Programska aplikacija za pretvaranje govora u tekst

Paić, Mate

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mate Paić

Zagreb, 2020. godina.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

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

Student:

Mate Paić

Zagreb, 2020. godina.

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

Zahvaljujem se mentoru doc. dr. sc. Tomislav Stipančić, dipl. ing. Na zadanom povjerenju, stručnoj pomoći, motivaciji te uloženom trudu, vremenu i strpljenju.

Također želim zahvaliti i svojim roditeljima na strpljenju, potpori i motivaciji tijekom studija i svim kolegama sa kojima sam dijelio divne uspomene tijekom studija.

Mate Paić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Mate Paić** Mat. br.: 0035211383

Naslov rada na hrvatskom jeziku: **Programska aplikacija za pretvaranje govora u tekst**

Naslov rada na engleskom jeziku: **Software application for speech to text convertion**

Opis zadatka:

Računalna analiza i prepoznavanje govora je sposobnost računalnog agenta da prepoznaje riječi i izraze u govornom jeziku i pretvara ih u ljudima čitljivi tekst. Primjena sustava za prepoznavanje govora je sve veća u različitim područjima umjetne inteligencije uključujući: web aplikacije za automatizirano prevođenje govora u tekst, posebna računalna sučelja koja omogućuju interakciju čovjeka i sustava, kognitivnu robotiku u sklopu aplikacija za kontekstualnu percepciju okoline, sustave sveprisutnog računarstva, itd.
U radu je potrebno:

a) objasniti principe rada algoritama za prepoznavanje govora,
b) načiniti pregled sustava za prepoznavanje govora te uz obrazloženje odabrati odgovarajući,
c) razviti modularnu aplikaciju za prepoznavanje govora koristeći Python programski jezik i prikladnu programsku biblioteku,
d) načiniti bazu ključnih riječi prepoznatog teksta.

Programska aplikacija mora biti izvedena tako da najprije prepoznaje govor sa snimljenih materijala te potom u sklopu primjene uživo koristeći mikrofon.
Rad predati u pisanom i elektroničkom obliku uz prikladne upute za korištenje razvijene aplikacije.
U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
28. studenog 2019.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 21. veljače 2020.
2. rok (izvanredni): 1. srpnja 2020.
3. rok: 17. rujna 2020.

Predviđeni datumi obrane:
1. rok: 24.2. – 28.2.2020.
2. rok (izvanredni): 3.7.2020.
3. rok: 21.9. - 25.9.2020.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

SADRŽAJ I
POPIS SLIKA III
POPIS TABLICA V
POPIS OZNAKA VI
SAŽETAK VII
SUMMARY VIII
1. UVOD 1
2. POVIJEST I FONETIKA 3
2.1. Povijest i razvoj 3
2.2. Proces nastajanja govora 4
2.2.1. Anatomija govornog sustava 4
2.2.2. Formiranje govora 5
2.3. Proces nastajanja govora 6
3. ALGORITMI PREPOZNAVANJA GOVORA 8
3.1. Proces nastajanja govora 8
3.2. Osnovna jednadžba sustava za prepoznavanje govora 10
3.3. Arhitektura sustava za prepoznavanje govora 11
3.4. Obrada zvučnog signala i izdvajanje značajki govornog signala 12
3.4.1. Fourierova transformacija 14
3.4.2. Mel skupina filtri 15
3.4.3. Primjena logaritamske operacije 16
3.5. Akustični model 17
3.5.1. Skriveni Markovljevi lanci u akustičnom modelu 17
3.5.1.1. Markovljevi lanci 17
3.5.1.2. Skriveni Markovljevi lanci- HMM 18
3.5.1.3. Osnovni problemi skrivenih Markovljevih lanaca 19
3.5.1.4. Skriveni Markovljevi lanci u ASR-u 20
3.5.2. Dinamičko savijanje vremena - DTW 23
3.5.3. Umjetne neuronske mreže 23
3.5.3.1. Učenje unaprijednih neuronskih mreža 23
3.5.3.2. Generiranje sonena iz segmenata 24
3.5.3.3. Povratne neuronske mreže 25
3.5.3.4. LSTM (engl. Long Short-Term Memory) 26
3.6. Jezični model 27
3.6.1. N-gram model 27
3.6.2. Ocjenjivanje jezičnog modela 30
3.6.3. Napredne tehnologije jezičnog modela 32
3.6.3.1. Jezični model baziran na klasam riječi 32
3.6.3.2. Jezični model s umjetnim neuronskim mrežama 32
3.7. Dekodiranje govora 35
3.8. *End-to-end* sustav za automatsko prepoznavanje govora 40

4. PRIMJENA SUSTAVA ZA PREPOZNAVANJE GOVORA 42
4.1. Appleova Siri…..……..…….……..……..…42
4.2. Amazonova Alexa……..…….……..…..…43
4.3. Microsoftova Cortana……...…...……...……...…43
4.4. Google Assistent……...…...……..……..…...…44
4.5. Sustav za prepoznavanje govora u automobilima..……..…….……...…...…45
5. MODULARNA APLIKACIJA ZA PREPOZNAVANJE GOVORA 46
5.1 Podloga za rad…...…….……..……...……...…46
5.2 Izbor potrebnih biblioteka.…...……..…..…….…..…46
5.3 Odabir potrebne metode……...……..…..……..……47
5.4 Prepoznavanje govora iz zvučnih datoteka..…….…….…..…48
5.5 Prepoznavanje govora sa mikrofona……..……...……..…49
5.6 Aplikacija pretvorbe govora u tekst sa bazom ključnih riječi prepoznatog teksta.49
6. ZAKLJUČAK..... 59
LITERATURA..... 60
PRILOZI..... 62

POPIS SLIKA

Slika 1. Robotski sustav “Plaea“ 2
Slika 2. “Shoebox“ 3
Slika 3. Shematski prikaz organa koji su dio govorni aparata 5
Slika 4. Presjek i osnovni dijelovi vokalnog trakta koji sudjeluju u produkciji govornog signala 6
Slika 5. Građa sustava za prepoznavanje govora 11
Slika 6. Koraci provođenja MFCC analize 12
Slika 7. Valni zapis riječi 'Pozdrav' 13
Slika 8. 'Window' funkcija 14
Slika 9. Fourierova transformacija 15
Slika 10. Frekvencijska domena riječi 'pozdrav' 15
Slika 11. Mel skup filtri 16
Slika 12. Rezultat MFCC analize 16
Slika 13. Markovljev lanac prvog reda 17
Slika 14. Markovljev lanac drugog reda 18
Slika 15. Markovljev lanac sa tri stanja 18
Slika 16. Shematski prikaz modeliranja fona /b/ s tri stanja uporabom HMM-a 20
Slika 17. Povezivanje fona 'brod'…… 21
Slika 18. Kontinuirani model prepoznavanja govora 22
Slika 19. Dijagram ćelija i vrata LSTM mreže 27
Slika 20. Trigram model 28
Slika 21. Shematski prikaz unaprijedne neuronske mreže korištene u jezičnom modelu 33
Slika 22. Shematski prikaz rada povratne neuronske mreže u jezičnom modelu 34
Slika 23. Shematski prikaz H konačnog pretvornika 36
Slika 24. Shematski prikaz L konačnog pretvornika 36
Slika 25. Shematski prikaz G konačnog automata 37
Slika 26. HCLG graf 39
Slika 27. Model end-to-end 40
Slika 28. CTC struktura 41
Slika 29. Apple HomePad 43
Slika 30. Amazon Echo 43
Slika 31. Microsoft Invoke 44
Slika 32. Google Home 45
Slika 33. Potrebni paketi za rad aplikacije 47
Slika 34. Pip install 47
Slika 35. Inicijalizacija Recognizer klase 47
Slika 36. Kod za pretvorbu sadržaja zvučne datoteke u tekst 48
Slika 37. Kod pretvorbe govora u tekst pomoću mikrofona 49
Slika 38. Pretvorba sa govora snimljenih materijala u tekst 50
Slika 39. Rezultat pretvorbe govora u tekst zvučne datoteke 'speech.wav' 50
Slika 40. Izbor jezika u aplikaciji 51
Slika 41. Upit za pomoć 52
Slika 42. Razgovor sa govornom asistenticom “Plaea 53
Slika 43. Kod i rezultat za izradu alarma(prije) 53
Slika 44. Kod i rezultat za izradu alarma(poslije) 54

Slika 45. Kod za vremensku prognozu 54
Slika 46. Funkcija plaea_speak za izradu glasovne asistentice “Plaea 55
Slika 47. Kod za snimit govor 56
Slika 48. Funkcija record_audio() 56
Slika 49. Rezultat pretvorbe govora u tekst sa bazama ključnih riječi n hrvatskom jeziku . 57
Slika 50. Rezultat pretvorbe govora u tekst sa bazama ključnih riječi n engleskom jeziku 57
Slika 51. Rezultat pretvorbe govora u tekst sa bazama ključnih riječi n njemačkom jeziku 58

POPIS TABLICA

Tablica 1. Primjeri grešaka prilikom prepoznavanja riječi……..9

Tablica 2. Značajke dobrog i lošeg jezičnog modela…….31

Tablica 3. Slijed dekodiranja(odozdo prema gore)……....38

POPIS OZNAKA

Oznaka	Jedinica	Opis
A		matrica tranzicijskih vrijednosti
ASR		automatsko prepoznavanje govora (engl. *Automatic Speech Recognition*)
B		matrica emitiranih vrijednosti
CTC		konekcionistička vremenska klasifikacija (engl. *Connectionist Temporal Classification)*
DNN		duboka neuronska mreža(engl. *Deep Neural Network*)
E		funkcija cilja
FSA		konačni automati (engl. *finite state automata)*
FST		konačni pretvornici(engl. *finite state transdures*)
GMM		Gausov miješani model(engl. *Gaussian Mixed Model*)
HMM		skriveni Markovljevi lanci(engl. *Hidden Mark Model*)
LM		jezični model(engl. *language model*)
LSTM		povratna mreža s dugom kratkotrajnom memorijom (engl. *Long Short-Term Memory*)
M		broj različitih opservacijskih simbola
N		broj mogućih stanja kod skrivenih Markovljevih modela
O		skup opservacija
P		matrica prijelaznih vjerojatnosti
q_t		stanje skrivenog Markovljevog modela u vremenskom trenutku t
W		niz prepoznatih riječi
RTF		faktor stvarnog vremena(engl. *real-time factor*)
RNN		povratna neuronska mreža(engl. *Recurrent Neural Network*)
S		skup stanja kod Markovljevih lanaca
T		vremenski skup kod Markovljevih lanaca
WER		postotak pogrešno prepoznatih riječi(engl. *Word Rate Error*)
w(n)		prozorska funkcija
X		skup slučajnih varijabli kod Markovljevih lanaca, niz akustičnih značajki
$X_m(f)$		Fourierova transformacija
Π		Inicijalna raspodjela vjerojatnosti stanja kod skrivenih Markovljevih lanaca

SAŽETAK

Govor je najprirodniji oblik komunikacije među ljudima. Zadatak sustava pretvorbe govora je da pretvori ono što čovjek kaže u tekstualni oblik. Aplikacije za pretvorbu govora omogućuju ljudima da koriste govor kao drugi, lakši način interakcije sa određenim računalima ili strojevima. Već preko tri desetljeća, provedeno je mnoštvo istraživanja o različitim aspektima djela umjetne inteligencije koja se bavi prepoznavanjem govora i njegove primjene. Danas mnogi proizvodi uspješno koriste automatsko prepoznavanje govora za komunikaciju između ljudi i strojeva. Detaljna analiza sustava za prepoznavanje govora je provedena u ovom završnom radu koja uključuje samu arhitekturu, algoritme, primjene i probleme samog sustava za prepoznavanje govora.

Ključne riječi: sustav za prepoznavanje govora, govor, značajke signala govora, fonem, akustični model, jezični model, neuronske mreže, skriveni markovljevi lanci, Python

SUMMARY

Speech is the most natural communication mode for human beings. The task of the speech conversion system is to convert what a person says into textual form. Speech conversion applications allow people to use speech as a antoher, easier way to interact with specific computers or machines. For more than three decades, a great amount of research was carried out on various aspects of artificial inteligence dealing with speech recognition and its applications. Today, many products successfully use automatic speech recognition to communicate between humans and machines. A detailed analysis of the speech recognition system is carried out in this final thesis which includes the architecture, algorithm, applications and problems of the automatic speech recognition.

Key words: speech, speech recognition system, speech signal characteristics, phonem, acoustic model, language model, neural networks, hidden markov model, Python

1. UVOD

Danas su računala postala sastavni dio života. Nalazimo ih u različitim oblicima, od stolnih računala i laptopa do mobitela i kućnih pomagala, kao i industrijskih robota i navigacije na brodovima. Kao što je rečeno u drugom poglavlju, govor je ljudima instinktivan način komunikacije pa je logično da se razvijaju sustavi bazirani na govoru kao sredstvu komunikacije. Prepoznavanje govora te razvijanje sustava za automatsko prepoznavanje je multidisciplinaran problem koji objedinjuje znanja lingvistike, računarstva, strojnog učenja itd. Ovako jednostavno objašnjenje više odgovara algoritmima čiji je zadatak u osnovi transkripcija. Oni određene zvučne valove povezuju s određenim slovima „bez ikakvog razmišljanja“. U drugom poglavlju je dan kratak pregled povijesti i razvoj sustava za prepoznavanje govora radi stjecanja dojma i upoznavanja s problematikom s kojom su se istraživači u ovom području morali nositi i osnove fonetike koje su nam bitne radi lakšeg raspoznavanja riječi stroju. Nadalje su podrobnije opisani algoritmi koji se koriste u sustavima za prepoznavanje govora. Priroda govornih signala omogućuje stvaranje značajki signala određenim algoritmima koji su opisani u nastavku. Algoritmi koji se koriste za analizu značajki te konačan odabir prepoznatih glasovnih jedinica čine ključan dio sustava za prepoznavanje govora, stoga su im posvećena posebna potpoglavlja. Ovaj cilj još uvijek nije u potpunosti ostvaren te se u razvitku svih budućih sustava radi na njegovom ostvarenju. Zatim su nabrojani sustavi koji koriste pretvorbu govora kao sredstvo komunikacije. I za kraj je razvijena modularna aplikacija u programskom jeziku Python u kojoj je eksperimentalno napravljan sustav za prepoznavanje govora.

Kao motiv ovom završnom radu bio je razvoj sustava za prepoznavanje govora koji će se implementirati na robotski sustav nazvan “Plaea“. Radi mogućnosti prepoznavanja govora “Plaea“ ima potencijala za još mnogo više. Može se razvijati tehnologija u kojoj “Palea“ ima razvijenu kontekstualnu percepciju okoline kao i sposobnost komunikacije sa čovjekom. Primjerice, na osnovu akustike ljudskog govora , ona je sposobna odrediti emociju u čovjeku. Izgled robotskog sustava “Plaea“ prikazan je na slici [Slika 1].

Slika 1. Robotski sustav “Plaea“

2. POVIJEST I FONETIKA

2.1. Povijest i razvoj

Razvoj sustava za prepoznavanja govora seže nedaleko u povijest. Intenzivniji razvoj započinje tek u drugoj polovici dvadesetog stoljeća što je uzrokovano razvojem tehnologije. U ovom kratkom pregledu povijesti razvoja sustava za prepoznavanje govora spomenuti su važniji događaji koji su dovodili do razvoja sustava za prepoznavanje govora.[1]

Godine 1952. Bell Laboratories su dizajnirali sustav "Audrey" koji je mogao prepoznati prvi put u povijesti glas. Nije mogao prepoznati riječi, već samo brojke. Deset godina kasnije , IBM JE predstavio "Shoebox", prikazan na slici [Slika 2] koji je prepoznavao 16 izgovorenih riječi engleskog jezika te znamenke od 0 do 9.

Širom svijeta druge su države razvile hardver koji je mogao prepoznati zvuk i govor. A do kraja 60-ih tehnologija bi mogla podržati riječi s četiri samoglasnika i devet suglasnika.

Slika 2. "Shoebox"

Prepoznavanje govora napravilo je nekoliko značajnih koraka tijekom 70-ih godina 20. stoljeća. Za to su zaslužni američko Ministarstvo obrane i DARPA. Program istraživanja razumijevanja govora (SUR) koji su vodili bio je jedan od najvećih u povijesti prepoznavanja govora. Govorni sustav Carnegie Mellon "Harpy", koji je proizašao iz ovog programa, uspio je razumjeti preko 1000 riječi, što je otprilike isto kao rječnik trogodišnjaka.

Također značajno u 70-ima bilo je uvođenje sustava Bell Laboratories koji je mogao tumačiti više glasova.

‘80 -ih je rječnik prepoznavanja govora prešao s nekoliko stotina riječi na nekoliko tisuća riječi. Jedno od otkrića proizašlo je iz statističke metode poznate kao „Skriveni Markovljevi lanci (HMM)“. Umjesto da samo koristi riječi i traži zvučne obrasce, HMM je procijenio vjerojatnost da nepoznati zvukovi zapravo jesu riječi.

Prepoznavanje govora je napredovalo u '90-ima velikim dijelom zbog razvoja osobnih računala. Brži procesori omogućili su šire korištenje softvera poput Dragon Dictate, koji je mogao prepoznati 100 riječi u minuti. Program je zahtijevao treniranje prije korištenja koje je trajalo 45 minuta.

BellSouth je predstavio glasovni portal (VAL) koji je bio interaktivan sustav na pozivanje koji je davao informacije na osnovu toga što je korisnik rekao na telefonu.

2000-tih godina se pojavljuje širok raspon programskih podrški za prepoznavanje govora. Programske podrške prepoznavanja govora za osobna računala u prosjeku su imale postotak prepoznavanja oko 80%. Tadašnji sustavi su dobre rezultate prepoznavanja pokazivali samo za relativno ograničene vokabulare. Windows Vista i Mac OS X su dolazili sa sustavom za prepoznavanje koji je omogućavao upravljanje glasovnim komandama. Međutim, većina korisnika nije koristila ove značajke, što zbog nepouzdanosti, što zbog složenosti korištenja.

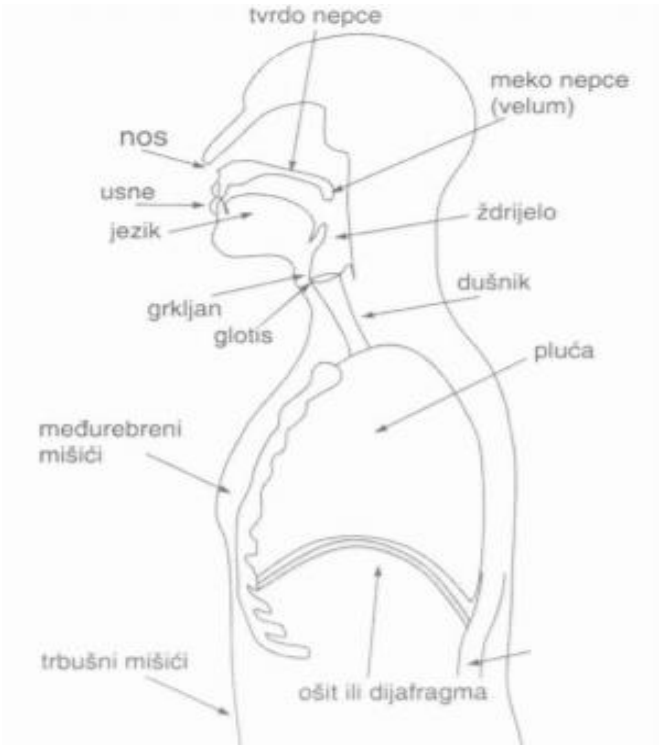
2.2. Proces nastajanja govora

Ljudi usmeno izražavaju misli, osjećaje i ideje kroz niz složenih pokreta koji mijenjaju i oblikuju osnovni ton stvoren glasom u specifične zvukove koji se mogu dekodirati. Govor se proizvodi precizno koordiniranim djelovanjem mišića u glavi, vratu, prsima i trbuhu. Razvoj govora je postupan proces koji zahtijeva godine prakse.

2.2.1. Anatomija govornog sustava

Ždrijelna i oralna šupljina, zajednički poznate kao vokalni trakt, dinamički se skupljaju te opuštaju, stvarajući sve vrste zvukova kroz rezonancu. Nazalna šupljina otvara još jednu rupu za zrak kako bi stvorila ono što lingvisti nazivaju nazalni glasovi. Zajedno, ove šupljine

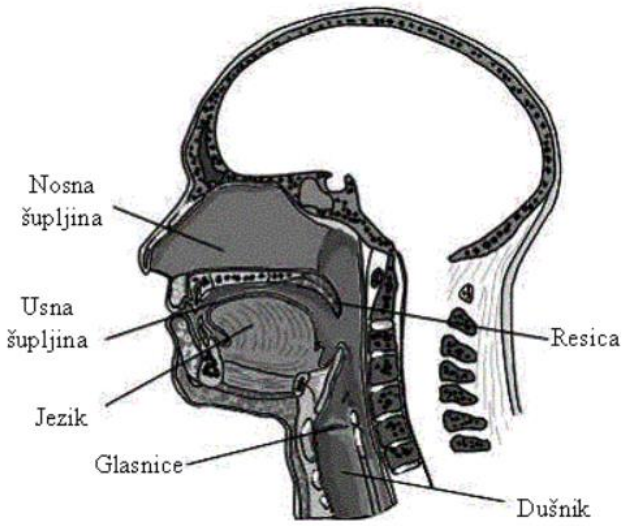
karakteriziraju zvukove koje ljudi proizvode.[2] Shematski prikaz organa za govor dan je na slici [Slika 3].



Slika 3. Shematski prikaz organa koji su dio govorni aparata

2.2.2. Formiranje govora

Artikulatorima nazivamo organe koji sudjeluju u formiranju govora koji sadrži informaciju koja se želi prenijeti (jezik, glasnice, itd.)[3]. Važnu ulogu u nastajanju govora imaju i pluća govornika koja se pod djelovanjem mišića prsnog koša stišću i potiskuju zrak kroz vokalni trakt. Značajnu ulogu u procesu formiranja govora imaju glasnice. Smještene su na vrhu dušnika (engl. *trachea*) te djeluju u kombinaciji sa strujanjem zraka. Osciliranje glasnica prilikom strujanja zraka slično je ponašanju piska (engl. *reed*) puhačkih instrumenata. Na frekvenciju titranja glasnica najviše utječu tlak zraka iz pluća te napetost samih glasnica koju je moguće kontrolirati. Vokalni trakt se ponaša kao svojevrsni filtar koji određuje koje se spektralne komponente pojačavaju a koje prigušuju. Geometrijski je oblik vokalnog trakta naravno promjenjiv, a određuje ga položaj artikulatora govornika.[Slika 4]



Slika 4. Presjek i osnovni dijelovi vokalnog trakta koji sudjeluju u produkciji govornog signala

2.3. Fonetika

Desetljeća lingvističkih i fonetskih studija prethodile su sustavu prepoznavanja govora. U ovom potpoglavlju će se kratko razjasniti način stvaranja govora i način kako se predočuje.

Fonetika ili glasoslovlje jezikoslovna znanstvena je disciplina koja se bavi proučavanjem artikulacijskih i akustičkih obilježja glasova i govora[4].

Postoje tri važne podjele na fonetiku, jednako važne za pretvorbu govora[5]:

- artikulacijsku fonetiku - proučava djelovanje artikulacijskih organa tj. govornog prolaza. Pri proizvodnji govornih glasova razlikuju se mjesto i način izgovora. Stoga su za opis glasova uspostavljena tri kriterija: zvučnost, način artikulacije i mjesto artikulacije.
- akustičku fonetiku - proučava akustičke osobine glasova i govora. U glavne se osobine govornih zvukova ubrajaju: amplituda(glasnoća), frekvencija(visina tona), trajanje(duljina glasa) i spektar.
- perceptivnu ili auditivnu fonetiku - proučava načine tumačenja glasova i govora

Fonemi se dijele na samoglasnike i suglasnike.

