Automatsko snimanje i transkribiranje snimaka

Battelli, Piero

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

AUTOMATSKO SNIMANJE I TRANSKRIBIRANJE SNIMAKA

Rijeka, ožujak 2024.

Piero Battelli

0069082557

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

AUTOMATSKO SNIMANJE I TRANSKRIBIRANJE SNIMAKA

Mentor: Prof. dr. sc. Ivo Ipšić

Rijeka, ožujak 2024.

Piero Battelli

0069082557

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 13. ožujka 2023.

Zavod: **Zavod za računarstvo**
Predmet: **Računalna obrada govora i jezika**

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Piero Battelli (0069082557)**
Studij: Sveučilišni diplomski studij računarstva
Modul: Računalni sustavi

Zadatak: **Automatsko snimanje i transkribiranje snimaka**

Opis zadatka:

Za potrebe razvoja sustava za automatsko raspoznavanje hrvatskog govora realizirajte programsku opremu za snimanje vremenskih izvješća dostupnih putem Internet stranica HRT-a. Snimke je potrebno formatirati i organizirati sukladno postojećim snimcima u bazi VEPRAD. Pomoću programskog alata otvorenog koda HTK i razvijenih akustičnih i jezičnim modela hrvatskog govora potrebno je generirati transkripcije snimljenih izvješća.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Prof. dr. sc. Ivo Ipšić

Predsjednik povjerenstva za diplomski ispit:

Prof. dr. sc. Miroslav Joler

Izjava o samostalnoj izradi rada

Izjavljujem da sam diplomski rad izradio samostalno.

Rijeka, ožujak 2024. _____

Piero Battelli

Sadržaj

1. UVOD

Tema ovog diplomskog rada je proširenje baze govornih snimaka VrEmenske Prognoze-RADio (VEPRAD) sa novim snimkama i njihovim transkripcijama na razini riječi i fonema. Uz to, za ubrzanje samog postupka prikupljanja podataka potrebno je razviti Python skriptu koja će se koristiti kao snimalica vremenskih prognoza kojom se automatski pokreće snimanje na dnevnoj bazi. Prije opisa samih postupaka potrebnih za realizaciju ovog zadatka u nastavku će biti opisano raspoznavanje govora kao znanstvena disciplina, njezina primjena i značajke te najčešće korišteni algoritmi. Zatim će biti opisan povijesni razvoj prepoznavanja govora, korišteni alati, izrada skripte za automatizaciju snimanja, izrada transkripcija snimaka, rezultati rada te zaključak.

1.1 Prepoznavanje govora

Prepoznavanje govora poznatije kao "speech-to-text" je sposobnost stroja ili programa da identificira izgovorene riječi i pretvori ih u čitljiv tekst [1]. Osnovni softver za prepoznavanje govora ima ograničeni vokabular te može identificirati isključivo riječi i fraze čistog izgovora dok napredniji softver može raditi s prirodnim govorom, različitim izgovorima riječi i naglascima te različitim jezicima. Potrebno je razlikovati prepoznavanje govora od prepoznavanja glasa jer je prepoznavanje glasa biometrijska tehnologija za identifikaciju glasa pojedinca što nema nikakve veze s područjem prepoznavanje govora kojeg se ovaj rad dotiče. Prepoznavanje govora temelji se na računalnim algoritmima koji procesiraju i tumače izgovorene riječi i pretvaraju ih u tekst tj. softversko rješenje pretvara zvuk sniman mikrofonom u pisani jezik razumljiv računalu i čovjeku. Softver korišten za prepoznavanje govora mora biti prilagodljiv na veliku raznolikost i kontekstualnu specifičnost ljudskog jezika te se stoga trenira nad različitim uzorcima i stilovima govora, jezicima, frazama itd., a kako bi sustav za prepoznavanje govora zadovoljavao navedene uvjete koristi dva tipa modela: akustični i jezični model. Akustični model predstavlja vezu između jezičnih jedinica govora i audio signala dok pojam jezičnog modela predstavlja povezivanje zvuka sa sekvencama riječi kako bi se moglo razlikovati riječi sličnog izgovora.

