Programska aplikacija za prepoznavanje emocija iz govora

Malnar, Alan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Alan Malnar

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Doc. dr. sc. Tomislav Stipančić, dipl. ing. Alan Malnar

Zagreb, 2020.

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

Zahvaljujem se mentoru doc. dr. sc. Tomislavu Stipančiću na ukazanom povjerenju, pomoći i savjetima tijekom izrade ovog rada.

Od srca se zahvaljujem svojoj obitelji i prijateljima na razumijevanju i velikoj podršci koju su mi pružali tijekom izrade ovog rada i kroz cijelo razdoblje studiranja.

Alan Malnar

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602 - 04 / 20 - 6 / 3	
Ur. broj: 15 - 1703 - 20 -	

DIPLOMSKI ZADATAK

Student: **ALAN MALNAR** Mat. br.: 0035199624

Naslov rada na hrvatskom jeziku: **Programska aplikacija za prepoznavanje emocija iz govora**

Naslov rada na engleskom jeziku: **A speech emotion recognition software application**

Opis zadatka:

Govor predstavlja prirodan način izražavanja kod ljudi. Emocije pomažu ljudima da bi kontekstualno razumjeli jedni druge. Prepoznavanje emocija iz govora objedinjuje skup metodologija koje obrađuju i klasificiraju zvučne signale kako bi se otkrile uključene emocije. Trenutni zaključci (hipoteze) o značenju emocija se temelje na prikupljenim i analiziranim akustičkim značajkama govora te na estimaciji značenja pojedinih riječi u rečenici u ovisnosti gdje se one nalaze.

U radu je potrebno izraditi cjelovito softversko rješenje afektivnog virtualnog agenta koji pretvara govor u tekst te na temelju toga zaključuje o emocionalnom stanju osobe koja govori, analizirajući pritom:

1. lingvističke značajke koje su izvučene iz sadržaja govorne poruke bazirajući se na nekoj od dostupnih baza podataka za prepoznavanje emocija, te
2. akustičke značajke koje predstavljaju mjeru verbalnog iskaza govornika.

Dobiveno softversko rješenje je potrebno eksperimentalno evaluirati uključivši ljudske subjekte.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
30. travnja 2020.	2. srpnja 2020.	6. srpnja do 10. srpnja 2020.

Zadatak zadao: Predsjednica Povjerenstva:

doc. dr. sc. Tomislav Stipančić prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	Jedinica	Opis
net		Težinska suma svih ulaza u neuron
x_i		Ulazni signali
y		Izlaz prijenosne funkcije
θ		Vrijednost praga
ω_i		Težinski faktor

SAŽETAK

Govor je ljudima primarni način komunikacije, a emocije koje se njime prenose pomažu boljem međusobnom razumijevanju. Zbog sve učestalije komunikacije između ljudi i strojeva, postaje bitno da i strojevi budu u mogućnosti prepoznati ljudske emocije. U ovom radu napravljeno je softversko rješenje virtualnog agenta koji prepoznaje emocije na temelju govora. U tu svrhu korištene su akustičke i lingvističke značajke govora. Najvažniji dio softvera čine umjetne neuronske mreže koje su učene na dostupnim bazama podataka. Izrađeni softverski model eksperimentalno je evaluiran na ljudskim subjektima.

Ključne riječi: govor, emocije, akustičke značajke, lingvističke značajke, umjetne neuronske mreže.

SUMMARY

Speech is primary means of communication between people, and the emotions it conveys help them to better understand each other. Due to more frequent communication between humans and machines, it is becoming important for machines to be able to recognize human emotions as well. In this paper, a software solution of a virtual agent which recognizes emotions based on speech is developed. For this purpose, acoustic and linguistic speech features were used. The most important part of the software consists of artificial neural networks which have been trained on available databases. The developed software model was experimentally evaluated on human subjects.

Key words: speech, emotions, acoustic features, linguistic features, artificial neural networks.

1. UVOD

1.1. Umjetna inteligencija

Proteklih nekoliko desetljeća svjedoci smo brojnim istraživanjima u svrhu razumijevanja ljudskog mozga te izgradnji sustava koji oponašaju ljudsku inteligenciju. Ljudski mozak je kompleksni organ koji služi kao vječna inspiracija za istraživanje umjetne inteligencije. Neuronske mreže ljudskog mozga vrlo su kompetentne za učenje apstraktnih koncepata visoke razine iz informacija niske razine dobivenih osjetilima. Učenje jezika, razumijevanje govora i prepoznavanje lica samo su neki od primjera koji pokazuju nevjerojatnu snagu ljudskog mozga u učenju koncepata visoke razine. Glavni cilj područja umjetne inteligencije je razviti inteligentne sustave koji su u stanju racionalno misliti i ponašati se slično kao ljudi. Postoje razna područja istraživanja koja se smatraju dijelovima umjetne inteligencije. Robotika, strojno učenje, računalni vid, obrada prirodnog jezika i automatsko zaključivanje neka su od njezinih glavnih područja. Razvijanje strojeva koji mogu komunicirati s ljudima razumijevanjem govora otvara put za izgradnju sustava koji su opremljeni inteligencijom sličnom čovjeku. Govor je najprirodniji i najprikladniji način na koji ljudi komuniciraju, a razumijevanje govora jedan je od najintrigantnijih procesa koje obavlja ljudski mozak. Sadrži lingvističke i paralingvističke informacije, a emocije su jedan ključan primjer paralingvističkih informacija koje se dijelom prenose govorom. Razvijanje strojeva koji razumiju paralingvističke informacije, poput emocija, olakšava komunikaciju između čovjeka i stroja jer istu čini jasnijom i prirodnijom [1].

1.2. Što su emocije?

U literaturi ne postoji konsenzus o definiciji emocija. Jedna od definicija glasi da su emocije složena psihološka stanja koja sadrže tri različite komponente: subjektivno iskustvo, fiziološki odgovor te bihevioralni ili izražajni odgovor [2]. Osim definiranja što su emocije, istraživači su također pokušali identificirati i klasificirati različite vrste emocija. Tako su nastala dva glavna pristupa, dimenzijski model i diskretni model. Kod dimenzijskog modela, emocije su reprezentirane količinom pobuđenosti i ugode, gdje pobuđenost određuje intenzitet emocije, dok ugoda određuje je li emocija pozitivna ili negativna. S druge strane, diskretni model nastoji emocije svrstati u određene kategorije kao što su sreća, tuga, ljutnja, strah i gađenje [3], [4].

Svaki deficit u percepciji paralingvističkih informacija loše utječe na kvalitetu komunikacije. Tvrdi se da djeca koja nisu u stanju razumjeti emocionalna stanja govornika, razvijaju slabe socijalne vještine i u nekim slučajevima pokazuju psihopatološke simptome. Ovo pokazuje

važnost prepoznavanja emocionalnih stanja u komunikaciji. Stoga je razvijanje strojeva koji razumiju paralingvističke informacije, kao što su emocije, presudno za uspostavljanje jasne i učinkovite komunikacije [1].

1.3. Prepoznavanje emocija

Prepoznavanje emocija predmet je istraživanja već dugi niz godina. Zaključivanje emocija na temelju izraza lica i bioloških mjerenja, poput otkucaja srca ili otpora kože, bili su okosnica istraživanja u prepoznavanju emocija. U novije vrijeme, prepoznavanje emocija iz govornog signala dobiva sve veću pažnju. Tradicionalni pristup ovom problemu temeljio se na činjenici da postoje odnosi između akustičkih značajki i emocija. Drugim riječima, emocija je kodirana akustičkim i prozodijskim značajkama govornih signala poput brzine govora, naglaska, intonacije, osnovne frekvencije titranja glasnica, intenziteta glasa, parametara vokalnog trakta dobivenih iz spektralne distribucije glasa te harmonijske strukture glasa [1], [4].

Znanstvenici su proučavali razne algoritme strojnog učenja koji bi klasificirali emocije na temelju nabrojanih zvučnih značajki u govoru. Skriveni Markovljevi modeli (engl. *Hidden Markov Models*, HMM), modeli s Gaussovim mješavinama (engl. *Gaussian Mixture Models*, GMM), strojevi s potpornim vektorima (engl. *Support Vector Machines*, SVM), klasifikator k-najbližih susjeda (engl. *k-Nearest Neighbor*, k-NN) i umjetne neuronske mreže su neki primjeri klasifikatora koji se široko koriste za razvrstavanje emocija na temelju zvučnih značajki govora. Učinkovitost ovih klasifikatora uglavnom ovisi o značajkama koje se smatraju vidljivim za određenu emociju, ali i o tehnikama izvlačenja istih. Iako govor prenosi velik dio emocionalnih informacija, nije dovoljan za prepoznavanje afektivnih stanja ljudi u svakodnevnim životnim situacijama. Razlog leži u tome što je prepoznavanje emocija inherentno multimodalni proces. Ostali modaliteti, poput vizualnog ili lingvističnog, također pridonose prenošenju informacija potrebnih za prepoznavanje emocija. Drugim riječima, pored govora, ljudi koriste i druge paralingvističke znakove kao što su izraz lica, govor tijela, semantika ili kontekst kako bi identificirali osjećaje kod drugih. Smatra se da 55 % poruke prilikom međusobne ljudske komunikacije prenosi govor tijela [1], [4]. Stoga je vidljivo da je za uspješno strojno prepoznavanje emocija potrebno uključiti što je moguće više modaliteta.

2. PRISTUP PROBLEMU

2.1. Preduvjeti

Kako bi se dobio uspješan sustav za prepoznavanje emocija iz govora, potrebno je riješiti tri ključne stvari, a to su:

- izbor dobre baze podataka za emocionalni govor,
- izdvajanje učinkovitih značajki,
- izrada pouzdanog klasifikatora pomoću algoritma strojnog učenja.

Izdvajanje značajki je glavni problem u sustavima za prepoznavanje emocija iz govora. Mnogi su istraživači predložili bitna govorna obilježja koja sadrže informacije o emocijama, kao što su energija, osnovna frekvencija, LPCC koeficijenti (engl. *Linear Prediction Cepstral Coefficients*), MFCC koeficijenti (engl. *Mel-scale Frequency Cepstral Coefficients*) i MSF značajke (engl. *Modulation Spectral Features*). Zbog toga se često može naići na radove u kojima se koristi kombinirani skup sastavljen od više vrsta značajki kako bi se dobilo što više informacija o emocijama iz govora. Međutim, korištenje velikog kombiniranog skupa značajki dovodi do prekomjernih dimenzija i suvišnosti govornih značajki, što proces učenja može učiniti složenim za većinu algoritama strojnog učenja i povećava vjerojatnost prenaučenosti (engl. *overfitting*), odnosno mreža napamet nauči neke značajke pa stoga pokazuje loše rezultate na novim podacima [5].

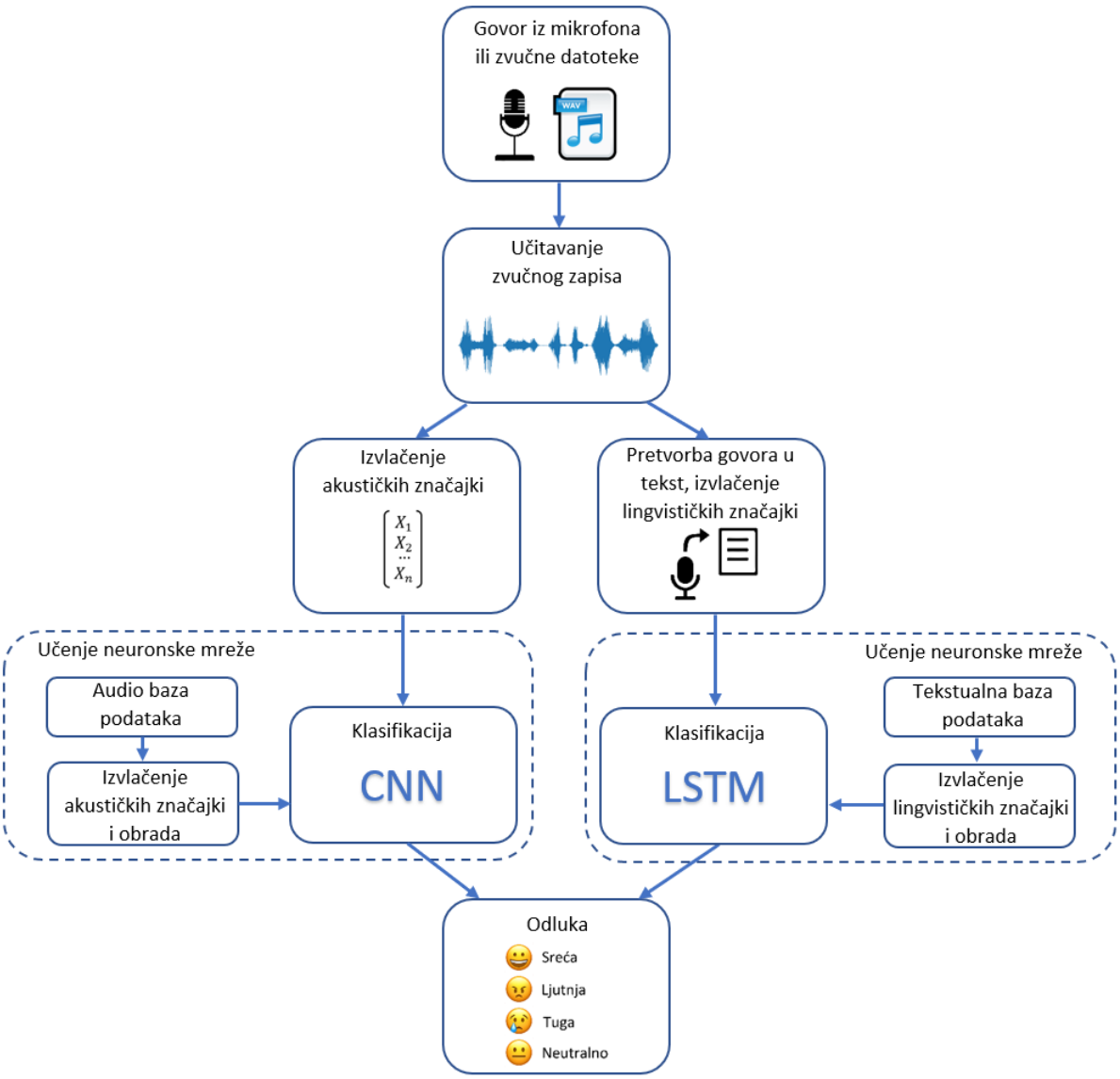
2.2. Struktura rada

Rad je strukturiran na način da čitatelju prvo pruži osnovnu teorijsku podlogu kako bi mogao lakše razumjeti principe na kojima se temelji ovaj zadatak. Zadatak je podijeljen na dva glavna dijela. U prvom dijelu obrađuje se akustički model gdje se iz govora izdvajaju akustičke značajke te se na temelju njih donosi zaključak o emocionalnom stanju osobe. U drugom dijelu obrađuje se lingvistički model gdje se govor pretvara u tekstualni oblik iz kojeg se zatim očitavaju emocije na temelju lingvističkih značajki. Oba problema rješavaju se pomoću umjetnih neuronskih mreža treniranih na dostupnim bazama podataka. Pri tome treba imati na umu sljedeće:

- emocije su subjektivne te ih ne doživljavaju svi ljudi na isti način. U uvodnom poglavlju spomenuto je kako postoje poteškoće u definiranju pojma emocije, što ima određen utjecaj i na rješavanje problema zadatka,

- budući da se neuronska mreža trenira na postojećim bazama podataka, konačan rad modela izravno ovisi o kvaliteti samih baza podataka.

Kako bi se lakše vizualizirao cijeli tijek rada ovog zadatka, na slici [Slika 1] prikazana je pojednostavljena reprezentacija u obliku dijagrama.



Slika 1. Pojednostavljeni princip rada konačnog modela

2.3. Radno okružje

2.3.1. Python

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. Do kraja 1998., Python je već imao bazu od 300 000 korisnika, a od 2000. su ga prihvatile mnoge ustanove kao što su MIT, NASA, IBM, Google, Yahoo i druge.

Python ne donosi neka nova revolucionarna obilježja u programiranju, nego na optimalan način objedinjuje sve najbolje ideje i principe rada drugih programskih jezika. On je istovremeno jednostavan i moćan programski jezik koji omogućuje programeru više razmišljanja o rješenju problema nego o samom jeziku. Na neki način može ga se smatrati hibridom jer se nalazi između tradicionalnih skriptnih jezika (kao što su Tcl, Scheme i Perl) i sistemskih jezika (kao što su C, C++ i Java). To znači da nudi jednostavnost i lako korištenje skriptnih jezika, poput Matlab-a, uz napredne programske alate koji se obično nalaze u sistemskim razvojnim jezicima. Python je besplatan za neprofitnu upotrebu i akademske ustanove, otvorenog je kôda, s izuzetno dobrom podrškom, literaturom i dokumentacijom. Također, sadrži brojne biblioteke koje olakšavaju rad pa je zbog svega toga odabran za izradu rješenja ovog rada [6].

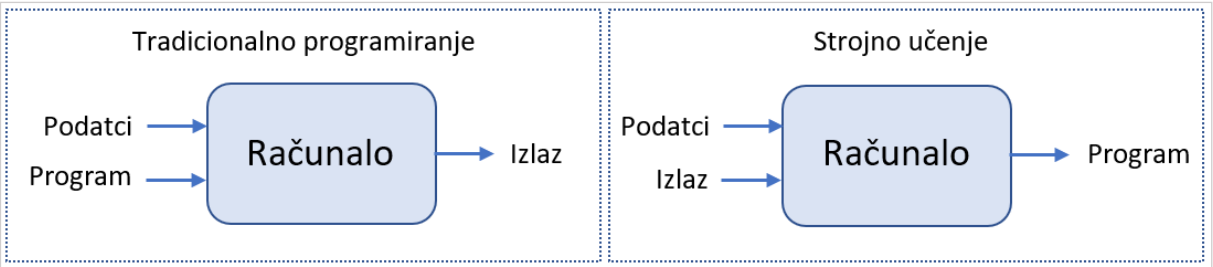
2.3.2. Jupyter Notebook

Python kôd za potrebe ovog rada pisan je unutar Jupyter Notebook-a. Jupyter Notebook je besplatna interaktivna web-aplikacija koja omogućuje stvaranje i razmjenu dokumenata koji sadrže računalni kôd i bogate tekstualne elemente (odlomak, jednadžbe, slike, poveznice itd.). 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. Navedena web-aplikacija često se koristi za obradu i vizualizaciju podataka, provođenje numeričkih simulacija, statističko modeliranje, strojno učenje i još mnogo toga. Podržava preko 40 programskih jezika, uključujući i Python [7], [8].

3. TEORIJSKA PODLOGA

3.1. Strojno učenje

Velik dio današnje digitalne ekonomije izgrađen je na zamršenim lancima osnovnih algoritama koji su marljivo spajani generacijama inženjera. No, većina tih sustava nije sposobna prilagoditi se promjenama. Sve konfiguracije i njihove izmjene moraju izvoditi visoko obučeni inženjeri, što sustav čini krhkim. Rješenje tog problema nudi strojno učenje. Strojno učenje je dio računalne znanosti koje računalima daje sposobnost učenja bez da su eksplicitno programirana. Taj pojam i definiciju osmislio je Arthur Samuel 1959. godine. Takav pristup omogućuje sustavima fleksibilnost i sposobnost da se dinamički prilagođavaju, a željeno ponašanje uče iz baza podataka. S dolaskom novih podataka, takvi se sustavi mogu redovito usavršavati. Na taj način vrlo sofisticirani softverski sustavi, pogonjeni strojnim učenjem, u mogućnosti su promijeniti svoje ponašanje bez većih promjena u kôdu, potrebno je samo izmijeniti podatke o obuci. Ovaj trend ima tendenciju drastičnog ubrzavanja jer alati za strojno učenje postaju sve jednostavniji za korištenje [9].



Slika 2. Usporedba tradicionalnog programiranja i strojnog učenja

[Slika 2] prikazuje pojednostavljenu usporedbu tradicionalnog programiranja i strojnog učenja. Tradicionalno programiranje je ručni proces. Da bi se dobio željeni izlaz, odnosno rezultat, potrebno je napraviti program za obradu ulaznih podataka na određen način. Umjesto da ljudi pišu program koji će pružati željene rezultate, računalu se daju ulazni podatci i željeni izlazni podatci na temelju kojih će ono samostalno napraviti program koji tu zadaću obavlja i to je osnovna ideja strojnog učenja.