Samoglasnik je fonem koji nastaje slobodnim prolaskom zračne struje između govornih organa. Glasnice pritom titraju, pa je samoglasnik nužno zvučan. U hrv. jeziku postoji pet samoglasnika a, e, i, o, u, a neki lingvisti tomu nizu dodaju i dvoglas ie, koji je nastao od nekadašnjeg jata. Najistaknutiji glas u slogu, središte sloga, slogotvorni glas, glas na kojem se

nalazi naglasak. Veličina i oblik usne šupljine stvaraju različite samoglasnike. Čeljust može biti podignuta ili spuštena, jezik položen više ili niže, uzdignut na vrhu ili na kraju, usnice razmaknute ili zaokružene (zaobljene), glas može rezonirati samo u usnoj šupljini ili i u nosnoj šupljini[4].

Suglasnik je glas koji se artikulira uz relativno veliku zaprjeku dahu kroz glasovni prolaz (suženje ili zatvor). Za razliku od samoglasnika, neki suglasnici su nezvučni fonemi i zbog toga nemaju izraženu frekvenciju, što će se vidjet kasnije u dijagramu.

Za izgovor riječi se rastavljaju na slogove. Slog je obično sastavljen od jezgre(najčešće samoglasnik) sa početkom i krajem(obično suglasnik).

Fonem je najmanja sljedbena jedinica u jezičnome sustavu koja služi za sporazumijevanje tako da razlikuje značenje, iako je sama bez značenja. Riječi se tvore spajanjem jednog ili više fonema. Fonemi su stalne i apstraktne jedinice jezika, dok su foni njihovo promjenjivo ostvarenje, odnosno konkretni glasovi. Ideja fonetskog pisma je da jedno slovo predstavlja jedan glas. No često se radi o više slova koja samostalno predstavljaju jedan glas, a u određenoj kombinaciji predstavljaju drugi. Fonem može uključivati donekle različite zvukove ili alofone. Npr. slovo "m" se u riječima **m**ama i tra**m**vaj različito izgovara. Dakle, fonemi su pojmovi u lingvističkom jeziku kako bi se razlikovale riječi, a foni su način na koji ih izgovaramo.[5]

U sustavima prepoznavanja govora se sakupljaju tekstualni korpus koja su fonetski prepisana i vremenski usklađena (označeno je vrijeme početka i završetka svakog telefona).

TIMIT je jedan popularni korpus koji sadrži izgovore 630 sjevernoameričkih govornika.

Razlog zašto je bitno razumjeti osnove fonetike leži u sljedećem. Posljednjih nekoliko desetljeća ključno je bilo pronalaženje najvjerojatnijeg niza riječi s obzirom na zvuk. Drugim riječima, princip je pojednostavljen za pronalaženje niza riječi W s najvećom vjerojatnošću s obzirom na promatrane audio signale. Više o tome u narednim poglavljima.[6]

3. ALGORITMI PREPOZNAVANJA GOVORA

Automatsko prepoznavanje govora (ASR) je skup računalnog hardvera i softverskih algoritama namijenjeni za identifikaciju i obradu ljudskog glasa. Koristi se za prepoznavanje riječi izrečenih od strane govornika ili za provjeru autentičnosti identiteta osobe koja govori u sustav.

Raspoznavanje govora prvenstveno se koristi za pretvaranje izgovorenih riječi u tekst spremljen na računalu. U pravilu, sustav zahtijeva unaprijed konfigurirane ili spremljene glasove primarnih korisnika. Ljudi trebaju trenirati takav sustav pohranjivanjem obrazaca govora i rječnika u sustav.

3.1. Ocjenjivanje sustava

Kada gradimo sustav za prepoznavanje veoma je važno testirati sustav. Sustavi za prepoznavanje govora ocjenjuju se s obzirom na uspjeh prepoznavanja riječi ili s obzirom na značaj. Najpoznatija mjera za mjerenje točnosti ASR-a je Word Error Rate(WER) koja se mjeri u postotku prepoznatih riječi[7]. Postoje tri vrste greške koje sustav može napraviti[Tablica 1]:

1. greške supstitucije(engl. *substitution*) - kada sustav pogrešno prepozna jednu riječ kao neku drugu riječ,
2. greške brisanja (engl. *deletion*) – kada sustav ne prepoznaje riječ a ona se nalazi u govoru,
3. greške umetanja(engl. *insertion*) - kada sustav doda riječ koja ne postoji na snimci i transkriptu govora.

Formula za računanje WER-a glasi:

$$WER = \frac{N_{\text{sub}} + N_{\text{ins}} + N_{\text{del}}}{N_{\text{ref}}} \quad \textbf{(1)}$$

gdje je N_{sub} broj grešaka supstitucije, N_{del} broj grešaka brisanja, N_{ins} broj grešaka umetanja, N_{ref} broj riječi u referentnom transkriptu govora. Brojčana vrijednost WER-a je uvijek između 0 i 1.

Tablica 1. Primjeri grešaka prilikom prepoznavanja riječi

Govor	Detekcija	Greška
however	how	substitution
	never	insertion
a		deletion

Ponekad se sve tri vrste grešaka ne smatraju jednako bitnima pa se izraz (1) modificira određenim faktorima. U pravilu se WER računa za svaku rečenicu pojedinačno, a za ukupni WER cijelog govora zbrajaju se pojedinačni za rečenice i dijele s brojem rečenica.

Stopa pogreške u rečenici (SER) rjeđe je korištena mjerna vrijednost koja svaku rečenicu tretira kao jedan uzorak koji je točan ili netočan. Ako je bilo koja riječ u rečenici pretpostavljena pogrešno, rečenica se ocjenjuje netočnom. SER se izračunava jednostavno kao udio netočnih rečenica u ukupnom broju rečenica.

Druga metoda ocjenjivanja performansi sustava za prepoznavanje govora je testiranje statističkog značaja (engl. *significance testing*). Ova metoda uključuje mjerenje razlike između dva eksperimenta ili dva algoritma te se ispituje je li pogreška posljedica samog algoritma, varijabilnih podataka, eksperimentalnog okruženja ili nekih drugih faktora. Najčešće korištena metoda koja se koristi zove se MAPSSWE (engl. *Matched Pairs Sentence-Segment Word Error*). Prilikom testiranja ove metode, testni set je podijeljen na segmente uz pretpostavku da su pogreške u jednom segmentu statistički neovisne jedna o drugoj. Ova se pretpostavka dobro podudara s tipičnim eksperimentima prepoznavanja govora gdje se mnogi izgovori prevode jedan po jedan.

Osim točnosti, sustavi za prepoznavanje govora ocjenjuju se i prema brzini i latentnosti. Brzina prepoznavanja se mjeri s obzirom na stvarno vrijeme, a označava se s faktorom stvarnog vremena (engl. *real-time factor, RTF*). Ako je RTF = 1, onda se radi o prepoznavanju u stvarnom vremenu, odnosno za 10 sekundi govora potrebno je 10 sekundi procesiranja. Ako je RTF > 1, onda sustavu treba više vremena za procesiranje no što je duljina snimljenog govora. Ovakvi sustavi su korisni kada je točnost važnija od brzine. Ako je RTF < 1, onda predviđa riječi brže no što ulazni podaci dolaze. To je korisno kada je pokrenuto više sustava na jednom računalu jer je onda moguće procesirati više zvučnih zapisa istovremeno. Ovakvi sustavi mogu „uhvatiti“ stvarno vrijeme, odnosno sakriti svoju latentnost iza svoje brzine.

3.2. Osnovna jednadžba sustava za prepoznavanje govora

Primarni cilj sustava za prepoznavanje je pretpostaviti najvjerojatniji niz simbola iz valjanog niza riječi u jeziku koji dolazi iz ljudskog govora tj. akustičnog ulaza. Akustične značajke se promatraju ka niz diskretnih promatranja:

$$O = o_1, o_2, o_3, \ldots o_N, \quad \textbf{(2)}$$

za koje se traži najvjerojatniji niz riječi $W = \{W_1, \ldots W_M\}$[7].

Potrebno je pronaći maksimum vjerojatnosti P(R|X), odnosno niz riječi koji najvjerojatnije odgovara nizu akustičnih značajki dobivenih prilikom akustične analize:

$$W = \text{argmax}_w P(W|O) \quad \textbf{(3)}$$

Da bismo riješili ovaj izraz, koristimo Bayesovo pravilo[8]:

$$P(W|O) = \frac{P(X|O)}{P(O)} \quad \textbf{(4)}$$

Budući da niz riječi ne ovisi o graničnoj vjerojatnosti promatranja P(O), ovaj se pojam može zanemariti. Sve komponente izraza (4) definiraju se odgovarajućim raspodjelama vjerojatnosti koje se dobivaju učenjem akustičnog i jezičnog modela.

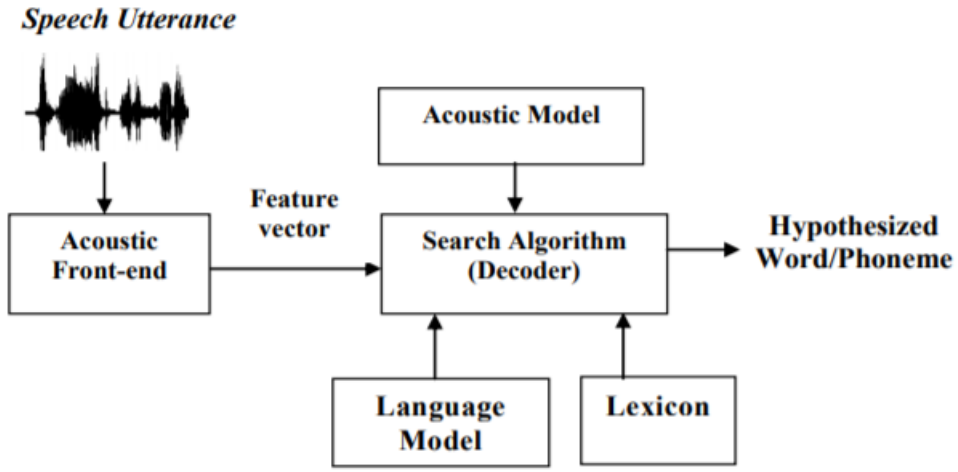
$$W = \text{argmax}_W P(O|W)P(W) \quad \textbf{(5)}$$

P(O|W) je komponenta jednadžbe koja opisuje akustični model sustava za prepoznavanje, koja opisuje akustična promatranja u danom nizu riječi. Akustični model odgovoran je za modeliranje načina na koji se nizovi riječi pretvaraju u akustične realizacije, a zatim u akustička promatranja u sustavu za prepoznavanje govora.

P(W) je komponenta jednadžbe koja opisuje jezični model sustava za prepoznavanje, koja predstavlja vjerojatnost da se određene riječi nađu zajedno u uređenom nizu W. Može se i vježbati sa riječima koje se očekuju da će biti potrebi u svakodnevnoj potrebi.

3.3. Arhitektura sustava za prepoznavanje govora

Svaki sustav za prepoznavanje govora se sastoji od sustava akustične analize(engl. *Acoustic Front-end*), akustičnog modela(engl. *Acoustic model*), jezičnog modela(engl. *Language model*), rječnika izgovora(engl. *Lexicon*) i dekodera kao što je prikazano na slici [Slika 5] [9]



Slika 5. Građa sustava za prepoznavanje govora

Prvi korak u prepoznavanju govora je pretvorba govora u oblik s kojim računalo može raditi, odnosno A/D pretvorba zvučnog signala. U trenutku kada je govor izgovoren, on postaje zvučni val i kao takav dolazi u kontakt sa sustavom za prepoznavanje govora.

Sustav akustične analize pretvara govorni signal u odgovarajuće značajke koje pružaju korisne informacije za prepoznavanje. Pretvorba ulaza govornog signala u obliku valnog oblika(engl. *waveform*) iz mikrofona u niz fiksnih akustičnih vektora da bi računalo moglo razumjeti je proces koji se naziva izdvajanje akustičnih značajki(engl. *feature extraction*).

Značajke govornog signala su određen skup vrijednosti koji su dobiveni određenim metodama i algoritmima nad određenim vremenskim isječkom digitaliziranog govornog signala. Značajke govornog signala predstavljaju statističke vrijednosti koje sadrže informaciju o trenutnom stanju signala. Omogućuju stvaranje i sažimanje informacije ključne za raspoznavanje osnovnih glasovnih jedinica – fonema. Osim fonema, mogu poslužiti i za raspoznavanje većih jezičnih jedinica, ovisno o primijenjenom algoritmu.

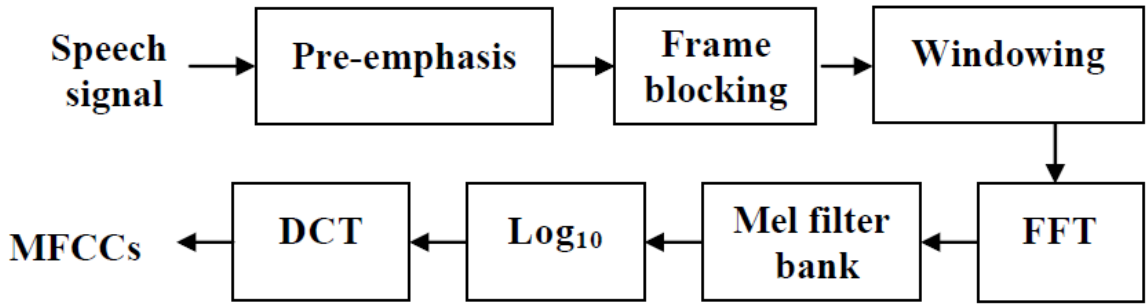
Ovako zapisani govorni signal dolazi do dekodera gdje se vrši uspoređivanje ulaznog signala s postojećim obrascima kako bi se pronašao najvjerojatniji par. Prilikom uspoređivanja, sustav se koristi akustičnim modelom, jezičnim modelom i rječnikom izgovora. Akustični

model služi za usporedbu fonema, jezični model koristi se za provjeru pravila jezika (gramatika i sintagme), a rječnik izgovora daje vezu između niza fonema i pisane riječi. Potrebna je faza 'učenja' u kojoj govornik čita sve riječi koje razmatra za potencijalno korištenje u budućnosti. Te riječi se skladište u rječnik i kasnije kada se nova riječ treba prepoznati uzima se iz rječnika i uspoređuje se sa najsličnijom riječi.

Nakon što dekoder sustava odabere najvjerojatniji niz riječi s obzirom na sve modele (akustični, jezični i rječnik izgovora), dobiveni niz riječi se ocjenjuje pa potvrđuje ili ispravlja po potrebi. Izlaz sustava je niz potvrđenih prepoznatih riječi.

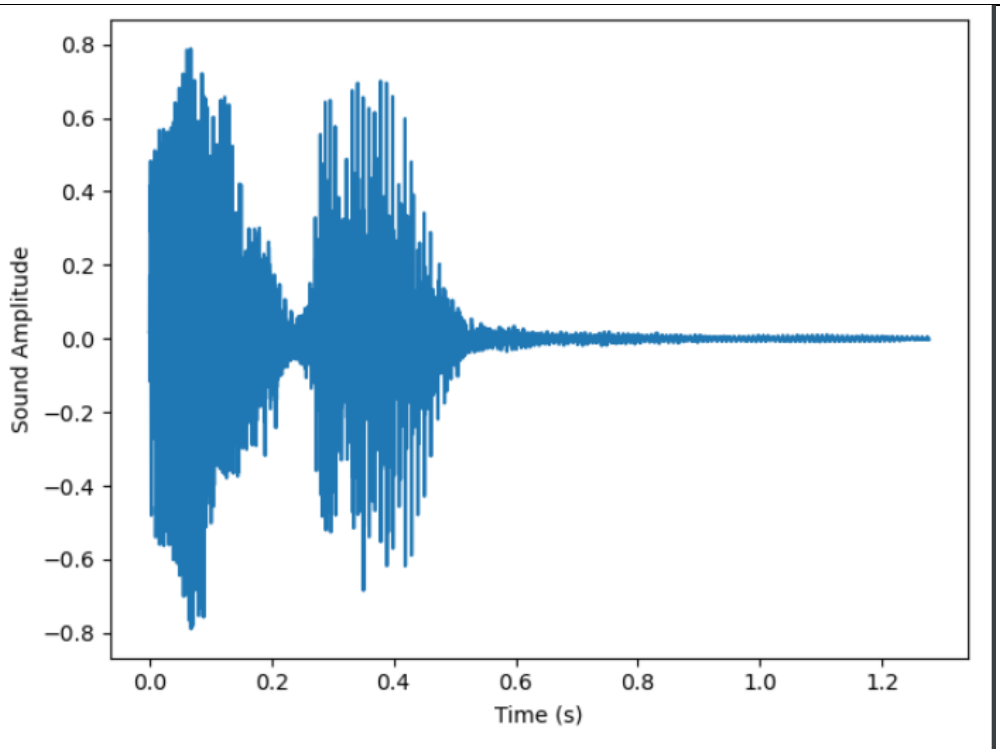
3.4. Obrada zvučnog signala i izdvajanje značajki govornog signala

Govorni signal potrebno je obraditi i izdvojiti značajke govora, odnosno provesti akustičnu analizu[10]. Akustična analiza sastoji se od sljedećih koraka[Slika 6]:



Slika 6. Koraci provođenja MFCC analize

Govorni zvučni valovi prodiru kroz zrak i dolaze do mikrofona koji pretvara akustične vibracije u električnu energiju. Prvi korak je prenaglašavanje koji primjenjuje visoko frekvencijski filter prije samog izdvajanja značajki kako bi se ispostavilo svojstvo da naglašen govor u pravilu ima veću energiju pri nižim frekvencijama, no što je ima nenaglašen govor pri visokim frekvencijama [10]. Ta električna energija uzrokuje valne oblike koji opisuju signal. Govorni signali imaju frekvenciju od oko 8000 Hz pa se koristi uzrokovanje od 16000Hz. Na slici [Slika 7] dan je valni zapis riječi 'Pozdrav' koje sam ja izgovorio. Pomoću Python jezika snimio sam i pretvorio u audio file i vizualizira svoj govor u valni oblik.



Slika 7. Valni zapis riječi 'Pozdrav'

Informacija u govornom signalu je zapravo predstavljena u kratkoročnim promjenama amplitude spektra valnog oblika signala. Vidi se da bezvučni dijelovi(suglasnici) imaju oblik nosa, dok zvučni dijelovi(samoglasnici) imaju uzdužno gibanje zbog vibriranja glasnica.

Značajke signala se upravo izvlače iz amplituda spektara kratkotrajnih odsječaka (okvira) signala. Značajke signala omogućuju sažimanje informacije govornog signala koje olakšavaju usporedbu značajki signala s drugim signalima, što je korisno za primjene prepoznavanja govora. Iz razloga velike raznovrsnosti govornih signala(različiti naglasci govornika, dijalekti, emocije...), provodi se izdvajanje akustičnih značajki radi smanjenja raznovrsnosti[11]

Sa slike [Slika 7] uočava se da govor nije stacionaran signal, što znači da mu se svojstva mijenjaju tijekom vremena. Radi toga, da bi mogli pravilno analizirati govorni signal, trebamo ga podijeli na manje dijelove, segmente signala, u kojima se ide sa pretpostavkom da je proces stacionaran. U pravilu u sustavu za prepoznavanje govora upotrebljavaju segmenti signala u trajanju od 25ms sa preklapanjem od 10ms[7]. Time je 1 sekunda raspoređena na 100 segmenata.

25ms je dovoljno veliko da se dohvati informacija da značajke unutar signala i dalje ostale relativno stacionarne. Ideja je da izvučene značajke budu neovisne o naglasku govornika i da zanemare potencijalnu okolnu buku. Popularna metoda izvlačenja značajki koja se koristi

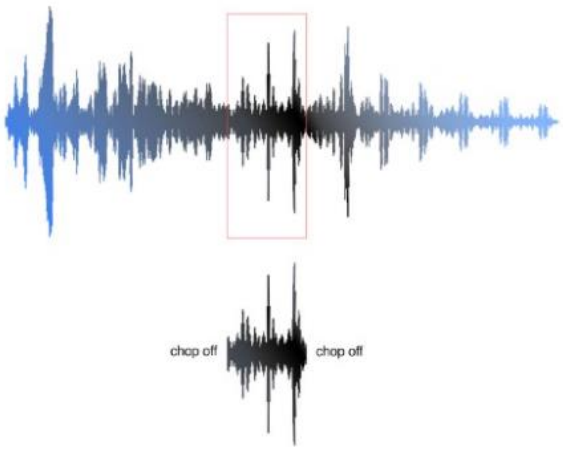
naziva se Mel frekvencijski kepstralni koeficijenti(engl. *Melfrequency cepstral coefficients* – MFCC).

3.4.1. Fourierova transformacija

Zato što izvlačimo segmente iz dužeg signala, potrebno je na njihove rubove primijeniti neku *'window'* funkciju.[Slika 8] 'Window funkcija osigurava da ne dođe do diskontinuiteta signala između susjednih segmenata. Najčešće se koristi Hamming funkcija kod koje amplituda opada na rubu koja glasi[10]:

$$w(n) = 0{,}54 - 0{,}46\cos\left(2\pi\frac{n}{N}\right), 0 \leq \mathrm{n} \leq N \quad \textbf{(6)}$$

gdje je N broj segmenata signala.

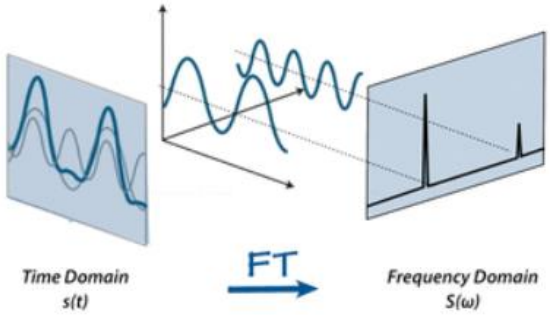


Slika 8. 'Window' funkcija

Svaki segment signala u vremenu pretvaramo u frekvencijsku domenu koristeći diskretnu Fourierovu transformaciju(u većini sustava implementirana je kao brza Fourierova transformacija) koja glasi[7]:

$$X_m[k] = \sum_{n=0}^{N-1} \mathrm{w[n]x[mN + n]}e^{-j2\pi fn} \quad \textbf{(7)}$$

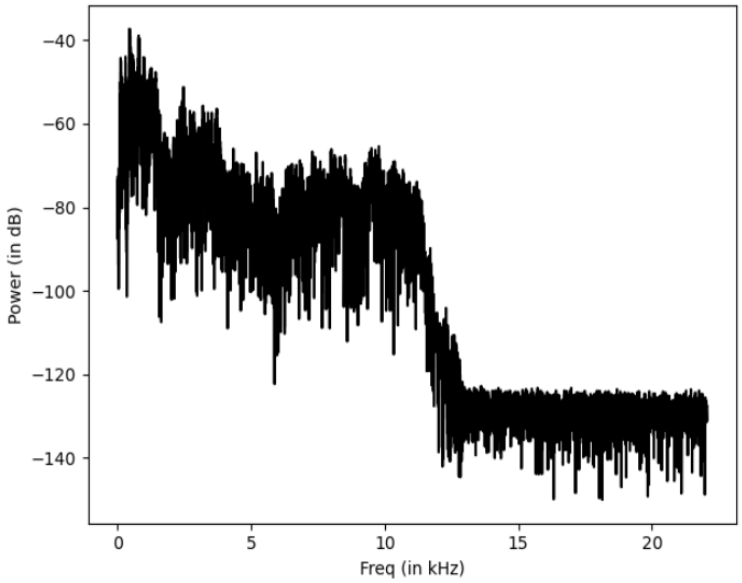
gdje je $x(n)$ predstavlja zvučni signal, w(n) neku prozorsku funkciju, a m je indeks segmenta.