1.2. Primjena i značajke prepoznavanja govora

Najpoznatija primjena prepoznavanja govora je u brojnim aplikacijama korištenih na mobilnim uređajima. Oni nam omogućuju unos glasovnih naredbi za slanje poruka, pokretanje poziva, speech-to-text procesiranje i slično. Osim toga, na današnjim pametnim telefonima postoje tzv. virtualni asistenti poput Siri na iOS operacijskom sustavu koji koriste prepoznavanje govora za komunikaciju s korisnikom. Prepoznavanje govora još se koristi na područjima obrazovanja, zdravstva, customer service-a, hands-free komunikacije, prepoznavanje emocija u glasu i slično. Neke od ključnih značajki prepoznavanje govora su "vaganje" jezika tj. ova značajka govori algoritmu da mora veću pozornost posvetiti pojedinim riječima koje se često izgovaraju ili riječima jedinstvenim trenutnom razgovoru. Osim toga s pomoću prepoznavanja govora moguće je ukloniti ambijentalni šum koji kvari kvalitetu audio datoteka, moguće je označavati pojedine govornike koji sudjeluju u razgovoru te filtrirati proste ili neželjene riječi.

1.3. Algoritmi za prepoznavanja govora

Kako bi uspješno mogli realizirati prepoznavanje govora potrebno je koristiti neke od provjerenih i efikasnih algoritama. Konkretno u sklopu ovog rada korišteni su skriveni Markovljevi modeli poznatiji kao HMM (Hidden Markov model). Oni su korišteni u autonomnim sustavima gdje su stanja sustava samo djelomično poznata ili u slučaju da nemamo svu potrebnu informaciju dostupnu, ali nešto detaljnije o skrivenim Markovljevim modelima kasnije. Često se u svrhu prepoznavanja koriste i algoritmi iz svijeta umjetne inteligencije poput neuronskih mreža, NLP tj. natural language processing i slični algoritmi [1].

2. POVIJEST RAZVOJA SUSTAVA ZA RASPOZNAVANJE GOVORA

Već 50-ih godina prošlog stoljeća započeo je razvoj modernih tehnologija za prepoznavanje govora od kojih prvi nastaje sustav Audrey dizajniran od strane Bell Laboratories. Radi se o sustavu koji je svojim dimenzijama zauzimao čitavu prostoriju, a mogao je prepoznati isključivo brojeve od jedan do devet izgovorene od strane samog programera stroja s preciznošću od čak 90%. Stroj je primjenjivan za olakšanje telefonskih poziva, ali zbog velike cijene i raznolikosti glasova nije bio previše praktičan. Desetak godina kasnije, 1962. godine, predstavljen je IBM-ov Shoebox koji je mogao raspoznati ukupno 16 riječi, no radi se o sustavu koji je još uvijek bio jako ograničen i zahtijevao je spor izgovor i pauzu među riječima kako bi mogao zabilježiti što je čuo [4]. Početkom 70-ih godina Ministarstvo obrane SAD-a prepoznaje značaj tehnologija prepoznavanja govora i potencijal koji su te tehnologije imale u vojsci. Upravo zbog toga to desetljeće obilježeno je izrazito brzim rastom vokabulara te nastaje program Harpy koji je bio u stanju prepoznati preko tisuću riječi što predstavlja vokabular malog djeteta. U to vrijeme nastaje i sve više igračaka za djecu koje su prepoznavale različite glasove i služile kao pomoć pri učenju pisanja riječi. Napredak u ovome području stagnirao je dugačak niz godina zbog ograničenosti dostupnih tehnologija koje su zaostajale za idejama inovatora. Najveća ograničenja predstavljali su snaga procesora i memorija. No, Google je 2010. godine široj javnosti predstavio Google Voice Search aplikaciju koja je označavala prekretnicu i značajno ubrzala razvoj prepoznavanja govora. Cilj navedene aplikacije bio je olakšati ljudima tako da umjesto tipkanja po sitnim ekranima pametnih telefona tog vremena mogu jednostavno reći svome uređaju što žele napisati u npr. poruci ili pretraživanju. Već iduće godine Apple predstavlja Siri, a radi se o virtualnom asistentu koji je u stanju odgovoriti na jako veliki broj pitanja i posjeduje sposobnost vođenja razgovora s korisnikom. U današnje vrijeme postoji jako puno virtualnih asistenata te se radi na razvoju softvera za diktiranje i brojnim integracijama prepoznavanja govora u našu svakodnevicu.