Postoje tri glavne kategorije strojnog učenja: nadzirano, nenadzirano i podržano/ojačano učenje (engl. *supervised, unsupervised, reinforcement learning*). Kod nadziranog učenja unaprijed su poznate labele (oznake, kategorije) kojima ulazni podatci pripadaju. Algoritam nadziranog strojnog učenja korištenjem skupa ulaznih parova *(X, y) = (ulazni podatci, labele)* stvara matematički model kako bi pronašao funkciju koja će pravilno mapirati različite ulaze na željene izlaze: $y = f(X)$. Nenadzirano učenje koristi skup ulaznih podataka koji nisu označeni i

prepušta se algoritmu da samostalno pronađe određene obrasce u podatcima koji imaju slične karakteristike. Kod podržanog/ojačanog učenja, algoritam poduzima određenu radnju kako bi maksimizirao numeričku nagradu. Ne zadaje mu se što mora poduzeti, nego on samostalno na temelju prijašnjeg iskustva mora otkriti koje radnje rezultiraju najvećom nagradom [10].

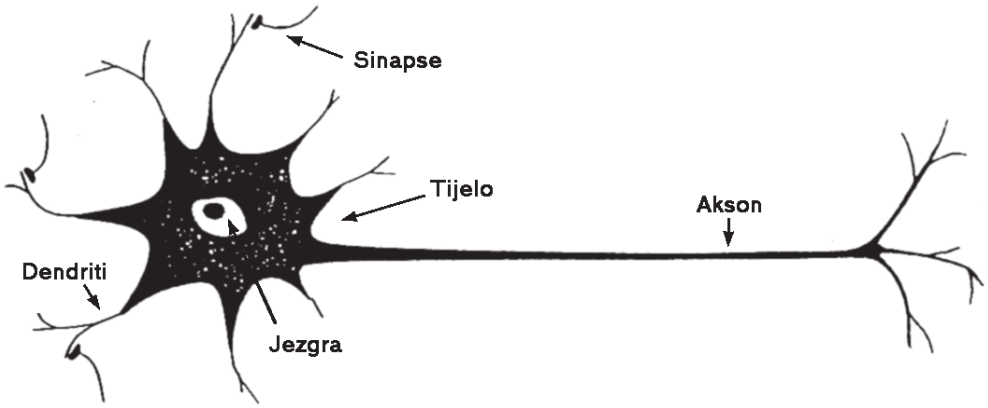
Mnoga su područja u kojima se strojno učenje može primijeniti, primjerice: pretraživanje weba, računalna biologija, medicina, internet trgovina, istraživanje svemira, robotika, izvlačenje informacija, socijalne mreže i drugo.

3.2. Umjetna neuronska mreža

Umjetna neuronska mreža jedan je od algoritama strojnog učenja, a inspirirana je biološkim živčanim sustavima. Stoga ćemo se kratko osvrnuti na njih.

3.2.1. Biološki i umjetni neuron

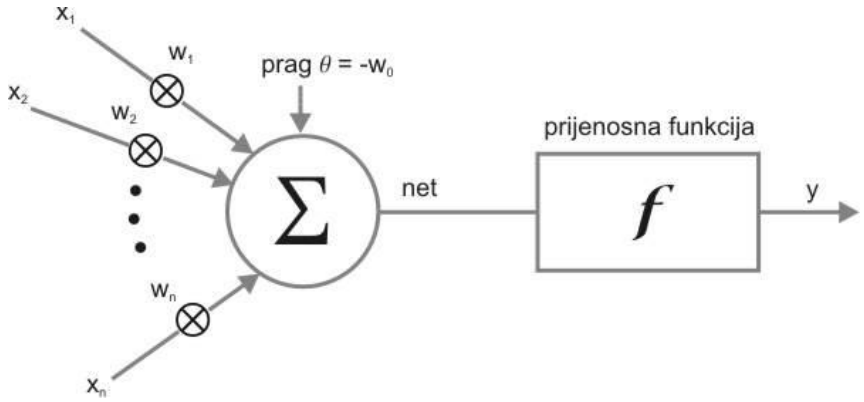
Sastavni dio mozga čine neuroni, odnosno živčane stanice, a ljudski mozak ima ih oko 10^{11}. Postoji više od 100 vrsta neurona koji su adekvatno prema svojoj funkciji konfigurirani u točno definiranom rasporedu. U prosjeku, svaki je neuron povezan s 10^4 drugih neurona. Sastoji od tri osnovna dijela: tijela stanice, dendrita i aksona [11]. [Slika 3] prikazuje građu neurona.



Slika 3. Građa biološkog neurona [12]

Tijelo stanice sadrži informaciju predstavljenu električkim potencijalom između unutrašnjeg i vanjskog dijela stanice. Na sinapsama, spojnim mjestima između dva neurona, primaju se informacije od drugih neurona u obliku post-sinaptičkog potencijala koji utječe na potencijal stanice povećavajući ga ili smanjujući. U tijelu stanice zbrajaju se post-sinaptički potencijali tisuća susjednih neurona. Ako ukupni napon prijeđe određeni prag, neuron se „pali“ i stvara tzv. akcijski potencijal. Kada se informacija akcijskim potencijalom prenese do završnih članaka, oni proizvode i otpuštaju kemikalije (neurotransmitere) - ovisno o veličini potencijala, što opet započinje niz opisanih događaja u idućim neuronima [11].

McCulloch-Pitts model umjetnog neurona [Slika 4], tzv. *Threshold Logic Unit* (TLU), imitira funkcionalnost biološkog neurona. Model koristi sljedeću analogiju: signali su reprezentirani numeričkim iznosom te se na ulazu u neuron množe težinskim faktorom, koji opisuje jakost sinapse. Signali pomnoženi težinskim faktorima se zatim zbrajaju analogno zbrajanju potencijala u tijelu stanice. Ako je dobiveni iznos iznad definiranog praga, neuron daje izlazni signal. Općenito, umjetni neuron umjesto praga ima neku funkciju, tzv. prijenosnu ili aktivacijsku funkciju [11].



Slika 4. Model umjetnog neurona [11]

Ulazni signali označeni su s x_1, x_2, … x_n, a težine s ω_1, ω_2, … ω_n. U općem slučaju, ulazni signali su realni brojevi u intervalu [-1, 1] ili [0, 1]. Težinska suma *net* dana je izrazom:

$$net = \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n - \theta, \quad \textbf{(1)}$$

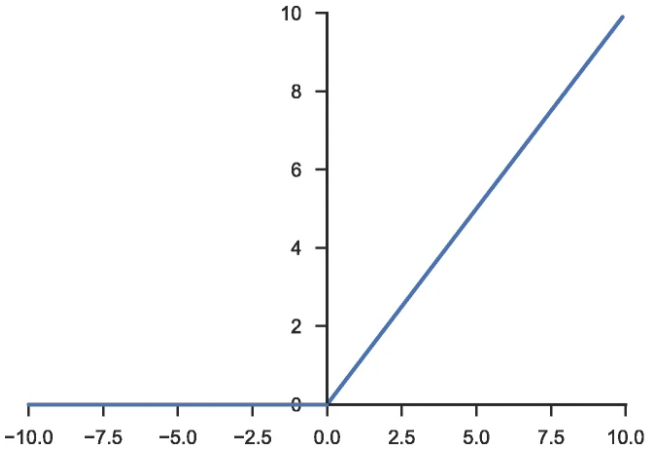
no često se zbog jednostavnosti uzima da je vrijednost praga $\theta = -\omega_0$ i dodaje se ulazni signal x_0 s fiksnom vrijednošću 1, što se onda može napisati ovako:

$$net = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n = \sum_{i=0}^{n} \omega_i x_i \,. \quad \textbf{(2)}$$

Izlaz prijenosne funkcije izgleda ovako:

$$y = f(net). \quad \textbf{(3)}$$

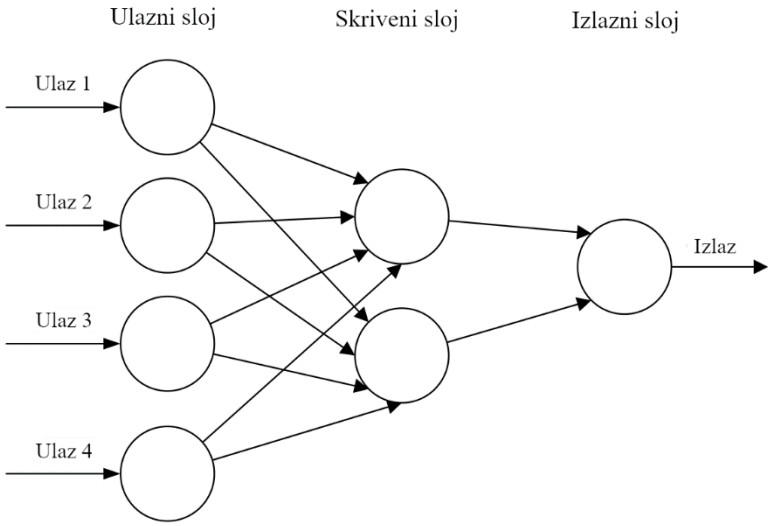
Postoji više vrsta aktivacijskih funkcija, a među najčešće korištenim je *ReLU* (engl. *Rectified Linear Unit*) [Slika 5]. Ona jednostavno računa izraz $f(x) = \max(0, x)$, što znači da na izlazu daje nulu ako je ulaz manji od nule, a u suprotnom prosljeđuje vrijednost ulaza [13].



Slika 5. *ReLU* aktivacijska funkcija [13]

3.2.2. Svojstva umjetne neuronske mreže

U širem smislu riječi, umjetna neuronska mreža je pojednostavljena replika ljudskog mozga kojom se pokušava simulirati postupak učenja. Ona je skup međusobno povezanih jednostavnih procesnih elemenata, jedinica ili čvorova [Slika 6], čija se funkcionalnost temelji na biološkom neuronu.



Slika 6. Unaprijedna neuronska mreža [14]

Cilj neuronske mreže je aproksimacija neke funkcije. Pri tome je snaga obrade mreže pohranjena u vezama između pojedinih neurona tj. težinama. One se dobivaju postupkom učenja iz skupa podataka za učenje. Učenje je proces kojim se nastoji smanjiti pogreška na izlazu. Neuronske mreže imaju mnoge prednosti u odnosu na konvencionalne metode obrade podataka:

- vrlo dobro procjenjuju nelinearne odnose uzoraka,
- mogu raditi s velikim brojem varijabli ili parametara,

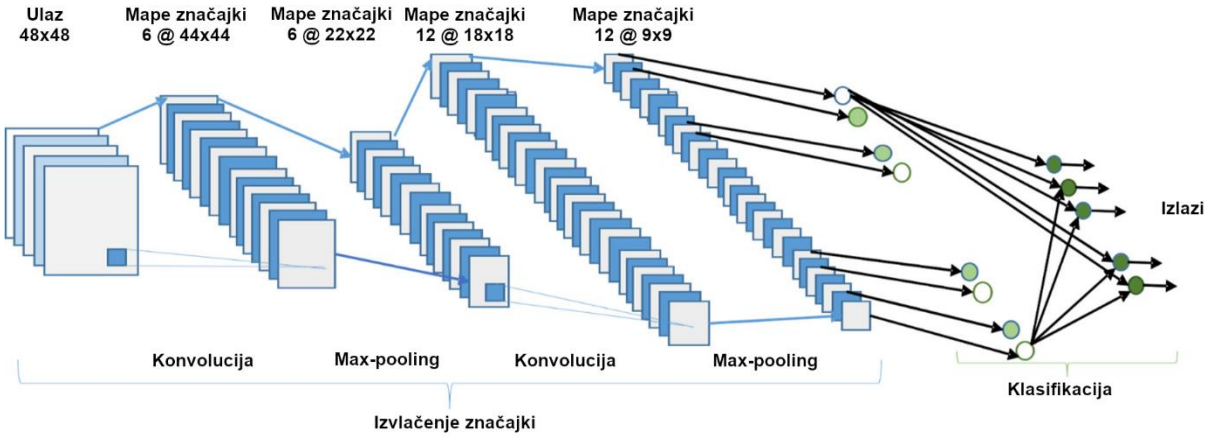
- robusne su na pogreške u podatcima te mogu raditi s nepotpunim podatcima,
- stvaraju vlastite odnose između podataka koji nisu zadani na eksplicitan simbolički način,
- sposobne su formirati znanje učeći iz iskustva,
- prilagodljive su okolini.

Neuronske mreže odlične su u rješavanju problema klasifikacije i predviđanja, odnosno svih problema kod kojih postoji odnos između ulaznih i izlaznih varijabli, bez obzira na nelinearnost i visoku složenost te veze [11].

3.3. Neuronske mreže korištene u ovom radu

3.3.1. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže (engl. *Convolutional Neural Networks*, CNN) su jedan od popularnijih oblika umjetnih neuronskih mreža [Slika 7]. Sadrže više skrivenih slojeva i stoga pripadaju kategoriji dubokih neuronskih mreža (engl. *Deep Neural Networks*).



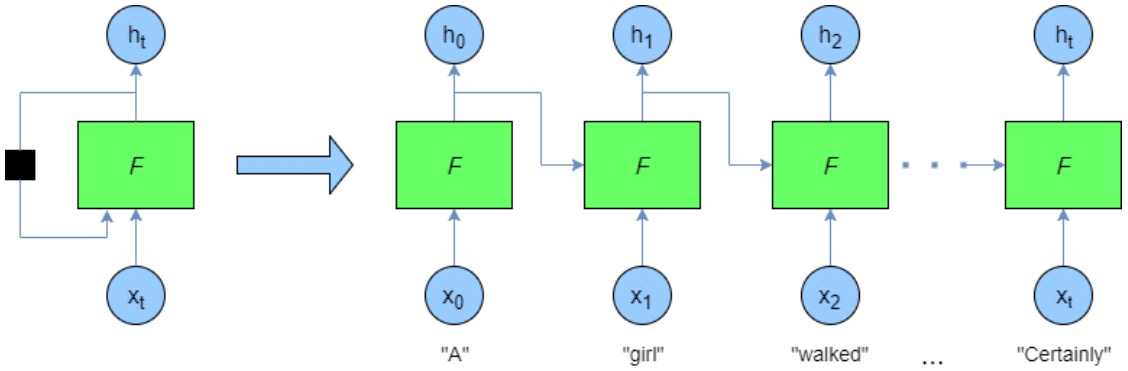
Slika 7. Arhitektura konvolucijske neuronske mreže [15]

Konvolucijske mreže mogu sadržavati tri glavne vrste sloja, konvolucijski sloj (engl. *convolutional layer*), sloj sažimanja (engl. *max-poolig layer*) te potpuno povezani sloj (engl. *dense layer*). Prva dva sloja vrše izvlačenje značajki, dok treći preslikava izvučene značajke na izlaz, kao što je npr. klasifikacija. U konvolucijskom sloju vrši se linearna matematička operacija s matricama, zvana konvolucija, na način da se filterom prolazi kroz ulazne podatke. Kada se pronađe određeni uzorak u podatcima, on se preslikava u prostor značajki karakterističan za taj element. Na kraju konvolucijskog sloja nalazi se aktivacijska funkcija koja prilagođava raspon vrijednosti podataka. Sloj sažimanja, jednostavno rečeno, smanjuje količinu informacija potrebnih za učenje na način da uzima dominantne vrijednosti, dok ostale odbacuje. Svi spomenuti slojevi mogu se kombinirati na različite načine kako bi se postigle što bolje

performanse mreže. S obzirom na protok informacija kroz strukturu, konvolucijske neuronske mreže su unaprijednog tipa. Kod unaprijednih neuronskih mreža (engl. *Feedforward Neural Networks*) ne postoje povratne petlje, stoga informacije putuju samo naprijed kroz mrežu, najprije kroz ulazne čvorove, zatim kroz skrivene čvorove i na kraju kroz izlazne čvorove. Naučena mreža će za pojedini ulazni podatak dati odgovarajući izlaz, odnosno klasu, no ta odluka neće imati utjecaja na iduću klasifikaciju. Zbog svojih karakteristika, konvolucijske neuronske mreže se pretežno koriste u raspoznavanju uzoraka unutar slika [13].

3.3.2. Povratne neuronske mreže

Povratne neuronske mreže (engl. *Recurrent Neural Networks*, RNN) su posebne po tome što za razliku od unaprijednih mreža imaju povratnu vezu. To im omogućuje da izlaz iz prethodnog vremenskog koraka prosljeđuju na ulaz u trenutni vremenski korak i na taj način utječu na rezultat. Zbog toga je model povratne neuronske mreže dinamičan i nakon procesa učenja. Na slici [Slika 8] može se vidjeti da mreža istovremeno za podatke uzima trenutni ulaz i izlaz iz prethodnog vremenskog koraka.

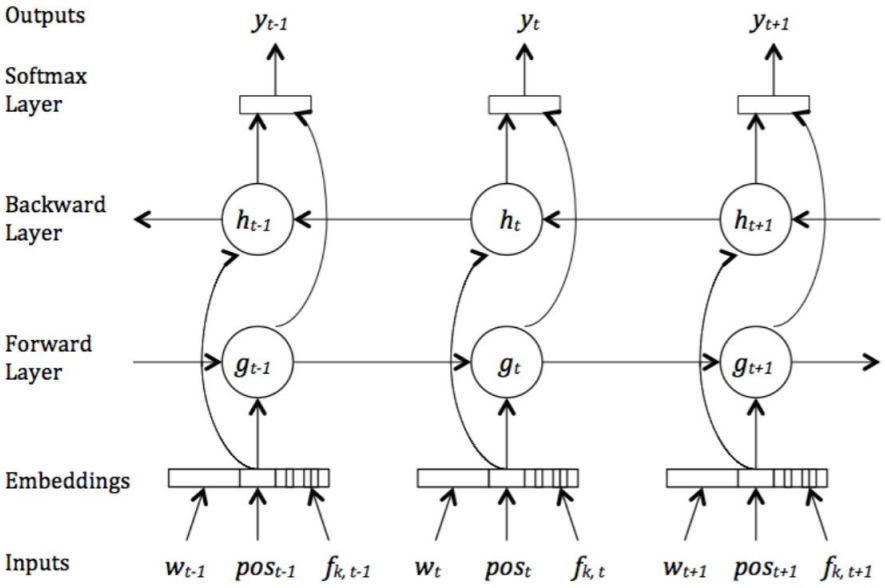


Slika 8. Povratna neuronska mreža [16]

Ove mreže koriste se za učenje sekvencijalnih podataka kao što su tekst, zvuk i podatci sa senzora. [Slika 8] također prikazuje i „odmotanu“ mrežu kojom je demonstriran način unošenja sekvencijalnih podataka. Pamteći prethodne informacije dok obrađuju trenutne, u mogućnosti su povezati niz događaja. Problem je što nakon određenog broja sekvenci, mreža počne „zaboravljati“ starije podatke, što se naziva problem nestajućeg gradijenta (engl. *vanishing gradient problem*). Taj problem rješava naprednija verzija povratne mreže, a to je LSTM (engl. *Long Short-Term Memory*, duga kratkoročna memorija). Ona je sposobna pamtiti bitne informacije kroz mnogo koraka i zbog toga može dobro razumjeti kontekst [17], [18].

Kao i kod konvolucijske neuronske mreže, LSTM neuronska mreža sastoji se od više slojeva. Ugradbeni sloj (engl. *embedding layer*) na efikasan način pretvara ulazne podatke u vektorske zapise koji sadrže informaciju o međusobnoj ovisnosti i odnosima. U LSTM sloju vrši se učenje

odnosa među podacima kako bi se shvatio kontekst. [Slika 9] prikazuje dva LSTM sloja koji podatke uče sekvencijalno u oba smjera. Takva konfiguracija naziva se dvosmjerna LSTM mreža (engl. *bidirectional LSTM network*) i sastoji se od unaprijednog i unazadnog sloja (engl. *forward, backward layer*). Unaprijedni sloj uči sekvencijalne podatke redom kako dolaze, dok unazadni sloj uči obrnutu kopiju ulaznih podataka. Takav način dodatno poboljšava shvaćanje konteksta i omogućuje brže učenje [19]. Na kraju mreže nalazi se izlazni sloj, a u slučaju klasifikacije podataka, taj sloj sadrži *Softmax* aktivacijsku funkciju. Spomenuta funkcija dodjeljuje vjerojatnost pripadanja svakoj od mogućih klasa. Klasa s najvećom vjerojatnošću uzima se kao konačna predikcija.