Slika 9. Fourierova transformacija

Apsolutna vrijednost Fourierove transformacije $X_m(f)$ koristi se za spektralni prikaz signala u obliku 2D spektrograma

Pomoću Python jezika se riječ 'Pozdrav' iz valnog oblika, kojem je na apscisi zadano vrijeme, pretvorila u graf frekvencijske domene kao što je prikazano na slici [Slika 10]

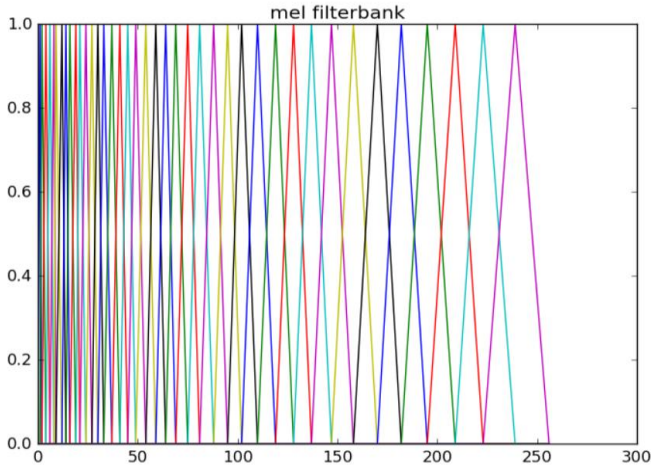


Slika 10. Frekvencijska domena riječi 'pozdrav'

3.4.2. Mel skupina filtri

Kako bi se uklonila raznolikost uzrokovana harmonijskom strukturom u govoru, izvodi se posebna operacija izglađivanja prema spektru magnituda. Najčešće se koristi Mel skup filtera (engl. *mel filterbank*). Primijenjeni u otprilike logaritamskoj skali na os s frekvencijama, ti

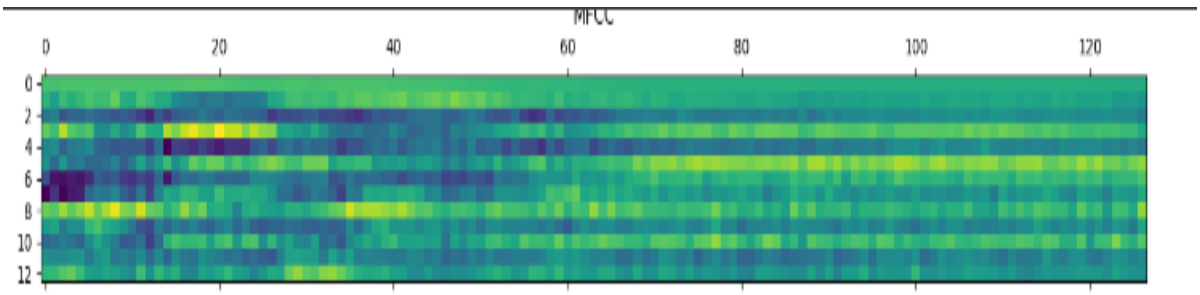
filtri postaju širi i međusobno udaljeniji s povećanjem frekvencije. Mel filtri mogu se zapisati u obliku matrice, gdje svaki red predstavlja jedan filter.[7]



Slika 11. Mel skup filtri

3.4.3. Primjena logaritamske operacije

Posljednji korak izdvajanja značajki signala je primjena logaritamske operacije. Ona sažima dinamički raspon signala, a kao izlaz daje koeficijente skupina filtri, koji pomoću nelinearne Mel-frekvencijske skale aproksimiraju karakteristike ljudskog slušnog sustava. Kako se najčešće koriste MFCC raspona 40, izlazni spektrogram nakon logaritmiranja prikazuje koeficijente filtriranja u obliku 40- dimenzionalnog skupa filtera. Na slici 3.7 je prikazano konači rezultat MFCC analize riječi 'pozdrav' pomoću Python jezika.[Slika 12]



Slika 12. Rezultat MFCC analize

Dijelovi s visokom energijom prikazani su žuto, a dijelovi s niskom energijom zeleno i plavo.

3.5. Akustični model

Većina današnjih sustava za pretvorbu govora koristi umjetne neuronske mreže i skrivene Markovljeve lance(HMM) kao algoritme prepoznavanja.[12]

Umjetne neuronske mreže se koriste za pretpostavljanje na razini fonema, dok skriveni Markovljevi lanci pretpostavljene foneme pretvaraju u pretpostavljeni niz koji čini riječ.

3.5.1. Skriveni Markovljevi lanci u akustičnom modelu

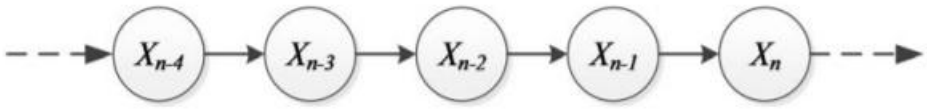
3.5.1.1. Markovljevi lanci

Prije proučavanja HMM-a, bilo bi korisno ukratko proći Markovljeve lance. Markovljev lanac predstavlja niz stanja sustava. U svakom trenutku sustav može prijeći u neko novo stanje ili može ostati u istom stanju. Promjene stanja nazivaju se tranzicije. Niz diskretnih slučajnih varijabli X_0, X_1, … . zvat ćemo stohastičkim lancem. Slučajne varijable uzimaju vrijednosti u konačnom skupu $S = \{s_0, s_1, \dots, s_n\}$[13]

Lanac X_0, X_1, … je Markovljev lanac prvog reda [Slika 13], ako za sve izbore stanja s_0, s_1, …, $s_n \in S$ vrijedi:

$$P(X_n = s_n|X_{n-1} = s_{n-1}, \dots, X_0 = s_0) = P(X_n = s_n|X_{n-1} = s_{n-1}) \quad \textbf{(8)}$$

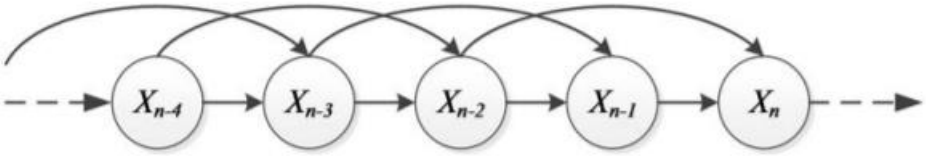
Ovdje trenutak n predstavlja sadašnjost, a 0,..., n-1 prošlost. Sadašnje stanje s_n ovisi samo o prethodnom s_{n-1}, ali ne i o načinu na koji je proces dospio u prethodno stanje tj. vrijednostima procesa u ranijim trenutcima.



Slika 13. Markovljev lanac prvog reda

Markovljev lanac drugog reda [Slika 14] ovisi o dvama prethodnim stanjima s_{n-1} i s_{n-1} te vrijedi:

$$P(X_n = s_n|X_{n-1} = s_{n-1}, \dots, X_0 = s_0) = P(X_n = s_n|X_{n-1} = s_{n-1}, X_{n-2} = s_{n-2}) \quad \textbf{(9)}$$

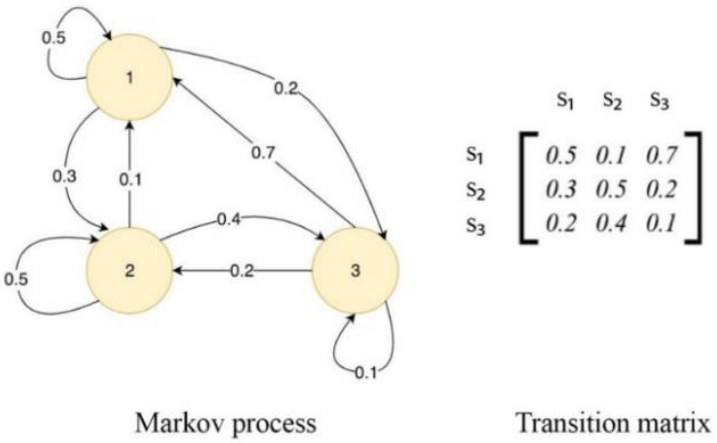


Slika 14. Markovljev lanac drugog reda

Veza između slučajnih varijabli X*n* i X*n*-1 zadana je prijelaznim vjerojatnostima. Vjerojatnost prijelaza iz stanja s_i u stanje s_j je p_{ij}.

Matrica s elementima pij označava se s P i naziva se matrica prijelaznih vjerojatnosti.

$$P = (p_{ij})i, j \in \{1, 2, \dots, k\} \quad \textbf{(10)}$$



Slika 15. Markovljev lanac sa tri stanja

3.5.1.2. Skriveni Markovljevi lanci- HMM

Kod markovljevih lanaca imali smo slučaj da je proces vidljiv tj. da u svakom trenu znamo stanje sustava. U slučaju skrivenog markovljevog lanca (engl. *Hidden Markov Model*, HMM) nemamo informaciju o stanju sustava, već imamo samo opservacije. HMM je MM sa skrivenom sekvencom stanja. To je dvostruki stohastički proces koji se sastoji od skrivenog stohastičkog proces (sekvenca stanja) i vidljivog stohastičkog procesa koji generira sekvencu opažanja.

Za svaki skriveni Markovljev model možemo definirati pet parametara koji ga jednoznačno određuju[14]:

1. N – broj stanja u kojima se proces može nalaziti

   $S = \{S_1, S_2, \dots, S_N\}$ **(11)**

   Kod prepoznavanja govora se ograničava tranzicija stanja na model „s lijeva na desno“ (engl. *left-to-right*) što znači da se samo stanje koje je neposredno ispred utječe na trenutno stanje. Ovakav način modeliranja idealan je za prepoznavanje fonema.[12]

2. M – broj različitih opservacijskih simbola

   $T = \{b_1, b_2, \dots, b_M\}$ **(12)**

   U prepoznavanju govora ovaj parametar najčešće odgovara broju slova u abecedi jezika prema kojem je sustav modeliran.

3. A - matrica tranzicijskih vjerojatnosti prelaska proces iz stanja u stanje

   $A = [a_{ij}], a_{ij} = P(q_{t+1} = S_j | q_t = S_i), 1 \leq i, j \leq N$ **(13)**

   Stanje modela u nekom vremenskom trenutku t označava se s q_t.

4. B - matrica vjerojatnosti emitiranja simbola

   $B = [b_j(k)], b_j(k) = P(O_t = b_k | q_t = S_i), 1 \leq j \leq N, 1 \leq k \leq M$ **(14)**

5. Π - matrica inicijalnih tranzicijskih vrijednosti

   $\Pi = [\pi_i] \pi_i = P(q_0 = S_i), 1 \leq i \leq N$ **(17)**

3.5.1.3. Osnovni problemi skrivenih Markovljevih lanaca

Skriveni Markovljevi modeli omogućuju jednostavno modeliranje procesa iz stvarnog svijeta, ali da bi ipak bili korisni u praski potrebno je riješiti tri osnovna problema.[7]

Prvi od sa kojim se susrećemo je problem evaluacije. Kod evaluacijskog problema pitamo se koja je vjerojatnost da je zadani model generirao uočeni niz opservacija. Ovaj problem evaluacije može se riješiti zbrajanjem vjerojatnosti nad svim mogućim vrijednostima skrivenog niza stanja. Unaprijedni algoritam(engl. *forward algorithm*) je učinkovito rješenje za dinamičko programiranje. Kao što mu samo ime govori, obrađuje niz u jednom prolazu. Pohranjuje do N vrijednosti u svakom vremenskom koraku i smanjuje računsku složenost.

Zatim dolazi problem dekodiranja. Rješavanjem problem dekodiranja trudimo se otkriti najvjerojatniji niz skrivenih stanja koji su mogli producirati dati niz opservacija. Koristeći jednostavnu metodu, moglo bi razmišljati na način da se pronađe svaka moguća kombinacija

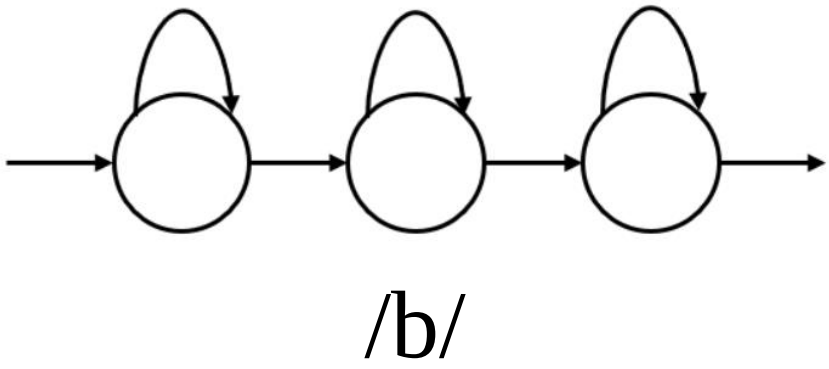
stanja, tj. svaki mogući put kroz model i zatim, za svaku tu kombinaciju stanja tj. izračunati vjerojatnosti emitiranja datog niza opservacija. Primjenjuje jednostavniji algoritam poznat pod nazivom Viterbijev algoritam.

Iza kraj nam preostaje riješiti problem treniranja. Osnovni zadatak treniranja skrivenih Markovljevih lanaca je određivanje parametara a_{ij} i b_{ij} koristeći skup uzoraka. Ovaj se problem može učinkovito riješiti pomoću algoritma Baum-Welch, koji uključuje Naprijed-Natrag algoritam (engl. *Forward-Backward* algorithm). Rezultat unaprijednog algoritma je taj da izračunava vjerojatnost da bude u stanju i trenutku t s obzirom na sva opažanja do uključujući vremena t. Algoritam unatrag ima sličnu strukturu, ali izračunava vjerojatnost da bude u stanju i trenutku t s obzirom na sva buduća promatranja koja počinju od t + 1. Ova dva algoritma se ujedinjuju u jedan da bi se povećala šansa da budu točni na vrijeme u svim mogućim opservacijama.

3.5.1.4. Skriveni Markovljevi lanci u ASR-u

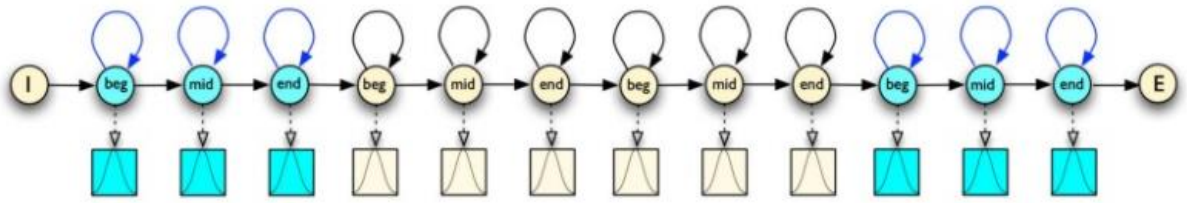
U prepoznavanju govora, skriveni Markovljevi modeli koriste se za modeliranje akustičkih značajki na razini fonema. Rječnik izgovora (engl. *pronunciation lexicon*) se modelira skrivenim Markovljevim lancima.

Treba se odrediti iz akustičkih značajki koliko segmenata odgovara jednom fonu. Zato što foni nisu homogeni tijekom vremena, dijelimo ih na još manje dijelove. Tipično se svaki fon modelira s tri stanja kako bi se odvojilo modeliranje početka, sredine i kraja izgovora fona. Svako stanje ima mogućnost prijelaza u samo sebe ili u iduće stanje, kao što se vidi na slici [Slika 16]



Slika 16. Shematski prikaz modeliranja fona /b/ s tri stanja uporabom HMM-a

Primjerice riječ 'brod' se povezuje sa 4 fonema od kojih svaki ima tri stanja.



Slika 17. Povezivanje fona 'brod'

Stoga je visokokvalitetni izgovorni rječnik koji svaku riječ u sustavu "piše" svojim fonima izuzetno važan za uspješno akustičko modeliranje. Tihe zvukove puno je teže uhvatiti. Možemo ga modelirati s 5 unutarnjih stanja umjesto s tri. Za neke ASR-ove također možemo koristiti različite fone različite tišine.[12]

Povijesno se za raspodjelu vjerojatnosti po stanjima koristio Gaussov miješani model (engl. *Gaussian Mixture Model*, GMM). Današnji sustavi više ne koriste Gaussov miješani model, već jednu duboku neuronsku mrežu kojoj je izlaz niz vrijednosti koji predstavlja sva stanja skrivenog Markovljevog modela za sve moguće foneme. Takvi sustavi se nazivaju hibridni sustavi ili DNN-HMM sustavi Na primjeru hrvatskog jezika koji ima 32 fonema, izlazni vektor neuronske mreže sadržavao bi 96 (32 x 3) vrijednosti.

Kako foni uvelike ovise o položaju u riječi, ne bi ih smili razmatrati neovisno o kontekstu riječi. Segment zvučnog signala bi trebao imati svoj fon i kontekst (engl. *context-dependent phone*)[15]. Gleda se prethodni fon, trenutni i sljedeći, odnosno radi se o tri uzastopna fona koji se jednom riječju nazivaju trifoni. Kada se koriste fonemi u kontekstu dolazi do velikog porasta stanja s obzirom na broj fonema u jeziku. Broj trifona je N3 , gdje N predstavlja broj fonema. Kada se koristi po tri stanja za svaki trifon, za hrvatski jezik je broj stanja 98304.

Ovakav porast broja stanja dovodi do dva velika problema:

- puno manje podataka prema kojima bi se učili trifoni
- neki trifoni se neće pojaviti pri učenju mreže, ali pojavit će se prilikom testiranja ili upotrebe

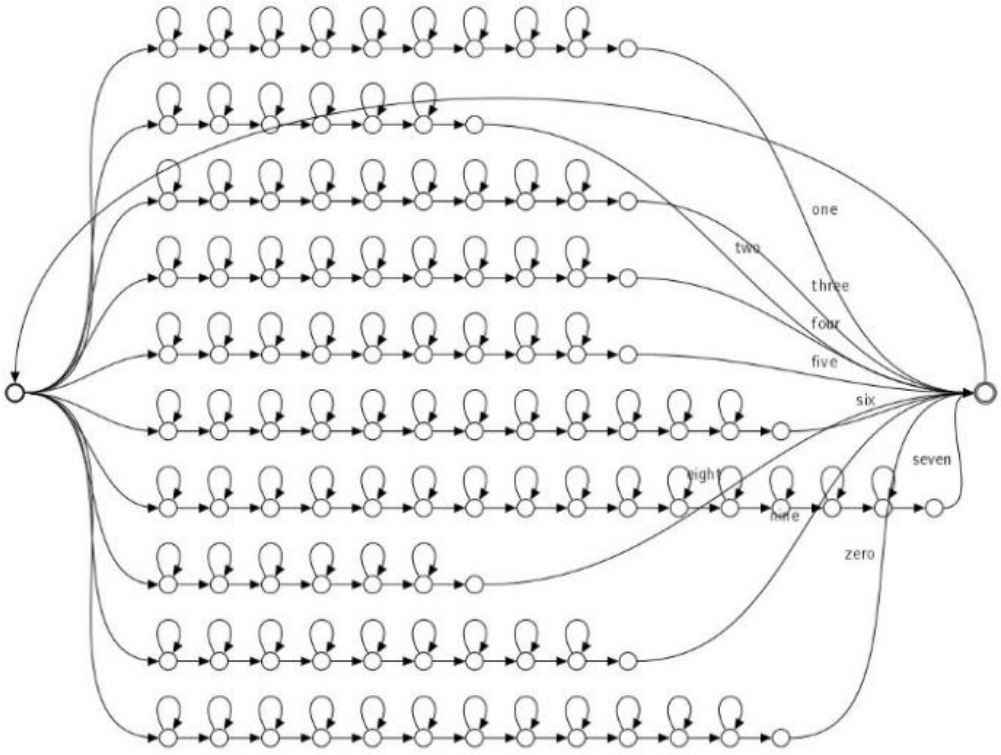
Rješenje ovih problema uključuje povezivanje više međusobno slična trifona i spajajući ih u jedno HMM stanje i time se smanjuje broj potrebnih stanja. Ovo povezano stanje se naziva senon.

Postupak stvaranja senona odvija se pomoću drva odluka. Drvo odluke se konstruira za svako stanje svakog fona ovisnom o kontekstu.

Prvo se povežu svi trifoni sa sličnim središnjim fonom i oni predstavljaju korijen drva. Zatim se drvo širi prema pitanjima o lijevim i desnim, odnosno prethodnim i sljedećim fonima. Primjerice, koliko su naglašeni ili kojoj skupini fona pripadaju. Drvo se širi po ovom modelu sve dok vjerojatnost pojavljivanja bilo kojeg trifona nije iznad predefinirane granice. Po završetku rasta drva, krajevi drva predstavljaju senone.

Ovaj postupak rješava oba gore navedena problema. Prvo, podaci se sada mogu dijeliti između nekoliko stanja trifona, tako da su procjene parametara prilično točne. Drugo, ako je u vrijeme ispitivanja potreban trifon koji se nije vidio u fazi učenja, potrebne se senone može se pronaći hodajući stablom odluka i odgovarajućim odgovorima na pitanja o razdvajanju. Skoro svi aktualni sustavi za prepoznavanje govora koriste senone u ovisnosti o kontekstu.

Koncept prepoznavanja govora s jednom riječju može se proširiti na kontinuirani govor s modelom HMM[Slika 18]. Dodamo lukove da bismo povezali riječi zajedno u HMM-u.



Slika 18. Kontinuirani model prepoznavanja govora

3.5.2. Dinamičko savijanje vremena - DTW

Dinamičko savijanje vremena (engl. *dynamic time wrapping*) pristup je koji se u prošlosti koristio za prepoznavanje govora, ali koji je sada uglavnom potisnut uspješnijim pristupom temeljenim na HMM-u.[[16]

DTW algoritam uspoređuje parametre nepoznate riječi s parametrima postojećih predložaka svake riječi. Prednost DTW-a je to što nalazi mogućnost prepoznavanja govora pri različitim brzinama izgovora. Nedostatak ove metode je to što vrlo mali pomaci u točkama usporedbe signala mogu voditi to netočnog rezultata. Osim toga, nemoguće je sekvence pohraniti u obliku grafova kao kod skrivenih Markovljevih 12 modela te tako omogućiti složenije algoritme pretraživanja s manjim bazama za pohranu značajki govora.

3.5.3. Umjetne neuronske mreže

Jedan od najznačajnijih pomaka u prepoznavanju govora u posljednjih nekoliko godina je uporaba dubokih neuronskih mreža u zvučnom modelu. Kao što je ranije spomenuto, hibridni DNN(engl. *deep neural network*) sustavi zamjenjuju zbirku GMM-a (po jedan za svaki senon) jednom dubokom neuronskom mrežom s izlaznim oznakama koje odgovaraju senonima. [7]

3.5.3.1. Učenje unaprijednih neuronskih mreža

Najčešće korištena umjetna neuronska mreža u akustičnom modelu je standardna unaprjedna neuronska mreža. Tijekom korištenja unaprijedne neuronske mreže, ona se uči kako bi klasificirala svaki segment ulaznog signala. Prilikom klasifikacije korisno je stvoriti kontekstualni prozor (engl. *context window*) za svaki segment koji će poslužiti kao ulaz u mrežu. Za segment u signalu t, ulaz u mrežu je simetrični prozor N segmenata prije i N segmenata kasnije. Ako je x_t vektor značajki u trenutku t, ulazni vektor mreže glasi:

$$x_t = [x_{t-N}, x_{t-N-1}, \dots, x_t, \dots, x_{t+N-1}, x_{t+N}] \quad \textbf{(19)}$$

U pravilu se N kreće između vrijednosti 5 i 10, ovisno o količini dostupnih podataka za učenje. Veći kontekstualni prozor osigurava više informacija, ali zahtjeva veću ulaznu matricu značajki, što može biti nezgodno u situacijama s malom količinom podataka za učenje. Često se i vektor značajki proširuje sa svojim vremenskim derivacijama koje se ponekad nazivaju i delta značajkama (engl. *delta features*). Jednostavan primjer računanja delta značajki glasi:

$\Delta x_t = x_{t+2} - x_{t-2}$, **(20)**

$\Delta^2 x = \Delta x_{t+2} - \Delta x_{t-2}$, **(21)**

Ulaz u mrežu kontekstualni prozor koji se sastoji od originalnog vektora značajki, delta značajki i delta-delta značajki(x_t, Δx_t i $\Delta^2 x_t$).

