 'KakoSeZoves', 'DobroPitanje']

    def __init__(self):
        self.machine = Machine(model=self, states=FiniteStateMachine.states,
                               initial='Slusam')
        self.machine.add_transition('unosBok', 'Slusam', 'Pozdrav')
        self.machine.add_transition('unosKakoseZoves1', 'Pozdrav', 'KakoSeZoves')
        self.machine.add_transition('unosKakoseZoves2', 'KakoSeZoves', 'DobroPitanje
', after='exit')

    def exit(self):
        sys.exit()

m = FiniteStateMachine()

import sys

while True:
    ulaz = input("Korisnik: ")
    if m.state == 'Slusam':
        print(chatbot1.get_response(ulaz))
        m.unosBok()
    elif m.state == 'Pozdrav':
        print(chatbot1.get_response(ulaz))
        m.unosKakoseZoves1()
    elif m.state == 'KakoSeZoves':
        print(chatbot2.get_response(ulaz))
        m.unosKakoseZoves2()
```

```
Korisnik: Bok
Bok i tebi
Korisnik: Kako se zoveš?
Ja sam ChatBot. Što prvo pitaš nekoga koga si tek upoznao?
Korisnik: Kako se zoveš?
Dobro pitanje
```

*Slika 6: Primjer izvođenja programskog koda 2.4.2.1.*

Na ovaj način omogućili smo da program u svakom trenutku zna o čemu razgovara s korisnikom i kakav korisnikov unos treba očekivati. Ovakav pristup postavljanja razumijevanja korisnikovog unosa korišten je i u sustavu B.A.R.I.C.A asistentica u kojem su prednosti tog pristupa još više došle do izražaja zbog složenije strukture razgovora.

### 2.4.3. Ostale tehnologije

Nabrojat ćemo i kratko objasniti i ostale važne tehnologije korištene u sustavu B.A.R.I.C.A asistentica za koje je sada možemo napomenuti čemu služe, a kasnije će biti opisano za što su korištene u sustavu.

- Hovercraft [14] - Python skripta za izradu HTML prezentacija. Na temelju RST datoteke[5] u koju je moguće uključiti i raznovrsne druge datoteke kao što su slike ili video zapisi koji će se kasnije vidjeti u HTML prezentaciji, ali i datoteke za upravljanje HTML-om kao što su JavaScript i CCS datoteke.
- Simple Websocket Server [15] – Python biblioteka za upravljanje porukama preko WebSocketa.
- Selenium [16] – Python biblioteka za automatizirano pretraživanje Interneta preko Internet preglednika.
- JQuery [17] – JavaScript biblioteka za upravljanje i manipuliranje HTML dokumentom.
- CrazyTalk [18] – Softver za animiranje avatara koji govore na temelju uvezene slike i audio zapisa.
- Voice notebook [19] – Aplikacija za pretvaranje govora u tekst. Podržava više jezika od kojih je jedan i hrvatski.

[5] Kratica od (engl.) reStructuredText datoteka, a odnosi se na format datoteke s jednostavnim i intuitivnim tekstom koji ukazuje na strukturu. [20]

# 3. Opis istraživanja

Opis istraživanja sastoji se od pregleda povezane literature i opisa sustava B.A.R.I.C.A asistentica usmjerenog na problem i ciljeve istraživanja.

## 3.1. Pregled povezane literature

Korištenje FSM-a prilikom izgradnje softvera za razgovor već je prepoznato u nekoliko istraživanja. Tako su Sangroya, Anantaram, Saini i Rawat [21] koristili FSM za vođenje komunikacije između softvera za razgovor i kupaca automobila prilikom iznošenja prigovora. Prepoznali su problem nerelevantnog razgovora od strane softvera za razgovor s nezadovoljnim kupcima pa je njihov cilj bio je iz kupčevog prigovora prepoznati stvaran problem i kupčevo emotivno stanje te na temelju toga korisno, relevantno i u pravom kontekstu nastaviti komunikaciju. FSM su koristili tako da su stanja FSM-a predstavljala teme razgovora koje je u nekom trenutku trebalo ili preskočiti ili ispitati ovisno o tome je li kupac to već rekao i je li to uopće relevantno za razgovor. Rezultati njihovog istraživanja pokazali su da su oni u 68.47% slučajeva uspjeli prepoznati kupčev problem i voditi komunikaciju kako je kupac to i očekivao.

Klopfenstein, Delpriori i Ricci [22] u svojem istraživanju pokušavaju pronaći rješenje problem nemogućnosti izrade agenata za razgovor i njihovu integraciju na Internet platforme bez naprednih vještina programiranja. Njihovo rješenje temelji se na sustavu Bottery. Sustav Bottery služi za generiranje modela kontekstualne konverzacije uz pomoć FSM-a. Agent se u tom sustavu može modelirati definiranjem stanja u kojima može biti tijekom konverzacije i pravilima prelaska iz stanja u stanje. Prilikom ulaska u neko stanje može izvršiti određenu akciju čime se dobiva mogućnost konverzacije. U ovom istraživanju pokazalo se da je sustav Bottery, koji koristi FSM za vođenje razgovora, pogodan za izradu agenata za strukturirani razgovor.

Yi i Jung [23] se u svojem istraživanju bave izradom agenta za razgovor koji se temelji na kombinaciji FSM-a, dohvaćanja informacija iz različitih izvora i razgovora na inicijativu agenta tako da on može bolje upravljati komunikacijom. Njihov agent za razgovor nije specijaliziran već služi za razgovor o popularnim temama kao što su sport, filmovi, glazba, znanost i sl. FSM se sastoji od stanja i tranzicija uz pomoć kojih se upravlja razgovorom.

Meštrović i sur. [24] razvili su sustav za razgovor na hrvatskom jeziku. Oni također, kao glavne probleme, spominju problem modeliranja kontinuiranog, povezanog razgovora i razvoj na

hrvatskom jeziku za kojeg tehnologije u NLP-u nisu dosta razvijene. Navode da je razlog nerazvijenosti NLP tehnologija za hrvatski jezik taj što hrvatskim jezikom govori mali broj ljudi, nije široko rasprostranjen i ima specifičnu vrlo kompliciranu strukturu. Njihov sustav služi za davanje informacija o vremenu na različitim područjima Republike Hrvatske. U njemu se za razgovor, za razliku od sustava za razgovor u ovom radu, za upravljanje komunikacijom koriste gramatike. Korištenje gramatika podrazumijeva znanje spremljeno u okvire razgovora kao što su npr. pitanje/odgovor ili pozdrav/pozdrav kojima se komunikacija raščlanjuje na dijelove i tako dobiva kontekst razgovora.

Vidimo da neki sustavi iz navedenih radova, kao i sustav B.A.R.I.C.A asistentica, upravljaju komunikacijom uz pomoć FSM-a. Za razliku od ostalih sustava, sustav B.A.R.I.C.A korištenje FSM-a kombinira s više instanca modela strojnog učenja i to tako da se ovisno o stanju FSM-a koristi određena instanca. Osim toga, dodatna vrijednost sustava B.A.R.I.C.A asistentica leži u tome što se komunikacija odvija na hrvatskom jeziku kojem je svakako potreban napredak u modeliranju usmene komunikacije između korisnika i računala te razvoju softvera za razgovor.

## 3.2. Opis sustava B.A.R.I.C.A asistentica

Aplikacijski softver B.A.R.I.C.A asistentica[6] je razvijen u Laboratoriju za umjetnu inteligenciju na Fakultetu organizacije i informatike u Varaždinu. B.A.R.I.C.A asistentica je agent za razgovor koji koristi umjetnu inteligenciju, a specijaliziran je za razgovor o FOI-u tako da svojim korisnicima pruža informacije kao što su općenite informacije o fakultetu, informacije o nastavnicima, dvoranama i raspored.