Slika 9. Arhitektura LSTM neuronske mreže [20]

3.4. Prenaučenost neuronskih mreža

Prenaučenost je čest problem kod neuronskih mreža. Događa se kada neuronska mreža podatke za učenje nauči toliko dobro da izgubi svojstvo generalizacije. Drugim riječima, mreža nauči napamet određene obrasce koji se pojavljuju u podatcima za učenje, a irelevantni su za ostale podatke. Cilj neuronskih mreža je dobiti model koji radi dobro i na podatcima za učenje i na novim podatcima koje će model koristiti za predikciju [21]. Postoji više metoda kojima se nastoji smanjiti prenaučenost, a ovdje će biti spomenute neke od najčešćih.

3.4.1. Učenje mreže na više primjera

Prednost dubokih neuronskih mreža je u tome što se njihov učinak nastavlja poboljšavati s porastom količine podataka. Stoga je prvi korak za suzbijanje prenaučenosti prikupljanje veće količine podataka. Porastom podataka, model nije u mogućnosti „prenaučiti“ sve primjere i na taj način se prisiljava na generalizaciju [22]. Problem je što nije uvijek jednostavno prikupiti

dovoljnu količinu podataka. U tome djelomično može pomoći metoda proširenja podataka (engl. *data augmentation*) kojom se od postojećih podataka umjetno nastoje dobiti novi podatci koristeći razne tehnike poput pomicanja, zrcaljenja, dodavanja distorzija i sl.

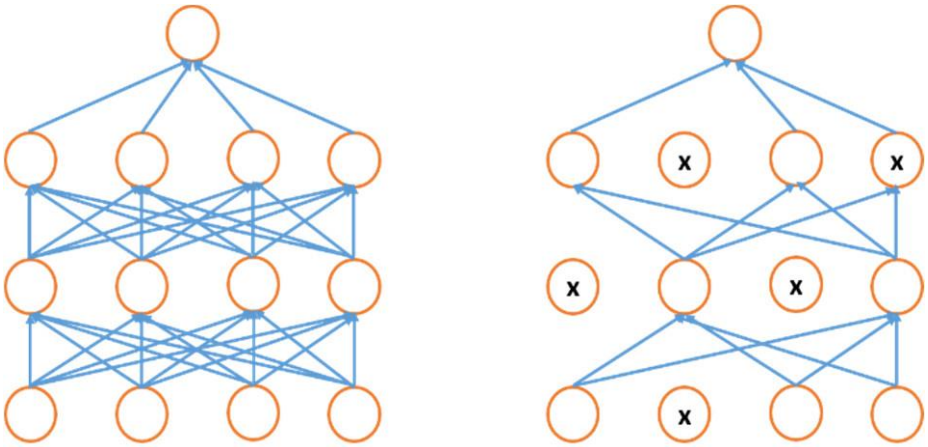
3.4.2. Rano zaustavljanje

Prilikom učenja neuronske mreže, jedno od važnih pitanja je kada prestati s učenjem. Ako se prestane prerano, model neće uspjeti naučiti dovoljno značajki. S druge pak strane, predugo učenje dovodi do prenaučenosti. Zbog toga se koristi tehnika ranog zaustavljanja (engl. *early stopping*). Podatci za učenje podijele se na dio za učenje i dio za validaciju. Validacija služi za otkrivanje trenutka kod kojeg dolazi do prenaučenosti. Kada greška validacije počne rasti, to je znak da dolazi do prenaučenosti i tada je poželjno zaustaviti proces učenja [Slika 10] [22].

Slika 10. Rano zaustavljanje tijekom učenja [23]

3.4.3. Isključivanje

Isključivanje (engl. *Dropout*) je jednostavna i efikasna metoda za sprječavanje prenaučenosti i može se upotrijebiti u gotovo svakoj vrsti mreže. Tijekom učenja, nasumično se zanemaruju izlazi pojedinih čvorova u mreži, odnosno njihova vrijednost postavlja se na nulu [Slika 11].



Slika 11. Slikovita reprezentacija *Dropout* koncepta [15]

To za posljedicu ima stvaranje novih konfiguracija neuronskih mreža u svakom koraku učenja. Na taj način mreža je prisiljena naučiti značajke koje su robusnije [24].

3.4.4. Normalizacija serije

Normalizacija serije (engl. *batch normalization*) je relativno nova metoda koja također pokazuje dobre rezultate. Podatci se unutar slojeva linearno transformiraju kako bi srednja vrijednost bila nula, a varijanca jedan. Osim što pomaže kod prenaučenosti, normalizacija također ubrzava proces učenja i čini ga stabilnijim [15].

4. AKUSTIČKI MODEL

U ovom poglavlju obradit će se potrebni koraci za analizu akustičkih značajki govora. Prvo će se dati kratki pregled audio baza podataka korištenih u ovom radu. Zatim će se te baze pripremiti kako bi bile prikladne za upotrebu. Nakon toga će se izvući potrebne značajke koje će činiti skup podataka za učenje neuronske mreže. Izradit će se model konvolucijske neuronske mreže koja će zatim učiti navedene podatke. Kao što je već spomenuto, sav programski dio rada izrađen je u web okruženju Jupyter Notebook koristeći Python programski jezik. Redom će biti prikazani i objašnjeni važniji dijelovi kôda, dok se u prilogu nalazi izlistan cjeloviti kôd.

4.1. Audio baze podataka

4.1.1. SAVEE baza podataka

Surrey Audio-Visual Expression Emotion (SAVEE) [25] je britanska baza podataka emocionalnog govora koja sadrži audio datoteke sa sedam oznaka emocija: sreća, tuga, bijes, strah, gađenje, iznenađenje i neutralno. Četiri muška glumca izvela su 15 izjava za svaku emociju. Baza sveukupno ima 480 audio zapisa u *.wav* formatu čija su imena sastavljena na način da prva dva slova označavaju inicijale osobe koja je izgovorila tu izjavu, nakon inicijala slijedi prvo slovo emocije izvedene tom izjavom i na kraju dolazi redni broj izjave. Na primjer, „DC_a01.wav“ označava prvu izjavu ljutnje koju je izvela osoba DC.

4.1.2. CREMA-D baza podataka

Crowd-sourced Emotional Mutimodal Actors Dataset (CREMA-D) [26] je baza podataka emocionalnog govora koja sadrži 7442 audio zapisa sa šest emocija: ljutnja, gađenje, strah, sreća, tuga i neutralno. Te zapise su izveli 43 ženska i 48 muških glumaca u dobi između 20 i 74 godine, različitih rasa i etničkih pripadnosti. Svaka osoba izvela je 12 rečenica za pojedinu emociju u nekoliko intenziteta. Slično kao i u prethodnom primjeru, imena audio zapisa izvedena su na način da u sebi sadrže identifikacijsku oznaku osobe, oznaku emocije i još neke dodatne informacije.

4.1.3. RAVDESS baza podataka

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [27] je baza podataka emocionalnog govora koja sadrži 1440 audio zapisa. 24 profesionalnih glumaca (12 žena i 12 muškaraca) izveli su dvije izjave u sedam emocija i dva intenziteta. Imena audio zapisa su brojčano kodirana i iz njih se mogu očitati informacije o emocijama.

4.1.4. TESS baza podataka

Toronto emotional speech set (TESS) [28] je baza podataka emocionalnog govora koja sadrži 2800 audio zapisa. Izveli su ih dvije glumice od 26 i 64 godine starosti, iskazujući sedam emocija. Nazivi emocija sadržani su u imenima zvučnih zapisa.

4.1.5. Emo-DB baza podataka

Berlin Database of Emotional Speech (Emo-DB) [29] je njemačka baza podataka emocionalnog govora koja sadrži 535 audio zapisa. Pet žena i pet muškaraca izgovorilo je 10 izjava za svaku emociju. Za razliku od prethodnih baza podataka, ova ima izjave izrečene na njemačkom jeziku, no svejedno je uzeta u obzir budući da je potrebno što više podataka. U imenima zvučnih zapisa nalazi se početno slovo emocije na njemačkom jeziku.

4.2. Pripremanje baze podataka

Analizom signala zvučnih zapisa, utvrđeno je da postoje razlike u glasnoći među pojedinim bazama podataka, a to može loše utjecati na rad neuronske mreže. Zbog toga je potrebno provesti normalizaciju svih zvučnih zapisa kako bi bili ujednačene glasnoće. U tu svrhu korišten je program Audacity. To je besplatan softver otvorenog kôda za obradu zvučnih zapisa [30].

Nakon provedene normalizacije, iz svake se baze podataka uzimaju izjave sa sljedećim emocijama: ljutnja, sreća, neutralno i tuga (engl. *anger, happiness, neutral, sadness*). Iako baze sadrže i izjave drugih emocija, spomenute četiri su odabrane zbog jednostavnijeg procesa učenja. Osim podjele na emocije, koristi se i posebna kategorija za spol, tako da ukupno postoji 8 kategorija. Razlog tome je što postoje značajnije razlike u karakteristikama muškog i ženskog glasa, pa bi neuronska mreža davala lošije rezultate ako ne bi bilo podjele prema spolu.

Na početku svakog programa potrebno je učitati sve biblioteke s kojima će se raditi [Slika 12].

```
1 # Ucitavanje potrebnih biblioteka
2 import pandas as pd
3 import os
```

Slika 12. Učitavanje potrebnih biblioteka

Zatim se na jednom mjestu definira relativna putanja svake baze [Slika 13].

```
1 # Definiranje lokacije svake baze
2 TESS = "./database/tess/TESS Toronto emotional speech set data/"
3 RAV = "./database/ravdess-emotional-speech-audio/audio_speech_actors_01-24/"
4 SAVEE = "./database/surrey-audiovisual-expressed-emotion-savee/ALL/"
5 CREMA = "./database/cremad/AudioWAV/"
6 EmoDB = "./database/emodb/wav/"
```

Slika 13. Definiranje lokacije korištenih baza podataka

Nakon što se učita lokacija pojedine baze, pomoću *for* petlje prolazi se kroz imena svih zvučnih zapisa te se iz njih očitava pripadna emocija. Ako se očitana emocija podudara sa željenim emocijama, onda se ona sprema u listu *emotions* (emocije) s odgovarajućim imenom. Svaka druga emocija sprema se u listu pod imenom *unknown* (nepoznato). Po potrebi, može se promijeniti koje emocije se spremaju, a koje ne. Također, u svakom koraku se u listu *path* (put) sprema put do trenutne datoteke. Na posljetku se ime svake datoteke zajedno s pripadnim putom i emocijom sprema u tablicu [Slika 14]. Princip je sličan i za ostale baze, uz potrebne prilagodbe zbog različitog načina označavanja emocija.

```
# Dohvacanje lokacije SAVEE baze podataka
dir_list = os.listdir(SAVEE)

# Ocitavanje emocija na temelju imena datoteka
emotion = []
path = []

for i in dir_list:
    if i[-8:-6] == '_a':
        emotion.append('male_angry')
    elif i[-8:-6] == '_h':
        emotion.append('male_happy')
    elif i[-8:-6] == '_n':
        emotion.append('male_neutral')
    elif i[-8:-6] == 'sa':
        emotion.append('male_sad')
    else:
        emotion.append('unknown')
    path.append(SAVEE + i)

# Postavljanje podataka u tablicu i provjera raspodjele labela
SAVEE_df = pd.DataFrame(emotion, columns = ['labels'])
SAVEE_df['source'] = 'SAVEE'
SAVEE_df = pd.concat([SAVEE_df,pd.DataFrame(path, columns = ['path'])], axis=1)
SAVEE_df.labels.value_counts()
```

Slika 14. Očitavanje emocija na temelju imena zvučnih zapisa

Nakon što se postupak napravi i za ostale baze, potrebno je sve prikupljene podatke spojiti u jednu zajedničku tablicu. Uklanjaju se svi oni podaci koji se ne odnose na željene emocije te se tablica sprema lokalno u obliku datoteke za kasniju upotrebu [Slika 15].

```python
# Spajanje svih podataka u jednu zajednicku tablicu
df = pd.concat([SAVEE_df, RAV_df, TESS_df, CREMA_df, EmoDB_df], axis = 0)

# Uklanjanje nepotrebnih labela
df = df[df.labels != 'unknown']
df = df[df.labels != 'female_unknown']
df = df[df.labels != 'male_unknown']

print(df.labels.value_counts())
df.head()
df.to_csv("data_path.csv", index=False)
```

```
female_angry      1163
female_happy      1140
female_sad        1133
female_neutral    1000
male_angry         887
male_happy         854
male_sad           852
male_neutral       782
Name: labels, dtype: int64
```

Slika 15. Zajednička baza podataka

Na slici [Slika 15] također je prikazana i konačna raspodjela pojedinih emocija nakon što su obrađene sve baze podataka. Iako postoji određena razlika u distribuciji, taj odnos je zadovoljavajući. Na slici [Slika 16] vidljivo je prvih pet ispisanih redaka iz spremljene datoteke.

	labels	source	path
0	male_angry	SAVEE	./database/surrey-audiovisual-expressed-emotio...
1	male_angry	SAVEE	./database/surrey-audiovisual-expressed-emotio...
2	male_angry	SAVEE	./database/surrey-audiovisual-expressed-emotio...
3	male_angry	SAVEE	./database/surrey-audiovisual-expressed-emotio...
4	male_angry	SAVEE	./database/surrey-audiovisual-expressed-emotio...

Slika 16. Izvadak iz zajedničke baze podataka

4.3. Izdvajanje značajki

Nakon što su učitane potrebne biblioteke, potrebno je učitati prethodno kreiranu datoteku u kojoj su spremljene emocije i putanje za svaki zvučni zapis [Slika 17].

```python
# Ucitavanje baze podataka
database = pd.read_csv("data_path.csv")
database.head()
```

Slika 17. Učitavanje zajedničke baze podataka

Iz svakog zvučnog zapisa potrebno je izvući značajke kod kojih je govor na neki način povezan s emocijama. Glavna stvar koju treba razumjeti u govoru jest da glas koji stvara čovjek nastaje prolaskom zraka iz pluća kroz glasnice. Titranjem glasnica formira se harmoničan signal koji

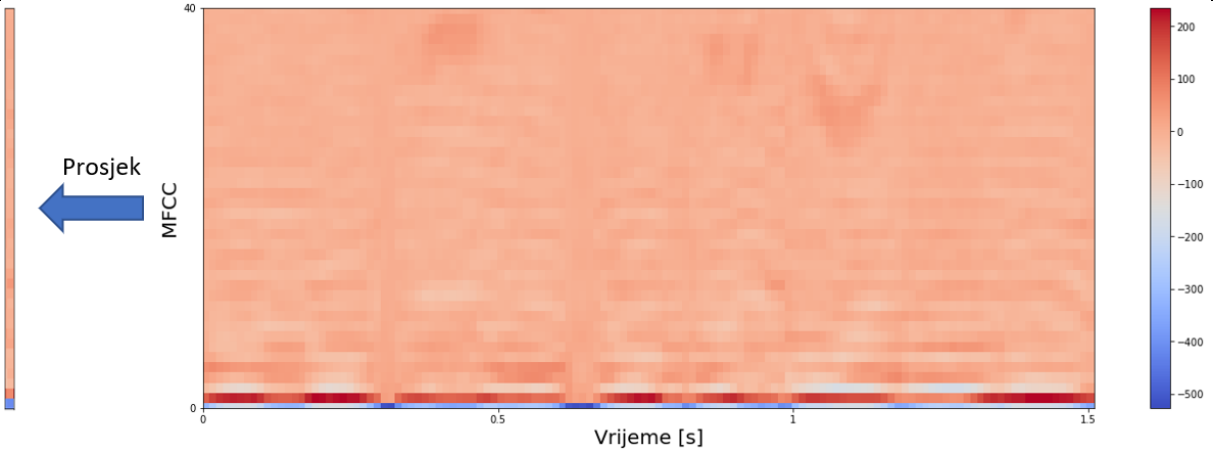
putuje dalje kroz vokalni trakt, uključujući jezik, zube itd. Oblik tog vokalnog trakta određuje kakav zvuk će izaći. Ako se može ispravno odrediti taj oblik, dobiva se točna reprezentacija fonema koji nastaju. Oblik vokalnog trakta očituje se u ovojnici kratkoročnog spektra snage, a to je prikazano u MFCC koeficijentima (engl. *Mel-scale Frequency Cepstral Coefficients*) [31]. Stoga je MFCC odabran za prvu značajku. Druga značajka je spektrogram, koji u suštini prikazuje spektar frekvencija zvučnog signala kroz vrijeme. Obje značajke nastaju pretvorbom iz vremenske domene u frekvencijsku postupkom brze Fourierove transformacije na pojedinom vremenskom okviru. Koristi se Mel skala jer ona odgovara načinu na koji ljudi čuju zvuk. Drugim riječima, ljudi ne doživljavaju frekvencije na linearnoj skali, nego bolje uočavaju razlike pri nižim frekvencijama.

Pomoću *for* petlje prolazi se kroz učitanu datoteku te se na temelju upisane putanje do pojedinog zvučnog zapisa iz istog izvlače potrebne značajke koristeći *LibROSA* biblioteku. Pri tome se svakom zvučnom zapisu prethodno uklone dijelovi bez zvuka na početku i kraju pomoću spomenute biblioteke [Slika 18]. *LibROSA* je vrlo snažna biblioteka za analizu i obradu zvučnih zapisa [32].

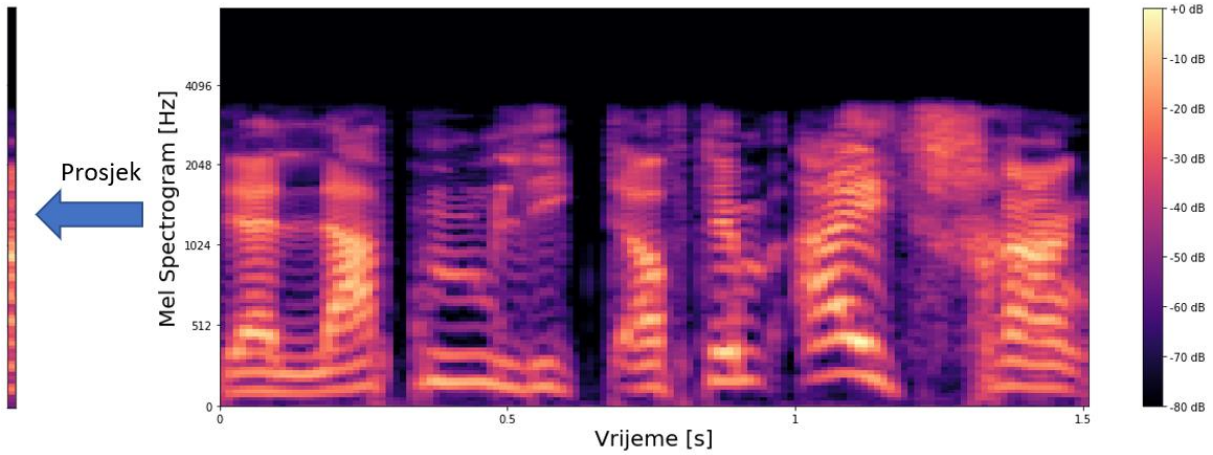
```
# Mjesto u koje ce se spremati znacajke
df = pd.DataFrame(columns=['feature'])

# Iterativno izvlacenje znacajki iz cijele baze podataka
counter=0
for index, path in enumerate(database.path):
    print(path)
    # Ucitavanje pojedine datoteke
    data, sample_rate = librosa.load(path, res_type='kaiser_fast', sr=44100)
    sample_rate = np.array(sample_rate)
    # Uklanjanje tisine sa pocetka i kraja datoteke
    data, _ = librosa.effects.trim(y=data, top_db = 20)

    result=np.array([])
    # Izvlacenje MFCC znacajki
    mfccs=np.mean(librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40),axis=1)
    result=np.hstack((result, mfccs))
    # Izvlacenje MEL znacajki
    mel=np.mean(librosa.feature.melspectrogram(data, sr=sample_rate), axis=1)
    result=np.hstack((result, mel))

    df.loc[counter] = [result]
    counter=counter+1
```

Slika 18. Izvlačenje akustičkih značajki

Potrebno je napomenuti kako se ovdje uzima srednja vrijednost značajki u čitavom vremenu trajanja zvučnog zapisa, što je simbolički prikazano na slikama [Slika 19] i [Slika 20]. Time se dobiva skup vrijednosti značajki čije su konačne dimenzije neovisne o duljini trajanja zvučnog zapisa.