3.5.3.2. Generiranje sonena iz segmenata

Najčešća funkcija cilja koja se koristi za učenje unaprijednih neuronskih mreža je funkcija prijelazne entropije (engl. *cross entropy*) koja glasi:

$$E = -\sum_{i=1}^{M} t_m \log(y_m), \quad \textbf{(22)}$$

gdje je M broj klasa senona, t_m je oznaka (1 ako pripada klasi m i 0 ako ne), a y_m je izlaz mreže. Odnosno, za svaki segment potrebno je generirati vektor dimenzija M x 1 koji se sastoji od nula, osim jedne jedinice koja odgovara točnom senonu. To znači da trebamo dodijeliti svaki segment svakog izgovora jednom senonu kako bismo generirali ove oznake.

Funkcija prijelazne entropije zahtjeva da svaki segment bude klasificiran, što može dovesti do problema jer se u govoru javljaju pauze i prekidi. Ovo se rješava uporabom povezane vremenske klasifikacije (engl. *Connectionist Temporal Classification*, CTC) koja se sastoji od dva koraka. Prvo svrstava tj. spaja zvukovne informacije segmenta u fone. Nadalje povezuje nizove fona sa optimalnim riječima što nas oslobađa od drva odluke. Za klasificiranje segmenata zvučnog zapisa općenito se koristi metoda prisilnog poravnavanja (engl. *forced alignment*) koja generira najvjerojatniji senon za dani segment. Ova metoda zahtjeva već postojeći sustav za prepoznavanje govora kako bi funkcionirala. To može biti sustav s Gaussovim miješanim modelom ili sustav s dubokim neuronskim mrežama koje su već naučene. Izlaz ove metode je datoteka u kojoj je zapisano sljedeće: početni segment i vrijeme početka izgovorene riječi unutar njega, krajnji segment i vrijeme završetka izgovorene riječi te odgovarajuća oznaka senona. Primjer ovakve izlazne datoteke je MLF datoteka koju koristi već spomenuti softver za prepoznavanje govora, HTK. Stupci MLF datoteke interpretiraju se kao:

1. vrijeme početka u 100 ns,
2. vrijeme završetka u 100 ns,

3. oznaka senona,
4. ocjena akustičnog modela za taj segment senona,
5. HMM trifon model,
6. ocjena akustičnog modela za trifon,
7. transkript izgovorene riječi (pojavljuje se na početku izgovora).

3.5.3.3. Povratne neuronske mreže

Za razliku od unaprijednih neuronskih mreža, povratne neuronske mreže RNN(engl. *recurrent neural network*) podatke na ulazu obrađuju u obliku niza dok se težine mreže međusobno vremenski ovisne. Postoji više standardnih oblika povratnih mreža. Konvencionalna povratna neuronska mreža ima skriven sloj izlaza koji se može izraziti:

$$h_t^i = f(W^i h_t^{i-1} + U^i h_{t-1}^i + c^i)\,, \tag{23}$$

gdje $f(\cdot)$ predstavlja nelinearnu funkciju (npr. sigmoidalnu), i je sloj mreže, t je oznaka broja segmenta ili vremenski indeks, a ulaz x je ekvivalentan izlazu nultog sloja, $h_t^i = x_t$. Slojevi u povratnim mrežama ovise o trenutnom ulazu te izlazu prijašnjeg vremenskog koraka.

Povratne neuronske mreže iznimno su pogodne za sustave za prepoznavanje govora zbog svoje vremenske ovisnosti. U akustičnom modeliranju povratne neuronske mreže mogu naučiti vremenske uzorke značajki fonema zapisanih u obliku vektorskog niza. Zato što mreža uči korelaciju vremena i podataka, upotreba kontekstualnog prozora više nije potrebna.

Učenje povratnih mreža se isto može odvijati pomoću funkcije cilja prijelazne entropije s malo modificiranom metodom izračuna gradijenta. Koristi se varijacija učenja s povratnim rasprostiranjem pogreške, algoritam učenja povratnog rasprostiranja pogreške u vremenu (engl. *back-propagation through time,* BPTT). Kao i standardni algoritam učenja s povratnim rasprostiranjem pogreške, BPTT optimizira parametre mreže korištenjem algoritma najstrmijeg pada primjenjujući pritom gradijent pogreške. Prilikom učenja dolazi do množenja velike količine gradijenata što se može odraziti na vrijednost izraza. Moguća su dva krajnja slučaja koje treba izbjegavati prilikom učenja. Prvo, izraz može poprimiti vrijednost

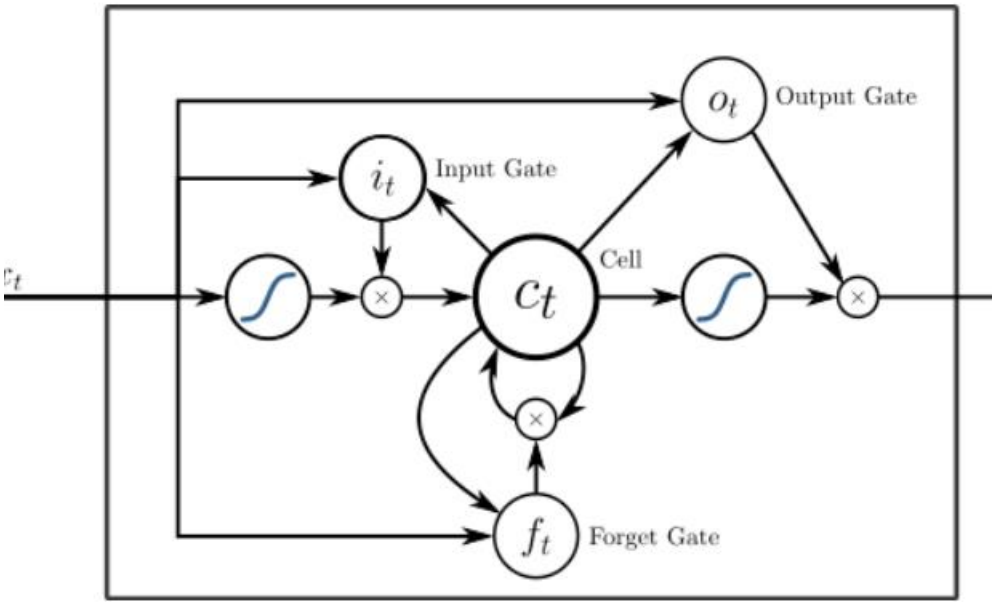
blizu nule, tzv. nestajući gradijent (engl. *vanishing gradient*), pri kojemu se ne događa učenje. Drugo, izraz može poprimiti iznimno velike vrijednosti, tzv. eksplodirajući gradijent (engl. *exploding gradient*), koji dovodi do nestabilnosti i divergencije. Za rješavanje problema nestajućeg gradijenta postoje dvije metode:

1. korištenje specifične povratne strukture, poput LSTM-a ili
2. ograničiti BPTT algoritam da gleda samo određeni broj prethodnih vrijednosti.

Problem eksplodirajućeg gradijenta rješava se metodom podrezivanja (engl. *gradient clipping*) kojom se određuje gornja granica apsolutne vrijednosti gradijenta za bilo koji parametar mreže. Svi gradijenti koji su veći od gornje granice postavljaju se na vrijednost gornje granice i učenje se nastavlja.

3.5.3.4. LSTM (engl. Long Short-Term Memory)

Da bi riješili probleme nestajućih i eksplodirajućih gradijenata i dugoročno bolje znali odnose u podacima o učenju koristimo povratna mreža s dugom kratkotrajnom memorijom, LSTM. LSTM koristi koncept ćelije, koja poput memorije pohranjuje podatke o stanju. Informacija može ostati sačuvana u vremenu ili može biti promijenjena u drugu s obzirom na operacije koje se odvijaju unutar četiri povratna sloja. Ove matematičke operacije nazivaju se vrata (engl. *gates*). Ulazna vrata (engl. *input gate*) odlučuje hoće li propustiti informaciju u trenutnom vremenu u ćeliju. Vrata zaborava(engl. *forget gate*) odlučuje hoće li ostaviti ili izbrisati trenutne sadržaje u ćeliji. Izlazna vrata(engl. *output gate*) odlučuje hoće li informaciju proslijediti dalje u mrežu. Zbog svoje uspješnosti u mnogim zadacima, LSTM povratne mreže su danas najčešće korišteni tip povratnih neuronskih mreža, posebice u sustavima za prepoznavanje govora. Dijagram ćelije i vrata LSTM mreže dan je slikom [Slika 19].



Slika 19. Dijagram ćelija i vrata LSTM mreže

3.6. Jezični model

U ovom potpoglavlju govorit će se o jezičnom modelu (engl. *language model*, LM) koji je komponenta sustava za prepoznavanje govora koji procjenjuje moguće vjerojatnosti riječi iz govora.[7] U osnovnoj jednadžbi sustava za prepoznavanje, koja je dana izrazom (3.3), P(W) predstavlja jezični model. Ideja jezičnog modela je da predviđa riječ prije nego je govornik izgovorio. Jezični model dodjeljuje najvjerojatniji slijed riječi koji se određuje prema gramatici i semantici te prema podacima za učenje[9]. Ne koriste se nefleksibilno programirana pravila gramatike i semantike zbog nepredvidivosti spontanog govora govornika. Iako postoje riječi kojima slično zvuči fon, ljudima ne predstavlja veliki problem prepoznati slijed riječi. Razlog tome je što ljudi prepoznaju kontekst rečenice i pretpostavljaju tu riječ koju nisu dobro čuli. Davanje konteksta sustavu za prepoznavanje je glavna svrha jezičnog modela.

3.6.1. N-gram model

Prvo je potrebno dodijeliti vjerojatnost svakom mogućem nizu riječi:

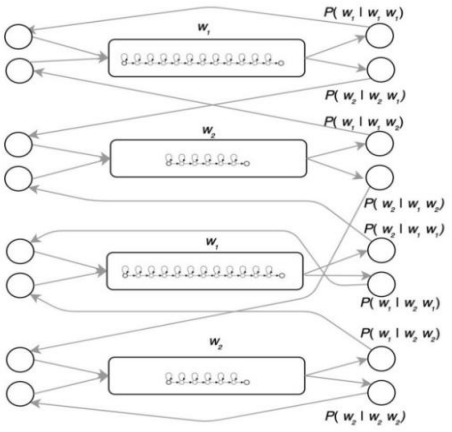
$W = w_1 w_2 \ldots w_n$, **(23)**

gdje je n broj riječi koji može biti neograničen, ali radi pojednostavljenja određujemo konačnu vrijednost koja je u osnovi rječnik sustava. Taj rječnik je u biti rječnik sustava za prepoznavanje govora. Riječ se ne može prepoznati ako ona nije u rječniku. Ako u govoru imamo riječi koje nisu u sustavu rječnika dolazi do pogreške (engl. *error*). Stoga treba imati rječnik da minimaliziramo šansu da se riječ ne prepozna. Očigledna strategija je odabrati riječi s najvećom vjerojatnošću da će se pojaviti, prema procjeni podataka učenja. Često postoji optimalna veličina rječnika koja balansira brzinu sustava s njegovom točnošću. Veliki rječnici povećavaju točnost, ali smanjuju brzinu dekodiranja riječi, katkad čak i ometaju rad akustičnog sustava što rezultira neželjenim pogreškama.

Čak i sa konačnim rječnikom i dalje postoji neograničen niz riječi tako da ne može se generirati jezični model grubim nadodavanjem vjerojatnosti svakoj rečenici. Umjesto toga koristimo lančano pravilo vjerojatnosti za niz određene duljine. Time je moguće zapisati rečenicu kao umnožak vjerojatnosti riječi, odnosno trenutnu riječ povezati s vjerojatnostima riječi koje su joj prethodile i koristeći Markovljeve lance ograničiti broj stanja.

$$P(W) = P(w_1)\ x\ P(w_2|w_1)\ x\ P(w_3|w_1w_2)\ x\ P(w_n|w_1 \ \dots \ w_{n\text{-}1}) \qquad \textbf{(24)}$$

Iako ovakav zapis odgovara Markovljevim lancima, u modeliranju jezika se koristi izraz N-gram model. Primjerice, bigram model predviđa sljedeću riječ samo prema prvoj prethodnoj, trigram[Slika 20] prema dvije prethodne i tako dalje. Generalno, N-gram model predviđa riječ prema N-1 prethodnoj riječi. U praksi se rijetko koriste N-grami veći od 4-grama ili 5-grama jer nisu pokazali značajan rast u točnosti.



Slika 20. Trigram model

Kako N-gram model dodaje vrijednost nizu riječi konačne duljine N, postavlja se pitanje kako predvidjeti kraj rečenice. U tu svrhu koristimo posebnu oznaku *end-of-sentence tag*</s> koja označava kraj niza i ubacujemo je u rječnik. Drugim riječima LM generira riječi sa lijevo na desno i staje kad naiđe na </s> oznaku. Također potrebno je ubaciti i oznaku <s> prije početka prve riječi w_1 čime se stvara kontekst za prvu stvarnu riječ. Ovo je korisno jer postoje riječi koje se učestalo pojavljuju na početku rečenica; primjerice upitne rečenice u hrvatskom najčešće će početi prilogom, upitnom zamjenicom ili upitnom česticom. Gotova vjerojatnost rečenice prema bigram modelu sada glasi:

$$P(W) = P(w_1|< s >) \times P(w_2|w_1) \times \ldots \times P(w_n|w_{n-1}) \times P(< /s >|w_n) \quad \textbf{(25)}$$

Uvjetna vjerojatnost N-grama se jednostavno može procijeniti prema učestalosti ponavljanja riječi. Općenito, za k-gram model vrijedi:

$$P(rk\,|r1\ \ldots\ rk-1) = \frac{c(r1 \ldots rk)}{c(r1 \ldots rk-1)}, \quad \textbf{(26)}$$

gdje je $c(r1 \ldots rk)$ broj ponavljanja niza duljine k.

Relativna učestalost ponavljanja kao procjena vjerojatnosti ima jedan veliki nedostatak: pridodaju vrijednost 0 za svaku riječ koja nije razmatrana u podacima za učenje (brojač u jednadžbi postane 0). Kako su podaci za učenje konačne veličine, ali same mogućnosti govora nisu, niti jedna kombinacija niza riječi ne bi smjela biti nemoguća. Postupkom uglađivanja jezičnog modela (engl. *language model smoothing*) svakom neopaženom N-gramu pridodaje se vjerojatnost veća od nule. To se može predočiti tako da se N-gram zamisli kao rupa u modelu koju treba ispuniti riječju. Postoji puno metoda uglađivanja jezičnog modela, a najpoznatija od njih je Witten-Bell uglađivanje.

Princip tog modela je tretirati neviđene riječi kao zaseban događaj koji se uračunava sa viđenim riječima. Svaki put kad se za vrijeme učenja vidi nepoznata riječ broji se kao nova riječ. Jednadžba točnog iznosa vjerojatnosti pojave prepoznatih riječi glasi:

$$P(w_k|w_1\ \ldots\ w_k-1) = \frac{c(w_1 \ldots w_k)}{c(w_1 \ldots w_k-1)+V(w_1 \ldots w_{k-1})}, \quad \textbf{(27)}$$

gdje je c(w_1… w_k) zbroj svih riječi(duljina teksta u učenju), a V je vokabular broj 'prvo viđenih riječi'. Dodatna vrijednost V u nazivniku upućuje na to da smanjuje vrijednost viđenih nizova riječi i raspodjeljuje po neviđenim nizovima s obzirom na broj prvi put viđenih riječi. To se generalizira u N-gramu dužine k tako da prvu k-1 riječ predstavimo ka kontekst za zadnju riječ.

Druga metoda uglađivanja je *back-off* metoda koja vraća u kraću verziju konteksta tj. smanjuje N-gram za jedan kad se riječ da bi se povećala vjerojatnost pronalaska tražene riječi.

To se može predočiti kao:

$$P_{bo}(w_k|w_1 \dots w_{k-1}) = \begin{cases} P(w_k|w_1 \dots w_{k-1}), c(w_1 \dots w_k) > 0 \\ P_{bo}(w_k|w_2 \dots w_{k-1})\alpha(w_2 \dots w_{k-1}), c(w_1 \dots w_k) = 0 \end{cases} \quad \textbf{(28)}$$

P_{bo} je nova procjena za sve N – grame. Ako su N-grami uočeni, onda se koristima procjenom P. Ako nisu uočeni u fazi učenja onda se procjenjuje skraćenim kontekstom(izostaje se w_1) i množi sa α koja je funkcija konteksta tako da zbroj procjena svih riječi w_k bude jedan. α je vjerojatnost neviđenih riječi u kontekstu w_1... w_{k-1}, podijeljena zbrojem istih vjerojatnosti neviđenih riječi prema raspodjeli P_{bo} (· | w_2... w_{k-1}).

3.6.2. Ocjenjivanje jezičnog modela

Pri razmatranju dva modela postavlja se pitanje koji je bolji. Ako jedan model daje veće vjerojatnosti pronađenih riječi od drugog, onda je taj model bolji. To se određuje na sljedeći način:

$$\log P(w_1 \dots w_n) = \log P(w_1| < s >) + \log P(w_2|w_1) + \dots + \log P(< /s > | w_n)\,, \quad \textbf{(29)}$$

gdje log P predstavlja vjerojatnost (engl. *likelihood*). Vjerojatnosti su uvijek negativne jer je ukupni broj vjerojatnosti u fazi učenja (w_1 … w_n) manji od 1. Model se obavezno testira na podacima koji su neovisni o podacima za učenje. Prilikom ispitivanja vjerojatnosti gleda se kako je vjerojatnost raspoređena po viđenim i neviđenim riječima te nizovima riječi. Entropija modela računa se prema izrazu:

$$-\frac{1}{N}\log P(w_1 \dots w_n) \quad , \quad \textbf{(30)}$$

gdje je N broj riječi u rječniku sustava.

Entropija predstavlja mjeru informativnosti niza riječi, odnosno daje prosječan broj bitova potrebnih za rad s određenim nizom riječi. Kompleksnost (engl. *perplexity*) je recipročna vrijednost srednje vjerojatnosti raspodijeljene po riječima tj. broj riječi u rječniku s istom vjerojatnošću pojavljivanja. Za računanje kompleksnosti koristi se geometrijska sredina jer su vjerojatnosti povezane množenjem, a ne zbrajanjem. Prema tome kompleksnost je suprotna entropiji:

$$P(w_1 \dots w_n)^{-\frac{1}{n}} \tag{31}$$

Katkada je potrebno ograničiti veličinu jezičnog modela, posebice zato što model raste gotovo linearno s brojem riječi uporabom N-grama. Neki se modeli ciljano smanjuju do određene veličine tako da manje važni parametri budu izbačeni i stvori se višak prostora. To se računa se s obzirom na entropiju ili kompleksnost. Za svaki N-gram računa se koliko on utječe na ukupnu entropiju i kompleksnost sustava te ako je razlika ispod nekog predodređenog praga, parametar se odbacuje. Nakon što se iz modela izbace nepotrebni parametri, potrebno je ponovo normalizirati vjerojatnosti preostalih parametara. Odnos karakteristika i kvalitete jezičnog modela prikazan je u tablici [Tablica 2]:

Tablica 2. Značajke dobrog i lošeg jezičnog modela

	vjerojatnost	entropija	kompleksnost
DOBAR MODEL	velika	mala	mala
LOŠ MODEL	mala	visoka	visoka

U slučaju da se nađe dva jezična modela koja su već prošla fazu učenja koji već imaju vjerojatnost procjene P_1 i P_2 sa tim da jedan ima više podataka iz učenja od drugog postavlja se pitanje kako kombinirat modele za bolju procjenu N-grama vjerojatnosti. Ne može se izvući podaci učenja iz modela i koristiti za novi zajednički jer potencijalno jedan model može imati više podataka iz učenja. U tu svrhu se koristi metoda interpolacijske procjene. Model interpolacijske procjene je veoma učinkovit u smanjenju ukupne kompleksnosti bez potrebe prikupljanja podataka iz faze učenja.

Ona ukazuje da je bolje kombinirati postojeće modele na razini vjerojatnosti. To znači da se pomnoži prosječni parametri u procjeni vjerojatnosti:

$$P(w_k|w_1 \ldots w_{k-1}) = \lambda\, P_1(w_k|w_k \ldots\, w_{k-1}) + (1 - \lambda)\, P_2(w_k|w_1 \ldots w_{k-1}) \quad \textbf{(32)}$$

Parametar λ kontrolira relativni utjecaj komponenti modela. Vrijednost blizu 1 znači da dominira prvi model dok vrijednost blizu 0 daje veći dio težine drugom modelu. Metoda interpolacijske procjene se koristi i na slučajevima sa više modela (λ_1, λ_2, … λ_M).

3.6.3. Napredne tehnologije jezičnog modela

Ovo potpoglavlje govori o dvije naprednije tehnike u modeliranju jezika na visokoj razini koje se danas široko koriste u praksi. Metode u modeliranju jezika neprestano se razvijaju, a istraživanja u ovom području vrlo su aktivna.

3.6.3.1. Jezični model baziran na klasam riječi

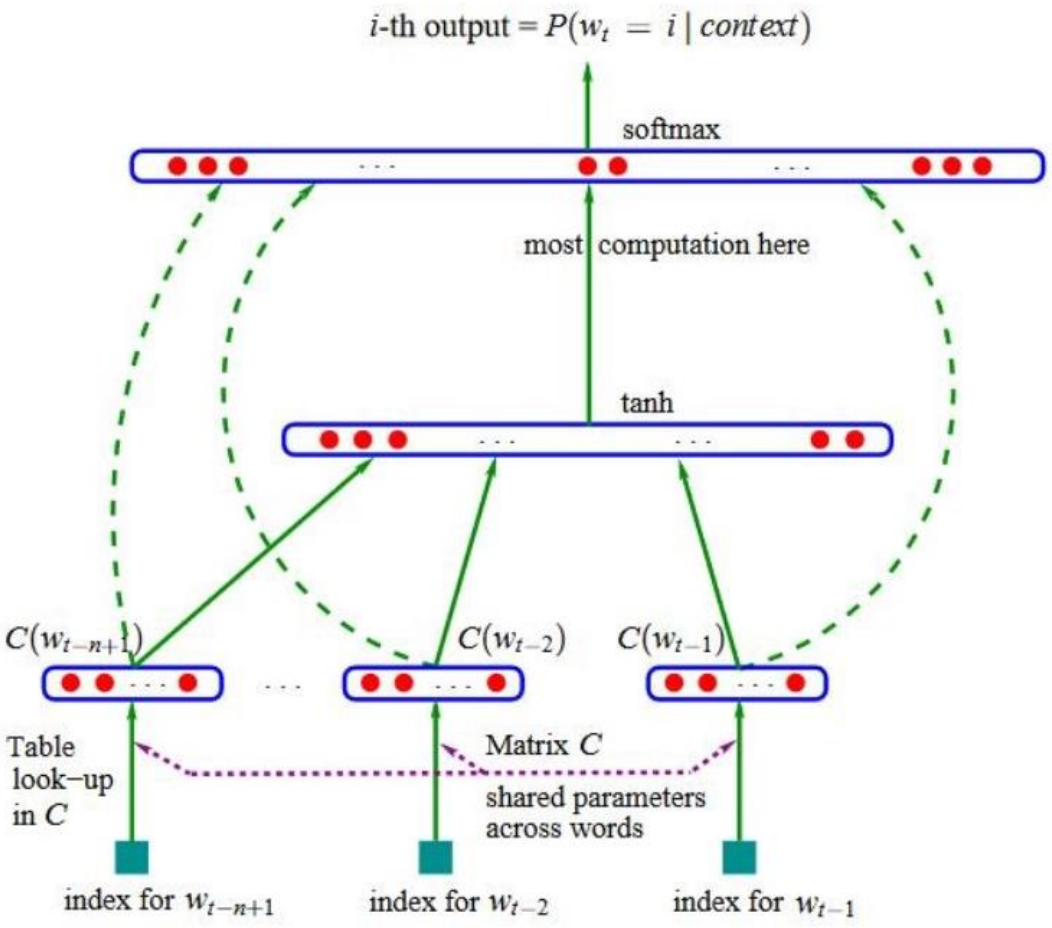
Jedan od velikih nedostataka N-grama je što sve riječi tretira u potpunosti različito. Tek nakon što se model dovoljno puta u fazi učenja susretne s određenom riječi, može naučiti sa kojim N-gramima se pojavljuje. Tako ljudi ne koriste jezik, već povezuju riječi ne samo prema sintaksama, nego i prema značenjima. Sličnost među riječima može se iskoristiti za generaliziranje jezičnih modela. Na tom principu su građeni jezični modeli bazirani na klasama riječi koji određene skupine riječi klasificiraju. Korištenjem klasa riječi smanjuje se broj N-grama. Ovaj koncept grupira riječi u klase riječi, stoga N-grami više nisu čiste riječi nego oznake za riječi. Primjerice ako definiramo klasu 'Mjeseci u godini', u kojoj se nalazu svi mjeseci(siječanj, veljača, ožujak, … , prosinac), N-gram se više neće ponašati tako da otvara točan mjesec npr. 'veljača' nego samo oznaku klase koja je u ovom slučaju 'Mjesec u godini'. N-gram se ponaša tako da može dohvatiti bilo koju riječ u klasi. Unutar klasa riječi računa se vjerojatnost pojave određene riječi u klasi.