[6] Dostupno u GitHub repozitoriju: https://github.com/tajsokec/barica


*Slika 7: Naslovna stranica sučelja sustava B.A.R.I.C.A asistentica*

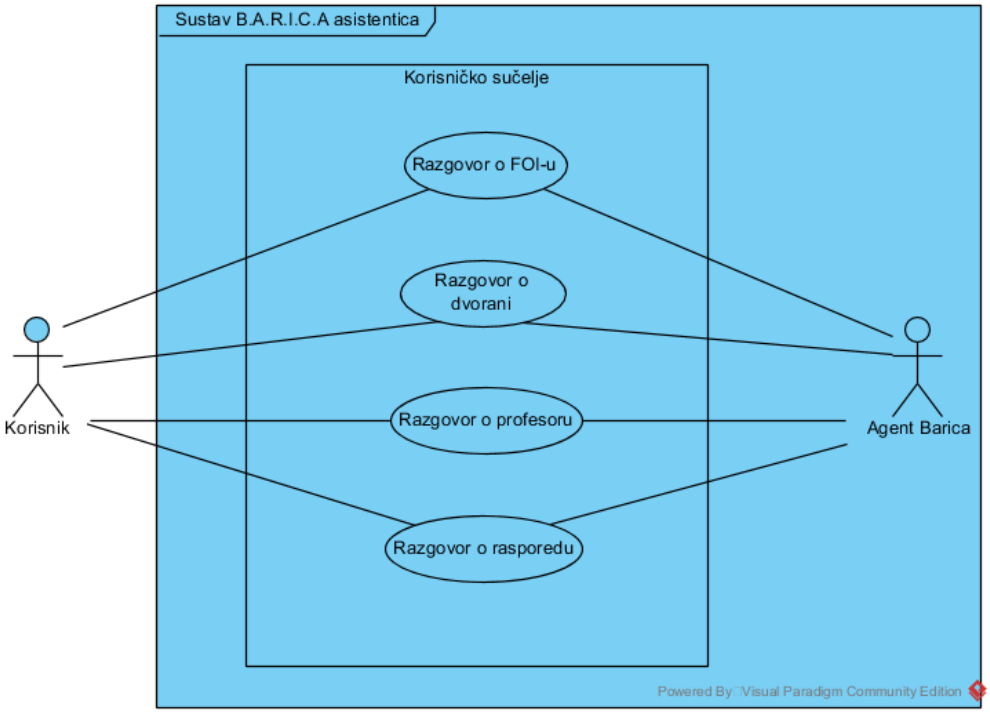
Sustav je nastao prema primjeru drugog sustava[7] napravljenog u Laboratoriju za umjetnu inteligenciju u Varaždinu. Taj sustav služio kao prezentacija na temu umjetne inteligencije s humanoidnom vizualizacijom robota koji na poziv govori i sudjeluje u prezentaciji. Iz tog sustava su preuzete ideje za korištenje nekih od tehnologija kao što su CrazyTalk, Hovercraft, Voice Notebook, JQuery, Python biblioteka Simple Websocket Server i ChatterBot. Također je preuzet dio unutarnje logike Python i JavaScript programskog koda koji se odnosi na upravljanje Websocket porukama. Ono što je značajnije nadograđeno je modeliranje komunikacije (dodavanje više instanci modela strojnog učenja i upravljanje komunikacijom uz pomoć FSM-a) te sposobnost Chatbota da samostalno pronalazi informacije za korisnika na Internetu. I naravno, sve je prilagođeno promijeni teme razgovora s umjetne inteligencije na temu dohvaćanja informacija o studiranju na FOI-u u korist studenata.

[7] Dostupno u GitHub repozitoriju: https://github.com/AILab-FOI/B.A.R.I.C.A./tree/master/BARICA-asistentica

### 3.2.1. Slučajevi korištenja

Korisnički zahtjevi sustava B.A.R.I.C.A asistentica bili su definirani prije početka izgradnje sustava, a prema njima je trebalo napraviti agenta koji će preko korisničkog sustava razgovarati s korisnicima koji će u prvom redu biti studenti FOI-a. Tim korisnicima trebalo je pružiti korisne informacije koje bi oni mogli tražiti. Tako je odlučeno da će, za početak, korisne informacije koje će agent moći davati svojim korisnicima biti općenite informacije o FOI-u, informacije o dvoranama, profesorima i raspored nastave. Prema tim korisničkim zahtjevima modeliran je FSM koji upravlja komunikacijom, a bit će od velike važnosti u nastavku rada, pa možemo ukratko skicirati i opisati korisničke zahtjeve.

Zahtjeve računalnih sustava možemo skicirati pomoću UML dijagrama slučajeva korištenja (engl. *Use Case Diagram*). Na slici 6 je prikazan dijagram slučajeva korištenja na kojem vidimo da korisničko sučelje sustava B.A.R.I.C.A asistentica omogućava razgovor između sudionika (engl. *Actors*) „Korisnik“ i „Agent Barica“ i to na četiri teme od kojih je svaka predstavljena jednim slučajem korištenja (engl. *Use Case*), a „Korisničko sučelje“ i „Agent Barica“ su u vlasništvu paketa (engl. *Package*) „Sustav B.A.R.I.C.A asistentica“.




*Slika 8: UML dijagram slučajeva korištenja za sustav B.A.R.I.C.A asistentica*

Dijagram slučajeva korištenja za sustav B.A.R.I.C.A asistentica mogao bi izgledati i drugačije. Agent za razgovor „Agent Barica“ mogao bi umjesto sudionika biti dio sustava koji sadrži svoje slučajeve korištenja jer je agent za razgovor ustvari softver i dio sustava koji se opisuje dijagramom slučajeva korištenja. Međutim, definiranjem agenta „Agent Barica“ kao sudionika u dijagramu slučajeva korištenja postigli smo dvije stvari. Prva je ta da je dodatno naglašena umjetna inteligencija agenta time što je on prikazan kao korisnikov sugovornik, a druga je ta da smo slučajeve korištenja definirali tako da se izravno prema njima može modelirati komunikacija uz pomoć FSM-a o čemu će riječi biti kasnije. U tablici 3 nalazi se kratki opis svakog od slučaja korištenja.

*Tablica 3: Popis slučaja korištenja za sustav B.A.R.I.C.A asistentica*

| Naziv | Opis |
|---|---|
| Razgovor o FOI-u | Korisnik od agenta može zatražiti da mu reče nešto o FOI-u na što će mu agent izrecitirati kratki govor o FOI-u. |
| Razgovor o dvorani | Ako korisnik želi pronaći neku dvoranu na FOI-u, može agenta pitati gdje je. |
| Razgovor o profesoru | Ako korisnik za nekog od profesora na fakultetu želi znati kad ima konzultacije, gdje mu se nalazi kabinet i sl., može to pitati agenta. |
| Razgovor o rasporedu | Korisnik može od agenta zatražiti raspored na kojem može vidjeti i gdje se pojedino predavanje održava. |

### 3.2.2. Arhitektura sustava

Na prikazu arhitekture sustava B.A.R.I.C.A asistentica na slici 9 sustav je podijeljen u četiri ključna dijela. Dio sustava „Python programska podrška“ upravlja radom sustava te sadrži agenta za razgovor. Razgovorom upravlja FSM izgrađen pomoću Python biblioteke Transitions opisan u sljedećem poglavlju.