Slika 19. Grafički prikaz MFCC koeficijenata za jedan zvučni zapis



Slika 20. Spektrogram jednog zvučnog zapisa

Izvlačenje značajki je procesorski poprilično zahtjevan zadatak koji traje određeno vrijeme budući da se iz svakog pojedinog zvučnog zapisa očitavaju potrebne informacije. Zbog toga je poželjno spremiti navedene značajke u datoteku kako bi se izbjeglo ponavljanje prethodno opisanog procesa. Te značajke se zatim pridružuju postojećoj tablici s oznakama emocija i njihovim putanjama [Slika 21].

```
# Spremanje izvucenih znacajki
df.to_pickle('df_extracted_features.pkl')
df[:5]
```

```
# Ucitavanje izvucenih znacajki
df = pd.read_pickle('df_extracted_features.pkl')
```

```
# Pridruzivanje izvucenih znacajki u postojecu tablicu
df = pd.concat([database, pd.DataFrame(df['feature'].values.tolist())],axis=1)
df[:5]
```

	labels	source	path	0	1	2	3	4	
0	male_angry	SAVEE	./database/surrey-audiovisual-expressed-emotio...	-335.922413	133.345593	22.253245	26.773340	28.770041	11.46
1	male_angry	SAVEE	./database/surrey-audiovisual-expressed-	-258.783070	186.191041	14.535849	-4.115234	21.572367	14.63

Slika 21. Spremanje izvučenih značajki i pridruživanje postojećoj tablici

Idući korak je podjela podataka na dio za učenje i dio za validaciju. To znači da će neuronska mreža učiti na temelju podataka iz skupa za učenje te će periodički provjeravati svoju učinkovitost na skupu za validaciju. Na taj način ispituje se prava učinkovitost mreže budući da ista radi s novim podacima koje nikad nije vidjela tijekom učenja. Podjela je izvedena tako da je 20 % od ukupnog broja podataka izdvojeno za validaciju [Slika 22].

```
# Podjela na dio za ucenje i testiranje
X_train, X_test, y_train, y_test=train_test_split(df.drop(['path','labels','source'],axis=1)
                                                  , df.labels
                                                  , test_size=0.2
                                                  , shuffle=True
                                                 )


X_train[:10]
```

Slika 22. Podjela podataka na skupove za učenje i validaciju

X_train predstavlja skup značajki za učenje, dok je *y_train* skup oznaka (labela), odnosno emocija za učenje. Analogno tome vrijedi i za *X_test* te *y_test*, samo za validaciju. Te je skupove potrebno pretvoriti u oblik prikladan za učenje. Sve oznake se kodiraju iz imena emocije u odgovarajuću numeričku reprezentaciju naredbom *fit_transform()*. Tako primjerice, svaka oznaka *female_angry* postaje reprezentirana brojem 0, *female_happy* brojem 1 i tako redom. Zatim se svaki od tih brojeva pretvara u svojevrsni binarni zapis naredbom *to_categorical()*. Tako broj 0 postaje [1, 0, 0, 0, 0, 0, 0, 0], broj 2 postaje [0, 0, 1, 0, 0, 0, 0, 0] i sl. Taj postupak zove se *One-Hot Encoding* [Slika 23].

```
# Pripremanje u oblik prikladan za Keras
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)

# One Hot encoding
le = LabelEncoder()
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.fit_transform(y_test)

y_train_cat = np_utils.to_categorical(y_train_encoded)
y_test_cat = np_utils.to_categorical(y_test_encoded)

print('X_train dimension: ', X_train.shape)
print('\nX_test dimension: ', X_test.shape)
print('\nClasses:\n', le.classes_)
print('\ny_train_categorical:\n', y_train_cat)
```

Slika 23. Pretvorba oznaka u potreban oblik

Na slici [Slika 24] može se uočiti kako je iz svakog zvučnog zapisa izvučeno ukupno 168 vrijednosti. One u sebi nose informaciju koja predstavlja određenu emociju. Dio za treniranje sadrži 6248 zapisa, a dio za validaciju 1563.

```
X_train dimension:  (6248, 168)

X_test dimension:  (1563, 168)

Classes:
 ['female_angry' 'female_happy' 'female_neutral' 'female_sad' 'male_angry'
 'male_happy' 'male_neutral' 'male_sad']

y_train_categorical:
 [[0. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]]
```

Slika 24. Veličina podataka, klase i izgled oznaka nakon pretvorbe

X_train i *X_test* je također potrebno prilagoditi u oblik prikladan za konvolucijsku neuronsku mrežu. To se postiže proširivanjem dimenzija dodavanjem još jedne osi [Slika 25].

```
# Prosirivanje dimenzija kakve trazi CNN
X_train_exp = np.expand_dims(X_train, axis=2)
X_test_exp = np.expand_dims(X_test, axis=2)
print('X_train_exp dimension: ', X_train_exp.shape)
print('X_test_exp dimension: ', X_test_exp.shape)
```

```
X_train_exp dimension:  (6248, 168, 1)
X_test_exp dimension:  (1563, 168, 1)
```

Slika 25. Pretvorba podataka u potreban oblik

4.4. Izrada modela konvolucijske neuronske mreže i učenje

Model konvolucijske neuronske mreže izrađuje se pomoću biblioteke *Keras*. To je alat visoke razine namijenjen za izgradnju mreža dubokog učenja. Arhitektura slojeva napravljena je po uzoru na [33]. Postavljeno je 8 jednodimenzionalnih konvolucijskih slojeva od kojih svaki ima *ReLU* aktivacijsku funkciju. Kako bi se spriječila prenaučenost mreže, odnosno učenje određenih značajki napamet, dodana su dva *Dropout* sloja [Slika 26].

```
# Definiranje modela
model = Sequential()
model.add(Conv1D(256, 8, padding='same',input_shape=(X_train_exp.shape[1],1)))
model.add(Activation('relu'))
model.add(Conv1D(256, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(8))
model.add(Activation('softmax'))
opt = keras.optimizers.rmsprop(lr=0.00001, decay=1e-6)
model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

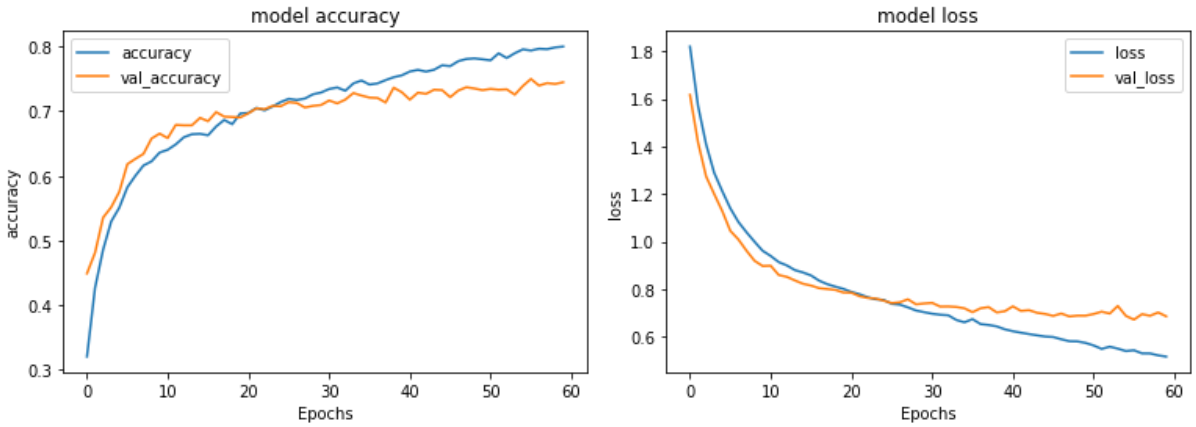
Slika 26. Model konvolucijske neuronske mreže

U modelu su postavljena i dva *BatchNormalization* sloja koja normaliziraju značajke što poboljšava performanse i stabilnost mreže. Na kraju mreže dolazi jedan potpuno povezani sloj sa *Softmax* aktivacijskom funkcijom koja daje distribuciju vjerojatnosti među klasama. Budući da imamo 8 klasa, taj sloj ima 8 izlaza. Za optimizaciju učenja odabran je *RMSprop* optimizator [Slika 26]. To je neobjavljen, ali vrlo popularan i robustan optimizator za adaptivno prilagođavanje stope učenja, koji omogućuje brzo i učinkovito učenje [34]. Učenje se vrši kroz 60 koraka (epoha) [Slika 27].

```
# Ucenje
model_history=model.fit(X_train_exp, y_train_cat, batch_size=16, epochs=60,
                        validation_data=(X_test_exp, y_test_cat))
```

Slika 27. Učenje neuronske mreže

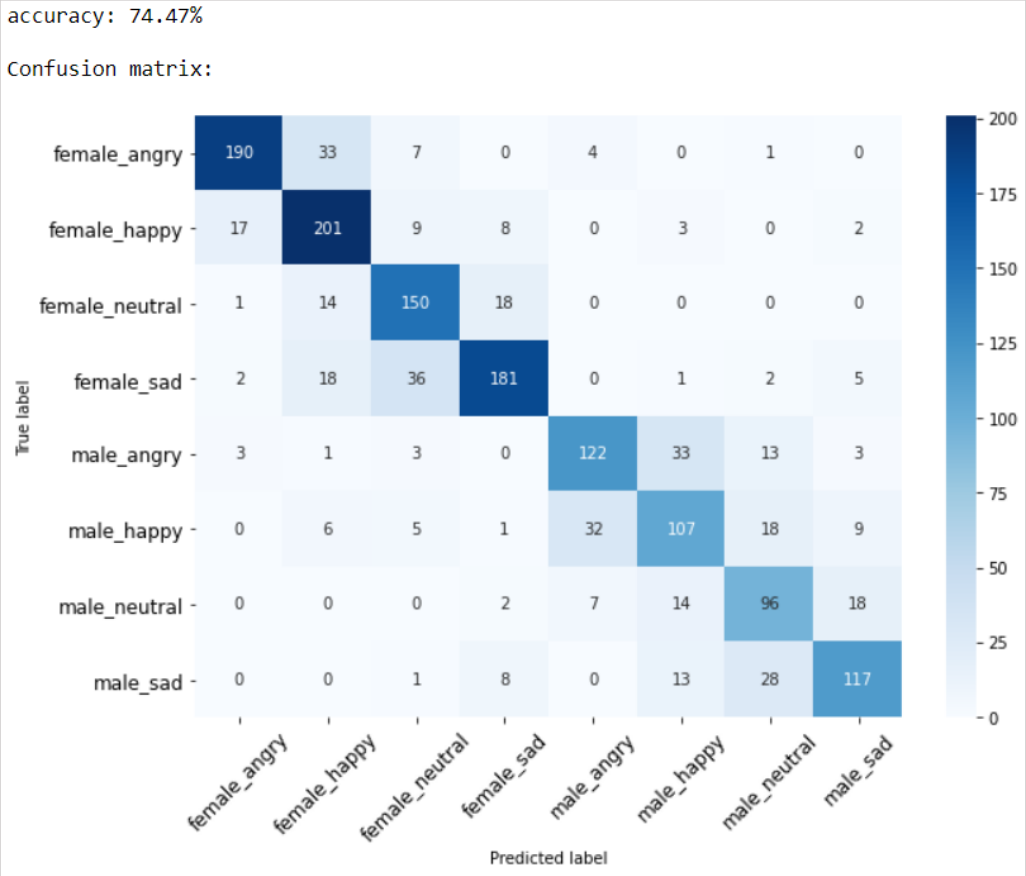
Rezultati učenja prikazani su na slici [Slika 28]. Točnost na skupu za validaciju dostigao je vrijednost nešto višu od 74 %. Na grafu gubitka vidljivo je da vrijednost kontinuirano pada tijekom učenja, što je poželjno jer je to indikacija da nema prenaučenosti.



Slika 28. Točnost akustičkog modela tijekom učenja (lijevo); gubitak modela tijekom učenja (desno)

Na slici [Slika 29] prikazana je matrica konfuzije. Matrica konfuzije, poznata i kao matrica pogrešaka, je sažeta tablica koja se upotrebljava za procjenu učinkovitosti klasifikacijskog modela. Iz nje se vrlo lako može očitati s kojim klasama model ima poteškoća u razlikovanju. Redovi matrice predstavljaju stvarne klase, dok stupci predstavljaju klase koje je model predvidio. Broj ispravno svrstanih klasa nalazi se na dijagonali matrice, stoga je cilj što manje rasipanje izvan dijagonale.

Vidljivo je kako mreža najviše problema ima s uočavanjem razlika između ljutnje i sreće, te tuge i neutralnog. To je očekivano budući da su i ljutnja i sreća emocije jakog intenziteta, dok su tuga i neutralno emocije slabijeg intenziteta. Zanimljivo je za uočiti kako model daje veću točnost klasifikacije za govore ženskog spola. Naravno, uzimajući u obzir činjenicu da u bazi podataka ima više ženskih zvučnih zapisa nego muških, kod muških zapisa i dalje postoji relativno veće rasipanje u odnosu na ženske. Razlog tome mogu biti dvije stvari. Zbog manjeg udjela muških zapisa, neuronska mreža ne može dovoljno dobro naučiti razliku između pojedinih emocija. Drugi razlog može biti da su muškarci lošije naglašavali pojedine emocije. Pomnijim proučavanjem baze doista se može primijetiti kako su ženske osobe vjernije dočaravale emocije, dok je kod muškaraca spektar izražavanja emocija bio nešto ograničeniji.



Slika 29. Matrica konfuzije za akustički model

5. LINGVISTIČKI MODEL

Kako bi model mogao iščitavati emocije na temelju samih izgovorenih riječi, potrebno je imati bazu podataka s mnogo različitih izjava od kojih je svaka označena onom emocijom koja u toj izjavi prevladava. Na osnovi te baze neuronska mreža će naučiti koje riječi, odnosno strukture niza riječi, se pretežno nalaze u pojedinim emocionalnim izjavama te će ih na temelju toga moći razlikovati.

5.1. Pripremanje tekstualne baze podataka

Ne postoji mnogo dostupnih tekstualnih baza podataka koje su označene na temelju diskretnih emocija. U ostalim radovima na temu klasifikacije emocija iz teksta, za podatke su najčešće upotrebljavane izjave s raznih društvenih mreža i mikro-*blogging* platformi kao što je Twitter [35], [36], [37], [38]. Odlučeno je da se u ovom radu neće koristiti takav pristup iz razloga što bi takvi podatci bili lošije kvalitete zbog mnogih gramatičkih i pravopisnih pogrešaka koje se vrlo često pojavljuju u takvim medijima, što bi na posljetku moglo utjecati na rad modela. Stoga je za ovaj rad odabrana *DailyDialog* baza podataka [39]. To je kvalitetna tekstualna baza koja sadrži dijaloge iz svakodnevnog života i napisana je upravo za potrebe istraživanja vezanih uz emocije i razgovor. Svaka izjava u dijalogu ručno je označena informacijom o emociji i namjeri. Originalna baza podataka dolazi u obliku dvije odvojene tekstualne datoteke. U jednoj su sadržane sve izjave [Slika 30], a u drugoj su pripadne emocije u obliku numeričke reprezentacije [Slika 31]. Postoji 13 118 različitih dijaloga (redaka), od kojih se svaki sastoji od nekoliko izjava. Završetak pojedine izjave označen je oznakom *__eou__* (engl. *end of utterance*). Pod izjavom se ovdje misli na cjelinu u dijalogu koju je jedna osoba izjavila bez promjene govornika te se jedna izjava može sastojati od nekoliko rečenica. Svaka izjava je reprezentirana svojom oznakom emocije u spomenutoj drugoj datoteci. Neutralna izjava označena je brojem 0, ljutnja brojem 1, gađenje brojem 2, strah brojem 3, sreća brojem 4, tuga brojem 5 i iznenađenje brojem 6. Odmah se može uočiti kako je cijela ovakva izvedba poprilično nezgodna za potrebe ovog rada, stoga ju je potrebno prilagoditi u prikladniji oblik.

```
5144 Why are you laughing ? __eou__ I can't help it . __eou__
5145 I'll be seeing you around . __eou__ OK.Take care . __eou__
5146 There will be a party in my company ; what shall I wear ? __eou__ Is it formal or informal ? __eou
5147 I broke my sister's mirror . I'm sure it'll get her back up . I really don't know how I can escape
5148 Do you like watching the Winter Olympic Games ? __eou__ Of course . It's the tradition game in wint
5149 Belinda , I ' m going to a party tonight . What shall I wear ? __eou__ Is it formal or informal ?
5150 What's a good place for sightseeing ? __eou__ Don't ask me . __eou__
5151 Freeze ! __eou__ Don't shoot . I give up . __eou__
5152 Do you have a light ? __eou__ Sorry , I don't smoke . __eou__
5153 Excuse me . Nature calls . __eou__ If you gotta go , you gotta go . __eou__
5154 The performance is amazing . I'm curious about how they did it . I mean the way they sing . __eou__
```

Slika 30. Datoteka s izjavama iz *DailyDialog* tekstualne baze podataka

```
5144  0 0
5145  0 0
5146  0 0 0 0 0 0 4 4
5147  0 0
5148  0 0 0 0 0 0 0 0 0 0 0 0 0 4 4
5149  0 0 0 0 0 4 4 0 0 4 4 0 0
5150  0 0
5151  0 0
5152  0 0
5153  0 0
5154  4 0 4 0 0 0 0 0 0 0 0 4 0 4 4 4 4
```

Slika 31. Datoteka s oznakama emocija iz *DailyDialog* tekstualne baze podataka

Odlučeno je da se svaka izjava odvoji na temelju *__eou__* oznake te da se onda te izjave spremaju u novu datoteku zajedno s pripadajućim oznakama emocija. Na taj način će baza biti preglednija i jednostavnija za uporabu te će sve biti na jednom mjestu. U tu svrhu napravljen je Python kôd koji će proći kroz obje tekstualne datoteke, izvući svaku izjavu zajedno s pripadajućom oznakom emocije i to spremiti u novu *.csv* datoteku koja običan tekst prikazuje u tabličnom obliku [Slika 32], [Slika 33].

```python
dialogues_file = './database_text/dialogues_text.txt'
with open(dialogues_file, 'r', encoding='utf8') as file:
    data = file.read().replace('__eou__', '\n')
    data = data.replace('\n\n', '\n')
    data = data.replace('’', '\'')
    data = data.replace(' \' t', '\'t')
    data = data.replace(' \' m', '\'m')
    data = data.replace(' \' re', '\'re')
    data = data.replace(' \' s', ' is')
    data = data.replace(' \' d', ' would')
    data = data.replace(' \' Ve', ' have')
    data = data.replace(' \' ll', ' will')
    data = data.replace(' .', '.')
    data = data.replace(' ,',',')
    data = data.replace(' ?', '?')
    data = data.replace(' !', '!')

expressions = data.splitlines() #stavlja svaku izjavu u listu

counter = 0
for row in expressions:
    counter += 1
print('Ukupan broj izjava:', counter)
```

```
Ukupan broj izjava: 102980
```

Slika 32. Razdvajanje izjava i provođenje korekcija

```python
# Slican postupak i za labele
labels_file = './database_text/dialogues_emotion.txt'
with open(labels_file, 'r', encoding='utf8') as file:
    data = file.read().replace(' ', '\n')
    data = data.replace('\n\n', '\n')

labels = data.splitlines() #stavlja svaku oznaku u listu

counter = 0
for row in labels:
    counter += 1
print('Ukupan broj labela:', counter)
```

```
Ukupan broj labela: 102979
```

Slika 33. Razdvajanje oznaka emocija

No, tijekom obavljanja zadatka, pojavio se problem. Naime, ukupan broj izjava nije se poklapao s ukupnim brojem oznaka emocija. Dobiveno je 102 980 izjava te 102 979 oznaka emocija, što znači da negdje postoji greška. Pretpostavka je da je problem nastao zbog jedne izjave viška ili nedostaje jedna oznaka emocije u nekom od dijaloga. Kako bi se lakše pronašao problem, izrađen je još jedan kratki Python program koji broji izjave i oznake u originalnim tekstualnim datotekama i uspoređuje ta dva broja. U slučaju da se naiđe na nepodudaranje u njihovom broju, program ispisuje na kojem se mjestu problem pojavljuje [Slika 34].