3.6.3.2. Jezični model s umjetnim neuronskim mrežama

Eksperimenti su pokazali da umjetne neuronske mreže osmišljene za jezične modele daju daleko bolje rezultate od klasičnih N-gram modela, ako imaju dovoljno podataka za učenje.

Za razliku od N-gram modela, jezični modeli s umjetnim neuronskim mrežama sposobni su generalizirati riječi.

Prva korištena umjetna neuronska mreža u jezičnom modelu bila je unaprijedna neuronska mreža koja je uspješno rješavala problem generalizacije pomoću smještaja riječi (engl. *word embedding*). Princip rješavanja ovom mrežom je dan na slici [Slika 21].

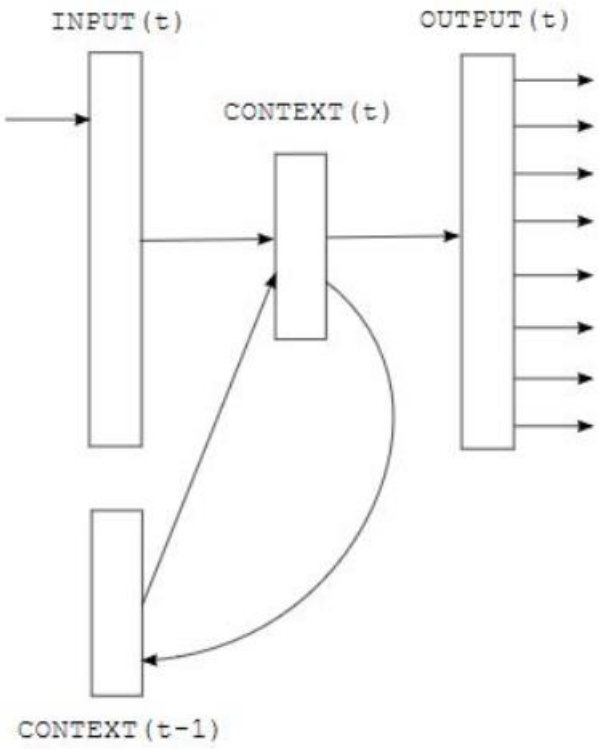


Slika 21. Shematski prikaz unaprijedne neuronske mreže korištene u jezičnom modelu[17]

Ulaz u mrežu je vektorski zapis N - 1 riječi koja tvori N-gram. Izlaz je vektor vjerojatnosti pretpostavljanih riječi. U oba slučaja su vektori dužine koje određuju za veličinu rječnika te se stoga lako mogu implementirati u sustave koji su do tada koristili jezični model baziran samo na N-gramima. Ulazne riječi se unutar mreže pretvaraju preko višedimenzionalne matrice u vektore ulaza. Kontekstualno slične riječi koje na isti način utječu na kontekst rečenice spremaju se blizu jedna drugoj unutar prostora matrice. Na taj način kontekstualno slične riječ

imaju slične vjerojatnosti u izlaznom vektoru, te se lakše prepoznaju u novim neviđenim nizovima riječi.

Još jedno ograničenje N-gram modela, kojeg su umjetne neuronske mreže riješile, je skraćivanje konteksta. U N-gram modelima riječ ovisi isključivo o N-1 prethodnih riječi. To je problem jer jezik dopušta gotove riječi, koje su kontekstualno povezane. N-gram model nema mogućnosti predvidjeti sve riječi u kontekstu. Ovaj nedostatak ispravljen je uporabom povratnih neuronskih mreža koje u trenutku t-1 aktiviraju funkcije skrivenog sloja kao ulaz u idući korak procesa u vremenu *t* kao što je prikazano na slici [Slika 22].Slika 22



Slika 22. Shematski prikaz rada povratne neuronske mreže u jezičnom modelu[18]

To omogućuje mreži da prosljeđuje informacije s jednog položaja riječi na drugo, bez strogog ograničenja u koliko dalekom vremenu potječu informacije koje se mogu koristiti za predviđanje trenutne sljedeće riječi. Postoje određeni matematički problemi koji se javljaju pri učenju ovih mreža, no i oni se uspješno rješavaju uporabom LSTM neuronskih mreža.

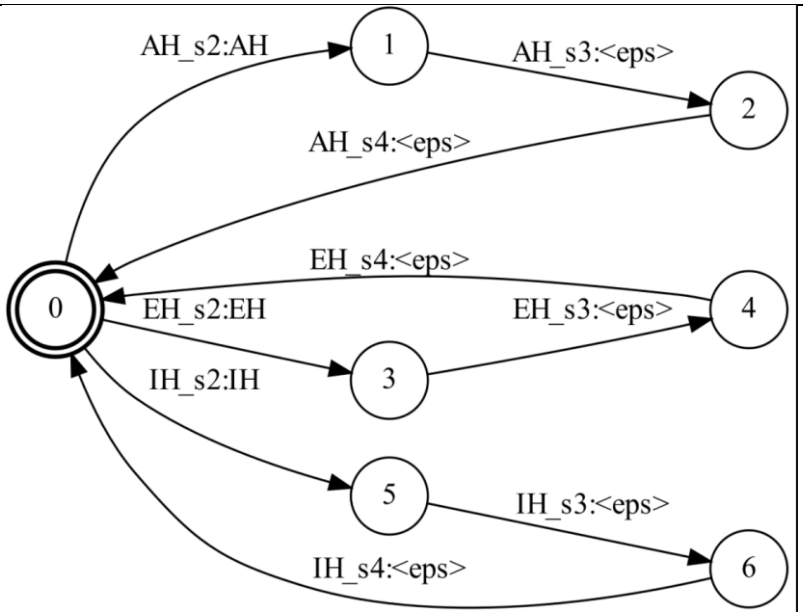
3.7. Dekodiranje govora

Akustični model na ulaz dekodera sustava za prepoznavanje govora šalje zapis o vjerojatnostima mogućih fonema unutar jednog segmenta. Jezični model šalje na ulaz dekodera zapis o vjerojatnostima mogućih nizova riječi. U fazi dekodiranja, zadatak je usporediti dobivene podatke i pronaći najvjerojatniji slijed riječi W s obzirom na niz opservacija O i akustičko-jezični model te proslijediti na izlaz sustava.

Viterbi algoritam se koristi pri dekodiranju jednostavnih HMM modela i malih rječnika. S druge strane pri pojavi velikog rječnika u ASR-u stvari postaju puno kompliciranije. Govor je neprekidna radnje gdje riječi ne poznaju granice, a sam prostor za traženje riječi je velik. U ovom poglavlju se opisuje koncept konačnog pretvornika s parametrima WFST (engl. *Weighted Finite-State Transducers*)[19]. On radi na način da pretvara više konačnih pretvornika i automata u jedan sustav i tako traži gramatički ispravnu rečenicu. WFST se sastoji od HMM modela, fona ovisnih o kontekstu, izgovorima i gramatici. Dekodiranje se modelira pomoću grafa te najvjerojatniji niz slova, odnosno riječi, pronaći pretraživanjem tog grafa. Za pretvorbu ulaznih podataka u graf najčešće se koriste konačni automati (engl. *finite state automata*, FSA) i konačni pretvornici (engl. *finite state transducers*, FST) te njihove težinske verzije. Konačni automati grafički preoblikuju nizove te ih je potom lakše pretraživati. Sastoje se od konačnog broja stanja, prijelaza između stanja i mogućih radnji između stanja. Konačni pretvornici mogu se predstaviti kao konačni automati s dvije trake između kojih se vrši preslikavanje iz jednog skupa znakova u drugi.

Rječnik izgovora je baza fonetskih zapisa svih riječi u rječniku sustava koja povezuje izgovor s napisanim oblikom riječi. Riječ je predstavljena nizom slova, koja su predstavljena nizom fonema, a niz fonema je predstavljen nizom akustičnih svojstva. Ulaz u rječnik izgovora je niz provjerenih fonema, a izlaz je napisana riječ. Određena riječ može imati veliki broj izgovora ovisno o govorniku. Uzrok tome može biti naglasak ili brzina govora. Nerijetko prilikom brzog pričanja, ljudi izostave fonem, ali sam smisao riječi i ona sama se ne mijenjaju zbog toga. Iz tog razloga rječnik izgovora ima nekoliko mogućih izgovora koji predstavljaju istu napisanu riječ. Svaki proces dekodiranja se sastoji od tri djela.

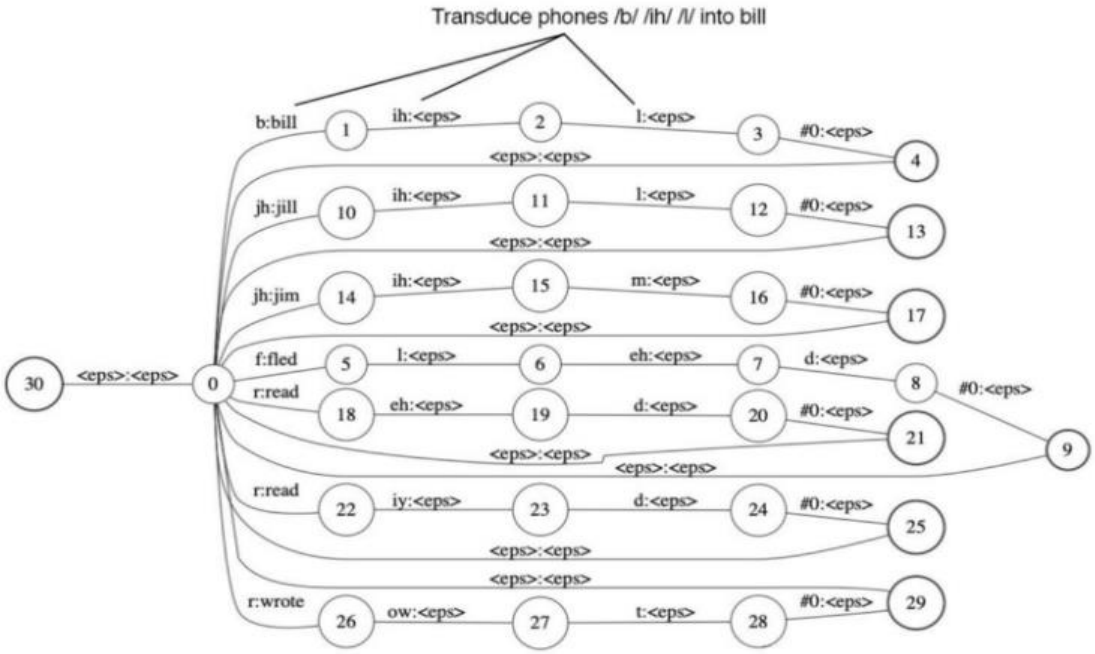
Prvi na redu dolazi HMM konačni pretvornik koji preslikava niz HMM stanja s oznakama senona u HMM modele s oznakama trifona. Struktura HMM pretvornika je dana na slici [Slika 23].



Slika 23. Shematski prikaz H konačnog pretvornika

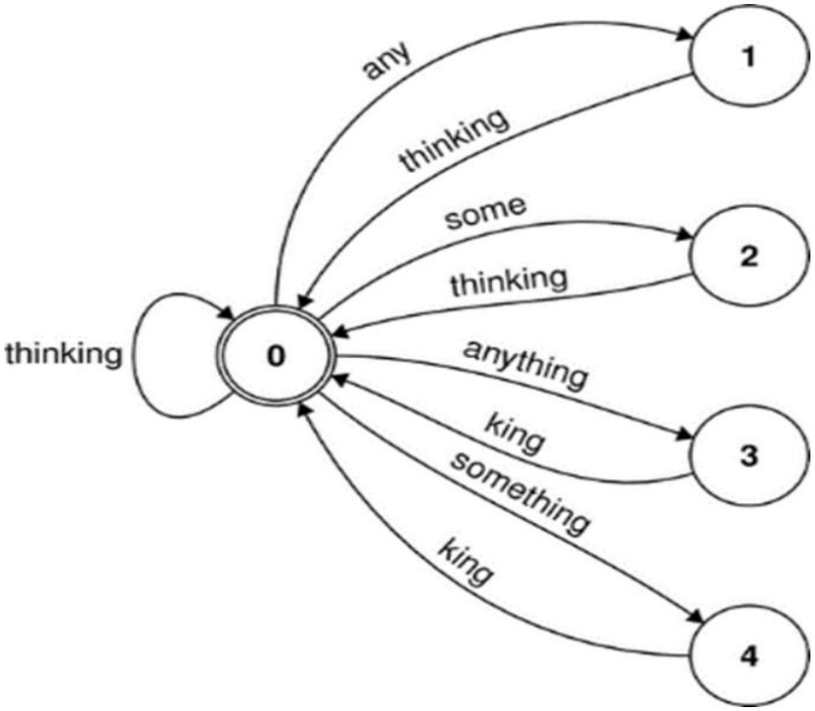
Svaki model fona sastoji se od petlje, gdje se nazivi akustičkog stanja, poput AH_s2, javljaju na ulaznoj strani, a odgovarajuća imena fona, poput AH, javljaju se na izlaznoj strani.

Zatim slijedi konačni pretvornik za rječnik izgovora(engl. *pronunciation lexicon*) L prikazan na slici [Slika 24] dekodira niz fona u riječ u akustičnom modelu pomoću konačnog pretvornika.



Slika 24. Shematski prikaz L konačnog pretvornika

I za kraj slijedi konačni automat G. Dijagram koji prikazuje konačni pretvornik G za pretvorbu riječi na temelju gramatike u jezičnom modelu prikazan je na slici [Slika 25].



Slika 25. Shematski prikaz G konačnog automata

Dijagram se sastoji od ulaza, izlaza i riječi između koje nalaze na putanji. Slika 25 upućuje na neka jednostavna pravila pri izradi konačnog pretvornika G:

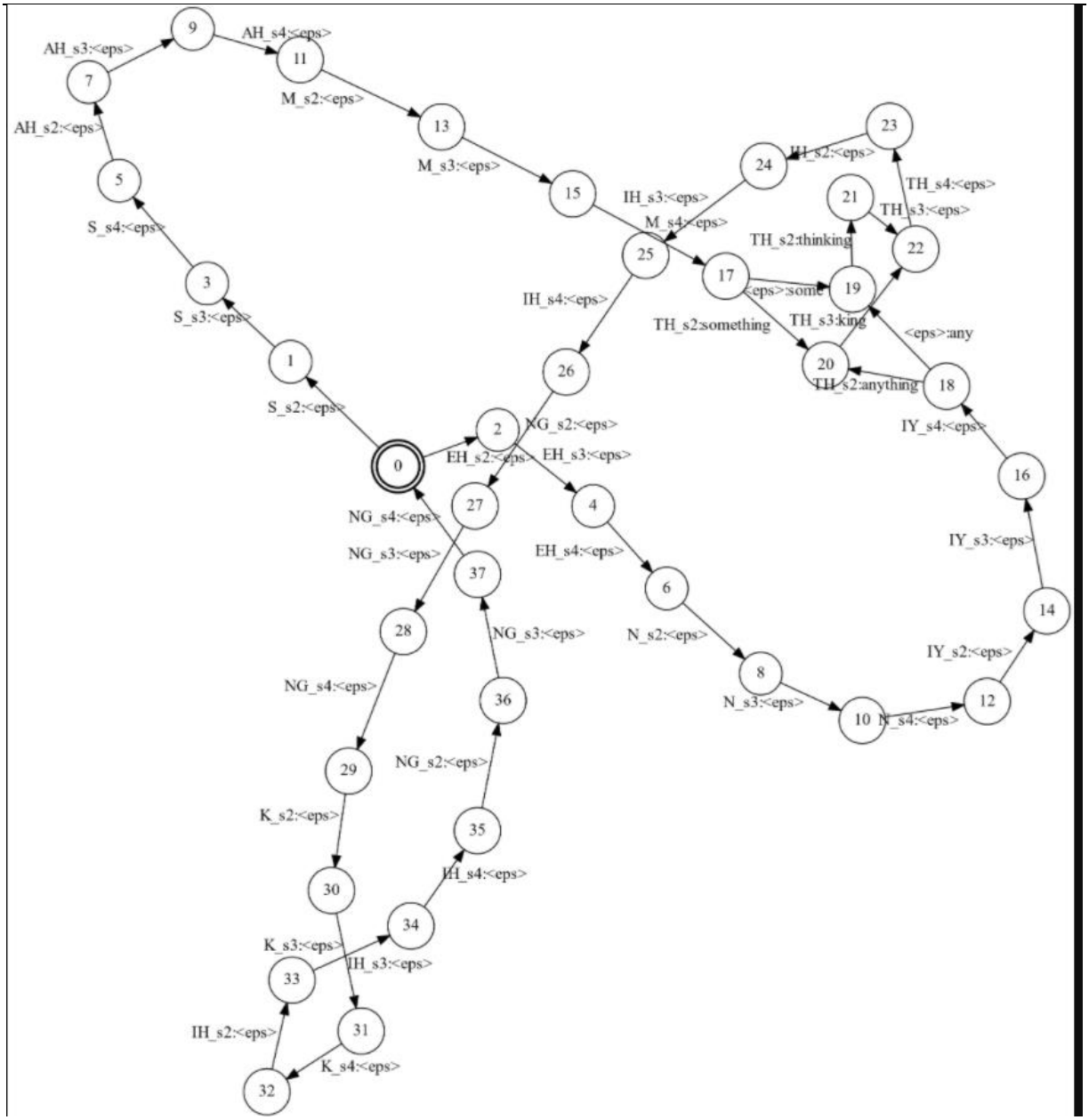
- Sve staze u početnom stanju započinju s oznakom 0, a završavaju u bilo kojem stanju s dvostrukim krugom,
- Lukovi između stanja su usmjereni i označeni jednom riječju iz rječnika,
- Pretpostavlja se da su parametri putanje jednaki nuli, osim ako nije drugačije određeno.
- Akceptori imaju jedan simbol

Graf dekodiranja objedinjuje nizove stanja akustičnog modela i nizove riječi. Većina modernih sustava za prepoznavanje govora koristi četiri konačna automata i pretvornika koji su predočeni tablicom [Tablica 3].

Tablica 3. Slijed dekodiranja(odozdo prema gore)

	pretvornik	ulaz	izlaz
G	gramatika	riječi	riječi
L	Rječnik izgovora	foni	riječi
C	Kontekstna ovisnost	Foni ovisni o kontekstu	foni
H	HMM	HMM stanja	Foni ovisni o kontekstu

Pretvornici se moraju koristiti redom kojim su prethodno navedeni tvoreći tako HCLG graf. Na slici [Slika 26] simbol „< eps >“ predstavlja dio niza gdje nema simbola. Na primjer, ako govornik izgovori dugo „a“ u nekoj riječi, zapis će izgledati kao „a< eps >“, umjesto „aaaaa…“. Pretraživanje grafa svodi se na pronalaženje najkraćeg puta za što se koristi algoritam širinskog pretraživanja (engl. *beam search algorithm*).



Slika 26. HCLG graf

3.8. *End-to-end* sustav za automatsko prepoznavanje govora

U ovom potpoglavlju se opisuje princip *end-to-end* tehnologije za prepoznavanje govora kao i razlike između *end-to-end* tehnologije sa tradicionalnim metodama prepoznavanja govora.[20]

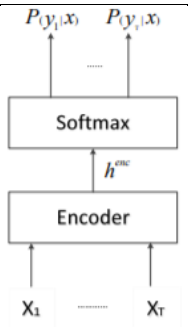
End-to-end je sustav koji izravno preslikava niz ulaznih zvučnih značajki u niz riječi. Za većinu poznatih ASR sustava potrebne su odvojene faze učenja akustičnog i jezičnog modela. Za odabir rječnika izgovora i definiranje niza fonema za određeni jezik potrebna su vješta znanja i vremenski je dugo kao što je objašnjeno u prethodnim potpoglavljima. *End-to-end* značajno pojednostavljuje kompleksnost ASR. Ideja je zamijeniti akustični i jezični model te rječnik izgovora s jednom dubokom povratnom neuronskom mrežom što je prikazano na slici [Slika 27].

Slika 27. Model end-to-end

Na taj način sustavi ne bi trebali biti prilagođeni određenom jeziku, već bi univerzalni sustav direktno iz zvučnog zapisa prepoznavao nizove slova bez potrebe za fonetskim preslikavanjem. Time bi se uklonila potreba za lingvističkim ekspertima prilikom stvaranja sustava za prepoznavanje govora.

Postoje više oblika *end-to-end* prepoznavanja od kojih je najpoznatiji (engl. *Connectionist Temporal Classification*).

U CTC-u za vrijeme faze učenja akustični modeli se uče bez segmenata signala između zvuka i ispisa riječi. Učenje akustičnog modela koji koristi CTC kao funkciju ne treba podatke unaprijed, već samo treba učiti ulazni niz i izlazni niz. Struktura je prikazana na slici [Slika 28].



Slika 28. CTC struktura

Ovim načinom nije potrebno označiti svaki podatak jedan po jedan i vjerojatnost izlaznog niza u CTC-u ne treba vanjsko induciranje. Koriste se prazni prostori (koji nema vrijednost segmenta), gdje svaka klasifikacija predviđanja odgovara riječju u govoru, a druga mjesta koja nemaju riječi smatraju se praznima. Izlaz je niz riječi neovisno o trajanju fonema. U slučaju da imamo x riječi, vjerojatnost da će izaći riječ y dana je formulom:

$$P(y|x) = \sum_{y \in B(y,x)} \prod_{t=1}^{T} P(y_t|x) \tag{33}$$

End-to-end sustavi aktualna su tema istraživanja i na njima se intenzivno radi. Zasada još uvijek ne mogu konkurirati sustavima s jezičnim modelom i rječnikom izgovora.

4. PRIMJENA SUSTAVA ZA PREPOZNAVANJE GOVORA

Sustav za prepoznavanje govora se gotovo ukorijenio u suvremeno društvo zahvaljujući jednostavnosti korištenja i olakšane mogućnosti potrebnih radnji. Čovjek može izreći 150 riječi u minuti, dok napisati u istom vremenu samo 40 riječi. Prepoznavanje govora nije jednostavna stvar. Još je to grana koja se razvija jer u određenim situacijama ASR krivo prepozna riječ. To su najčešće situacije kad je izgovor rečen dijalektom ili ako osoba ima govornu manu, ako više osoba priča od jednom ili ako je velika buka ili ako nije rečeno na engleskom. Razvijanje preciznosti ASR-a dovelo je do široke upotrebe glasovnih asistenata među ljudima. U uvodu je dan primjer samo jedne od mnogih upotreba ASR-a. U ovom poglavlju će se nabrojati par primjera primjene u svakodnevnom životu a neke će se detaljnije opisati [21]. Neke od primjena su:

- U medicini se pomoću samog pacijentovog govora prepoznati bolesti,
- U vojnoj tehnologiji se može u borbenim avionima govorom postavljati: sustav autopilota, koordinate određene lokacije, i aktiviranje raketa za ispaljivanje,
- Neke video igrice se mogu igrati bez uporabe kontrolera,
- Za vrijeme vožnje zrakoplovom, pilot umjesto da traži smjernice od osobe iz zračne luke on može to tražiti od govornog asistenta.