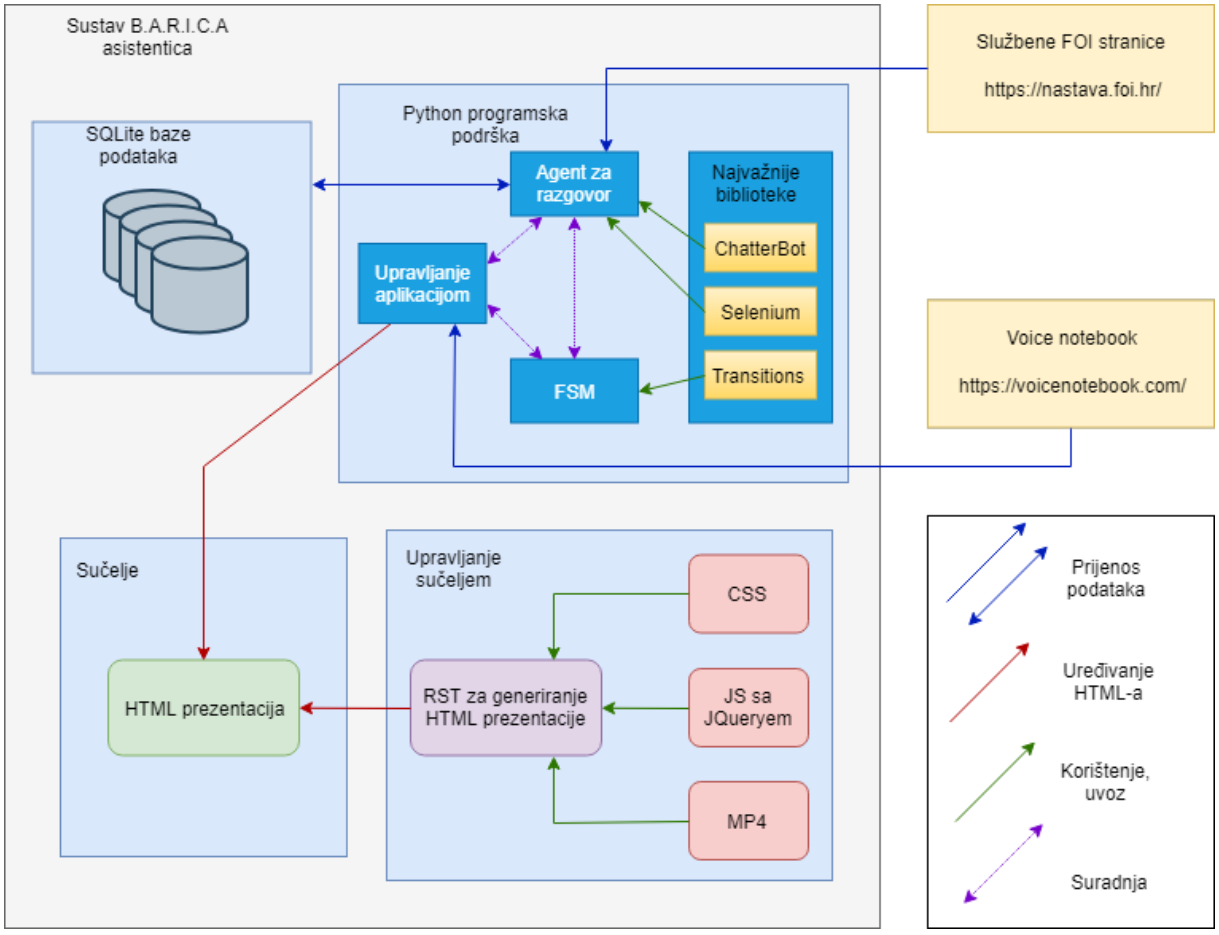
Agentovo znanje potrebno za razgovor je pohranjeno u „SQLite bazama podataka“, a trenutna verzija ima četiri SQLite baze podataka, po jednu za svaki od instanci modela strojnog učenja. Instance modela strojnog učenja su instance Python biblioteke ChatterBot i usko su povezane sa stanjima FSM-a pa će također biti opisane u idućem poglavlju.

Osim u bazama podataka, agent neke informacije koje traži korisnik nalazi i na Internetu. Tako na primjer, informacije o profesorima na FOI-u i o studentskim rasporedima uvozi sa „Službene FOI stranice“ uz pomoć Python biblioteke Selenium.

Za prevođenje ljudskog govora u računalu prepoznatljiv oblik koristi se aplikacija „Voice notebook“ koja osluškuje okruženje u kojem je pokrenuta, a potom ljudski govor pretvara u tekst. Dobiveni tekst sprema u međuspremnik računala pa nam je uz pomoć opcije „Zalijepi“ (engl. Paste) dostupna u Python programu. U Python djelu programa se ciklično u malom vremenskom periodu provjerava da li se u međuspremniku nalazi nešto novo što je potrebno obraditi kao korisnikovu naredbu.

Korisničko sučelje sustava čini HTML prezentacija koja se prilikom pokretanja aplikacije generira iz RST datoteke uz pomoć alata za izradu HTML prezentacija Hovercraft, a za vrijeme rada aplikacije Python dio sustava uređuje prezentaciju u stvarnom vremenu prema potrebi. To radi tako da se preko WebSocketa šalje strukturirane poruke Web stranici na kojoj se nalazi HTML prezentacija. Te strukturirane poruke obrađuju se pomoću JavaScripta ugrađenog u HTML koji koristi JQuery kako bi na HTML-u napravio potrebne preinake u realnom vremenu.

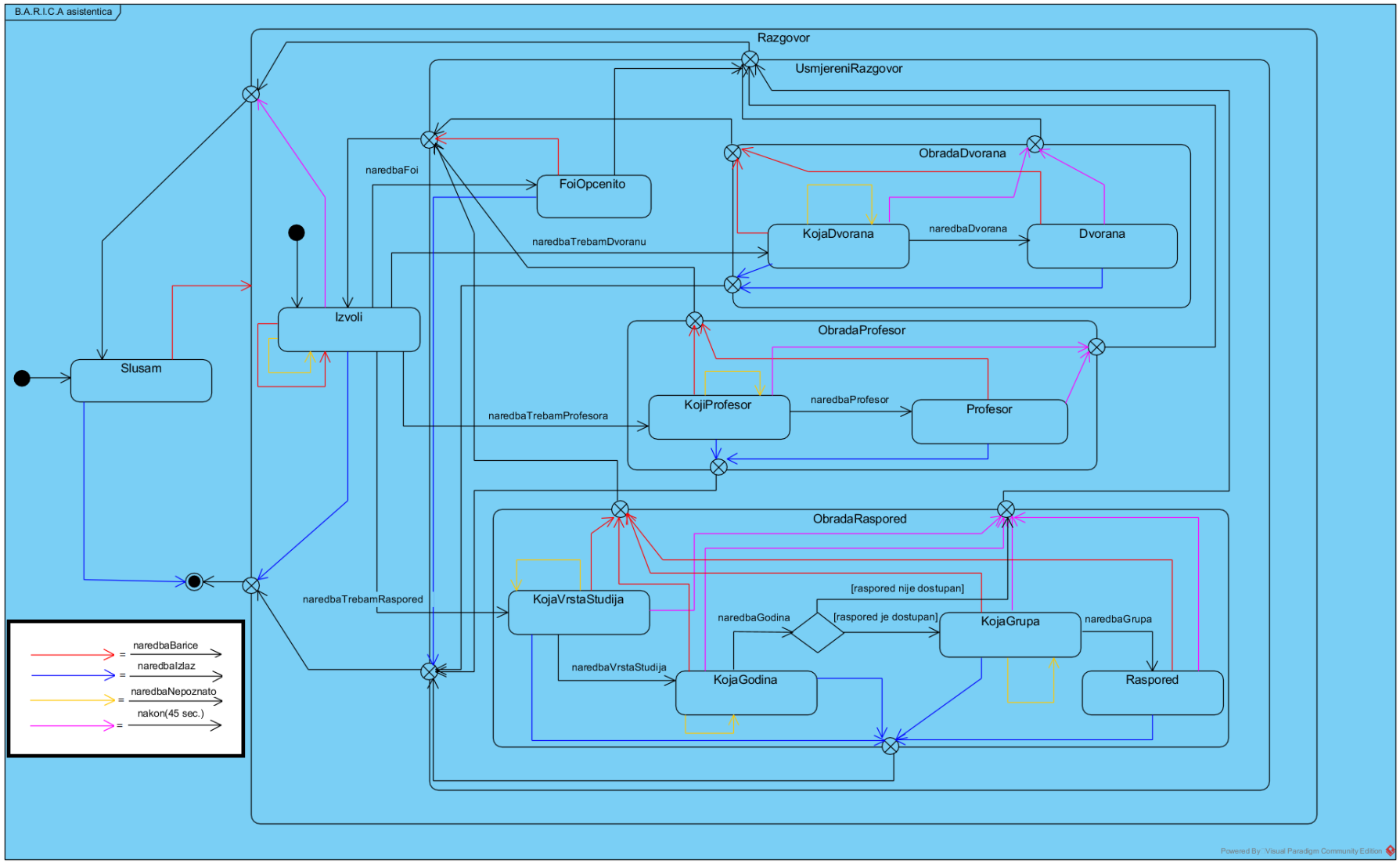
U RST datoteku je uključena MP4 datoteka s video zapisom koji sadrži vizualizaciju agenta za razgovor, odnosno video prikaz robota koji govori sve što agent za razgovor zna reći. Taj video prikaz generiran je u alatu CrazyTalk iz uvezene slike humanoidnog robota i audio zapisa sa svime što agent zna reći. Za vrijeme rada aplikacije JavaScript prema potrebi upravlja video zapisom tako da pokreće onaj dio video zapisa u kojem taj robot izgovara ono što korisnik u tom trenutku treba čuti ili se ponaša na onaj način na koji se u tom trenutku treba ponašati što daje privid konverzacije korisnika s agentom za razgovor.