```python
# Program broji koliko ima izjava u svakom retku (dijalogu) kako bi se
# usporedilo s brojem labela jer se ne poklapaju
import re

f1 = './database_text/dialogues_text.txt'
f2 = './database_text/dialogues_emotion.txt'
pattern1 = '__eou__'
pattern2 = ' '
counter = 0

with open(f1,'r',encoding='utf8') as f1, open(f2,'r',encoding='utf8') as f2:
    for line1, line2 in zip(f1, f2):
        counter += 1
        count1 = sum(1 for match in re.finditer(pattern1, line1))
        count2 = sum(1 for match in re.finditer(pattern2, line2))
        if count1 != count2:
            print('Nepodudaranje u %d. retku!' % counter)
```

```
Nepodudaranje u 673. retku!
```

Slika 34. Program za usporedbu izjava i oznaka

Program je pronašao nepodudaranje u 673. retku. I doista, vidljivo je kako taj dijalog ima 12 izjava te 11 oznaka emocija [Slika 35], [Slika 36].

```
673  Sam , can we stop at this bicycle shop ? __eou__ Do you want to buy a new bicycle ? __eou__
     Yes , and they have a sale on now . __eou__ What happened to your old one ? __eou__ I left it
     at my parent's house , but I need one here as well . __eou__ I've been using Jim's old bike
     but he needs it back . __eou__ Let's go then . __eou__ Look at this mountain bike . It is
     only £ 330 . Do you like it ? __eou__ I prefer something like this one - a touring bike , but
     it is more expensive . __eou__ How much is it ? __eou__ The price on the tag says £ 565 but
     maybe you can get a discount . __eou__ OK , let's go and ask . __eou__
```

Slika 35. Redak s izjavama u kojem postoji nepodudaranje

```
672  0 0 0 0 0 4
673  0 0 0 0 0 0 0 0 0 0 0 0
674  0 0 0 0 0 0 0 0 0 0 0 0
```

Slika 36. Redak s oznakama u kojem postoji nepodudaranje

Dodavanjem još jedne nule problem je riješen te je postupak spremanja izjava i oznaka emocija u *.csv* datoteku dovršen s ispravljenom verzijom. [Slika 37] prikazuje dio kôda koji spaja izjave i emocije te ih sprema u jednu zajedničku *.csv* datoteku.

```
1  import csv
2  combined = zip(labels, expressions)
3
4  new_file = './database_text/DailyDialog_database.csv'
5  with open(new_file,'w', encoding='utf8', newline='') as f:
6      writer = csv.writer(f)
7      for row in combined:
8          writer.writerow(row)
```

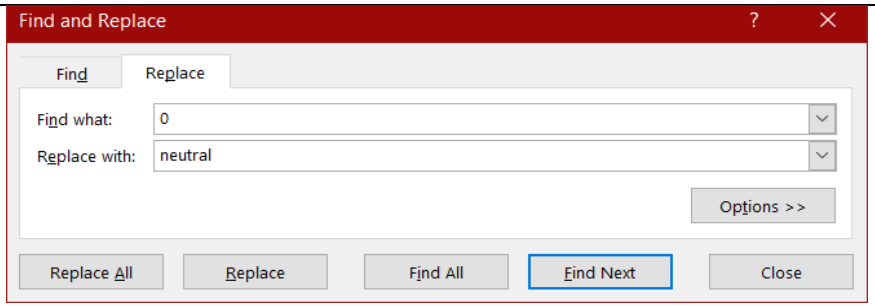
Slika 37. Spremanje tekstualne baze podataka

Kako bi se vizualizirao sadržaj novonastale *.csv* datoteke, ista je otvorena u programu Excel [Slika 38].

40803	0	I guess it is a formal one because the general director will give a speech there, and most of the staff will take part in.
40804	0	In that case, formal suit with a nice tie will be better.
40805	0	You are right. What about shoes?
40806	0	The brown leather shoes are OK.
40807	4	Thanks a lot.
40808	4	Don't mention it.
40809	0	I broke my sister's mirror. I'm sure it'll get her back up. I really don't know how I can escape the punishment.
40810	0	You'd better tell your sister what happened and face the music.
40811	0	Do you like watching the Winter Olympic Games?
40812	0	Of course. It's the tradition game in winter, and with good reason. Don't you like watching it?
40813	0	Of course I do. I love it. All the games are exciting and the competitors are respectable.
40814	0	That's true. For various reasons, it takes more to hold a fierce game for Winter Olympics than it does for Summer Olympic

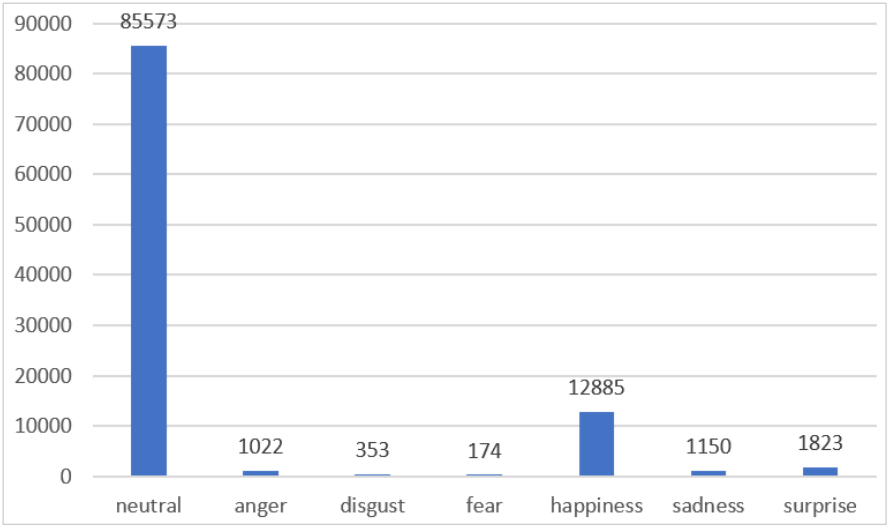
Slika 38. Izgled tekstualne baze podataka u programu Excel

Zbog jednostavnosti, sve brojčane oznake emocija zamijenjene su pripadajućim tekstualnim oznakama naredbom *Find and Replace* [Slika 39].



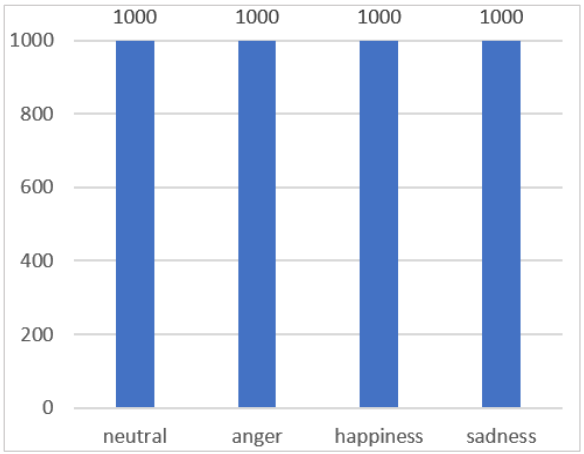
Slika 39. Pretvorba brojčanih oznaka emocija u riječi

Provjerom distribucije pojedinih emocija, uočen je veliki nesrazmjer [Slika 40]. To može imati loš utjecaj na rad neuronske mreže.



Slika 40. Distribucija izjava po svakoj emociji

Emocije gađenja, straha i iznenađenja (engl. *disgust, sadness, surprise*) se u ovom slučaju ionako ne uzimaju u obzir pa se zbog toga uklanjaju. Izjave preostalih emocija koje su od interesa potrebno je uskladiti kako bi im raspodjela bila jednaka. Iz svake kategorije emocija nasumičnim odabirom uzeo se jednak broj izjava, pa distribucija sada izgleda kao na slici [Slika 41]. Time je baza podataka pripremljena za učenje lingvističkih značajki.



Slika 41. Novonastala distribucija izjava po emociji

5.2. Izdvajanje značajki

Iz pripremljene baze podataka je potrebno je izdvojiti značajke koje će neuronska mreža učiti. Kao i inače, na početku se učitaju sve potrebne biblioteke. Zatim se naprave dvije prazne liste, jedna za spremanje izjava, a druga za spremanje oznaka. Napravljena je i posebna lista u koju se mogu staviti riječi koje se u tekstu često ponavljaju, a ne daju dodatnu informaciju o emocionalnoj naravi izjave. Sve riječi sadržane u toj listi zanemarit će se prilikom učitavanja baze podataka. No, pri tome valja biti oprezan kako se ne bi zanemarile one riječi koje ipak mogu izmijeniti emocionalno obilježje izjave.

Otvara se pripremljena baza podataka te se pomoću *for* petlje učitavaju sve izjave i oznake. Pri tome se provode dodatne obrade nad podatcima, kao što su uklanjanje znakova i brojeva, pretvorba velikih slova u mala, te uklanjanje nepotrebnih riječi [Slika 42].

```
# Inicijaliziranje mjesta za spremanje izraza i labela
expressions = []
labels = []

# Definiranje nepotrebnih rijeci
stopwords = ['i', 'you', 'he', 'she', 'it', 'we', 'they', 'i\'m', 'i\'ll',
             'you\'re', 'am', 'are', 'is', 'a', 'the', 'and']

# Otvaranje baze podataka
counter=0
text_database = './database_text/DailyDialog_database.csv'
with open(text_database, 'r', encoding="utf8") as csvfile:
    file = csv.reader(csvfile, delimiter=',')
    # Kroz bazu se ide red po red
    for row in file:
        counter +=1
        print(row[0])
        # Sprema se labela za trenutni redak
        labels.append(row[0])
        print(row[1])
        # Izdvaja se izjava, sva slova se pretvaraju u mala
        expression = row[1].lower()
        # Uklanjaju se nepotrebni znakovi
        expression = re.sub('[^a-z\']+', ' ', expression)
        stripped_expression = ''
         # Ide se rijec po rijec u izjavi
        for word in expression.split():
            # Ako se rijec nalazi unutar stopwords, onda se uklanja
            if word in stopwords: word = ''
            else: word = word + ' '
            # Sastavljanje nove ociscene recenice
            stripped_expression = stripped_expression + word
        print(stripped_expression, '\n')
        # Dodavanje recenice u pripremljeno mjesto
        expressions.append(stripped_expression)
print(len(labels))
print(len(expressions))
```

Slika 42. Učitavanje izjava i oznaka te njihova obrada

Izvadak rezultata navedenog postupka prikazan je na slici [Slika 43].

```
sadness
 Sorry. I'm afraid we can't. But you may order something else instead.
sorry afraid can't but may order something else instead

happiness
 Well, I have no regrets!
well have no regrets

happiness
 I'll have a good day once this day is over.
have good day once this day over
```

Slika 43. Izgled izjava prije i nakon obrade

Tehnike poput algoritama strojnog učenja preferiraju točno definirane ulaze i izlaze fiksne duljine, stoga je rad sa sirovim tekstom problematičan jer su u njemu riječi i rečenice različite duljine. Zbog toga se provodi tzv. tokenizacija, odnosno raščlanjivanje rečenica na riječi i zamjena svake riječi određenim tokenom. Ovdje je token jedan broj koji se dodjeljuje svakoj jedinstvenoj riječi. Riječ koja se najviše spominje u bazi podataka biti će označena brojem 1, iduća brojem 2 i tako redom. Konkretno u ovom slučaju, engleska riječ *sorry* (hrv. oprosti) dobila je broj 12 za token. Engleski jezik ima preko 171 000 različitih riječi [40], što je prevelika količina za potrebe ovog rada, stoga će se u obzir uzeti prvih 10 000 najčešćih riječi u bazi. Sve ostale riječi zamjenjuju se oznakom <*OOV*> koja predstavlja riječi izvan vokabulara (engl. *out of vocabulary*). Na taj način izrađen je vokabular od 10 000 različitih riječi koje su reprezentirane jedinstvenim brojevima. Cijeli ovaj proces zove se izvlačenje značajki ili kodiranje značajki (engl. *feature extraction, feature encoding*). Popularniji naziv izvlačenja značajki iz teksta je i „vreća riječi“ (engl. *bag-of-words*).

Zatim se u skladu s izjavama, koje su ranije spremljene unutar liste, iz „vreće riječi“ uzimaju odgovarajuće riječi reprezentirane brojem i sastavljaju se nove izjave ekvivalentne prijašnjim, ali u obliku nizova brojeva. Kao što je već napomenuto, za strojno učenje potrebni su ulazi fiksnih dimenzija. Budući da sve izjave nisu iste duljine, potrebno je nadopuniti one koje su kraće od definirane vrijednosti. To će se napraviti na način da se na kraj kraće izjave doda onoliko nula koliko je potrebno da bi se postigla ujednačena duljina (engl. *zero padding*). Isto tako, sve one izjave koje su dulje od definirane vrijednosti skratit će se na definiranu duljinu odbacivanjem viška (engl. *truncating*). Odlučeno je da duljina svake izjave bude postavljena na 100 riječi [Slika 44]. Izgled jedne izjave nakon provedenog postupka prikazan je na slici [Slika 45].

```
tokenizer = Tokenizer(num_words = vocab_size, oov_token='<OOV>')
tokenizer.fit_on_texts(expressions)
expressions_sequences = tokenizer.texts_to_sequences(expressions)
expressions_sequences_padded = pad_sequences(expressions_sequences, maxlen=100,
                                             padding='post', truncating='post')

dict(list(tokenizer.word_index.items())[:6])
```

```
{'<OOV>': 1, 'to': 2, 'that': 3, 'have': 4, 'my': 5, 'for': 6}
```

Slika 44. Pretvorba podataka u potreban oblik

```
[  38   29   33   35   72 2377  431 2378   77 1269    8 1044   29 2379
   10   27  229  725   27  804   42  459   24    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
```

Slika 45. Izgled jedne izjave nakon pretvorbe

Način tokenizacije potrebno je spremiti u datoteku kako bi se kasnije kod postupka prepoznavanja svaka nova rečenica mogla obraditi identičnom procedurom [Slika 46].

```
tokenizer_json = tokenizer.to_json()
with io.open('tokenizer.json', 'w', encoding='utf-8') as file:
    file.write(json.dumps(tokenizer_json, ensure_ascii=False))
```

```
X_train, X_test, y_train, y_test = train_test_split(expressions_sequences_padded
                                                    , labels
                                                    , test_size=0.2
                                                    , shuffle=True
                                                   )
```

Slika 46. Spremanje načina tokenizacije i podjela podataka na skupove za učenje i validaciju

Kao što je to bio slučaj kod akustičkog modela, i ovdje je potrebno raspodijeliti podatke na dio za učenje i dio za testiranje te provesti *One-Hot Encoding*. Podatci se raspodjeljuju slučajnim odabirom na 80 % za učenje i 20 % za testiranje, što daje skup za učenje od 3200 podataka i skup za testiranje od 800 podataka. Oznake emocija se prvo pretvaraju (kodiraju) u brojčanu reprezentaciju, a zatim u binarnu [Slika 47].

```
le = LabelEncoder()
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.fit_transform(y_test)

le_name_mapping = dict(zip(le.transform(le.classes_), le.classes_))
print('Label Encoding Classes as ')
print(le_name_mapping)

y_train_cat = np_utils.to_categorical(y_train_encoded)
y_test_cat = np_utils.to_categorical(y_test_encoded)
print('\nOne Hot Encoded class shape ')
print('Training set:', y_train_cat.shape)
print('Testing set:', y_test_cat.shape)
```

```
Label Encoding Classes as
{0: 'anger', 1: 'happiness', 2: 'neutral', 3: 'sadness'}

One Hot Encoded class shape
Training set: (3200, 4)
Testing set: (800, 4)
```

Slika 47. Pretvorba oznaka u potreban oblik

5.3. Izrada modela neuronske mreže i učenje

Neuronska mreža za učenje lingvističkih značajki bit će povratna LSTM mreža. Za njenu izradu također se koristi Keras biblioteka, a napravit će se sekvencijalni model po uzoru na [41]. Prvi sloj mreže je *embedding* sloj [Slika 48]. On pretvara numeričke reprezentacije riječi u vektorski zapis. Tijekom učenja vrijednosti vektora se mijenjaju na način da riječi koje imaju slično značenje ili među njima postoji svojevrsna povezanost, dobivaju slične vrijednosti. Tako će, primjerice, riječi "sreća" i "smijeh" imati puno bliži odnos u vektorskom prostoru nego što bi imale riječi "sreća" i "bol". Ovo je vrlo bitno jer takve informacije o odnosima među riječima omogućuju neuronskoj mreži da shvati kontekst. Nakon *embedding* sloja, slijedi *bidirectional* LSTM sloj. U njemu se uneseni podatci prvo propuštaju normalnim redoslijedom, a zatim obratnim redoslijedom. To pomaže LSTM sloju da bolje nauči dulje nizove riječi i njihovu ovisnost. U istom sloju postavljen je *Dropout* kako bi se izbjegla prenaučenost. Zatim se dodaje potpuno povezani sloj s *ReLU* aktivacijskom funkcijom. Na posljetku dolazi još jedan potpuno povezani sloj s četiri izlaza jer toliko ima klasa. Budući da provodimo klasifikaciju, aktivacijska funkcija u zadnjem sloju je *Softmax* koja daje distribuciju vjerojatnosti. Upotrijebljen je *Adam* optimizator koji adaptivno određuje stopu učenja i stoga je proces učenja brz i efikasan.

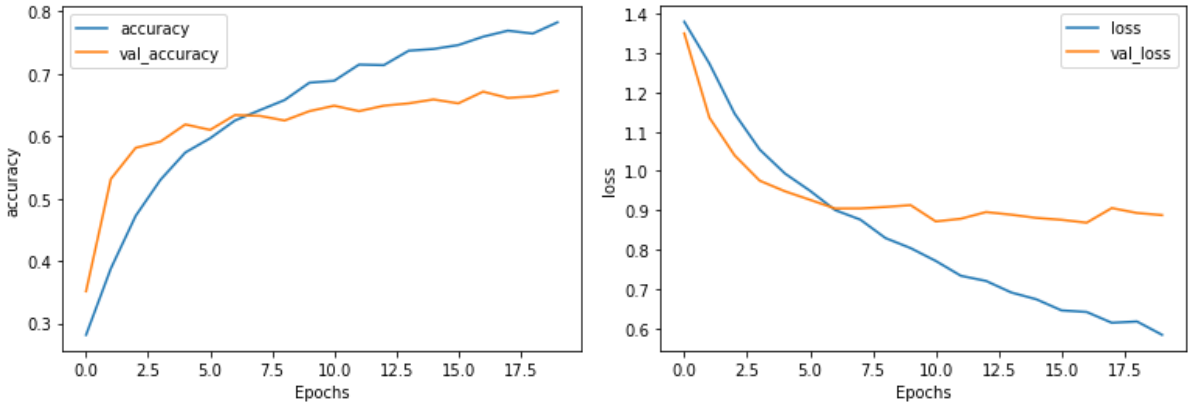
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embedding_dim
                                                       , dropout=0.9
                                                       )),
    tf.keras.layers.Dense(embedding_dim, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])

adam = optimizers.Adam(lr=0.001)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
model.summary()
```

Slika 48. Model LSTM neuronske mreže

Učenje se provodi u 20 koraka (epoha) [Slika 49]. Točnost i gubitak tijekom učenja prikazani su na slici [Slika 50]. Vidljivo je da točnost na skupu za validaciju konvergira vrijednosti oko 67 %. Učenje u više od 20 koraka ne bi mnogo poboljšalo rezultat jer bi gubitak počeo rasti i došlo bi do prenaučenosti.