4.1. Appleova Siri

Prvi primjer govornog asistenta predstavljen je 2011, od strane tehnološke tvrtke 'Apple'. Od tada implementirana je u svi njihove proizvode kao što su: iPhone, iPad, AppleWatch, HomePod[Slika 29], Mac računala, Apple TV itd. Preko mobitela je moguće koristiti Siri preko upravljačke ploče automobila (Apple's CarPlay). Pošto je prva inačica koja je izašla Siri se suočila sa mnogo tehničkih problema. Dolazilo bi do problema u prepoznavanju. Kada treba poslati poruku ili obaviti poziv ne nailazi poteškoće ali ako je zahtjev kompleksniji Siri nekad zakaže. Siri radi uspješno samo sa šest vrsti aplikacija: vožnja, poruke i pozivi, galerija, plaćanja, fitness i informativne obavijest.

Sirijeva prednost je to što ima širok mogućnost prepoznavanja u više jezika.

Slika 29. Apple HomePad

4.2. Amazonova Alexa

Uređaji koji koriste Alexu su: Amazon Echo[Slika 30] i Echo Show(govorom kontrolirani tablet)

Slika 30. Amazon Echo

Amazon je usavršio Alexu za kompleksnije stvari od Siri. U početku je Alexa bila na lošem glasu jer je slabo prepoznavala govor, ali Alexi treba vrijeme da se prilagodi na čovjekov glas. Alexa je na mnogo načina moćnija od Siri. Omogućuje integraciju sa pametnim uređajima u kući kao što su kamere, brave u vratima, svjetla, termostati itd. Kada se upita Siri da odradi kupovinu ona samo doda željeni proizvod u shopping listu, dok Alexa obavu kupovinu. Nedostatak Alexe je mala mogućnost izbora jezika(engleski i njemački samo).

4.3. Microsoftova Cortana

Cortana je Microsoftov digitalni osobni asistent predstavljen prvi put 2014. godine kao dio Windows Phone 8.1. Naziv Cortana potječe iz Microsoftovog serijala igara Halo, gdje je lik s

istim imenom također osobni digitalni asistent. Kasne 2017 Microsoft je obznanio da je postotak prepoznavanja greški svega 5.1%. Suparnički glasovni asistenti idu duboko u podatke s uređaja, pregledavajući korisnikovu povijest pretraživanja na internetu. Iako je to često korisno, može biti i neugodno u obliku neprestanih obavijesti ili jednostavno zastrašujuće što pametni sustav može toliko puno znati o vama. Cortana koristi tzv. bilježnicu za spremanje svih informacija prikupljenih o korisniku tijekom korištenja (npr. ponašanje u određenoj okolini, podsjetnike, alarme, kontakte, podatke o lokaciji, itd.). Daljnjim korištenjem Cortane, ona može automatski dodavati i brisati podatke o korisniku, a korisnik može i sam ukloniti neke netočne informacije. Slično Amazonu, i Microsoft je objavio vlastiti kućni pametni zvučnik Invoke[Slika 31] koji izvršava mnoge iste funkcije kao i njihovi suparnički uređaji. Microsoft ima još jednu ogromnu prednost kada je u pitanju doseg tržišta. Cortana je dostupna na svim Windows računalima i mobitelima koji rade na Windowsu 10.

Slika 31. Microsoft Invoke

4.4. Google Assistent

Od traženja prijevoda određenih fraza, do računanja matematičkih operacija Google Assistent ne samo da točno odgovara, već daje i dodatni kontekst te za informaciju navodi izvornu web adresu. Google Home[Slika 32], koji je sličan Amazonovom Echu je predstavljen krajem 2016. godine i već za godinu dana se nametnuo kao veliki suparnik ostalim konkurencijama. Krajem 2017. Google se pohvalio s 95% točnosti riječi za američki engleski; najviši od svih glasovnih asistenata koji su trenutno tamo. To znači da stopa pogrešaka riječi iznosi 4,9% - što Google čini prvim iz grupe koji je pao ispod praga od 5%.

Slika 32. Google Home

4.5. Sustav za prepoznavanje govora u automobilima

Uređaji s glasovnim aktiviranjem i digitalni glasovni asistenti nisu samo olakšavanje stvari.

Također je riječ i o sigurnosti - barem je tako kada je riječ o prepoznavanju govora u automobilu. Tvrtke kao Apple i Google rade na tim sustavima sa ciljem da vozač manje gleda po mobitelu za vrijeme vožnje. Umjesto da vozač tipka na mobitelu može reći automobilu da napravi radnju koja mu treba(obaviti poziv ili preko Google Karti pronaći lokaciju). Ako je automobili pri kraju za benzinom govorni asistent obavještava vozača i navodi ga do najbliže benzinske stanice. Kako je ovo nova tehnologija ljudima se teško naviknuti na tu novinu pa potencijalno može doći do još veće distrakcije za vrijeme vožnje. Ali kako svijet napreduje, za očekivat da će i ljudi u skladu sa tim koristiti sve benefite tehnologije.

5. MODULARNA APLIKACIJA ZA PREPOZNAVANJE GOVORA

U ovom poglavlju će biti objašnjena izrada modularne aplikacije u programu Python i rezultat bi trebao biti prepoznat govor tako da bude napisan tekst zvučnih datoteka dobivenih preko mikrofona ili audio datoteke. Za početak je objašnjen kod na jednostavnijim primjerima a kasnije cijela aplikacija sa rezultatima. Kasnije je obrađena aplikacija za pretvorbu koja sadrži glasovnu asistenticu imenom "Plaea".

5.1. Podloga za rad

Softversko rješenje ovog rada realizirano je u programskom jeziku Python. Python je interpreterski, interaktivni, objektno orijentirani programski jezik. Razvio ga je Guido van Rossum 1990. godine. Iz razloga što je interpreterski jezik, znatno je sporiji u odnosu na C i C++. Razlog zašto će se koristiti u ovom zadatku, ali i zašto se općenito koristi za zadatke ovoga tipa, je zbog velikog broja paketa specijaliziranih za različite projekte. Python kôd za potrebe ovog rada pisan je unutar PyCharm-a. PyCharm je besplatna interaktivna web-aplikacija koja omogućuje stvaranje i razmjenu dokumenata koji sadrže računalni kôd i bogate tekstualne elemente (odlomak, jednadžbe, slike, poveznice itd.) i specijalizirana je samo za Python. Na taj način, izrađeni programi su istovremeno izvršni dokumenti koji se mogu pokrenuti za analizu podataka i čitljivi dokumenti koji sadrže opise analiza i rezultate.

5.2. Izbor potrebnih biblioteka

Prije samog početka programiranja potrebna je priprema biblioteke koji će se koristiti. Za prepoznavanje govora postoji više biblioteka[22]. Neke od njih su :

- apiai
- assemblyai
- google-cloud-speech
- pocketsphinx
- SpeechRecognition
- Watson-developer-cloud
- Wit

U ovoj aplikaciji je korišten SpeechRecogniton koji je najpopularniji i najlakša biblioteka za prepoznavanje govora.

Prikaz svih nama potrebnih paketa dan je na [Slika 33]:

```
import speech_recognition as sr
import webbrowser as wb                 # link za internet
import time                             # za sleep koji nam služi da postavimo periodično vrijeme između dva razgovora
import playsound                        # da ne ode na druge filove kao npr. iTunes..
import os                               #  brisenja audia u direktoriju
import random
from gtts import gTTS                   # da napravi audiofile Plaea
from time import ctime                  # trenutno vrijeme
from configparser import ConfigParser   #  da može prepoznati API kljuc stranice OpenWheatherMap kojeg smo smjestili u file config.ini
import requests
from googletrans import Translator      # za prevođenje jezika
import datetime                         # paket za odredivanje sati i minuta pri postavljanju alarma
```

Slika 33. Potrebni paketi za rad aplikacije

Svi paketi se instaliraju na jednak način, a to je upis u terminal 'pip install [potrebni paket]'. [Slika 34]

```
Terminal:  Local  +
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019. Microsoft Corporation. Sva prava pridržana.

(venv) C:\Users\Korisnik\Desktop\python\zavrsni\speech recognition>pip install SpeechRecogniton
```

Slika 34. Pip install

5.3. Odabir potrebne metode

Sva čarolija prepoznavanja leži u *Recognizer* klasi koja se nalazi u SpeechRecognition paketu. Ona služi za samu pretvorbu. Njenom inicijalizacijom u varijabli r se postiže olakšano kodiranje[Slika 35.]

```
r = sr.Recognizer()  # inicijaliziramo klasu Recognizer()
```

Slika 35. Inicijalizacija Recognizer klase

Klasa Recognizer ima sedam metoda za pretvorbu govora iz zvučnih datoteka koristeći različiti API(aplikacijsko programsko sučelje). To su :

- recognize_bing(): Microsoft Bing Speech
- recognize_google(): Google Web Speech API
- recognize_google_cloud(): Google Cloud Speech
- recognize_houndify(): Houndify
- recognize_ibm(): IBM Speech to Text
- recognize_sphinx(): CMU Sphinx
- recognize_wit(): Wit.ai

Od sedam metoda samo recognize_sphinx radi offline, dok svi ostali moraju imati Internet vezu. Pošto ne treba API ključ za Google Web Speech API odabrana je metoda recognize_google(). Za ostalih šest metoda potrebna autentifikacija sa API ključem ili kombinacijom korisničkog imena i lozinke.

5.4. Prepoznavanje govora iz zvučnih datoteka

SpeechRecogniton lako radi sa zvučnim zapisima(engl. *audio file*) koristeći *AudioFile* klasu. SpeechRecognition podržava sljedeće formate zvučnih datoteka:

- WAV
- AIFF
- AIFF-C
- FLAC

Za pretvorbu iz zvučne datoteke prvo moramo inicijalizirati *AudioFile* klasu u neku varijablu('filename'). U AudioFile smo stavili neku zvučni datoteku('speech.wav'). Program otvara datoteku i čita sadržaj i pohranjuje te podatke u drugu varijablu ('source') radi jednostavnosti rada. Zatim, record() metoda snima podatke iz čitave datoteke u treću varijablu('audio'). Zatim pokrenemo metodu recognize_google(), u koju stavimo treću varijablu, koja služi za prepoznavanje govora. Cijeli kod je dan na [Slika 36]

```
r = sr.Recognizer()
filename = sr.AudioFile('speech.wav')
with filename as source:
    audio = r.record(source)
    text = r.recognize_google(audio)
    print(text)
```

Slika 36. Kod za pretvorbu sadržaja zvučne datoteke u tekst

U slučaju kada je velika buka program ima određenih poteškoća pri prepoznavanju govora.

Za taj slučaj postoji metoda adjust_for_ambient_noise() koja djelomično pomaže. Ona prvu sekundu trajanja zvuka ne prepoznaje nego se prilagođava Recognizer klasu prema buci u zvuku. Zbog toga se može dogoditi da početak teksta ne prepozna. U tu svrhu možemo staviti parametar 'duration' na 0.5s tako da prije obavi posao prilagodbe.

5.5. Prepoznavanje govora sa mikrofona

Da bi aplikacija sa mikrofonom radila, potrebno je instalirati PyAudio paket u Pythonu. Klasa Microphone() je potrebna za rad. Ona radi isto što i AudioFile() klasa, ali samo u ovom slučaju sa mikrofonom. Snima govor koji je došao iz mikrofona, pohranjuje ga i stavlja u varijablu 'source'. Umjesto metode record(), koristi se metoda listen() koja radi isto što i record(). Kod pretvorbe govora iz mikrofona u tekst je prikazan na [Slika 37].

```
r = sr.Recognizer()
mic = sr.Microphone()
with mic as source:
    print("Say something")
    r.adjust_for_ambient_noise(source, duration=0.5)
    audio = r.listen(source)
    text = r.recognize_google(audio)
    print(text)
```

Slika 37. Kod pretvorbe govora u tekst pomoću mikrofona

I ovdje je korištena metoda adjust_for_ambient_noise() radi veće otpornosti na buku.

5.6. Aplikacija pretvorbe govora u tekst sa bazom ključnih riječi prepoznatog teksta

U ovome radu je napravljen, ne samo sustav za prepoznavanje govora tj. pretvorbu govora u tekst, nego i interaktivni dio gdje govornik ima mogućnost traženja uputa od virtualne asistentice imenom “Plaea“. Aplikacija se sastoji od dva dijela: prvi dio je pretvorba sa snimljenih zvučnih datoteka, a drugi dio je sa mikrofona uživo govoreći. U prvom i drugom dijelu moguć je rad na engleskom, njemačkom ili hrvatskom jeziku.

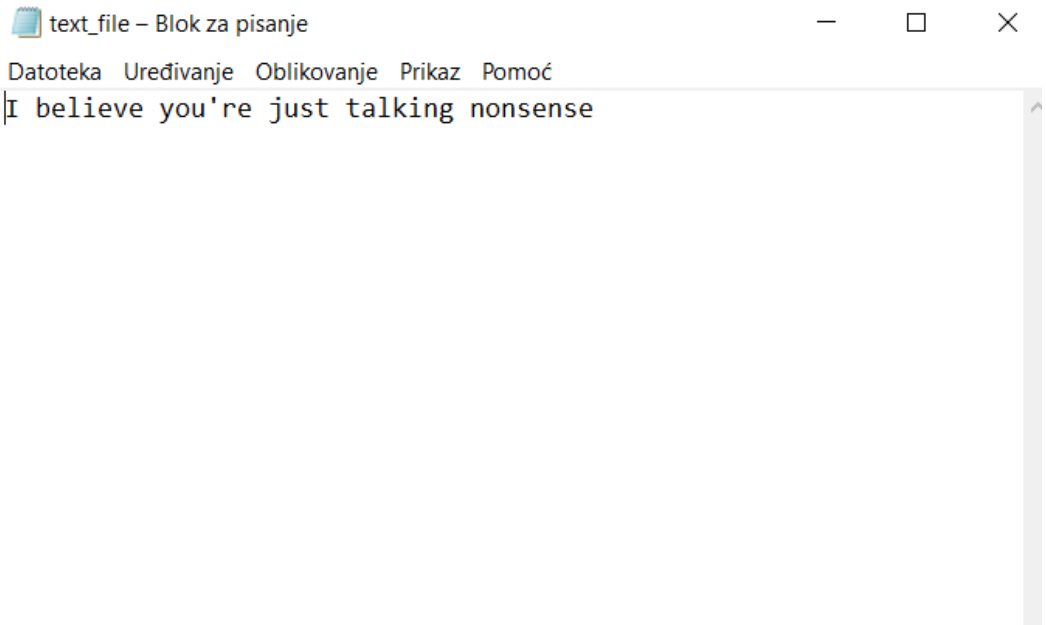
Odmah na početku dolazi do upita kakvu vrsta pretvorbe osoba treba. U prvom dijelu koji je objašnjen kako funkcionira u prethodnim potpoglavljima kod je na slici [Slika 38]. Funkcija

input nam služi da odredimo koji jezik želimo koristiti i koju zvučnu datoteku želimo pretvoriti u tekst.

```
r = sr.Recognizer()                                                    # inicijaliziramo klasu Recognizer()

while True:

    part = int(input(
        "What type of recognition are you interesting in(audio recorded(1), form microphone(2) or exit program(3)"))
    # dio sa snimljenih zvučnih zapisa
    if part == 1:
        language = input("Language(english, german or croatian): ")
        filename = input("What audio do you want to hear?")
        with sr.AudioFile(filename) as source:
            r.adjust_for_ambient_noise(source, duration=0.1)           # adaptacija za zvuk
            audio_data = r.record(source)
            if language == 'english':
                text = r.recognize_google(audio_data)                  # prepoznaje govor u audiu
            elif language == 'german':
                text = r.recognize_google(audio_data, language='de')
            else:
                text = r.recognize_google(audio_data, language='hr')
            file = open('text_file.txt', 'w')
            file.write(text.capitalize())
            file.close()
            print(text.capitalize())                                   # capitalize je za veliko prvo slovo u recenici
            continue
```

Slika 38. Pretvorba sa govora snimljenih materijala u tekst

U slučaju da odredimo jezik engleski u varijablu 'language' i postavimo u varijablu 'filename' zvučnu datoteku naziva 'speech.wav' , u kojoj se čuje kako osoba govori: " I believe you're just talking nonsense.', rezultat bi trebao ispasti [Slika 39]:

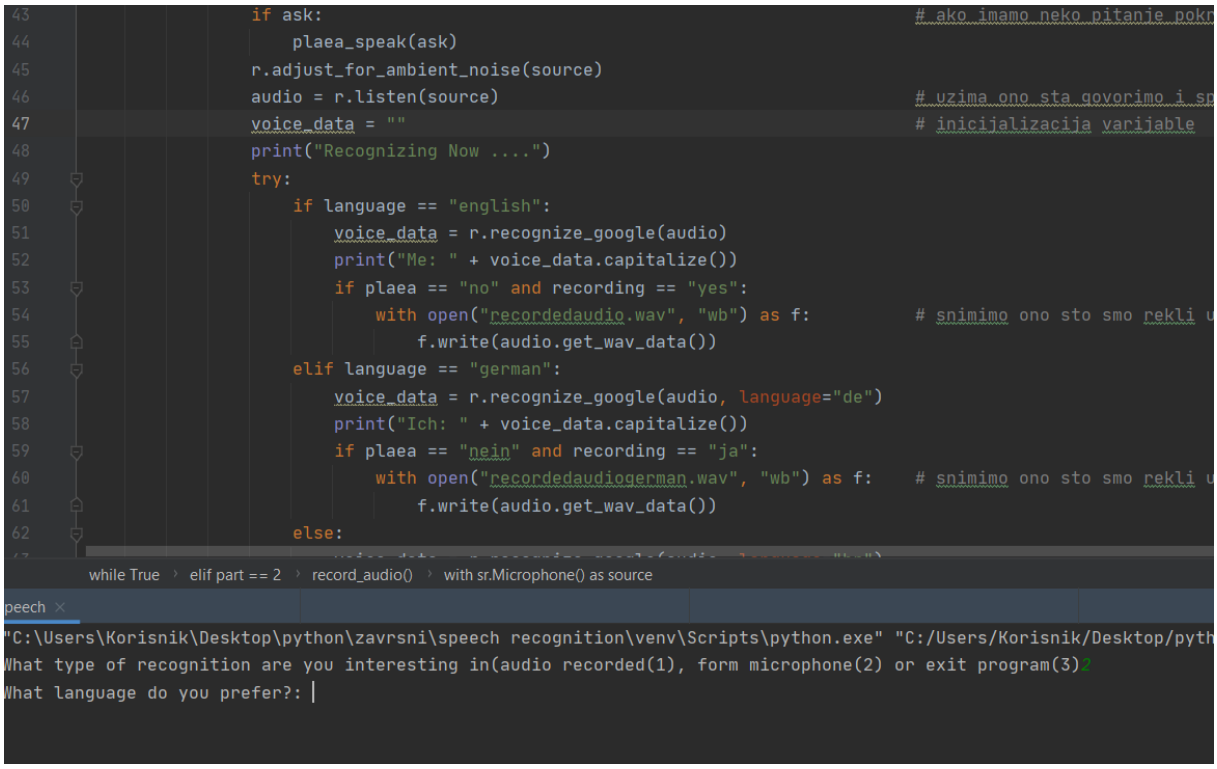


Slika 39. Rezultat pretvorbe govora u tekst zvučne datoteke 'speech.wav'

Ista stvar se može napraviti na njemačkom i hrvatskom jeziku.

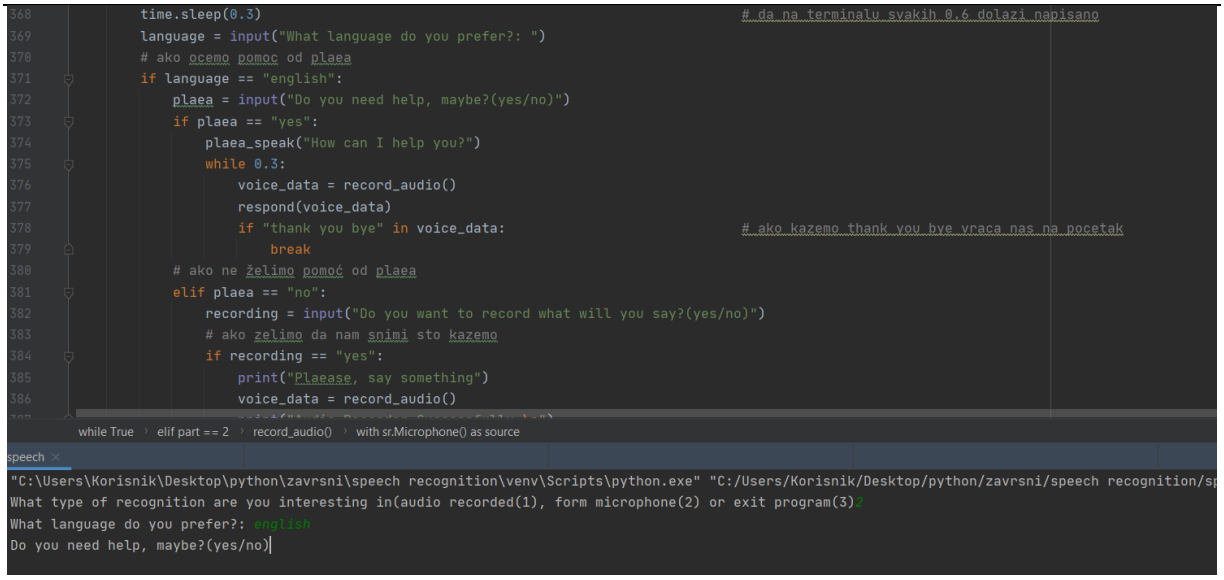
U drugom dijeli koji je pretvorba preko mikrofona je ubačena glasovna asistentica “Plaea“.

Nakon što se odabralo na početku drugi dio aplikacije postavlja se pitanje koji jezik govornik preferira.[Slika 40]



Slika 40. Izbor jezika u aplikaciji

Osoba upisuje jezik koji mu odgovara. Zatim se nudi opcija pomoći u obliku glasovne asistentice koja nudi pomoć.[Slika 41]



Slika 41. Upit za pomoć

Ako osoba prihvati pomoć nastupa glasovna asistentica “Plaea“. To izgleda tako da osoba njoj reče radnju koja je njemu potrebna i ona izbaci rezultate napisano i potvrdi to svojim govorom. Kad osoba izgovori radnju koja mu je potrebna svi se ti glasovi pretvaraju u riječi što je vidljivo na terminalu. Radnje koje “Plaea“ može izvršiti su na govornikove upite su[Slika 42]:

- odrediti datum i uru - govornik kaže 'time' “Plaea“ odredi koliko je sati i koji je datum
- googlati sadržaj kojeg govornika zanima- na govornikov upit 'search' “Plaea“ odlazi na google
- pronaći željenu lokaciju – na govornikov upit 'find location' “Plaea“ odlazi na Google Karte
- odrediti vremensku prognozu u bilom gradu na svijetu
- računati brojke kao jednostavni kalkulator
- prevoditi
- namjestiti alarm

```
Recognizing Now ....
Me: Hello what is your name
Hello. My name is Plaea.
Recognizing Now ....
Me: What time is it
Thu Sep 17 17:08:02 2020
Recognizing Now ....
Me: Search
What do you want to search for?
Recognizing Now ....
Me: Speech recognition
Here is what I found for speech recognition
Recognizing Now ....
Me: Do the math
Addition(1), subtraction(2), multiplication(3), division(4)
Recognizing Now ....
Me: Multiplication
Say the number 1
Recognizing Now ....
Me: 6
Say the number 2
Recognizing Now ....
Me: 5
The final number is 30
Recognizing Now ....
Me: Translate
choose language(1 en to de, 2 de to en, 3 en to hr, 4 hr to en5 de to hr, 6 hr to de, en to fr, 8 random
say what you want to translate...
Recognizing Now ....
Me: Good morning
guten Morgen
```

Slika 42. Razgovor sa govornom asistenticom “Plaea“

Za većinu ovih radnji je potrebna biblioteka koja je navedena i objašnjena u komentarima (oznaka #)[Slika 33] na početku ovog poglavlja. Dana su objašnjena kodova za alarm i vremensku prognozu jer su ta dva koda kompleksnija od ostalih.