*Slika 9: Prikaz arhitekture sustava B.A.R.I.C.A asistentica*

### 3.2.3. Konačni automat u sustavu

Sustav B.A.R.I.C.A asistentica koristi FSM koji upravlja komunikacijom. Za prikaz modela FSM-a korištenog u sustavu B.A.R.I.C.A asistentica koristit ćemo UML dijagrama stanja (engl. *State Machines Diagram*) koji možemo vidjeti na slici 10. On opisuje ponašanje objekta koji u programu predstavlja FSM. Na dijagramu možemo vidjeti u kojim diskretnim stanjima FSM može biti, koje mu je početno stanje, kako završava, kako prelazi iz stanja u stanje te koje akcije sustav izvodi u pojedinom stanju.




*Slika 10: UML dijagram stanja za sustav B.A.R.I.C.A asistentica*

FSM korišten u sustavu B.A.R.I.C.A asistentica, osim početnog pseudo-stanja i konačnog stanja koje ima svaki FSM, ima jedanaest diskretnih stanja koja upravljaju komunikacijom. Također ima četiri pomoćna složena stanja (engl. *Composite State*) koja služe za grupiranje stanja. U našem slučaju, složena stanja korištena su zbog smanjenja kompleksnosti dijagrama. Npr., da je dijagram napravljen bez hijerarhijskog prikaza sa složenim stanjima, u završno stanje bilo bi usmjereno jedanaest tranzicija, po jedna iz svakog diskretnog stanja. Popis i opis stanja nalazi se u tablici 4.

*Tablica 4: Popis stanja FSM-a*

| Naziv stanja | Opis |
|---|---|
| Stanja čekanja eksplicitne komande | |
| Slusam | Stanje u kojem agent osluškuje i čeka da ga korisnik pozove po imenu (Barice) što znači da korisnik želi razgovarati. Jedni valjani ulaz u ovom stanju je pozivanje agenta po imenu. |
| Izvoli | Stanje u kojem agent korisnika pita „Izvoli?“, a nakon toga čeka da mu korisnik reče o čemu želi razgovarati. Korisnik, prema četiri slučaja korištenja, može zatražiti da mu agent reče nešto o FOI-u, informacije o dvoranama na fakultetu, informacije o profesorima na fakultetu ili raspored. |
| KojaDvorana | Stanje u kojem agent korisnika pita koju dvoranu treba naći, a nakon toga čeka da mu korisnik odgovori. Korisnik može izgovoriti naziv neke od dvorana na FOI-u. |
| KojiProfesor | Stanje u kojem agent korisnika pita za kojeg profesora treba informacije, a nakon toga čeka da mu korisnik odgovori. Korisnik može izgovoriti ime nekog od profesora na FOI-u. |
| KojaVrstaStudija | Stanje u kojem agent korisnika pita za koju vrstu studija treba raspored, a nakon toga čeka da mu korisnik odgovori. Korisnik mora izgovoriti naziv neke od vrsti studija na FOI-u. |
| KojaGodina | Stanje u kojem agent korisnika pita za koju godinu studija treba raspored, a nakon toga čeka da mu korisnik odgovori. Korisnik mora izgovoriti godinu studija. |
| KojaGrupa | Stanje u kojem agent korisnika pita za koju grupu treba raspored, a nakon toga čeka da mu korisnik odgovori. Korisnik mora izgovoriti naziv grupe za koju želi raspored. |
| Stanje davanja konačne informacije | |

| | |
|---|---|
| FoiOpcenito | Stanje u kojem agent recitira kratki govor o FOI-u. Nakon završetka recitacije agent prelazi u stanje „Slusam“, a za vrijeme te recitacije korisnik može dati neku naredbu agentu što će prekinuti recitaciju i pokrenuti akciju ovisno o naredbi. |
| Dvorana | Stanje u kojem agent izgovara lokaciju dvorane za koju je korisnik to zatražio. Nakon završetka davanja informacije agent prelazi u stanje „Slusam“, a za vrijeme dok agent daje tu informaciju korisnik može dati neku naredbu agentu što će prekinuti davanje informacije i pokrenuti akciju ovisno o naredbi. |
| Profesor | Stanje u kojem agent korisniku kaže „Izvoli podatke o profesoru“ te prikaže informacije o profesoru za kojega je korisnik tražio informacije. Nakon što agent izgovori „Izvoli podatke o profesoru“ prelazi u stanje „Slusam“, a za vrijeme dok to izgovara korisnik može dati neku naredbu agentu što će prekinuti davanje informacije i pokrenuti akciju ovisno o naredbi. |
| Raspored | Stanje u kojem agent korisniku kaže „Izvoli raspored“ i prikaže raspored koji je korisnik tražio. Nakon što agent izgovori „Izvoli raspored“ prelazi u stanje „Slusam“, a za vrijeme dok to izgovara korisnik može dati neku naredbu agentu što će prekinuti davanje informacije i pokrenuti akciju ovisno o naredbi. |
| Složena stanja | |
| Razgovor | Složeno stanje u kojem agent i korisnik razgovaraju. Izvan tog stanja agent samo osluškuje i čeka na razgovor. |
| UsmjereniRazgovor | Složeno stanje u kojem se razgovor usmjerava prema pojedinom slučaju korištenja. |
| ObradaDvorana | Složeno stanje u kojem se razgovara o nekoj od dvorana. |
| ObradaProfesor | Složeno stanje u kojem se razgovara o nekom profesoru. |
| ObradaRaspored | Složeno stanje u kojem se razgovara o rasporedu. |

Spomenuli smo da su uz FSM usko povezane instance modela strojnog učenja jer se za davanje odgovora korisniku koristi instanca ovisno o stanju FSM-a. Možemo te instance modela strojnog učenja usporediti s temama razgovora. Ovisno o temi razgovora koristimo instancu strojnog učenja koja ja specijalizirana za tu temu razgovora. U bazi podataka za tu instancu modela strojnog učenja (svaka instanca modela strojnog učenja ima svoju bazu podataka s podacima za treniranje) se nalaze podaci za razgovor o toj temi. Tako neće biti zabune ako za neki korisnikov unos može biti više mogućih odgovora. Npr. prilikom izgovaranja dvorane koju

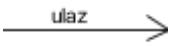
korisnik želi pronaći i prilikom izgovaranja godine za koju korisnik treba raspored valjan korisnikov unos je riječ „jedan“. Ako se prilikom ulaska korisnikove naredbe koja glasi „jedan“ FSM nalazi u stanju „KojaDvorana“ program će koristiti instancu strojnog učenja za razgovor o dvoranama i odgovoriti korisniku gdje se nalazi dvorana 1 na FOI-u, a ako se nalazi u stanju „KojaGodina“ koristit će instancu strojnog učenja za razgovor o rasporedu i pitati ga dalje za grupu. U trenutnoj verziji sustava napravljene su četiri instance modela strojnog učenja koje su opisane u tablici 5.