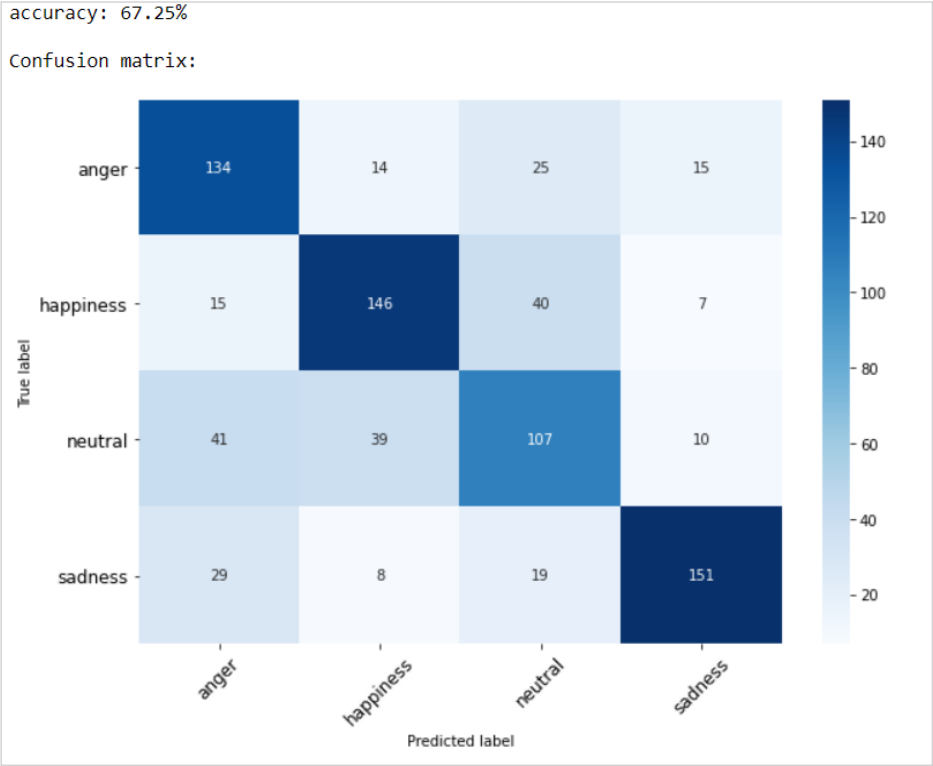
```
history = model.fit(X_train, y_train_cat, epochs=20,
                    validation_data=(X_test, y_test_cat), verbose=2)
```

Slika 49. Učenje neuronske mreže



Slika 50. Točnost lingvističkog modela tijekom učenja (lijevo); gubitak modela tijekom učenja (desno)

Na slici [Slika 51] je prikazana matrica konfuzije. Vidljivo je da mreža neutralne izjave često pogrešno protumači kao ljute ili sretne, dok tužne izjave u nekim slučajevima zamijeni s ljutim. Za bolje rezultate bilo bi potrebno koristiti mnogo veću bazu podataka ili pronaći kvalitetniju strukturu modela za klasificiranje. No, bez obzira na to, rezultat je relativno zadovoljavajući.



Slika 51. Matrica konfuzije za lingvistički model

6. PREPOZNAVANJE

Sada su gotova oba modela za klasifikaciju emocija iz govora. Preostaje još napraviti program koji će na temelju naučenih modela prepoznavati emocije iz novih zvučnih zapisa dobivenih putem mikrofona ili audio datoteke.

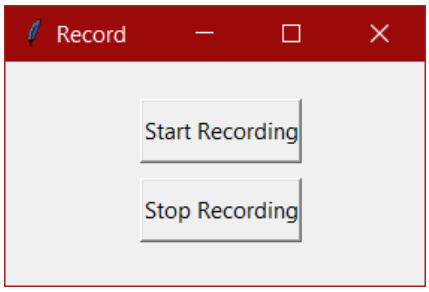
Ponovno se prvo učitaju potrebne biblioteke. Zatim je potrebno učitati prethodno naučene modele [Slika 52].

```
le = LabelEncoder()

# ucitavanje akusickog modela
json_file = open('saved_models/akusticki_model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# Ucitavanje tezina u model
loaded_model.load_weights('saved_models/akusticki_model.h5')
print('Model loaded from disk')

# Ucitavanje lingvistickog modela
model_txt = load_model('saved_models/lingvisticki_model.h5')
# ucitavanje tokena rijeci
with open('tokenizer.json') as f:
    data = json.load(f)
    tokenizer = tokenizer_from_json(data)
```

```
Model loaded from disk
```

Slika 52. Učitavanje naučenih modela

Pomoću *tkinter* biblioteke, koja služi za izradu grafičkih sučelja, napisana je funkcija po uzoru na [42] i njenim se pozivom dobije prozor s naredbama za početak i kraj snimanja [Slika 53] .



Slika 53. Prozor za snimanje zvučnih zapisa

Nakon toga potrebno je učitati snimljeni zvučni zapis i pomoću *LibROSA* biblioteke ukloniti dijelove tišine s početka i kraja zapisa [Slika 54].

```
file_name = 'test_audio/recording.wav'
data, sample_rate = librosa.load(file_name,res_type='kaiser_fast',sr=44100)
data, _ = librosa.effects.trim(y=data, top_db = 20)

plt.figure(figsize=(8, 4))
librosa.display.waveplot(data, sr=sample_rate)

ipd.Audio(file_name)
```

Slika 54. Učitavanje zvučnog zapisa i uklanjanje dijelova tišine

Zatim je potrebno izvući značajke na identičan način kao što se to napravilo za proces učenja [Slika 55].

```
features=np.array([])

mfccs = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40), axis=1)
features=np.hstack((features, mfccs))

mel=np.mean(librosa.feature.melspectrogram(data, sr=sample_rate), axis=1)
features=np.hstack((features, mel))

features = np.expand_dims(features, axis=0)
features = np.expand_dims(features, axis=-1)
features.shape
```

Slika 55. Izvlačenje značajki iz zvučnog zapisa

Učitane značajke se potom provode kroz CNN model koji daje distribuciju vjerojatnosti između 8 klasa, po 4 za oba spola. Budući da su ovdje od interesa samo emocije, muške i ženske klase će se u ovom trenutku zbrojiti tako da se u konačnici dobije klasifikacija samo na temelju emocija [Slika 56].

```
p = loaded_model.predict(features, batch_size=16, verbose=0)
classes=['anger', 'happiness', 'neutral', 'sadness']
pred_audio=[p[0][0]+p[0][4], p[0][1]+p[0][5], p[0][2]+p[0][6], p[0][3]+p[0][7]]

print('Predviđanje akustičkog modela:')
dict(zip(classes, pred_audio))
```

Slika 56. Prepoznavanje emocija iz zvučnog zapisa

Za provedbu drugog dijela prepoznavanja, govor je potrebno pretvoriti u tekst. U tu svrhu koristi se biblioteka *SpeechRecognition* koja vrši automatsku pretvorbu i podržava nekoliko aplikacijskih programskih sučelja (engl. *Application Programming Interface*, API) te je vrlo jednostavna za upotrebu. Odabran je *Google Speech Recognition* API jer je besplatan, a biblioteka u sebi već posjeduje ključ za pristup usluzi tako da nije potrebna prijava. Taj API ima i mogućnost za prilagodbu na okolnu buku. Prepoznati tekst sprema se u varijablu *text*, a ako govor nije razaznat, ispisuje se greška [Slika 57].

```python
# Prepoznavanje govora
r = speech_recognition.Recognizer()
with speech_recognition.AudioFile(file_name) as source:
    #r.adjust_for_ambient_noise(source, duration=1)
    audio_text = r.listen(source)
    try:
        text = r.recognize_google(audio_text, language = 'en-US')
        print('Converting into text...')
        print(text)

    except speech_recognition.UnknownValueError:
        print('Google Speech Recognition could not understand audio')

    except speech_recognition.RequestError as e:
        print('Google Speech Recognition service error; {0}'.format(e))
```

Slika 57. Prepoznavanje govora i pretvorba u tekst pomoću *Google Speech Recognition* API-a

Spremljeni tekst obrađuje se kao i za proces učenja te se prosljeđuje LSTM neuronskoj mreži na klasifikaciju [Slika 58].

```python
txt=[text]
seq = tokenizer.texts_to_sequences(txt)
padded = pad_sequences(seq, maxlen=100, padding='post', truncating='post')
pred_txt = model_txt.predict(padded)
classes=['anger', 'happiness', 'neutral', 'sadness']

print('Tekst:', text)
print('\nPredviđanje lingvističkog modela:')
dict(zip(classes, pred_txt[0]))
```

Slika 58. Obrada i prepoznavanje emocija iz teksta

Rezultate oba modela potrebno je kombinirati u konačan zajednički rezultat, a taj proces naziva se fuzija. Postoje različite metode i algoritmi za provođenje fuzije koji određuju pod kojim uvjetima i na koji način se spajaju rezultati. Ovdje će se upotrijebiti linearna fuzija na način da se rezultati oba modaliteta pomnože nekom fiksnom vrijednošću kojom se određuje koliki udio pojedini modalitet ima u konačnom rezultatu. Odlučeno je da akustički i lingvistički model imaju podjednak utjecaj, stoga se rezultati jednog i drugog modela množe vrijednošću 0,5 i zatim se zbrajaju. Konačna klasifikacija prikazuje se u obliku stupčastog dijagrama te se ispisuje dominantna emocija [Slika 59].

```
# Konacna predikcija
fusion = 0.5*pred_audio+0.5*pred_txt

plt.figure(figsize=(6,4))
plt.title('Predviđanje kombiniranog modela')
plt.plot()
plt.grid(False)
plt.yticks([])
bars = plt.bar(range(len(classes)), fusion[0], width=0.5)
plt.ylim([0, 1])
plt.xticks(range(4), classes)

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), -0.15, round(yval, 4))
plt.show()
print('Dominant emotion: {} {:0.2f}%'.format(classes[np.argmax(fusion)],
                                             100*np.max(fusion)))
```

Slika 59. Konačna predikcija emocije

7. EKSPERIMENTALNA EVALUACIJA

Ovdje će biti prikazan rad izrađenog softverskog rješenja na ljudskim subjektima. Budući da se na papiru ne mogu prikazati zvučni zapisi korišteni za eksperimentalnu evaluaciju, odlučeno je da se koriste snimke ljudskih subjekata dostupnih na *YouTube* video platformi, a za svaku će biti navedena poveznica. Tako čitatelji mogu znati na kojim podatcima je ovaj softver evaluiran.

Zapis 1, neutralna izjava, [43], (0:19 - 0:27)

```
Predviđanje akustičkog modela:

{'anger': 0.044754677,
 'happiness': 0.06600224,
 'neutral': 0.6376979,
 'sadness': 0.25154522}
```

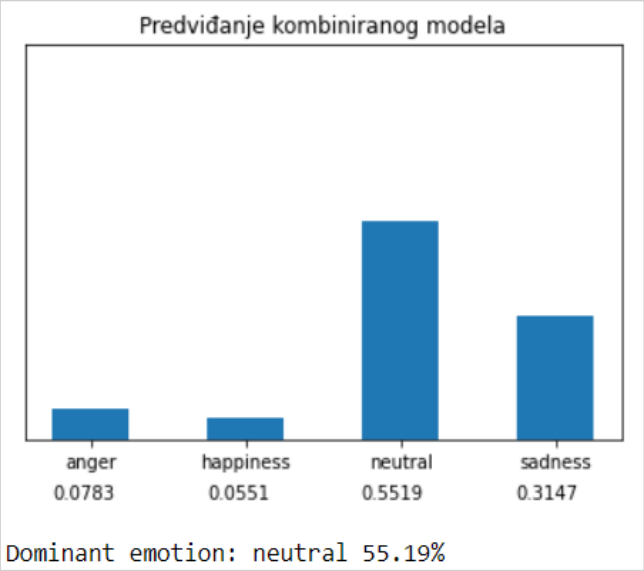
Slika 60. Predikcija emocije akustičkog modela za prvi zapis

Akustički model uspješno je prepoznao neutralnu emociju [Slika 60].

```
Tekst: we play hard and we work hard sometimes we play hard when we
should be working hard when it's Primary

Predviđanje lingvističkog modela:

{'anger': 0.1118566,
 'happiness': 0.044162706,
 'neutral': 0.4660434,
 'sadness': 0.37793732}
```

Slika 61. Predikcija emocije lingvističkog modela za prvi zapis

Lingvistički model pogrešno je prepoznao nekoliko riječi na kraju izjave, ali je svejedno očitao dobru emociju [Slika 61]. Kombinacijom oba modela dobiven je konačni rezultat prikazan na slici [Slika 62] s dominantnom emocijom neutralno.



Slika 62. Konačna predikcija emocije za prvi zapis

Zapis 2, neutralna izjava, [43], (0:42 - 0:50)

```
Predviđanje akustičkog modela:

{'anger': 0.0073651215,
 'happiness': 0.026101122,
 'neutral': 0.6966232,
 'sadness': 0.2699106}
```

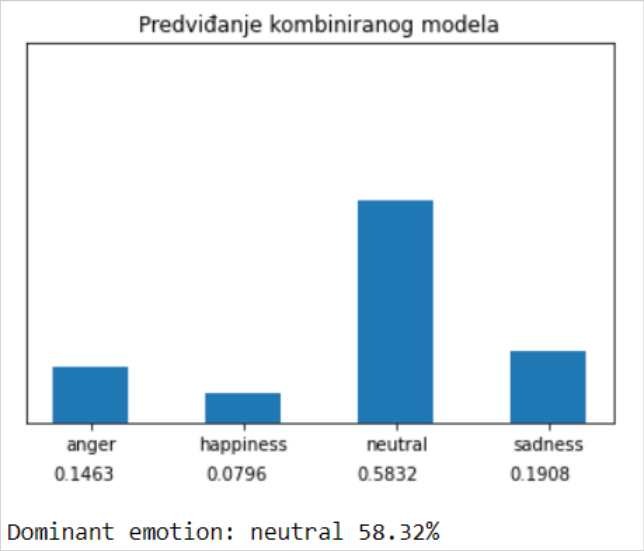
Slika 63. Predikcija emocije akustičkog modela za drugi zapis

Ovaj zapis također ima neutralnu emociju i to je akustički model dobro zaključio [Slika 63].

```
Tekst: the first trying to remind my boss that there is no wire and
paper

Predviđanje lingvističkog modela:

{'anger': 0.28532103,
 'happiness': 0.13308503,
 'neutral': 0.46985385,
 'sadness': 0.11174009}
```

Slika 64. Predikcija emocije lingvističkog modela za drugi zapis

Lingvistički model je prilikom pretvorbe govora u tekst preskočio nekoliko riječi u izjavi, no opet je unatoč tome zaključio dobru emociju [Slika 64]. Distribucija kombiniranog modela prikazana je na slici [Slika 65], a dominantna emocija je neutralno.



Slika 65. Konačna predikcija emocije za drugi zapis

Zapis 3, ljuta izjava, [44], (6:51 – 6:56)

```
Predviđanje akustičkog modela:

{'anger': 0.99348724,
 'happiness': 0.006251513,
 'neutral': 0.00023140997,
 'sadness': 2.9848481e-05}
```

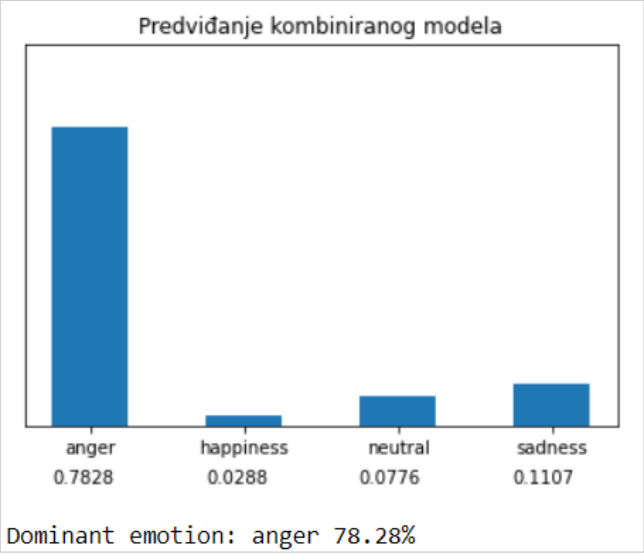
Slika 66. Predikcija emocije akustičkog modela za treći zapis

Ovaj zapis sadrži ljutitu izjavu i akustički model je u to potpuno siguran [Slika 66].

```
Tekst: is not a joke Jim millions the family supper every year

Predviđanje lingvističkog modela:

{'anger': 0.57206845,
 'happiness': 0.051429898,
 'neutral': 0.15505832,
 'sadness': 0.22144336}
```

Slika 67. Predikcija emocije lingvističkog modela za treći zapis

Nekoliko riječi s početka izjave nije pretvoreno u tekst te je napisano *supper* umjesto *suffer*. Unatoč tome, lingvistički model je poprilično siguran da se ovdje radi o ljutitoj izjavi [Slika 67]. Nakon kombinacije model je 78 % siguran da je dominantna emocija ljutnja [Slika 68].



Slika 68. Konačna predikcija emocije za treći zapis

Zapis 4, tužna izjava, [45], (3:12 – 3:18)

```
Predviđanje akustičkog modela:

{'anger': 0.00047233605,
 'happiness': 0.061196625,
 'neutral': 0.03632763,
 'sadness': 0.90200347}
```

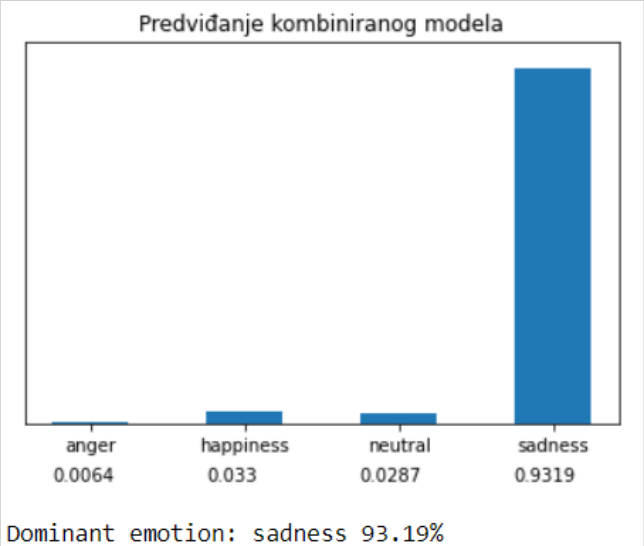
Slika 69. Predikcija emocije akustičkog modela za četvrti zapis

Ovdje imamo izjavu za koju je akustički model siguran da je tužna i to je točno [Slika 69].

```
Tekst: sorry if you misinterpreted things

Predviđanje lingvističkog modela:

{'anger': 0.012251313,
 'happiness': 0.0047462923,
 'neutral': 0.021117438,
 'sadness': 0.9618849}
```

Slika 70. Predikcija emocije lingvističkog modela za četvrti zapis

Lingvistički model je također siguran da se ovdje radi o tužnoj izjavi [Slika 70], a nakon kombinacije dominantna emocija je tuga sa sigurnošću od 93 % [Slika 71].



Slika 71. Konačna predikcija emocije za četvrti zapis

Zapis 5, tužna izjava, [45], (3:25 – 3:29)

```
Predviđanje akustičkog modela:

{'anger': 0.0005260697,
 'happiness': 0.026552342,
 'neutral': 0.7561162,
 'sadness': 0.21680535}
```

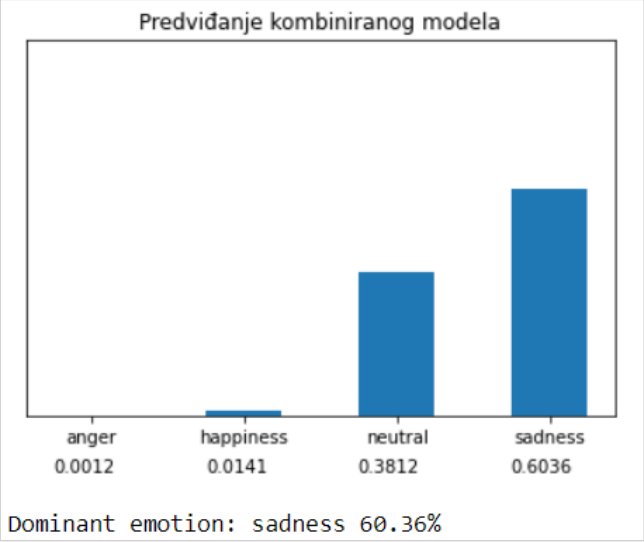
Slika 72. Predikcija emocije akustičkog modela za peti zapis

Kod ove izjave, zaključak o emociji ovisi o kontekstu u kojem se promatra, sadržajno je tužna, no izrečena je na način koji subjektivno više naginje prema neutralnom nego tuzi. Tako zaključuje i akustički model. On tvrdi da se radi o neutralnoj emociji sa 75,6 % sigurnosti, a nakon toga slijedi emocija tuge sa 21,7 % [Slika 72].

```
Tekst: sorry I misinterpreted our friendship

Predviđanje lingvističkog modela:

{'anger': 0.0017779215,
 'happiness': 0.0015853576,
 'neutral': 0.006229374,
 'sadness': 0.9904074}
```

Slika 73. Predikcija emocije lingvističkog modela za peti zapis

Gledajući isključivo u lingvističkoj domeni, model je u potpunosti siguran da se radi o emociji tuge s 99 % [Slika 73], a nakon provedene kombinacije rezultata, krajnji zaključak je da prevladava emocija tuge sa 60 % sigurnosti, dok je iza nje emocija neutralno s 38 % [Slika 74].