Kod za namjestiti alarm je prikazana na slici [Slika 43 i Slika 44].

```
if "set alarm" in voice_data:
    alarmHour = int(input("What hour do you want the alarm to ring"))
    alarmMinute = int(record_audio("What minute do you want the alarm to ring"))
    amPm = str(record_audio("am or pm?"))
    plaea_speak("Waiting for alarm {}h {}min {}".format(alarmHour, alarmMinute, amPm))
    if (amPm == "afternoon"):
        alarmHour = alarmHour + 12
    while (1 == 1):
        if (alarmHour == datetime.datetime.now().hour and
                alarmMinute == datetime.datetime.now().minute):
            print("Time to wake up")
            playsound.playsound('Alarm-Clock Sound.mp3')
            break

    print("exited")

if "exit" in voice_data:
    exit()
```

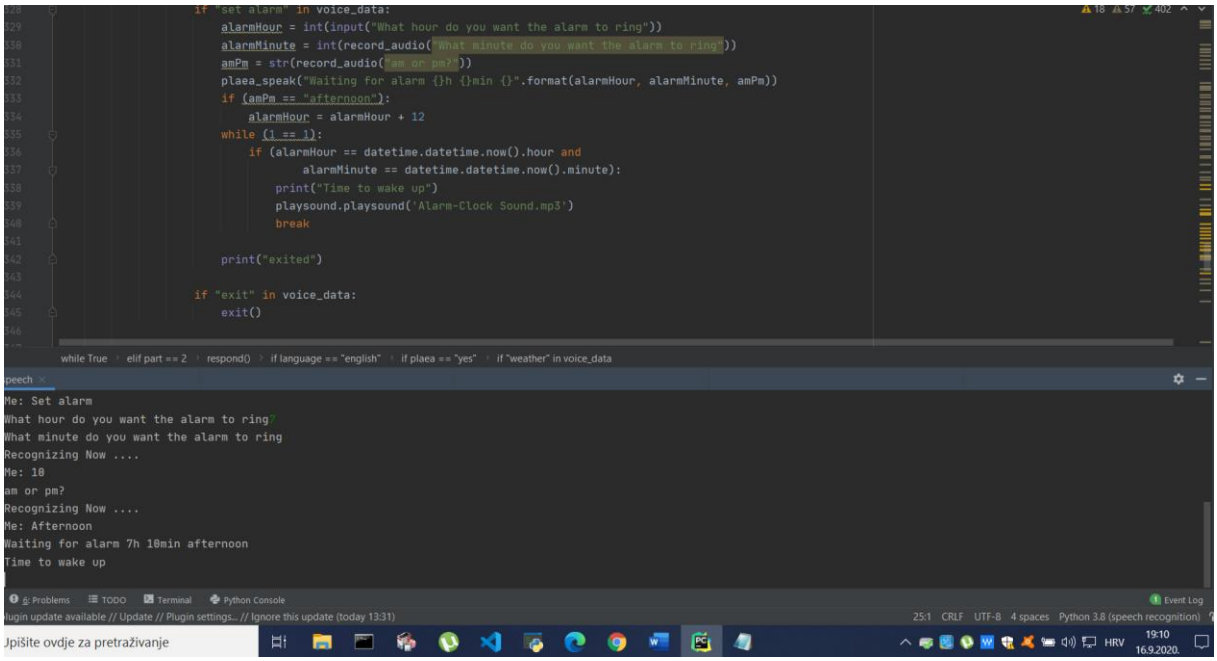
while True › elif part == 2 › respond() › if language == "english" › if plaea == "yes" › if "weather" in voice_data

speech

```
Recognizing Now ....
Me: Set alarm
What hour do you want the alarm to ring?
What minute do you want the alarm to ring
Recognizing Now ....
Me: 10
am or pm?
Recognizing Now ....
Me: Afternoon
Waiting for alarm 7h 10min afternoon
```

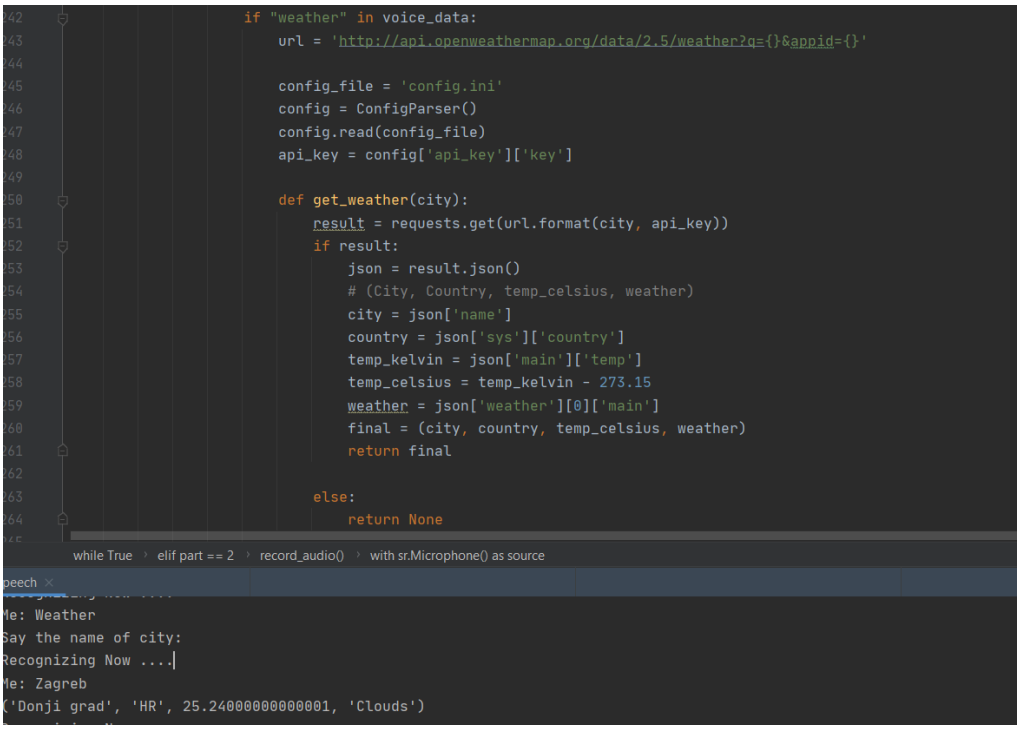
Slika 43. Kod i rezultat za izradu alarma(prije)

Ako je poslijepodne postavljen je brojač na +12 tako da dođe na popodnevno vrijeme(npr. 7h + 12 je 19h). Kad dođe trenutku koji je postavljen za alarm, on zvoni zvukom koji je postavljen('Alarm-Clock Sound.mp3) i izlazi natpis u terminalu: "Time to wake up".



Slika 44. Kod i rezultat za izradu alarma(poslije)

Kod za vrijeme je prikazan na slici [Slika 45]:



Slika 45. Kod za vremensku prognozu

Pomoću varijable url se povezujmo sa OpeanWheatherMapAPI stranicom i koristimo njezin API ključ, kojeg smo pohranili, u tekst dokument za određivanje trenutnog vremena u svijetu.

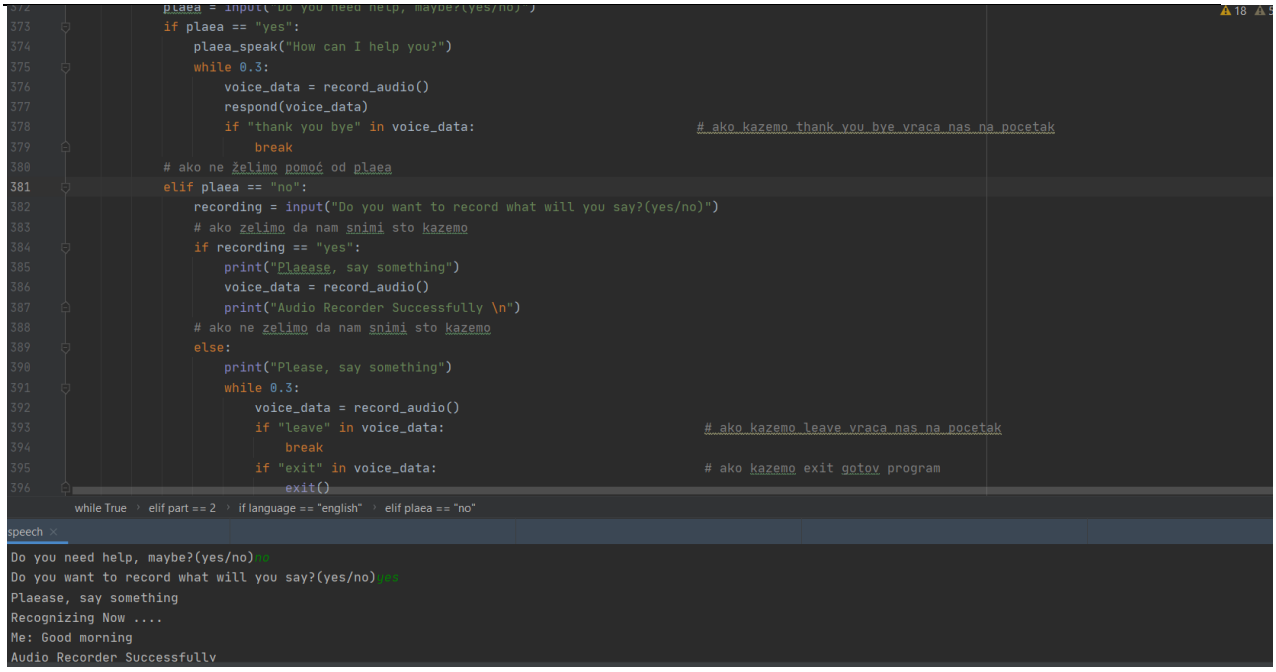
Kod za kreiranje glasovne asistentice “Plaea“ je prikazana na slici[Slika 46].

```
def plaea_speak(audio_string):
    tts = gTTS(text=audio_string, lang="en")                    # text to speech varijabla
    r = random.randint(1, 1000000)                              # tako da mozemo svakom audiufilu dati random broj
    audio_file = "audio-" + str(r) + ".mp3"                     # stvara audio file plaea
    tts.save(audio_file)                                        # snimamo audio file
    playsound.playsound(audio_file)                             # pokrenemo sto prica plaea
    print(audio_string)                                         # printamo sto kaze plaea
    os.remove(audio_file)                                       # brisemo audio file u direktoriju
```

Slika 46. Funkcija plaea_speak za izradu glasovne asistentice “Plaea“

Jednako tako su napravljene funkcije na njemačkom, francuskom i hrvatskom jeziku što je bitno za prijevod rečenica. “Plaea“ nije sposobna lijepo se izražavati na hrvatskom području. I to je jedan od glavnih nedostataka modernih sustava za prepoznavanje govora.

U slučaju da govornik ne traži pomoć već samo želi pretvoriti govor u tekst otvara mu se mogućnost da snimi svoj vlastiti govor u zvučnu datoteku, koja se sprema u isti direktorij sa Pyhton dokumentom. Kod prikazan na slici [Slika 47]:



```
plaea = input("Do you need help, maybe?(yes/no) ")
if plaea == "yes":
    plaea_speak("How can I help you?")
    while 0.3:
        voice_data = record_audio()
        respond(voice_data)
        if "thank you bye" in voice_data:                    # ako kazemo thank you bye vraca nas na pocetak
            break
# ako ne želimo pomoć od plaea
elif plaea == "no":
    recording = input("Do you want to record what will you say?(yes/no)")
    # ako zelimo da nam snimi sto kazemo
    if recording == "yes":
        print("Plaease, say something")
        voice_data = record_audio()
        print("Audio Recorder Successfully \n")
    # ako ne zelimo da nam snimi sto kazemo
    else:
        print("Please, say something")
        while 0.3:
            voice_data = record_audio()
            if "leave" in voice_data:                        # ako kazemo leave vraca nas na pocetak
                break
            if "exit" in voice_data:                         # ako kazemo exit gotov program
                exit()
```

while True › elif part == 2 › if language == "english" › elif plaea == "no"

speech

```
Do you need help, maybe?(yes/no)no
Do you want to record what will you say?(yes/no)yes
Plaease, say something
Recognizing Now ....
Me: Good morning
Audio Recorder Successfully
```

Slika 47. Kod za snimit govor

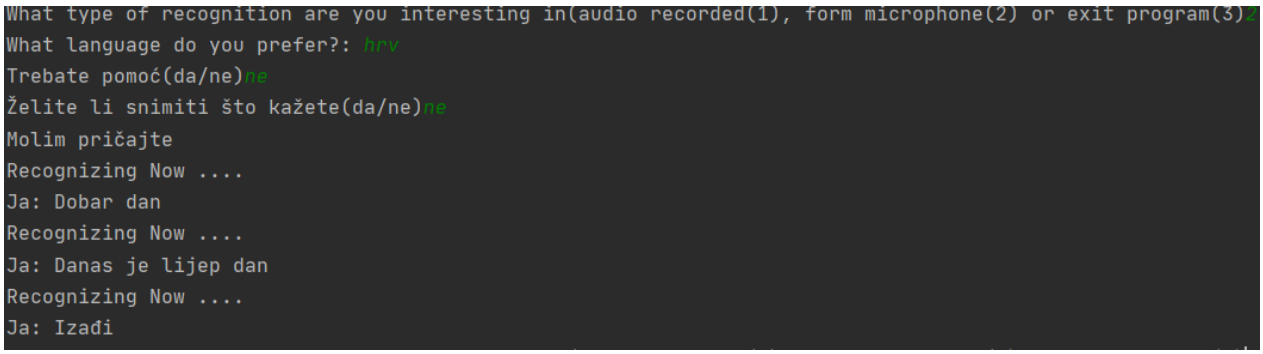
Ako je odabra snimit govor, u programu dolazi do if petlje u kojoj je funkcija record_audio() u kojoj su zadane naredbe za pretvorbu(klasa Microphone())[Slika 48]. U toj funkciji su i if naredbe koje određuju snima li se ili ne i kojim jezikom.

```
elif part == 2:
    def record_audio(ask=False):
        with sr.Microphone() as source:
            if ask:                                          # ako imamo neko pitanje pokrece se funkcija plaea_speak
                plaea_speak(ask)
            r.adjust_for_ambient_noise(source)
            audio = r.listen(source)                         # uzima ono sta govorimo i sprema dok ne dodje tisina
            voice_data = ""                                  # inicijalizacija varijable
            print("Recognizing Now ....")
            try:
                if language == "english":
                    voice_data = r.recognize_google(audio)
                    print("Me: " + voice_data.capitalize())
                    if plaea == "no" and recording == "yes":
                        with open("recordedaudio.wav", "wb") as f:      # snimimo ono sto smo rekli u audio file
                            f.write(audio.get_wav_data())
                elif language == "german":
                    voice_data = r.recognize_google(audio, language="de")
                    print("Ich: " + voice_data.capitalize())
                    if plaea == "nein" and recording == "ja":
                        with open("recordedaudiogerman.wav", "wb") as f:    # snimimo ono sto smo rekli u audio file
                            f.write(audio.get_wav_data())
                else:
                    voice_data = r.recognize_google(audio, language="hr")
                    print("Ja: " + voice_data.capitalize())
                    if plaea == "ne" and recording == "da":
                        with open("recordedaudiohrv.wav", "wb") as f:       # snimimo ono sto smo rekli u audio file
                            f.write(audio.get_wav_data())

            except sr.UnknownValueError:                     # ako se cuje preveliko suskanje ili buka ili nejasne rijeci
                if language == "english":
                    if plaea == "yes":                       # postavili smo uvjet kad zelimo pomoc
                        plaea_speak("Sorry, I did not get that")
                    else:
                        print("Sorry, I did not get that")
                elif language == "german":
                    if plaea == "ja":
                        plaea_speak_ger("Entschuldigung, ich habe das nicht verstanden")
                    else:
                        print("Entschuldigung, ich habe das nicht verstanden")
                else:
                    if plaea == "da":                        # postavili smo uvjet kad zelimo pomoc
                        plaea_speak_cro("Oprostite nisam razumio")
                    else:
                        print("Oprostite nisam razumio")
            except sr.RequestError:                          # ako nema interneta itd..
                if plaea == "yes":                           # ako smo odabrali pomoc od a ona kaze
                    plaea_speak("Sorry, my speech service is down")
                else:
                    print("Sorry, my speech service is down")           # ako nismo odabrali pomoc od Plaea samo se napise
            return voice_data
```

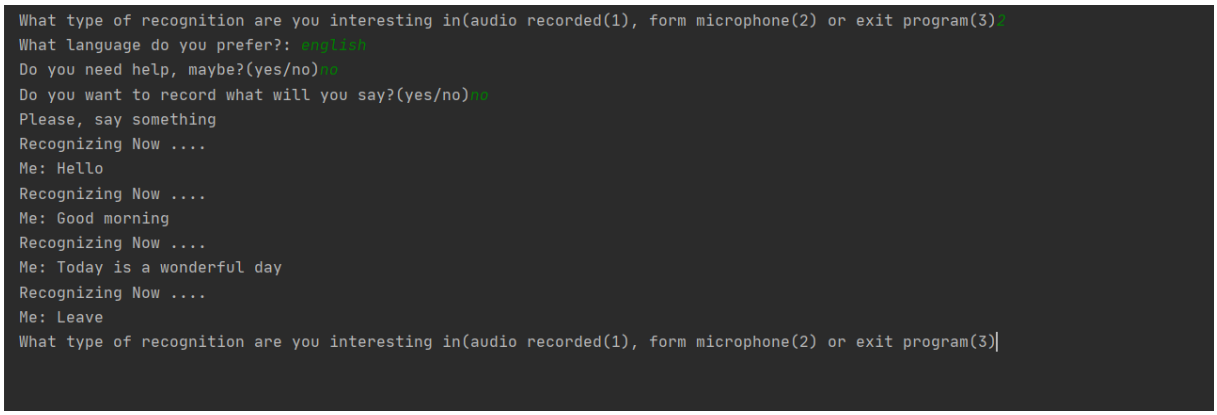
Slika 48. Funkcija record_audio()

Postavljen je try blok koji je aktiviran ako je pretvorba bila uspješna i except ako je bila neuspješna kao što je vidljivo sa slike.

Zadnja opcija ove aplikacije je čista pretvorba govora u tekst bez interakcije sa “Plaeaom“ i bez snimanja govora. Rezultati pretvorbe na tri jezika su prikazani na slikama [Slika 49, Slika 50, Slika 51]:



Slika 49. Rezultat pretvorbe govora u tekst sa bazama ključnih riječi n hrvatskom jeziku



Slika 50. Rezultat pretvorbe govora u tekst sa bazama ključnih riječi n engleskom jeziku

Slika 51. Rezultat pretvorbe govora u tekst sa bazama ključnih riječi n njemačkom jeziku

6. ZAKLJUČAK

Cilj ovog završnog rada bila je izrada aplikacije za pretvorbu govora u tekst u svrhu korištenja za potrebe robotskog sustava "Plaea". Navedene su tehnologije i opisan je način njihovog rada. Iz navedenih razmatranja zaključuje se da neovisno o mjestu implementacije, bio to jezični model, akustični model ili dekoder, umjetne neuronske mreže daju bolje rezultate, brže su i točnije od prijašnjih sustava, ali isključivo pod uvjetom da je učenje provedeno s dovoljnom količinom kvalitetnih podataka. Usprkos velikim postignućima umjetne inteligencije postoji dosta mjesta za napredak, pogotovo u prepoznavanju neeuropskih jezika. Prepoznavanje govora je kompleksno područje na kojem se svakodnevno intenzivno radi, posebno na modernijim sustavima poput *end-to-end* sutava. Iako još relativno novi, *end-to-end* su budućnost sustava za prepoznavanje jer koriste jednu neuronsku mrežu i time su brži i učinkoviti. Rast interesa za ovo područje među ljudima i tehnološki rast su ključni čimbenici još većeg razvoja sustava za prepoznavanje riječi.

Kako bi se još više povećala interaktivnost robotskog sustava "Plaea" potrebno je još rada na njemu. Prepoznavanje govora je tek važan kotačić u cjeloukupnom razvoju kontekstualne percepcije. Ostavlja se mogućnost izrade bolje aplikacije koja može prepoznati emocije u čovjeku.

LITERATURA

[1] A short history history of speech recognition, Sonix, https://sonix.ai/history-of-speech-recognition

[2] Pletikos E.: Kultura govora, Glas, https://fonet.ffzg.unizg.hr/pletikos/predav-kultura_gov/2_Glas-fonacija.pdf

[3] Osnove procesa nastajanja govora, http://dog.zesoi.fer.hr/predavanja/HTML/Osnove%20procesa%20nastajanja%20govora.htm

[4] Leksikografski zavod Miroslav Krleža, http://www.enciklopedija.hr

[5] Fonem, https://hr.wikipedia.org/wiki/Fonem

[6] Hui J. , Speech Recogniton – Phonetics, kolovoz 2019 https://medium.com/@jonathan_hui/speech-recognition-phonetics-d761ea1710c0,

[7] Speech recognition systems, Microsoft, edx https://courses.edx.org/courses/coursev1:Microsoft+DEV287x+2T2019/course/

[8] Stipančić T., Umjetna inteligencija – probabilistička perspektiva , https://e-ucenje.fsb.hr/pluginfile.php/84463/mod_resource/content/2/03_teorije_vjerojatnosti_umjetna_inteligencija.pdf

[9] E. H Chandra, A Review on Automatic Speech Recognition Architecture and Approaces, travanj 2016 file:///C:/Users/Korisnik/Downloads/A_Review_on_Automatic_Speech_Recognition_Architect%20(1).pdf

[10] Hui J. , Speech Recogniton – Feature Extraction MFCC & PLP, kolovoz 2019 https://medium.com/@jonathan_hui/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9, listopad 2019

[11] U. Shrawankar i V. Thakare, »Techniques for Feature Extraction in Speech Recognition, https://arxiv.org/ftp/arxiv/papers/1305/1305.1145.pdf

[12] Hui J., Speech Recogniton – GMM- HMM, rujan 2019 https://medium.com/@jonathan_hui/speech-recognition-gmm-hmm-8bb5eff8b196,

[13] Šekoranja B. , Podloge za vježbe iz kolegija Umjetna inteligencija, Markovljevi lanci, http://titan.fsb.hr/~bosekora/nastava/ui/UI_podloge.pdf

[14] Šmuc T. , Strojno učenje, Markovljevi modeli, svibanj 2012

file:///C:/Users/Korisnik/Downloads/Strojno%20uenje_HMMs_2012_v3.pdf

[15] Hui J. , Speech Recognition – Acoustic, Lexicon & Language Model, rujan, 2019 https://medium.com/@jonathan_hui/speech-recognition-acoustic-lexicon-language-model-aacac0462639,

[16] Speech recognition, Dynamic Time Wrapping(DTW), https://en.wikipedia.org/wiki/Speech_recognition#Dynamic_time_warping_(DTW)-based_speech_recognition

[17] Y. Bengio, R.ucharme, P. Vincent, C. Jauvin: A neural Probabilistic Language Model, Journal of Machine Learning Research, ožujak 2003, https://jmlr.org/papers/volume3/tmp/bengio03a.pdf

[18] T.Mikolov, M. Karafiat, L. Burget, J. Černocky, S. Khudanpur: Recurreent neural network based language model, Proc. Interspeech, 2010 , https://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

[19] Hui J. , Speech Recognition – Weighted Finite-State Transducers(WFST), rujan 2019, https://medium.com/@jonathan_hui/speech-recognition-weighted-finite-state-transducers-wfst-a4ece08a89b7

[20] Wang S. , Li G. , Overview of end-to-end speech recognition, 2019, https://iopscience.iop.org/article/10.1088/1742-6596/1187/5/052068/pdf

[21] Van der Valde, N. , Speech Recogntion Technology Overview, srpanj, 2019, https://www.globalme.net/blog/the-present-future-of-speech-recognition/

[22] Amos, D. , The Ultimate Guide To Speech Recognition With Python. , https://realpython.com/python-speech-recognition/