*Tablica 5: Opis instanci modela strojnog učenja*

| Opis instance modela strojnog učenja | Stanja FSM-a u kojima se koristi |
|---|---|
| Glavna instanca - koristi se za pozivanje agenta Barice, za određivanje smjera razgovora i razgovor o FOI-u općenito | • Slusam<br>• Izvoli<br>• FoiOpcenito |
| Instanca za razgovor o dvoranama | • KojaDvorana<br>• Dvorana |
| Instanca ta razgovor o profesorima | • KojiProfesor<br>• Profesor |
| Instanca za razgovor o rasporedu | • KojaVrstaStudija<br>• KojaGodina<br>• KojaGrupa<br>• Raspored |

Nadalje o dijagramu stanja, zbog povećanja preglednosti, tranzicije u dijagramu su podijeljene u nekoliko grupa. Zato na većini tranzicija ne mora ništa pisati jer se dodatno pojašnjenje nalazi na legendi. Tranzicije su u dijagramu stanja uvijek jednosmjerne te potječu od izvorišnog do odredišnog stanja s time da izvorišno i odredišno stanje može biti i isto stanje. Popis i opis tranzicija nalazi se u tablici 6.

*Tablica 6: Popis tranzicija FSM-a*

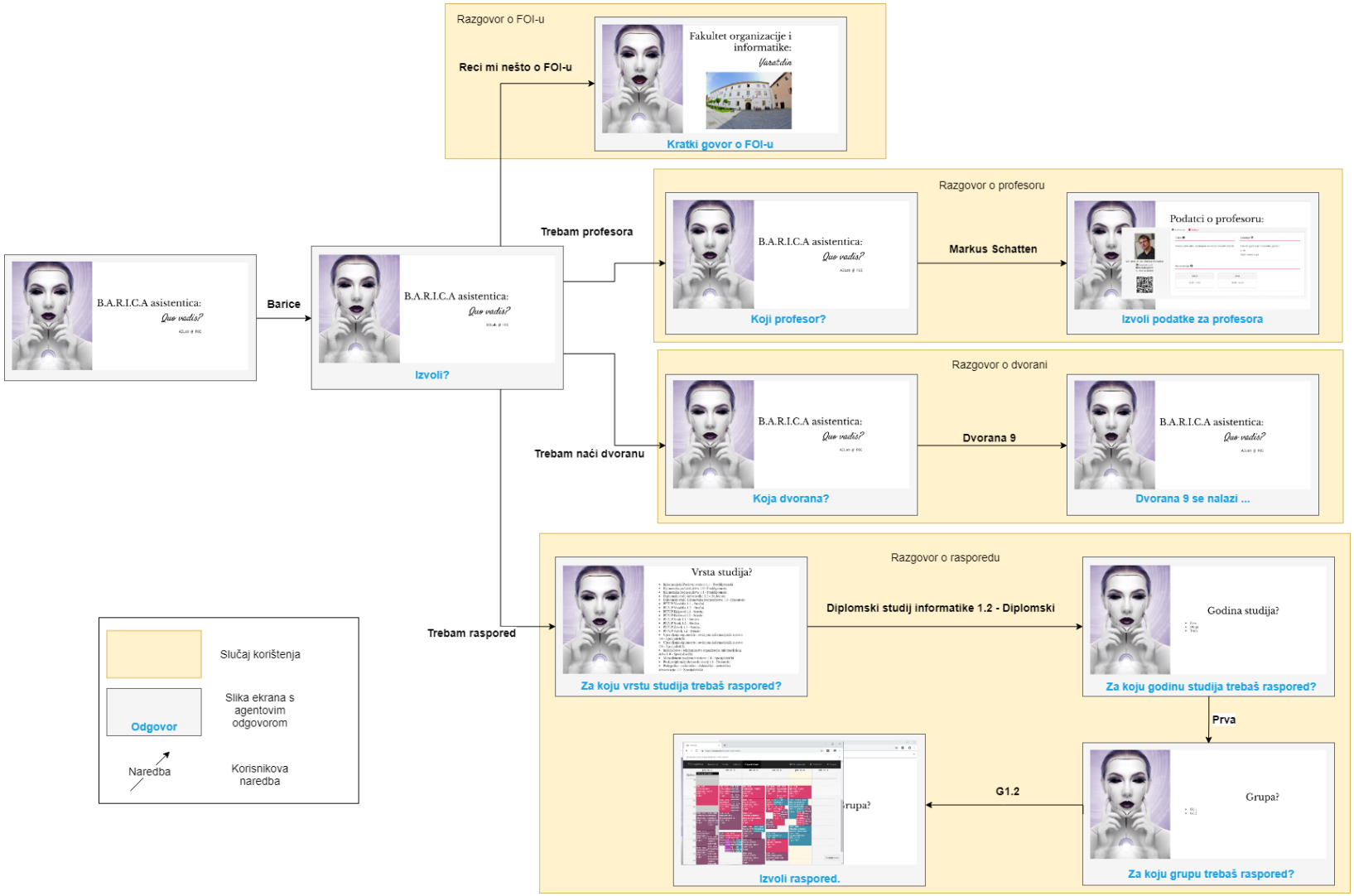
| Notacija | Opis |
|---|---|
|  | Prelazak iz izvorišnog stanja u odredišno kada izvorišno stanje završi. |
|  | Prelazak iz izvorišnog u odredišno stanje kada je u izvorišnom stanju dočekana neka eksplicitna naredba „ulaz“. Naredba „ulaz“ može biti:<br>• naredbaFoi – Korisnik traži od agenta da mu reče nešto o FOI-u.<br>• naredbaTrebamDvoranu – Korisnik kaže agentu da treba naći dvoranu.<br>• naredbaTrebamProfesora – Korisnik kaže agentu da treba profesora.<br>• naredbaTrebamRaspored – Korisnik kaže agentu da treba raspored.<br>• naredbaDvorana – Korisnik kaže agentu naziv dvorane koju želi naći.<br>• naredbaProfesor – Korisnik kaže agentu ime profesora za kojeg treba informacije.<br>• naredbaVrstaStudija – Korisnik kaže agentu naziv vrste studija za koji treba raspored.<br>• naredbaGodinaStudija – Korisnik kaže agentu godinu studija za koju treba raspored.<br>• naredbaGrupa – Korisnik kaže agentu grupu studija za koju treba raspored. |
|  | naredbaBarice - Prelazak iz izvorišnog u odredišno stanje kada je u izvorišnom stanju dočekana eksplicitna naredba „Barice“ kojom korisnik poziva agenta na razgovor izgovarajući riječ „Barice“. Ovom naredbom FSM prelazi u stanje „Izvoli“ iz bilo kojeg od ostalih diskretnih stanja. |
|  | naredbaIzlaz - Prelazak iz izvorišnog u odredišno stanje kada je u izvorišnom stanju dočekana eksplicitna naredba „Izlaz“. Korisnik izgovara riječ „Izlaz“. Ovom naredbom prekida se rad FSM-a, programa i aplikacije iz bilo kojeg od diskretnih stanja. |
|  | naredbaNepoznato - Ponovni ulazak u isto stanje kada je u izvorišnom stanju dočekana eksplicitna naredba ali je ona nepoznata u tom stanju. Prilikom dobivanja nepoznate naredbe agent se vraća u isto stanje u kojem je bio i traži da se naredba ponovi. |
|  | nakon(45 sec.) - Prelazak iz izvorišnog u odredišno stanje ako se eksplicitna naredba čeka dulje od 45 sekundi. Ovim se osigurava da agent bude spreman za sljedećeg korisnika ako trenutni korisnik tijekom razgovora odustane od razgovora. Drugim riječima, ako agent čeka da mu korisnik odgovori dulje od 45 sekundi smatra da je korisnik odustao od razgovora i postavlja se u stanje „Slusam“. |

### 3.2.4. Primjer korištenja sustava

Skica primjera korištenja sa slikama ekrana s otvorenim Internet preglednikom za vrijeme korištenja nalazi se na slici 11. Tu možemo primijetiti i sva četiri slučaja korištenja. Na tranzicijama s jedne slike ekrana na drugu nalazi se primjer korisnikove govorne naredbe dok se ispod svake slike ekrana nalazi primjer agentovog odgovora.