Slika 74. Konačna predikcija emocije za peti zapis

Zapis 6, tužna izjava, [45], (7:06 – 7:10)

```
Predviđanje akustičkog modela:

{'anger': 0.04605437,
 'happiness': 0.041384116,
 'neutral': 0.26103017,
 'sadness': 0.65153134}
```

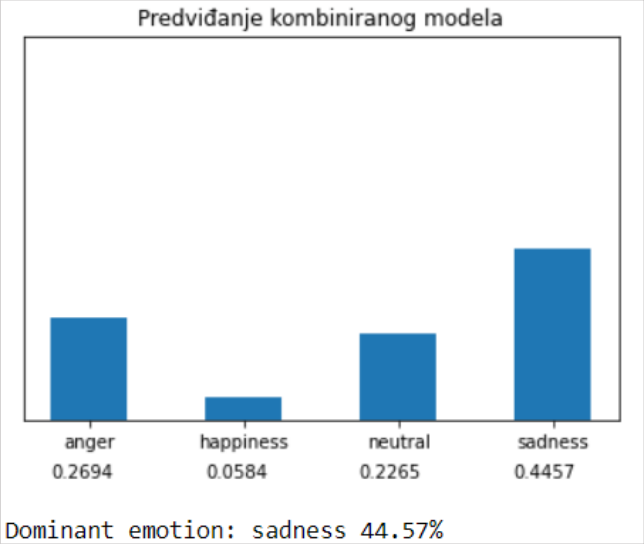
Slika 75. Predikcija emocije akustičkog modela za šesti zapis

Ponovno imamo tužnu izjavu i akustički model je to točno zaključio [Slika 75].

```
Tekst: why am I so sad

Predviđanje lingvističkog modela:

{'anger': 0.49282297,
 'happiness': 0.075317234,
 'neutral': 0.19192275,
 'sadness': 0.23993695}
```

Slika 76. Predikcija emocije lingvističkog modela za šesti zapis

Prva polovica izjave dobro je pretvorena u tekst, no druga polovica uopće nije pretvorena. Lingvistički model smatra da se ovdje radi o emociji ljutnje s 49 % sigurnosti, a nakon nje slijedi tuga s 24 % [Slika 76]. Kombinacijom se dobiva zaključak da ipak dominira emocija tuge [Slika 77].



Slika 77. Konačna predikcija emocije za šesti zapis

Zapis 7, sretna izjava, [46], (13:51 – 13:57)

```
Predviđanje akustičkog modela:

{'anger': 0.22165793,
 'happiness': 0.52092654,
 'neutral': 0.25441384,
 'sadness': 0.0030016096}
```

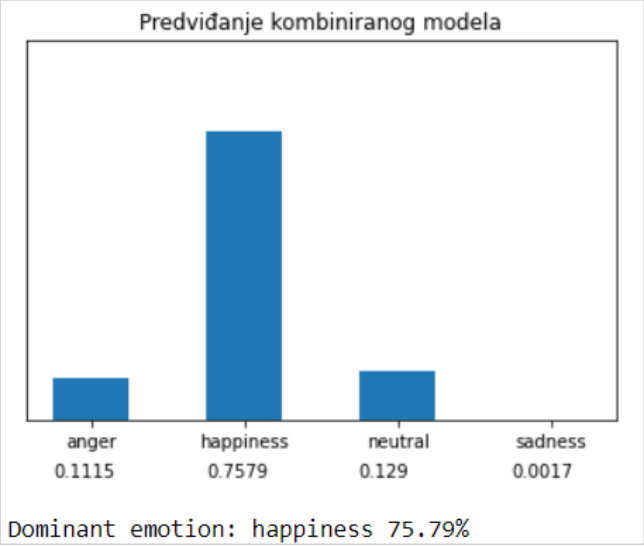
Slika 78. Predikcija emocije akustičkog modela za sedmi zapis

Ovdje imamo sretnu izjavu koju je akustički model dobro prepoznao [Slika 78].

```
Tekst: happy to be here it's very nice to see all of you

Predviđanje lingvističkog modela:

{'anger': 0.0012897124,
 'happiness': 0.99486864,
 'neutral': 0.0034999824,
 'sadness': 0.00034173013}
```

Slika 79. Predikcija emocije lingvističkog modela za sedmi zapis

Lingvistički model je u potpunosti siguran da se u ovom primjeru radi o emociji sreće [Slika 79], tako da i nakon kombiniranja dominantna emocija ostaje sreća [Slika 80].



Slika 80. Konačna predikcija emocije za sedmi zapis

Zapis 8, ljuta izjava, [46], (3:18 – 3:21)

```
Predviđanje akustičkog modela:

{'anger': 0.68899226,
 'happiness': 0.2587316,
 'neutral': 0.026133282,
 'sadness': 0.026142813}
```

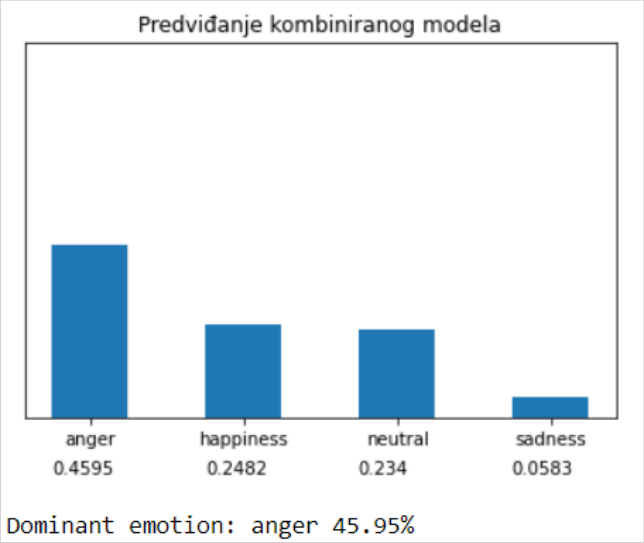
Slika 81. Predikcija emocije akustičkog modela za osmi zapis

I za kraj imamo još jednu ljutu izjavu. Akustički model je dobro pretpostavio da je u njoj prisutna emocija ljutnje [Slika 81].

```
Tekst: this is not funny it's totally unprofessional

Predviđanje lingvističkog modela:

{'anger': 0.22990896,
 'happiness': 0.23773776,
 'neutral': 0.44194013,
 'sadness': 0.09041315}
```

Slika 82. Predikcija emocije lingvističkog modela za osmi zapis

Ovdje lingvistički model, na temelju teksta, smatra da se radi o neutralnoj emociji s 44,2 % sigurnosti, nakon nje je sreća s 23,8 %, a tek na trećem mjestu je emocija ljutnje s 23 % [Slika 82]. No kombinacijom se dobiva točan zaključak da u ovoj izjavi ipak dominira emocija ljutnje [Slika 83].



Slika 83. Konačna predikcija emocije za osmi zapis

8. ZAKLJUČAK

U ovom radu korištena su dva načina prepoznavanja emocija iz govora. U prvom načinu upotrijebljena je konvolucijska neuronska mreža koja je učila akustičke značajke izvučene iz baze podataka sa zvučnim zapisima. U drugom načinu koristila se LSTM neuronska mreža koja je učila lingvističke značajke na temelju baze podataka s tekstualnim zapisima razgovora. Zatim su obje metode zajednički korištene za eksperimentalnu evaluaciju rada na ljudskim subjektima. Ovakav višemodalni pristup povećava točnost, no u nekim slučajevima može pogoršati rezultat. Primjerice, modul za pretvaranje govora u tekst nije savršen i neće uvijek ispravno prepoznati sve riječi, što može utjecati na točnost lingvističkog modela. To je poprilično izraženo kada osoba iskazuje intenzivne emocije i stoga govor bude manje razgovijetan. Pretvorba govora u tekst daje najbolje rezultate kada osoba govori jasno, umjerenom brzinom i neutralnim tonom, no tada će akustički model davati lošije rezultate budući da on zahtijeva izraženije emocije u govoru. Alati za pretvorbu govora u tekst počeli su se tek nedavno intenzivnije razvijati te će s vremenom davati sve bolje rezultate.

Prijedlozi za buduća poboljšanja su korištenje gotovih *Word Embedding* modela, kao što je npr. *Word2Vec*, koji već sadrže naučene vektore riječi i njihove odnose, što dodatno može poboljšati razumijevanje konteksta. Točnost prepoznavanja emocija potencijalno se može poboljšati i korištenjem nekih drugih kombinacija akustičkih značajki koje bolje opisuju različite emocije. Naprednije metode fuzije mogle bi pomoći u trenutcima kada su uvjeti rada za jedan od modaliteta nepovoljniji. Korištenje LSTM neuronske mreže i za akustičke i lingvističke značajke omogućilo bi prepoznavanje emocija u stvarnom vremenu paralelno s govorom osobe.

LITERATURA

[1] S. Shahsavarani, "Speech emotion recognition using convolutional neural networks", [diplomski rad], Lincoln: University of Nebraska; 2018.

[2] K. Cherry, "Emotions and Types of Emotional Responses", 2019., https://www.verywellmind.com/what-are-emotions-2795178 (pristupano: 29.6.2020.).

[3] P. S. Sreeja, G. S. Mahalakshmi, "Emotion Models: A Review", International Journal of Control Theory and Applications; 2017.

[4] B. Dropuljić, S. Skansi, L. Mršić, "Metodologija estimacije emocionalnih stanja na temelju akustičkih značajki govora", International Journal of DIGITAL TECHNOLOGY & ECONOMY; 2016.

[5] L. Kerkeni, Y. Serrestou, K. Raoof, C. Cléder, M. Mahjoub, M. Mbarki, "Automatic Speech Emotion Recognition Using Machine Learning", Social Media and Machine Learning; 2019.

[6] M. Essert, D. Ševerdija, I. Vazler, "Python - osnove", Digitalni udžbenik, Osijek; 2007.

[7] "Project Jupyter", https://jupyter.org/ (pristupano: 29.6.2020.).

[8] "Jupyter/IPython Notebook Quick Start Guide", https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html (pristupano: 29.6.2020.).

[9] B. Ramsundar, R. B. Zadeh, "TensorFlow for Deep Learning", O'Reilly Media; 2018.

[10] J. Brownlee, "14 Different Types of Learning in Machine Learning", https://machinelearningmastery.com/types-of-learning-in-machine-learning/ (pristupano 29.6.2020.).

[11] B. Dalbelo Bašić, M. Čupić, J. Šnajder, "Umjetne neuronske mreže", Zagreb: Fakultet elektrotehnike i računarstva; 2008.

[12] N. Bolf, "OSVJEŽIMO ZNANJE", Kem. Ind. 68 (5-6), 219–220; 2019.

[13] R. Yamashita, M. Nishio, R.K.G Do, K. Togashi, "Convolutional neural networks: an overview and application in radiology", Insights into Imaging 9, 611–629; 2018.

[14] K. T. O'Shea, R. Nash, "An Introduction to Convolutional Neural Networks", 2015.

[15] Md Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, Mst S. Nasrin et al., "A State-of-the-Art Survey on Deep Learning Theory and Architectures", 2019.

[16] "Keras LSTM tutorial – How to easily build a powerful deep learning language model", https://adventuresinmachinelearning.com/keras-lstm-tutorial/ (pristupano: 1.7.2020.).

[17] "Introduction to RNN and LSTM(Part-1)", 2020., https://mc.ai/introduction-to-rnn-and-lstmpart-1-2/ (pristupano: 29.6.2020.).

[18] J. Brownlee, "A Gentle Introduction to Long Short-Term Memory Networks by the Experts", 2017., https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ (pristupano: 29.6.2020.).

[19] J. Brownlee, "How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras", 2017., https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/ (pristupano: 1.7.2020.).

[20] V. Zayats, M. Ostendorf, H. Hajishirzi, "Disfluency Detection using a Bidirectional LSTM", 2016.

[21] J. Brownlee, "How to Avoid Overfitting in Deep Learning Neural Networks", 2018., https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/ (pristupano: 1.7.2020.).

[22] A. Sagar, "5 Techniques to Prevent Overfitting in Neural Networks", 2019. https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html (pristupano: 2.7.2020.).

[23] "Early Stopping with PyTorch to Restrain your Model from Overfitting", 2020., https://mc.ai/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting/ (pristupano: 1.7.2020.).

[24] A. Budhiraja, "Dropout in (Deep) Machine learning", 2016., https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (pristupano: 29.6.2020.).

[25] "Surrey Audio-Visual Expressed Emotion (SAVEE) Database", http://kahlan.eps.surrey.ac.uk/savee/Database.html (pristupano: 29.6.2020.).

[26] "Crowd Sourced Emotional Multimodal Actors Dataset (CREMA-D)", https://github.com/CheyneyComputerScience/CREMA-D (pristupano: 29.6.2020.).

[27] Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. https://doi.org/10.1371/journal.pone.0196391.

[28] Pichora-Fuller, M. Kathleen; Dupuis, Kate, 2020, "Toronto emotional speech set (TESS)", https://doi.org/10.5683/SP2/E8H2MF, Scholars Portal Dataverse, V1.

[29] "Berlin Database of Emotional Speech", http://emodb.bilderbar.info/docu/ (pristupano: 29.6.2020.).

[30] "Audacity", https://www.audacityteam.org/ (pristupano: 29.6.2020.).

[31] "Mel Frequency Cepstral Coefficient (MFCC) tutorial", http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ (pristupano: 29.6.2020.).

[32] "LibROSA", https://librosa.org/librosa/ (pristupano: 2.7.2020.).

[33] E. J. Lok, "Audio Emotion, Part 3 - Baseline model", 2019., https://www.kaggle.com/ejlok1/audio-emotion-part-3-baseline-model (pristupano: 29.6.2020.).

[34] V. Bushaev, "Understanding RMSprop — faster neural network learning", 2018., https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a (pristupano: 2.7.2020.).

[35] W. Li, H. Xu, "Text-based emotion classification using emotion cause extraction", Expert Systems with Applications, vol. 41, issue 4, part 2, p. 1742-1749; 2014.

[36] W. Wang, L. Chen, K. Thirunarayan, A. P. Sheth, "Harnessing Twitter "Big Data" for Automatic Emotion Identification," 2012 International Conference on Privacy, Security, Risk and Trust, Amsterdam, 2012, pp. 587-592, doi: 10.1109/SocialCom-PASSAT.2012.119.

[37] M. Hasan, E. Rundensteiner, E. Agu, "EMOTEX: Detecting Emotions in Twitter Messages", 2014 ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference, Stanford University, May 27-31, 2014.

[38] R. C. Balabantaray, M. Mudasir, S. Nibha, "Multi-Class Twitter Emotion Classification: A New Approach", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868; 2012.

[39] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, S. Niu, "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset", Asian Federation of Natural Language Processing; 2017. https://www.aclweb.org/anthology/I17-1099 (pristupano: 29.6.2020.).

[40] "English Language: Number of Words", https://wordinfo.info/unit/3925 (pristupano: 29.6.2020.).

[41] S. LI, "Multi Class Text Classification with LSTM using TensorFlow 2.0", 2019. https://towardsdatascience.com/multi-class-text-classification-with-lstm-using-tensorflow-2-0-d88627c10a35 (pristupano: 29.6.2020.).

[42] "Recording audio with pyaudio on a button click and stop recording on another buttonclick", https://stackoverflow.com/a/51229082 (pristupano: 2.7.2020.).

[43] "Audition Tapes - The Office cast", https://youtu.be/YHu9xcoo_qk (pristupano: 29.6.2020.).

[44] "The Office US - Best Moments - ALL SEASONS", https://youtu.be/DVPEsKCL20Q (pristupano: 29.6.2020.).

[45] "Saddest Moment - The Office US", https://youtu.be/S14aCP3e3yI (pristupano: 29.6.2020.).

[46] "The Office Season 2 INTRO COMPILATION!", https://youtu.be/dWFyRxZxE9A (pristupano: 29.6.2020.).

PRILOZI

I. CD-R disc

II. Python kôd

Pripremanje audio baze podataka.ipynb

```
# Ucitavanje potrebnih biblioteka
import pandas as pd
import os
```

```
# Definiranje lokacije svake baze
TESS = "./database/tess/TESS Toronto emotional speech set data/"
RAV = "./database/ravdess-emotional-speech-audio/audio_speech_actors_01-24/"
SAVEE = "./database/surrey-audiovisual-expressed-emotion-savee/ALL/"
CREMA = "./database/cremad/AudioWAV/"
EmoDB = "./database/emodb/wav/"
```

```
# Dohvacanje lokacije SAVEE baze podataka
dir_list = os.listdir(SAVEE)

# Ocitavanje emocija na temelju imena datoteka
emotion = []
path = []

for i in dir_list:
    if i[-8:-6] == '_a':
        emotion.append('male_angry')
    elif i[-8:-6] == '_h':
        emotion.append('male_happy')
    elif i[-8:-6] == '_n':
        emotion.append('male_neutral')
    elif i[-8:-6] == 'sa':
        emotion.append('male_sad')
    else:
        emotion.append('unknown')
    path.append(SAVEE + i)

# Postavljanje podataka u tablicu i provjera raspodjele labela
SAVEE_df = pd.DataFrame(emotion, columns = ['labels'])
SAVEE_df['source'] = 'SAVEE'
SAVEE_df = pd.concat([SAVEE_df,pd.DataFrame(path, columns = ['path'])], axis=1)
SAVEE_df.labels.value_counts()
```

```
# Dohvacanje lokacije RAVDESS baze podataka
dir_list = os.listdir(RAV)
dir_list.sort()

# Ocitavanje emocija na temelju imena datoteka
emotion = []
gender = []
path = []

for i in dir_list:
    fname = os.listdir(RAV + i)
    for f in fname:
        part = f.split('.')[0].split('-')
        emotion.append(int(part[2]))
        temp = int(part[6])
        if temp%2 == 0:
            temp = "female"
```

```
        else:
            temp = "male"
        gender.append(temp)
        path.append(RAV + i + '/' + f)

# Postavljanje podataka u tablicu i provjera raspodjele labela
RAV_df = pd.DataFrame(emotion)
RAV_df = RAV_df.replace({1:'neutral', 2:'unknown', 3:'happy', 4:'sad',
                         5:'angry', 6:'unknown', 7:'unknown', 8:'unknown'})
RAV_df = pd.concat([pd.DataFrame(gender),RAV_df], axis = 1)
RAV_df.columns = ['gender','emotion']
RAV_df['labels'] = RAV_df.gender + '_' + RAV_df.emotion
RAV_df['source'] = 'RAVDESS'
RAV_df = pd.concat([RAV_df,pd.DataFrame(path, columns = ['path'])], axis = 1)
RAV_df = RAV_df.drop(['gender', 'emotion'], axis = 1)
RAV_df.labels.value_counts()
```

```
# Dohvacanje lokacije TESS baze podataka
dir_list = os.listdir(TESS)
dir_list.sort()

# Ocitavanje emocija na temelju imena datoteka
path = []
emotion = []

for i in dir_list:
    fname = os.listdir(TESS + i)
    for f in fname:
        if i == 'OAF_angry' or i == 'YAF_angry':
            emotion.append('female_angry')
        elif i == 'OAF_happy' or i == 'YAF_happy':
            emotion.append('female_happy')
        elif i == 'OAF_neutral' or i == 'YAF_neutral':
            emotion.append('female_neutral')
        elif i == 'OAF_Sad' or i == 'YAF_sad':
            emotion.append('female_sad')
        else:
            emotion.append('unknown')
        path.append(TESS + i + "/" + f)

# Postavljanje podataka u tablicu i provjera raspodjele labela
TESS_df = pd.DataFrame(emotion, columns = ['labels'])
TESS_df['source'] = 'TESS'
TESS_df = pd.concat([TESS_df,pd.DataFrame(path, columns = ['path'])],axis=1)
TESS_df.labels.value_counts()
```

```
# Dohvacanje lokacije CREMA baze podataka
dir_list = os.listdir(CREMA)
dir_list.sort()

# Ocitavanje emocija na temelju imena datoteka
gender = []
emotion = []
path = []
female = [1002,1003,1004,1006,1007,1008,1009,1010,1012,1013,1018,1020,1021,
          1024,1025,1028,1029,1030,1037,1043,1046,1047,1049,1052,1053,1054,
          1055,1056,1058,1060,1061,1063,1072,1073,1074,1075,1076,1078,1079,
```

```
          1082,1084,1089,1091]

for i in dir_list:
    part = i.split('_')
    if int(part[0]) in female:
        temp = 'female'
    else:
        temp = 'male'
    gender.append(temp)
    if part[2] == 'SAD' and temp == 'male':
        emotion.append('male_sad')
    elif part[2] == 'ANG' and temp == 'male':
        emotion.append('male_angry')
    elif part[2] == 'HAP' and temp == 'male':
        emotion.append('male_happy')
    elif part[2] == 'NEU' and temp == 'male':
        emotion.append('male_neutral')
    elif part[2] == 'SAD' and temp == 'female':
        emotion.append('female_sad')
    elif part[2] == 'ANG' and temp == 'female':
        emotion.append('female_angry')
    elif part[2] == 'HAP' and temp == 'female':
        emotion.append('female_happy')
    elif part[2] == 'NEU' and temp == 'female':
        emotion.append('female_neutral')
    else:
        emotion.append('unknown')
    path.append(CREMA + i)

# Postavljanje podataka u tablicu i provjera raspodjele labela
CREMA_df = pd.DataFrame(emotion, columns = ['labels'])
CREMA_df['source'] = 'CREMA'
CREMA_df = pd.concat([CREMA_df,pd.DataFrame(path, columns = ['path'])],axis=1)
CREMA_df.labels.value_counts()
```

```
# Dohvacanje lokacije EmoDB baze podataka
dir_list = os.listdir(EmoDB)
dir_list.sort()

# Ocitavanje emocija na temelju imena datoteka
gender = []
emotion = []
path = []
female = ['08', '09', '13', '14', '16']

for i in dir_list:
    if i[0:2] in female:
        temp = 'female'
    else:
        temp = 'male'
    gender.append(temp)
    if i[5:6] == 'W' and temp == 'male':
        emotion.append('male_angry')
    elif i[5:6] == 'W' and temp == 'female':
        emotion.append('female_angry')
    elif i[5:6] == 'F' and temp == 'male':
        emotion.append('male_happy')
    elif i[5:6] == 'F' and temp == 'female':
```

```
        emotion.append('female_happy')
    elif i[5:6] == 'T' and temp == 'male':
        emotion.append('male_sad')
    elif i[5:6] == 'T' and temp == 'female':
        emotion.append('female_sad')
    elif i[5:6] == 'N' and temp == 'male':
        emotion.append('male_neutral')
    elif i[5:6] == 'N' and temp == 'female':
        emotion.append('female_neutral')
    else:
        emotion.append('unknown')
    path.append(EmoDB + i)

# Postavljanje podataka u tablicu i provjera raspodjele labela
EmoDB_df = pd.DataFrame(emotion, columns = ['labels'])
EmoDB_df['source'] = 'EmoDB'
EmoDB_df = pd.concat([EmoDB_df,pd.DataFrame(path, columns = ['path'])],axis=1)
EmoDB_df.labels.value_counts()
```

```
# Spajanje svih podataka u jednu zajednicku tablicu
df = pd.concat([SAVEE_df, RAV_df, TESS_df, CREMA_df, EmoDB_df], axis = 0)

# Uklanjanje nepotrebnih labela
df = df[df.labels != 'unknown']
df = df[df.labels != 'female_unknown']
df = df[df.labels != 'male_unknown']

print(df.labels.value_counts())
df.to_csv("data_path.csv", index=False)
df.head()
```