PRILOZI

I. CD-R disc

II. Python kod

Python kod za pretvorbu valnog oblika u frekvencijsku domenu

```python
import numpy as np
from scipy.io import wavfile
import matplotlib.pyplot as plt

# Read the input file
sampling_freq, audio = wavfile.read('recordedaudiohrv.wav')

# Normalize the values
audio = audio / (2.**15)

# Extract length
len_audio = len(audio)

# Apply Fourier transform
transformed_signal = np.fft.fft(audio)
half_length = int(np.ceil((len_audio + 1) / 2.0))
transformed_signal = abs(transformed_signal[0:half_length])
transformed_signal /= float(len_audio)
transformed_signal **= 2

# Extract length of transformed signal
len_ts = len(transformed_signal)

# Take care of even/odd cases
if len_audio % 2:
    transformed_signal[1:len_ts] *= 2
else:
    transformed_signal[1:len_ts-1] *= 2

# Extract power in dB
power = 10 * np.log10(transformed_signal)

# Build the time axis

x_values = np.arange(0, half_length, 1) * (sampling_freq / len_audio) / 1000.0

# Plot the figure
plt.figure()
plt.plot(x_values, power, color='black')
plt.xlabel('Freq (in kHz)')
plt.ylabel('Power (in dB)')
plt.show()
```

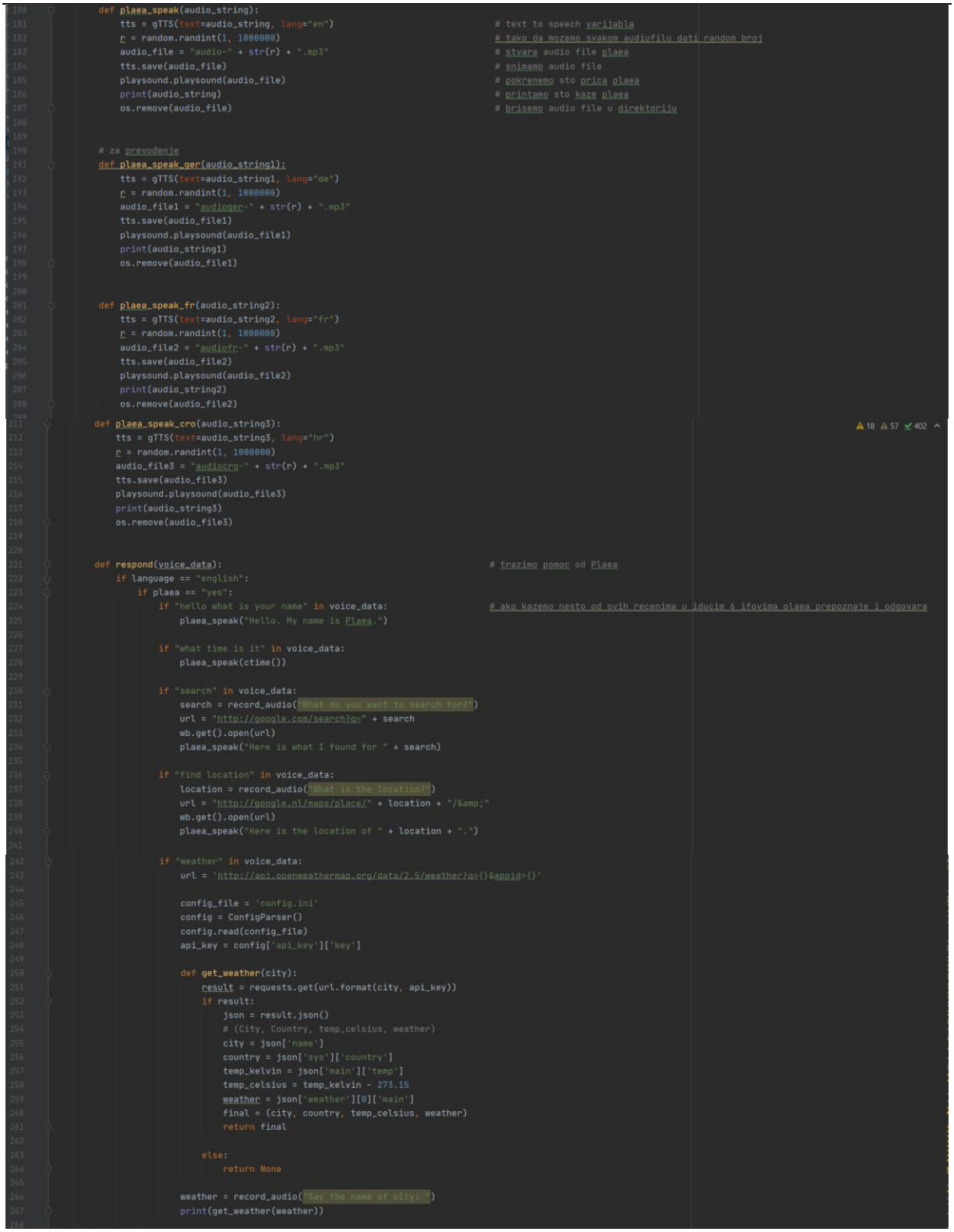
Python kod za izvlačenje značajki MFCC analizom

```
import matplotlib.pyplot as plt
from scipy.io import wavfile
from python_speech_features import mfcc, logfbank

# Read input sound file
sampling_freq, audio = wavfile.read("recordedaudiohrv.wav")

# Extract MFCC and Filter bank features
mfcc_features = mfcc(audio, sampling_freq)
filterbank_features = logfbank(audio, sampling_freq)

# Print parameters
print('\nMFCC: \nNumber of windows =', mfcc_features.shape[0])
print('Length of each feature =', mfcc_features.shape[1])
print('\nFilter bank: \nNumber of windows =', filterbank_features.shape[0])
print('Lenth of each feature =', filterbank_features.shape[1])

# Plot the features
mfcc_features = mfcc_features.T
plt.matshow(mfcc_features)
plt.title('MFCC')

filterbank_features = filterbank_features.T
plt.matshow(filterbank_features)
plt.title('Filter bank')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.show()
```

Python kod aplikacije za prepoznavanje govora i glasovnog asistenta 'Plea'

```
import speech_recognition as sr
import webbrowser as wb                                     # link za internet
import time                                                 # za sleep koji nam služi da postavimo periodično vrijeme između dva razgovora
import playsound                                            # da ne ode na druge filove kao npr. iTunes..
import os                                                   #  brisenja audia u direktoriju
import random
from gtts import gTTS                                       # da napravi audiofile Plaea
from time import ctime                                      # trenutno vrijeme
from configparser import ConfigParser                       # za prepoznati API kljuc stranice OpenWheatherMap kojeg smo smjestili u file config.ini
import requests
from googletrans import Translator                          # za prevođenje jezika
import datetime                                             # paket za odredivanje sati i minuta pri postavljanju alarma

r = sr.Recognizer()                                         # inicijaliziramo klasu Recognizer()

while True:

    part = int(input(
        "What type of recognition are you interesting in(audio recorded(1), form microphone(2) or exit program(3)"))
    # dio sa snimljenih zvučnih zapisa
    if part == 1:
        language = input("Language(english, german or croatian): ")
        filename = input("What audio do you want to hear?")
        with sr.AudioFile(filename) as source:
            r.adjust_for_ambient_noise(source, duration=0.1)        # adaptacija za zvuk
            audio_data = r.record(source)
            if language == 'english':
                text = r.recognize_google(audio_data)               # prepoznaje govor u audiu
            elif language == 'german':
                text = r.recognize_google(audio_data, language='de')
            else:
                text = r.recognize_google(audio_data, language='hr')
            file = open('text_file.txt', 'w')
            file.write(text.capitalize())
            file.close()
            print(text.capitalize())                                # capitalize je za veliko prvo slovo u recenici
            continue

    # dio sa mikrofona
    elif part == 2:
        def record_audio(ask=False):
            with sr.Microphone() as source:
                if ask:                                             # ako imamo neko pitanje pokrece se funkcija plaea.speak
                    plaea_speak(ask)
                r.adjust_for_ambient_noise(source)
                audio = r.listen(source)                            # uzima ono sta govorimo i sprema dok ne dodje tisina
                voice_data = ""                                     # inicijalizacija varijable
                print("Recognizing Now ....")
                try:
                    if language == "english":
                        voice_data = r.recognize_google(audio)
                        print("Me: " + voice_data.capitalize())
                        if plaea == "no" and recording == "yes":
                            with open("recordedaudio.wav", "wb") as f:          # snimimo ono sto smo rekli u audio file
                                f.write(audio.get_wav_data())
                    elif language == "german":
                        voice_data = r.recognize_google(audio, language="de")
                        print("Ich: " + voice_data.capitalize())
                        if plaea == "nein" and recording == "ja":
                            with open("recordedaudiogerman.wav", "wb") as f:    # snimimo ono sto smo rekli u audio file
                                f.write(audio.get_wav_data())
                    else:
                        voice_data = r.recognize_google(audio, language="hr")
                        print("Ja: " + voice_data.capitalize())
                        if plaea == "ne" and recording == "da":
                            with open("recordedaudiohrv.wav", "wb") as f:       # snimimo ono sto smo rekli u audio file
                                f.write(audio.get_wav_data())

                except sr.UnknownValueError:                                    # ako se cuje previsoko suskanje ili buka ili nejasne rijeci
                    if language == "english":
                        if plaea == "yes":                                      # postavili smo uvjet kad zelimo pomoc
                            plaea_speak("Sorry, I did not get that")
                        else:
                            print("Sorry, I did not get that")
                    elif language == "german":
                        if plaea == "ja":
                            plaea_speak_ger("Entschuldigung, ich habe das nicht verstanden")
                        else:
                            print("Entschuldigung, ich habe das nicht verstanden")
                    else:
                        if plaea == "da":                                       # postavili smo uvjet kad zelimo pomoc
                            plaea_speak_cro("Oprostite nisam razumio")
                        else:
                            print("Oprostite nisam razumio")
                    except sr.RequestError:                                     # ako nema interneta itd..
                        if plaea == "yes":                                      # ako smo odabrali pomoc od a ona kaze
                            plaea_speak("Sorry, my speech service is down")
                        else:
                            print("Sorry, my speech service is down")           # ako nismo odabrali pomoc od Plaea samo se napise
                    return voice_data
```

```python
    # za prevodenje
    def record_audio_ger(ask=False):
        with sr.Microphone() as source1:
            if ask:                                                 # ako imamo neko pitanje pokrece se funkcija Plaea_speak
                plaea_speak_ger(ask)
            r.adjust_for_ambient_noise(source1)
            audio1 = r.listen(source1)                              # uzima ono sta govorimo i sprema dok ne dodje tisina
            voice_data_ger = ""                                     # inicijalizacija varijable
            print("Jetzt erkennen ....")
            try:
                voice_data_ger = r.recognize_google(audio1, language='de')
                print("Ich: " + voice_data_ger.capitalize())
                if plaea == "no" and recording == "yes":
                    with open("recordedaudiogerman.wav", "wb") as f:          # snimimo ono sto smo rekli u audio file
                        f.write(audio1.get_wav_data())

            except sr.UnknownValueError:                            # ako se cuje preveliko suskanje ili buka ili nejasne rijeci
                if plaea == "yes":                                  # postavili smo uvjet kad zelimo pomoc
                    plaea_speak_ger("Entschuldigung, ich habe das nicht verstanden")
                else:
                    print("Entschuldigung, ich habe das nicht verstanden")
            except sr.RequestError:                                 # ako nema interneta itd..
                if plaea == "yes":                                  # ako smo odabrali pomoc od Plaea ona kaze
                    plaea_speak_ger("Entschuldigung, mein Sprachdienst ist ausgefallen")
                else:
                    print(
                        "Entschuldigung, mein Sprachdienst ist ausgefallen")  # ako nismo odabrali pomoc od Plaea samo se napise
            return voice_data_ger


    def record_audio_fr(ask=False):
        with sr.Microphone() as source2:
            if ask:                                                 # ako imamo neko pitanje pokrece se funkcija Plaea_speak
                plaea_speak_fr(ask)
            r.adjust_for_ambient_noise(source2)
            audio2 = r.listen(source2)                              # uzima ono sta govorimo i sprema dok ne dodje tisina
            voice_data_fr = ""                                      # inicijalizacija varijable
            print("Reconnaitre maintenant ....")
            try:
                voice_data_fr = r.recognize_google(audio2, language='fr')
                print("Moi: " + voice_data_fr.capitalize())
                if plaea == "no" and recording == "yes":
                    with open("recordedaudiofr.wav", "wb") as f:          # snimimo ono sto smo rekli u audio file
                        f.write(audio2.get_wav_data())

            except sr.UnknownValueError:                            # ako se cuje preveliko suskanje ili buka ili nejasne rijeci
                if plaea == "yes":                                  # postavili smo uvjet kad zelimo pomoc
                    plaea_speak_fr("Désolé, je n'ai pas compris")
                else:
                    print("Désolé, je n'ai pas compris")
            except sr.RequestError:                                 # ako nema interneta itd..
                if plaea == "yes":                                  # ako smo odabrali pomoc od Plaea ona kaze
                    plaea_speak_fr("Désolé, mon service vocal est en panne")
                else:
                    print(
                        "Désolé, mon service vocal est en panne")  # ako nismo odabrali pomoc od Plaea samo se napise
            return voice_data_fr


    def record_audio_cro(ask=False):
        with sr.Microphone() as source3:
            if ask:                                                 # ako imamo neko pitanje pokrece se funkcija plaea_speak
                plaea_speak_cro(ask)
            r.adjust_for_ambient_noise(source3)
            audio3 = r.listen(source3)                              # uzima ono sta govorimo i sprema dok ne dodje tisina
            voice_data_cro = ""                                     # inicijalizacija varijable
            print("Prepoznaje ....")
            try:
                voice_data_cro = r.recognize_google(audio3, language='hr')
                print("Ja: " + voice_data_cro.capitalize())
                if plaea == "no" and recording == "yes":
                    with open("recordedaudiocro.wav", "wb") as f:          # snimimo ono sto smo rekli u audio file
                        f.write(audio3.get_wav_data())

            except sr.UnknownValueError:                            # ako se cuje preveliko suskanje ili buka ili nejasne rijeci
                if plaea == "yes":                                  # postavili smo uvjet kad zelimo pomoc
                    plaea_speak_cro("Oprostite, ne razumijem")
                else:
                    print("Oprostite, ne razumijem")
            except sr.RequestError:                                 # ako nema interneta itd..
                if plaea == "yes":                                  # ako smo odabrali pomoc od plaea ona kaze
                    plaea_speak_cro("Oprostite pao je sustav")
                else:
                    print("Oprostite pao je sustav")                # ako nismo odabrali pomoc od plaea samo se napise
            return voice_data_cro
```



```
def plaea_speak(audio_string):
    tts = gTTS(text=audio_string, lang="en")                    # text to speech varijabla
    r = random.randint(1, 1000000)                              # tako da mozemo svakom audiufilu dati random broj
    audio_file = "audio-" + str(r) + ".mp3"                     # stvara audio file plaea
    tts.save(audio_file)                                        # snimamo audio file
    playsound.playsound(audio_file)                             # pokrenemo sto prica plaea
    print(audio_string)                                         # printamo sto kaze plaea
    os.remove(audio_file)                                       # brisemo audio file u direktoriju


# za prevodenje
def plaea_speak_ger(audio_string1):
    tts = gTTS(text=audio_string1, lang="de")
    r = random.randint(1, 1000000)
    audio_file1 = "audioger-" + str(r) + ".mp3"
    tts.save(audio_file1)
    playsound.playsound(audio_file1)
    print(audio_string1)
    os.remove(audio_file1)


def plaea_speak_fr(audio_string2):
    tts = gTTS(text=audio_string2, lang="fr")
    r = random.randint(1, 1000000)
    audio_file2 = "audiofr-" + str(r) + ".mp3"
    tts.save(audio_file2)
    playsound.playsound(audio_file2)
    print(audio_string2)
    os.remove(audio_file2)
def plaea_speak_cro(audio_string3):
    tts = gTTS(text=audio_string3, lang="hr")
    r = random.randint(1, 1000000)
    audio_file3 = "audiocro-" + str(r) + ".mp3"
    tts.save(audio_file3)
    playsound.playsound(audio_file3)
    print(audio_string3)
    os.remove(audio_file3)


def respond(voice_data):                                        # trazimo pomoc od Plaea
    if language == "english":
        if plaea == "yes":
            if "hello what is your name" in voice_data:         # ako kazemo nesto od ovih recenima u iducim 6 ifovima plaea prepoznaje i odgovara
                plaea_speak("Hello. My name is Plaea.")

            if "what time is it" in voice_data:
                plaea_speak(ctime())

            if "search" in voice_data:
                search = record_audio("What do you want to search for?")
                url = "http://google.com/search?q=" + search
                wb.get().open(url)
                plaea_speak("Here is what I found for " + search)

            if "find location" in voice_data:
                location = record_audio("What is the location?")
                url = "http://google.nl/maps/place/" + location + "/&amp;"
                wb.get().open(url)
                plaea_speak("Here is the location of " + location + ".")

            if "weather" in voice_data:
                url = 'http://api.openweathermap.org/data/2.5/weather?q={}&appid={}'

                config_file = 'config.ini'
                config = ConfigParser()
                config.read(config_file)
                api_key = config['api_key']['key']

                def get_weather(city):
                    result = requests.get(url.format(city, api_key))
                    if result:
                        json = result.json()
                        # (City, Country, temp_celsius, weather)
                        city = json['name']
                        country = json['sys']['country']
                        temp_kelvin = json['main']['temp']
                        temp_celsius = temp_kelvin - 273.15
                        weather = json['weather'][0]['main']
                        final = (city, country, temp_celsius, weather)
                        return final

                    else:
                        return None

                weather = record_audio("Say the name of city: ")
                print(get_weather(weather))
```

```
            if "do the math" in voice_data:
                choose_operation = str(record_audio("Addition(1), subtraction(2), multiplication(3), division(4)"))
                number1 = int(record_audio("Say the number 1"))
                number2 = int(record_audio("Say the number 2"))
                if choose_operation == "addition":
                    result = number1 + number2
                elif choose_operation == "subtraction":
                    result = number1 - number2
                elif choose_operation == "multiplication":
                    result = number1 * number2
                else:
                    result = number1 / number2
                plaea_speak("The final number is {}".format(result))

            if "Translate" in voice_data:
                choose_language = int(input("choose language"))
                if choose_language == 1:
                    sentence = str(record_audio("say what you want to translate..."))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='english', dest='german')
                    plaea_speak_ger(transleted_sentence.text)
                elif choose_language == 2:
                    sentence = str(record_audio_ger("Sagen Sie, was Sie übersetzen möchten..."))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='german', dest='english')
                    plaea_speak(transleted_sentence.text)
                elif choose_language == 3:
                    sentence = str(record_audio("say what you want to translate..."))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='english', dest='croatian')
                    plaea_speak_cro(transleted_sentence.text)
                elif choose_language == 4:
                    sentence = str(record_audio_cro("Recite što želite prevesti"))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='hr', dest='en')
                    plaea_speak(transleted_sentence.text)
                elif choose_language == 5:
                    sentence = str(record_audio_ger("Sagen Sie, was Sie übersetzen möchten..."))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='de', dest='hr')
                    plaea_speak_cro(transleted_sentence.text)
                elif choose_language == 6:
                    sentence = str(record_audio_cro("Recite što želite prevesti"))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='hr', dest='de')
                    plaea_speak_ger(transleted_sentence.text)
                elif choose_language == 7:
                    sentence = str(record_audio("say what you want to translate"))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src='en', dest='fr')
                    plaea_speak_fr(transleted_sentence.text)
                else:
                    from_lang = record_audio("From what language")
                    to_lang = record_audio("To what language")
                    sentence = str(input("say what you want to translate"))
                    translator = Translator()
                    transleted_sentence = translator.translate(sentence, src=from_lang, dest=to_lang)
                    print(transleted_sentence.text)

            if "set alarm" in voice_data:
                alarmHour = int(input("What hour do you want the alarm to ring"))
                alarmMinute = int(record_audio("What minute do you want the alarm to ring"))
                amPm = str(record_audio("am or pm?"))
                plaea_speak("Waiting for alarm {}h {}min {}".format(alarmHour, alarmMinute, amPm))
                if (amPm == "afternoon"):
                    alarmHour = alarmHour + 12
                while (1 == 1):
                    if (alarmHour == datetime.datetime.now().hour and
                            alarmMinute == datetime.datetime.now().minute):
                        print("Time to wake up")
                        playsound.playsound('Alarm-Clock Sound.mp3')
                        break

                print("exited")

            if "exit" in voice_data:
                exit()


        elif language == "german":
            if "wie heißen sie" in voice_data:
                plaea_speak_ger("Mein Name ist Plaea")
            if "wie spät ist es" in voice_data:
                plaea_speak_ger(ctime())
            if "video" in voice_data:
                video = record_audio_ger("Was willst du sehen?")
                url = "https://www.youtube.com/results?search_query=" + video
                wb.get().open(url)
                plaea_speak_ger("Hier ist, wofür ich gefunden habe " + video + ".")
            if "fertig" in voice_data:
                exit()
```

```
        else:
            if "kako se zoveš" in voice_data:
                plaea_speak_cro("Moje ime je Plaea")


    # gotove funkcije
    time.sleep(0.3)                                                    # da na terminalu svakih 0.6 dolazi napisano
    language = input("What language do you prefer?: ")
    # ako ocemo pomoc od plaea
    if language == "english":
        plaea = input("Do you need help, maybe?(yes/no)")
        if plaea == "yes":
            plaea_speak("How can I help you?")
            while 0.3:
                voice_data = record_audio()
                respond(voice_data)
                if "thank you bye" in voice_data:                      # ako kazemo thank you bye vraca nas na pocetak
                    break
        # ako ne želimo pomoć od plaea
        elif plaea == "no":
            recording = input("Do you want to record what will you say?(yes/no)")
            # ako zelimo da nam snimi sto kazemo
            if recording == "yes":
                print("Plaease, say something")
                voice_data = record_audio()
                print("Audio Recorder Successfully \n")
            # ako ne zelimo da nam snimi sto kazemo
            else:
                print("Please, say something")
                while 0.3:
                    voice_data = record_audio()
                    if "leave" in voice_data:                          # ako kazemo leave vraca nas na pocetak
                        break
                    if "exit" in voice_data:                           # ako kazemo exit gotov program
                        exit()

    elif language == "german":
        plaea = input("Brauchen Sie vielleicht helfen?(ja/nein): ")
        if plaea == "ja":
            plaea_speak_ger("Womit kann ich Ihnen behilflich sein")
            while 0.3:
                voice_data = record_audio()
                respond(voice_data)
                if "danke tschüss" in voice_data:
                    break
        elif plaea == "nein":
            recording = input("Möchten Sie aufzeichnen, was Sie sagen werden? (Ja / Nein)")
            if recording == "ja":
                print("Bitte, sprechen Sie")
                voice_data = record_audio()
                print("Audiorecorder erfolgreich \n")
            # ako ne zelimo da nam snimi sto kazemo
            else:
                print("Bitte, sprechen Sie")
                while 0.3:
                    voice_data = record_audio()
                    if "verlassen" in voice_data:
                        break
                    if "fertig" in voice_data:
                        exit()
    else:
        plaea = input("Trebate pomoć(da/ne)")
        if plaea == "da":
            plaea_speak_cro("Kako Vam mogu pomoći")
            while 0.3:
                voice_data = record_audio()
                respond(voice_data)
                if "hvala vidimo se" in voice_data:
                    break
        elif plaea == "ne":
            recording = input("Želite li snimiti što kažete(da/ne)")
            if recording == "da":
                print("Molim pričajte")
                voice_data = record_audio()
                print("Uspješno snimljeno \n")
            # ako ne zelimo da nam snimi sto kazemo
            else:
                print("Molim pričajte")
                while 0.3:
                    voice_data = record_audio()
                    if "izađi" in voice_data:
                        break
                    if "gotovo" in voice_data:
                        exit()

else:  # ako u pocetku ne odredimo 1 ili 2
    break
```