Možemo i pobliže pokazati slike korisničkog sučelja prilikom korištenja. Prikaz naslovne stranice već se nalazi gore na slici 7. Na toj stranici sustav se nalazi dok osluškuje okolinu i čeka da ga se pozove po imenu, dok ga se pozove po imenu, dok razgovara o dvoranama, i dok korisnik inicira razgovor o profesoru sve dok agent čeka da mu korisnik ukaže na određenog profesora.

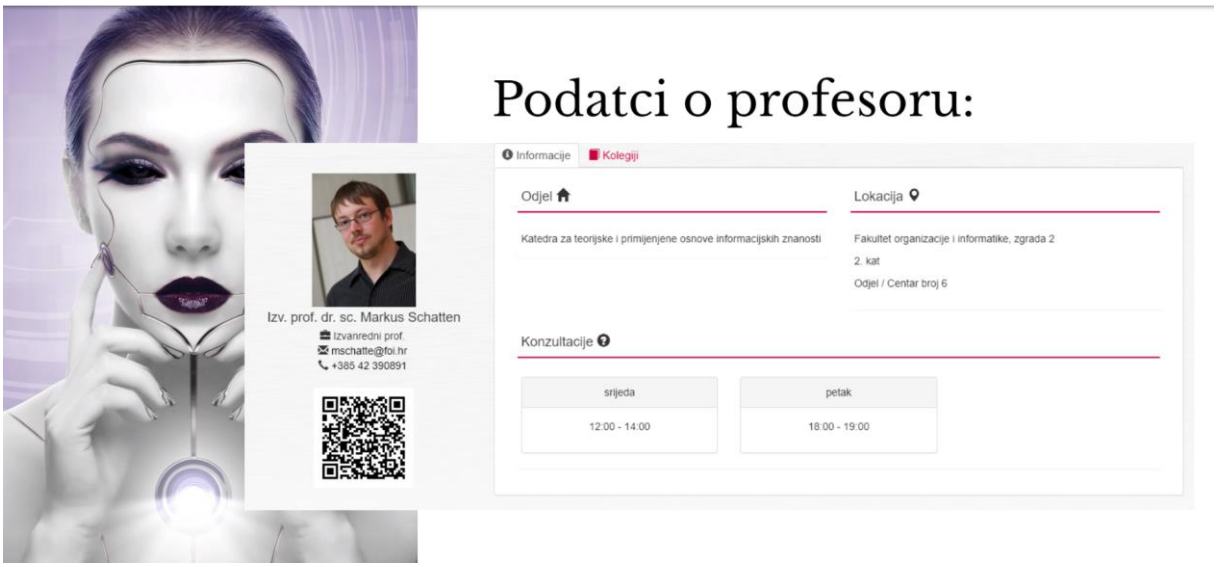
U slučaju da korisnik zatraži od agenta da mu reče nešto o FOI prikazuje se stranica korisničkog sučelja na slici 12. Na svakoj od stranica sustava lijevo se nalazi video humanoidnog robota kako izgovara relevantni tekst ili, ako ništa ne govori, izvodi mimiku slušanja kako bi se pokazalo da osluškuje okolinu i da je spreman na korisnikovu naredbu. U ovom slučaju robot recitira kratki govor o FOI-u, a desno se nalaze naziv i slika fakulteta.




*Slika 11: Prikaz primjera korištenja sustava B.A.R.I.C.A asistentic*


*Slika 12: Stranica sustava kada korisnik dobiva opće informacije o FOI-u*

Kad korisnik zatraži podatke za određenog profesora s FOI-a prikaže mu se stranica kao na slici 13. Robot kratko izgovori „Izvoli podatke za profesora“ i prikaže podatke o profesoru koje je pronašao na službenim stranicama fakulteta. Na ovaj način korisnik može dobiti korisne informacije o profesorima kao što su e-mail adresa, lokacija kabineta i termini konzultacija.

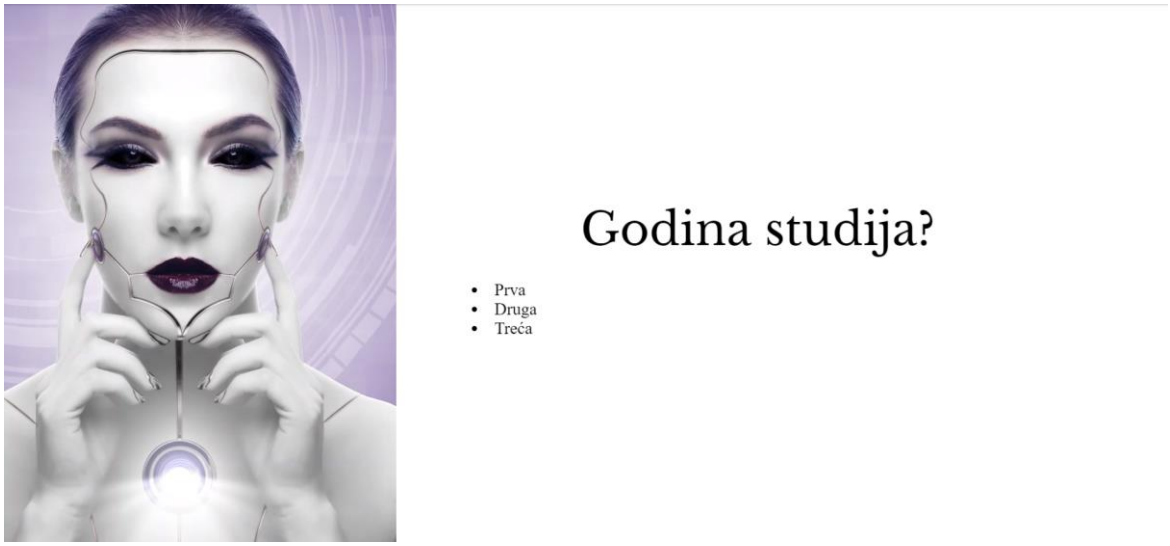



*Slika 13: Stranica sustava kada korisnik dobiva informacije za traženog profesora*

Nadalje možemo pokazati stranice sustava prilikom razgovora o rasporedu. Nakon što korisnik pita agenta za raspored, prikazuje se stranica kao na slici 14, robot izgovara pitanje „Za koju vrstu studija trebaš raspored“. Desno se nalaze sve vrste studija na FOI-u koje su ponuđene , a korisnik treba pročitati onu vrstu studija za koju treba raspored.