Ucenje akustickih znacajki.ipynb

```
# Ucitavanje potrebnih biblioteka
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Input, Flatten, Dropout, Activation, BatchNormalization
from keras.layers import Conv1D, MaxPooling1D
from keras.utils import np_utils, to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
import seaborn as sns
import librosa
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
import pickle
```

```
# Ucitavanje baze podataka
database = pd.read_csv("data_path.csv")
database.head()
```

```
# Mjesto u koje ce se spremati znacajke
df = pd.DataFrame(columns=['feature'])

# Iterativno izvlacenje znacajki iz cijele baze podataka
counter=0
for index, path in enumerate(database.path):
    print(path)
    # Ucitavanje pojedine datoteke
    data, sample_rate = librosa.load(path, res_type='kaiser_fast', sr=44100)
    sample_rate = np.array(sample_rate)
    # Uklanjanje tisine sa pocetka i kraja datoteke
    data, _ = librosa.effects.trim(y=data, top_db = 20)

    result=np.array([])
    # Izvlacenje MFCC znacajki
    mfccs=np.mean(librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40),axis=
1)
    result=np.hstack((result, mfccs))
    # Izvlacenje MEL znacajki
    mel=np.mean(librosa.feature.melspectrogram(data, sr=sample_rate), axis=1)
    result=np.hstack((result, mel))

    df.loc[counter] = [result]
    counter=counter+1
```

```
# Spremanje izvucenih znacajki
df.to_pickle('df_extracted_features.pkl')
df[:5]
```

```
# Ucitavanje izvucenih znacajki
df = pd.read_pickle('df_extracted_features.pkl')
```

```
# Pridruzivanje izvucenih znacajki u postojecu tablicu
df = pd.concat([database, pd.DataFrame(df['feature'].values.tolist())],axis=1)
df[:5]
```

```
print(df.shape)
```

```
# Podjela na dio za ucenje i testiranje
X_train, X_test, y_train, y_test=train_test_split(df.drop(['path','labels','sour
ce'],axis=1)
                                                  , df.labels
                                                  , test_size=0.2
                                                  , shuffle=True
                                                 )


X_train[:10]
```

```
# Pripremanje u oblik prikladan za Keras
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)

# One Hot encoding
le = LabelEncoder()
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.fit_transform(y_test)

y_train_cat = np_utils.to_categorical(y_train_encoded)
y_test_cat = np_utils.to_categorical(y_test_encoded)

print('X_train dimension: ', X_train.shape)
print('\nX_test dimension: ', X_test.shape)
print('\nClasses:\n', le.classes_)
print('\ny_train_categorical:\n', y_train_cat)
```

```
# Prosirivanje dimenzija kakve trazi CNN
X_train_exp = np.expand_dims(X_train, axis=2)
X_test_exp = np.expand_dims(X_test, axis=2)
print('X_train_exp dimension: ', X_train_exp.shape)
print('X_test_exp dimension: ', X_test_exp.shape)
```

```
# Definiranje modela
model = Sequential()
model.add(Conv1D(256, 8, padding='same',input_shape=(X_train_exp.shape[1],1)))
model.add(Activation('relu'))
model.add(Conv1D(256, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
```

```
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Flatten())
model.add(Dense(8))
model.add(Activation('softmax'))
opt = keras.optimizers.rmsprop(lr=0.00001, decay=1e-6)
model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accurac
y'])
model.summary()
```

```
# Ucenje
model_history=model.fit(X_train_exp, y_train_cat, batch_size=16, epochs=60,
                        validation_data=(X_test_exp, y_test_cat))
```

```
# Funkcija za crtanje grafova
def plot_graphs(model_history, string):
    plt.plot(model_history.history[string])
    plt.plot(model_history.history['val_'+string])
    plt.title('model '+string)
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()
```

```
# Grafovi tocnosti i gubitaka
plot_graphs(model_history, "accuracy")
plot_graphs(model_history, "loss")
```

```
score = model.evaluate(X_test_exp, y_test_cat, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))


# Predvidanja
predicted = model.predict(X_test_exp)
predicted = predicted.argmax(axis=1)
predicted = predicted.astype(int).flatten()
predicted = (le.inverse_transform((predicted)))

# Stvarne labele
true = y_test_cat.argmax(axis=1)
true = true.astype(int).flatten()
true = (le.inverse_transform((true)))

# Matrica konfuzije
cf_matrix = confusion_matrix(true, predicted)
plt.figure(figsize = (10,7))
heatmap=sns.heatmap(cf_matrix, annot=True, fmt="d", cmap='Blues')
heatmap.yaxis.set_ticklabels(le.classes_, rotation=0, fontsize=12)
heatmap.xaxis.set_ticklabels(le.classes_, rotation=45, fontsize=12)
```

```
plt.ylabel('True label')
plt.xlabel('Predicted label')
print('\nConfusion matrix:')
```

```
# Spremanje modela
model_name = 'akusticki_model.h5'
save_dir = os.path.join(os.getcwd(), 'saved_models')

if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Model saved to %s ' % model_path)

model_json = model.to_json()
with open('saved_models/akusticki_model.json', 'w') as json_file:
    json_file.write(model_json)
```

Pretvaranje iz TXT u CSV.ipynb

```
dialogues_file = './database_text/dialogues_text.txt'
with open(dialogues_file, 'r', encoding='utf8') as file:
    data = file.read().replace('__eou__', '\n')
    data = data.replace('\n\n', '\n')
    data = data.replace('’', '\'')
    data = data.replace(' \' t', '\'t')
    data = data.replace(' \' m', '\'m')
    data = data.replace(' \' re', '\'re')
    data = data.replace(' \' s', ' is')
    data = data.replace(' \' d', ' would')
    data = data.replace(' \' Ve', ' have')
    data = data.replace(' \' ll', ' will')
    data = data.replace(' .', '.')
    data = data.replace(' ,',',')
    data = data.replace(' ?', '?')
    data = data.replace(' !', '!')

expressions = data.splitlines() #stavlja svaku izjavu u listu

counter = 0
for row in expressions:
    counter += 1
print('Ukupan broj izjava:', counter)
```

```
# Slican postupak i za labele
labels_file = './database_text/dialogues_emotion_corrected.txt'
with open(labels_file, 'r', encoding='utf8') as file:
    data = file.read().replace(' ', '\n')
    data = data.replace('\n\n', '\n')

labels = data.splitlines() #stavlja svaku oznaku u listu

counter = 0
for row in labels:
    counter += 1
print('Ukupan broj labela:', counter)
```

```
import csv
combined = zip(labels, expressions)

new_file = './database_text/DailyDialog_database.csv'
with open(new_file,'w', encoding='utf8', newline='') as f:
    writer = csv.writer(f)
    for row in combined:
        writer.writerow(row)
```

Brojac izjava i labela.ipynb

```
# Program broji koliko ima izjava u svakom retku (dijalogu) kako bi se
# usporedilo s brojem labela jer se ne poklapaju
import re

f1 = './database_text/dialogues_text.txt'
f2 = './database_text/dialogues_emotion.txt'
pattern1 = '__eou__'
pattern2 = ' '
counter = 0

with open(f1,'r',encoding='utf8') as f1, open(f2,'r',encoding='utf8') as f2:
    for line1, line2 in zip(f1, f2):
        counter += 1
        count1 = sum(1 for match in re.finditer(pattern1, line1))
        count2 = sum(1 for match in re.finditer(pattern2, line2))
        if count1 != count2:
            print('Nepodudaranje u %d. retku!' % counter)
```

```
import tensorflow as tf
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras import optimizers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from keras.utils import np_utils
import seaborn as sns
import csv
import io
import json
import numpy as np
import matplotlib.pyplot as plt
import re
```

```
vocab_size = 10000
embedding_dim = 64
```

```
# Inicijaliziranje mjesta za spremanje izraza i labela
expressions = []
labels = []

# Definiranje nepotrebnih rijeci
stopwords = ['i', 'you', 'he', 'she', 'it', 'we', 'they', 'i\'m', 'i\'ll',
             'you\'re', 'am', 'are', 'is', 'a', 'the', 'and']

# Otvaranje baze podataka
counter=0
text_database = './database_text/DailyDialog_database.csv'
with open(text_database, 'r', encoding="utf8") as csvfile:
    file = csv.reader(csvfile, delimiter=',')
    # Kroz bazu se ide red po red
    for row in file:
        counter +=1
        print(row[0])
        # Sprema se labela za trenutni redak
        labels.append(row[0])
        print(row[1])
        # Izdvaja se izjava, sva slova se pretvaraju u mala
        expression = row[1].lower()
        # Uklanjaju se nepotrebni znakovi
        expression = re.sub('[^a-z\']+', ' ', expression)
        stripped_expression = ''
         # Ide se rijec po rijec u izjavi
        for word in expression.split():
            # Ako se rijec nalazi unutar stopwords, onda se uklanja
            if word in stopwords: word = ''
            else: word = word + ' '
            # Sastavljanje nove ociscene recenice
            stripped_expression = stripped_expression + word
        print(stripped_expression, '\n')
        # Dodavanje recenice u pripremljeno mjesto
```

```
        expressions.append(stripped_expression)
print(len(labels))
print(len(expressions))
```

```
tokenizer = Tokenizer(num_words = vocab_size, oov_token='<OOV>')
tokenizer.fit_on_texts(expressions)
expressions_sequences = tokenizer.texts_to_sequences(expressions)
expressions_sequences_padded = pad_sequences(expressions_sequences, maxlen=100,
                                             padding='post', truncating='post')

dict(list(tokenizer.word_index.items())[:6])
```

```
print(expressions_sequences_padded[100])
```

```
tokenizer_json = tokenizer.to_json()
with io.open('tokenizer.json', 'w', encoding='utf-8') as file:
    file.write(json.dumps(tokenizer_json, ensure_ascii=False))
```

```
X_train, X_test, y_train, y_test = train_test_split(expressions_sequences_padded
                                                    , labels
                                                    , test_size=0.2
                                                    , shuffle=True
                                                   )
```

```
le = LabelEncoder()
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.fit_transform(y_test)

le_name_mapping = dict(zip(le.transform(le.classes_), le.classes_))
print('Label Encoding Classes as ')
print(le_name_mapping)

y_train_cat = np_utils.to_categorical(y_train_encoded)
y_test_cat = np_utils.to_categorical(y_test_encoded)
print('\nOne Hot Encoded class shape ')
print('Training set:', y_train_cat.shape)
print('Testing set:', y_test_cat.shape)
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embedding_dim
                                                       , dropout=0.9
                                                       )),
    tf.keras.layers.Dense(embedding_dim, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])

adam = optimizers.Adam(lr=0.001)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accurac
y'])
model.summary()
```

```
history = model.fit(X_train, y_train_cat, epochs=20,
                    validation_data=(X_test, y_test_cat), verbose=2)
```

```
def plot_graphs(history, string):
    plt.plot(history.history[string])
    plt.plot(history.history['val_'+string])
    plt.xlabel("Epochs")
    plt.ylabel(string)
    plt.legend([string, 'val_'+string])
    plt.show()
```

```
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```

```
score = model.evaluate(X_test, y_test_cat, verbose=0)
print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))


# Predviđanja
preds = model.predict(X_test)
preds = preds.argmax(axis=1)
preds = preds.astype(int).flatten()
preds = (le.inverse_transform((preds)))

# Stvarne labele
actual = y_test_cat.argmax(axis=1)
actual = actual.astype(int).flatten()
actual = (le.inverse_transform((actual)))

cf_matrix = confusion_matrix(actual, preds)

plt.figure(figsize = (10,7))
heatmap=sns.heatmap(cf_matrix, annot=True, fmt="d", cmap='Blues')
heatmap.yaxis.set_ticklabels(le.classes_, rotation=0, fontsize=12)
heatmap.xaxis.set_ticklabels(le.classes_, rotation=45, fontsize=12)
plt.ylabel('True label')
plt.xlabel('Predicted label')
print('\nConfusion matrix:')
```

```
model.save('saved_models/lingvisticki_model.h5')
```

Prepoznavanje.ipynb

```
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.text import tokenizer_from_json
from tensorflow.keras.preprocessing.sequence import pad_sequences
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
from keras.models import model_from_json
import IPython.display as ipd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder
import speech_recognition
import json
import tkinter
import pyaudio
import wave
import os
import librosa
import librosa.display
```

```
le = LabelEncoder()

# ucitavanje akusickog modela
json_file = open('saved_models/akusticki_model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# Ucitavanje tezina u model
loaded_model.load_weights('saved_models/akusticki_model.h5')
print('Model loaded from disk')

# Ucitavanje lingvistickog modela
model_txt = load_model('saved_models/lingvisticki_model.h5')
# ucitavanje tokena rijeci
with open('tokenizer.json') as f:
    data = json.load(f)
    tokenizer = tokenizer_from_json(data)
```

```
class RecAUD:
    def __init__(self, chunk=1024, frmat=pyaudio.paInt16, channels=1,
                 rate=44100, py=pyaudio.PyAudio()):
        # Start Tkinter and set Title
        self.main = tkinter.Tk()
        self.collections = []
        self.main.geometry('280x150')
        self.main.title('Record')
        self.CHUNK = chunk
        self.FORMAT = frmat
        self.CHANNELS = channels
        self.RATE = rate
        self.p = py
        self.frames = []
        self.st = 1
        self.stream = self.p.open(format=self.FORMAT, channels=self.CHANNELS,
                                  rate=self.RATE, input=True,
```

```
                                    frames_per_buffer=self.CHUNK)
        # Set Frames
        self.buttons = tkinter.Frame(self.main, padx=40, pady=20)

        # Pack Frame
        self.buttons.pack(fill=tkinter.BOTH)

        # Start and Stop buttons
        self.strt_rec = tkinter.Button(self.buttons, width=10, padx=10, pady=5,
                                       text='Start Recording',
                                       command=lambda: self.start_record())

        self.strt_rec.grid(row=0, column=0, padx=50, pady=5)

        self.stop_rec = tkinter.Button(self.buttons, width=10, padx=10, pady=5,
                                       text='Stop Recording',
                                       command=lambda: self.stop())

        self.stop_rec.grid(row=1, column=0, columnspan=1, padx=50, pady=5)

        tkinter.mainloop()

    def start_record(self):
        self.st = 1
        self.frames = []
        stream = self.p.open(format=self.FORMAT, channels=self.CHANNELS,
                             rate=self.RATE, input=True,
                             frames_per_buffer=self.CHUNK)

        print("* recording")
        while self.st == 1:
            data = stream.read(self.CHUNK)
            self.frames.append(data)
            self.main.update()

        stream.close()

        wf = wave.open('test_audio/recording.wav', 'wb')
        wf.setnchannels(self.CHANNELS)
        wf.setsampwidth(self.p.get_sample_size(self.FORMAT))
        wf.setframerate(self.RATE)
        wf.writeframes(b''.join(self.frames))
        wf.close()

    def stop(self):
        self.st = 0
        print("* stop")
```

```
# Snimanje zvuka pomocu mikrofona
guiAUD = RecAUD()
```

Prepoznavanje akustickih znacajki

```
file_name = 'test_audio/Zapis_8.wav'
#file_name = 'test_audio/recording.wav'
data, sample_rate = librosa.load(file_name,res_type='kaiser_fast',sr=44100)
```

```
data, _ = librosa.effects.trim(y=data, top_db = 20)

plt.figure(figsize=(8, 4))
librosa.display.waveplot(data, sr=sample_rate)

ipd.Audio(file_name)
```

```
features=np.array([])

mfccs = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40), axis=1)
features=np.hstack((features, mfccs))

mel=np.mean(librosa.feature.melspectrogram(data, sr=sample_rate), axis=1)
features=np.hstack((features, mel))

features = np.expand_dims(features, axis=0)
features = np.expand_dims(features, axis=-1)
features.shape
```

```
p = loaded_model.predict(features, batch_size=16, verbose=0)
classes = ['anger', 'happiness', 'neutral', 'sadness']
pred_audio=[p[0][0]+p[0][4], p[0][1]+p[0][5], p[0][2]+p[0][6], p[0][3]+p[0][7]]
pred_audio = np.array(pred_audio)

print('Predviđanje akustičkog modela:')
dict(zip(classes, pred_audio))
```

Prepoznavanje lingvistickih znacajki

```
# Prepoznavanje govora
r = speech_recognition.Recognizer()
with speech_recognition.AudioFile(file_name) as source:
    #r.adjust_for_ambient_noise(source, duration=1)
    audio_text = r.listen(source)
    try:
        text = r.recognize_google(audio_text, language = 'en-US')
        print('Converting into text...')
        print(text)

    except speech_recognition.UnknownValueError:
        print('Google Speech Recognition could not understand audio')

    except speech_recognition.RequestError as e:
        print('Google Speech Recognition service error; {0}'.format(e))
```

```
txt=[text]
seq = tokenizer.texts_to_sequences(txt)
padded = pad_sequences(seq, maxlen=100, padding='post', truncating='post')
pred_txt = model_txt.predict(padded)
classes=['anger', 'happiness', 'neutral', 'sadness']

print('Tekst:', text)
print('\nPredviđanje lingvističkog modela:')
dict(zip(classes, pred_txt[0]))
```

Kombinirano prepoznavanje

```
# Konacna predikcija
fusion = 0.5*pred_audio+0.5*pred_txt

plt.figure(figsize=(6,4))
plt.title('Predviđanje kombiniranog modela')
plt.plot()
plt.grid(False)
plt.yticks([])
bars = plt.bar(range(len(classes)), fusion[0], width=0.5)
plt.ylim([0, 1])
plt.xticks(range(4), classes)

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x(), -0.15, round(yval, 4))
plt.show()
print('Dominant emotion: {} {:0.2f}%'.format(classes[np.argmax(fusion)],
                                             100*np.max(fusion)))
```