*Slika 14: Stranica sustava kada korisnik treba reći vrstu studija za koju treba raspored*

Kad korisnik odredi vrstu studija za koju treba raspored prikaže mu se stranica sustava kao na slici 15 gdje robot izgovara pitanje „Za koju godinu studija trebaš raspored?“. Korisnik treba izgovoriti neku od ponuđenih godina studija s desne strane.




*Slika 15: Stranica sustava s ponuđenim godinama studija za raspored*

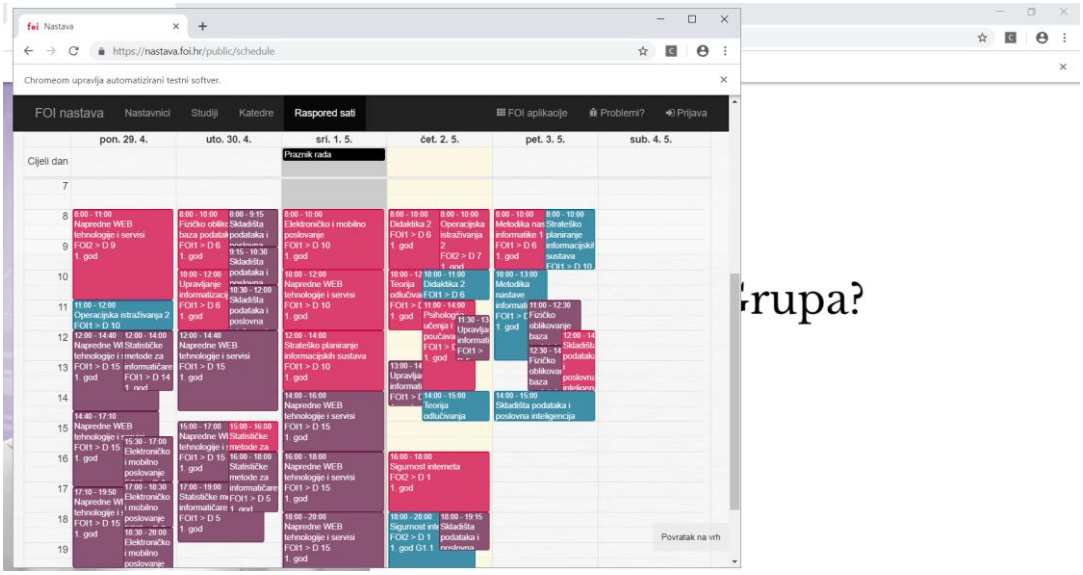
Ako za određenu vrstu i godinu studija postoji raspored korisniku se prikazuje stranica sustava kao ona slici 16. Robot izgovara pitanje „Za koju grupu trebaš raspored?“, a korisnik treba odgovoriti s nekom od desno ponuđenih grupa.


*Slika 16: Stranica sustava s ponuđenim grupama za raspored*

Ako raspored za određenu vrstu i godinu studija nije dostupan, prikaze se stranica sustava kao na slici 17, a robot kratko izgovara da ovakav raspored nije dostupan. To može biti kada npr., korisnik traži raspored za diplomski studij i treću godinu.


*Slika 17: Stranica sustava kada raspored nije dostupan za unesenu vrstu i godinu studija*

Kada korisnik izgovori neku od grupa robot kratko izgovori „Izvoli raspored“, otvara se raspored u novom prozoru Internet preglednika koji se pozicionira tako da se vidi cijeli raspored. Ovaj prozor se zatvara na iduću korisnikovu naredbu ili ako nema nikakve naredbe naredne dvije minute.




*Slika 18: Prikaz sustava kada se korisniku prikazuje raspored*

# 4. Rasprava

Vidjeli smo da korištenje FSM-a za razvoj agenta za razgovor može viti vrlo korisno, pogotovo kada se kao u našoj situaciji, kombinira s razbijanjem modela strojnog učenja u više dijelova koji se onda koriste ovisno o stanju FSM-a. Sustav B.A.R.I.C.A asistentica svojim je funkcionalnim zahtjevima odlučno odgovarao ovom istraživanju iz više razloga.

- U sustavu B.A.R.I.C.A asistentica je bilo potrebe za strukturiranom i povezanom komunikacijom. U takvim je slučajevima posebno korisno koristiti FSM za modeliranje komunikacije kako bi se na korisnikov unos dao pravilan odgovor za nastavak komunikacije u željenom smjeru.
- U sustavu B.A.R.I.C.A asistentica je bilo potrebe za stavljanjem korisnikovog unosa u kontekst razgovora jer smo željeli da agent bude svjestan da jedan te isti unos korisnika u različitim situacijama i dijelovima razgovora može značiti različito. Tu smo se poslužili metodom instanciranja više od jednog modela strojnog učenja od kojih je svaki specijaliziran za neko područje ili temu razgovora. Naravno, u kombinaciji s korištenjem FSM-a.
- U sustavu B.A.R.I.C.A asistentica komunikacija se odvija na hrvatskom jeziku pa i najmanje modeliranje takve komunikacije, proučavanje i istraživanje predstavlja napredak za NLP.

Iako je za ovaj sustav ovaj način modeliranja razgovora i upravljanja razgovorom bio dobar i koristan, ne treba pomisliti da ga treba koristiti za modeliranje svakog sljedećeg agenta za razgovor. Čak i na radu sustava B.A.R.I.C.A asistentica možemo vidjeti neke nedostatke korištenja ovog načina upravljanja komunikacijom. Broj dobrih agentovih odgovora se zasigurno povećava jer je korisnikov unos smješten u kontekst, ali je smanjeno nešto što je nekad vrlo važno kod agenata za razgovor. Korisnik treba znati što u nekom trenutku može reći agentu čime se gubi na prirodnom i spontanom toku razgovora, odnosno, komunikacija na ovaj način mora biti vrlo strukturirana i formulirana. Npr., u sustavu B.A.R.I.C.A asistentica potrebno je znati da se agenta prvo treba pozvati po imenu, zatim da treba inicirati razgovor na neku temu itd. Rezultati ovog nedostatka mogu se ublažiti što boljim modelom FSM-a. To se može postići dobrim definiranjem slučajeva korištenja i crtanjem dijagrama stanja prema slučajevima korištenja prije nego se krene s izradom sustava. Takvim modeliranjem i s dobrim uputama za korisnike praktična primjena ovakvog agenta za razgovor bi mogla biti ostvarena.

Postoje agenti za razgovor u kojima uopće ne bi ni imalo smisla koristiti ovaj način modeliranja razgovora s FSM-om i strojnim učenjem. Npr. agenti za razgovor kojima uopće nije važan kontekst korisnikovog unosa. To je moguće ako njihova komunikacija s korisnicima ne mora biti povezana, svaki korisnikov upit postoji sam za sebe i nije potrebno nastavljati komunikaciju na tu temu. Ili u slučaju ako komunikacija može biti slabo povezana ali svejedno nema potrebe za modeliranjem jer su korisnički unosi dosta različiti i nema straha di bi ih agent mogao poistovjetiti.

Dizajn sustava B.A.R.I.C.A asistentica zajedno s modelom FSM-a napravljen je tako da se u njega mogu ugraditi novi slučajevi korištenja ako se u budućnosti utvrdi da bi sustav studentima mogao pomoći i kod drugih stvari. I to prilično jednostavno, dodavanjem slučajeva korištenja, stanja i tranzicija FSM-a i treniranjem podataka. Također, veliki dio sustava mogao lako bi se primijeniti i za druge fakultete, bilo bi samo potrebno modificirati dio koji dohvaća informacije tako da agent dohvaća informacije za drugi fakultet.

# 5. Zaključak

Proučavanjem teorije o modeliranju agenata za razgovor i izgradnjom stvarnog agenta za razgovor na hrvatskom jeziku došlo je do sljedećeg zaključka. U određenim situacijama može biti od koristi razgovorom na hrvatskom jeziku između korisnika i agenta upravljati kombinacijom FSM-a i strojnog učenja. Pokazalo se da je agent Barica, u sustavu kojeg smo opisivali, u svakom trenutku svjestan teme razgovora, može postaviti korisnikov unos u kontekst i odgovoriti u skladu s time.

Osim što se kombinacija FSM-a i strojnog učenja pokazala uspješnom u radu je pokazan novi, inovativni način razvoja agenta za razgovor na hrvatskom jeziku. Možda bi ovaj način razvoja agenta za razgovor mogao potaknuti i ostale razvojne inženjere s područja umjetne inteligencije na neku inovativnu ideju razvoja hrvatskog NLP-a, bilo kakvog Chatbota na hrvatskom jeziku ili im jednostavno dati ideju za korištenje korisne tehnologije u ovom području.

# Zahvale

Zahvaljujem profesoru Izv. prof. dr. sc. Markusu Schattenu na mentorstvu te obitelji, zaručniku i prijateljima na podršci.

# Popis literature


[1] Weizenbaum, Joseph. "ELIZA---a computer program for the study of natural language communication between man and machine." Communications of the ACM 9.1 (1966): 36-45.

[2] Brandtzaeg, Petter Bae, and Asbjørn Følstad. "Why people use chatbots." International Conference on Internet Science. Springer, Cham, 2017.

[3] Wooldridge, Michael. An introduction to multiagent systems. John Wiley & Sons, 2009.

[4] Radziwill, Nicole M., and Morgan C. Benton. "Evaluating quality of chatbots and intelligent conversational agents." *arXiv preprint arXiv:1704.04579* (2017).

[5] Petrenko, A., N. Yevtushenko, and G. V. Bochmann. "Testing deterministic implementations from nondeterministic FSM specifications." Testing of Communicating Systems. Springer, Boston, MA, 1996. 125-140.

[6] Sarkar, Soumitra, et al. "Automated Incident Management for a Platform-as-a-Service Cloud." *Hot-ICE*. 2011.

[7] Hong, Pengyu, Matthew Turk, and Thomas S. Huang. "Gesture modeling and recognition using finite state machines." Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition. IEEE, 2000.

[8] Mitra, Sushmita, and Tinku Acharya. "Gesture recognition: A survey." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37.3 (2007): 311-324.

[9] Python dokumentacija. Dostupno na https://www.python.org/doc/.

[10] JavaScript dokumentacija — DevDocs. Dostupno na: https://devdocs.io/javascript/.

[11] About ChatterBot — ChatterBot 1.0.2 dokumentacija. Dostupno na: https://chatterbot.readthedocs.io/en/stable/.

[12] Mukhopadhyay, Sayan. Advanced Data Analytics Using Python: With Machine Learning, Deep Learning and NLP Examples. Apress, 2018. S. Mukhopadhyay, Advanced Data Analytics Using Python, Berkeley, CA: Apress, 2018.

[13] GitHub Transitions repozitorij. Dostupno na: https://github.com/pytransitions/transitions.

[14] GitGub Hovercraft repozitorij. Dostupno na: https://github.com/regebro/hovercraft.

[15] GitHub SimpleWebsocketServer repozitorij. Dostupno na: https://github.com/dpallot/simple-websocket-server.

[16] Selenium dokumentacija. Dostupno na: https://www.seleniumhq.org/docs/.

[17] JQuery dokumentacija. Dostupno na: https://jquery.com/.

[18] Talking Avatar and Facial Animation Software – CrazyTalk. Dostupno na: https://www.reallusion.com/crazytalk/.

[19] Voice Notebook. Dostupno na: https://voicenotebook.com/.

[20] ReStructuredText dokumentacija. Dostupno na http://docutils.sourceforge.net/rst.html.

[21] Sangroya, Amit, et al. "Extracting Latent Beliefs and using Epistemic Reasoning to Tailor a Chatbot." IJCAI. 2018.

[22] Klopfenstein, Lorenz Cuno, Saverio Delpriori, and Alessio Ricci. "Adapting a Conversational Text Generator for Online Chatbot Messaging." International Conference on Internet Science. Springer, Cham, 2018.

[23] Yi, Sanghyun, and Kyomin Jung. "A Chatbot by Combining Finite State Machine, Information Retrieval, and Bot-Initiative Strategy."

[24] Meštrović, Ana, et al. "A croatian weather domain spoken dialog system prototype." Journal of computing and information technology 18.4 (2010): 309-316.

# Popis slika

# Popis tablica

# Sažetak

Ime autora: Tajana Šokec

Naslov rada: Modeliranje kontekstualno svjesnog agenta za razgovor na hrvatskom jeziku uz pomoć konačnog automata i strojnog učenja

U ovom radu proučava se način modeliranja agenta za razgovor na hrvatskom jeziku temeljen na kombinaciji korištenja konačnog automata (engl. *Finite State Machine*) i više instanci modela strojnog učenja (engl. *Machine Learning*). Tom kombinacijom svaki korisnikov unos postavlja se u kontekst razgovora na sljedeći način. Stanja konačnog automata opisuju stanje razgovora. Svaka instanca modela strojnog učenja koristi svoju bazu podataka s podacima za treniranje, a specijalizirana je za neki dio razgovora odnosno temu razgovora. Prilikom davanja odgovora korisniku koristi se instanca ovisno o tome u kojem se stanju nalazi konačni automat. Istraživanje se izvodi teorijski i praktično izradom softvera na inače slabo pokrivenom području izrade softvera za razgovor na hrvatskom jeziku.

Ključne riječi: agent za razgovor, konačni automat, strojno učenje

# Summary

Author: Tajana Šokec

Title: Modeling of a context  aware conversational agent on Croatian language based on finite state machine and machine learning

This paper examines the modeling of a conversational agent based on a combination of a finite state machine and multiple instances of machine learning models. With this combination, each user's input is set up in the conversation context as follows. The states of the final state machine describes the status of the conversation. Each instance of the machine learning model uses its own training database and is specialized in some part of the conversation or topic of the conversation. When responding to a user, a specific instance is used depending on the current state of the final state machine. The research is performed theoretically and practically by software development on a generally poorly crafted area of conversational agent development in the Croatian language.

Key words: conversational agent, finite state machine, machine learning

# Kratki životopis

Moje ime je Tajana Šokec. Rođena sam 28.7.1995. godine u Virovitici, a odrasla i živim u Pitomači, Virovitičko-Podravska županija. Srednju školu, prirodoslovno-matematičku gimnaziju, završila sam u Gimnaziji Petra Preradovića u Virovitici. Trenutno studiram u Varaždinu, na Fakultetu organizacije i informatike. Druga sam godina diplomskog studija na smjeru Informacijsko i programsko inženjerstvo. Pohađala sam stručnu praksu u Laboratoriju za umjetnu inteligenciju na fakultetu koji pohađam i trenutno radim na diplomskom radu na temu Singularna dekompozicija matrica i prepoznavanje lica.