3. PROGRAMSKI ALATI ZA OBRADU GOVORA

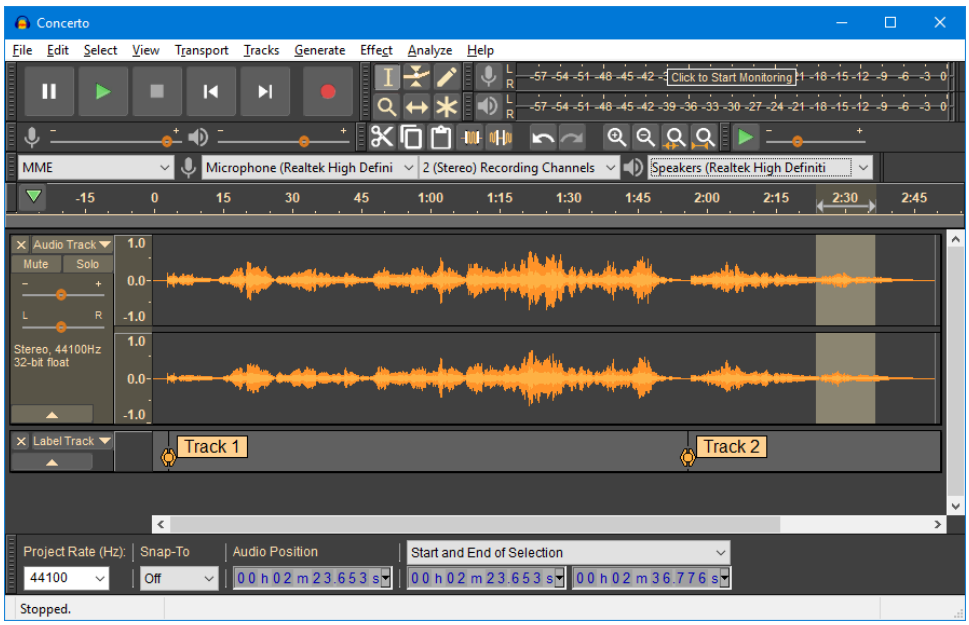
Prije početka rada bilo je potrebno upoznati se s alatima i tehnologijama koji će biti korištene, a u nastavku ovog poglavlja slijedi njihov opis. Uz opis navedene će biti i hiperveze za preuzimanje tih alata na vlastita računala.

3.1 Python

Python je iznimno popularan high-level programski jezik opće namjene s fokusom na čitkost samog koda. On podržava višestruke programske paradigme uključujući strukturirano, objektno-orijentirano i funkcijsko programiranje, a također sadržava jako velik broj standardnih knjižnica funkcija. U današnje vrijeme većina ljudi koristi Python 3 izdan 2008. godine iako još uvijek postoji mogućnost korištenja Python 2 programskog jezika za jako specifične zadatke. Samo neke od prednosti Python 3 programskog jezika su jednostavnost učenja i čitanja koda, njegova iznimno raznolika primjena, podržava jako velik broj standardnih i proširenih knjižnica, jako ga je lako integrirati s drugim jezicima poput C-a ili Jave, sadrži jako opširnu dokumentaciju itd. U ovome se radu Python koristi zbog njegovih sposobnosti vezanih uz automatizaciju i izradu skripti kako bi izradili skriptu koja će snimati vremenske prognoze u zadanim intervalima. Automatizacija se odnosi na proces automatiziranja ponavljajućih zadataka ili procesa koristeći softver ili strojeve. Postoje brojne knjižnice funkcija u sklopu Pythona koje to omogućuju, a neke od najpoznatijih su Selenium, Pandas, Twilio, Pillow te PyAutoGUI koja je korištena u sklopu ovog rada, ali detaljnije o tome nešto kasnije. Python je moguće preuzeti na vlastito računalo sa službene web stranice Welcome to Python.org.

3.2 Audacity

Audacity je audio editor koji je besplatan i jako jednostavan za korištenje, a dostupan je na brojnim operacijskim sustavima poput Windows-a, MacOS-a, GNU/Linux-a itd. Kako je izvorni kod Audacity-a dostupan svima na proučavanje ili korištenje može se zaključiti da se radi o open source softveru [6]. Audacity se može koristiti za snimanje zvuka s pomoću mikrofona ili za digitalizaciju snimki s drugih medija, omogućuje raznolike formate zvuka od 16, 24 i 32 bita, jako je jednostavno uređivanje audio datoteka s pomoću osnovnih cut, copy, paste i delete alata, moguće je dodavati efekte na sam zvuk, postaviti odabranu temu i urediti izgled editora itd. Primarno se u ovome radu koristi Audacity za snimanje vremenskih prognoza i pohranu dobivenih audio datoteka nakon skraćivanja čitave vremenske prognoze na manje isječke. Softver je moguće preuzeti sa službene stranice [Audacity ® | Free, open source, cross-platform audio software for multi-track recording and editing. (audacityteam.org)]



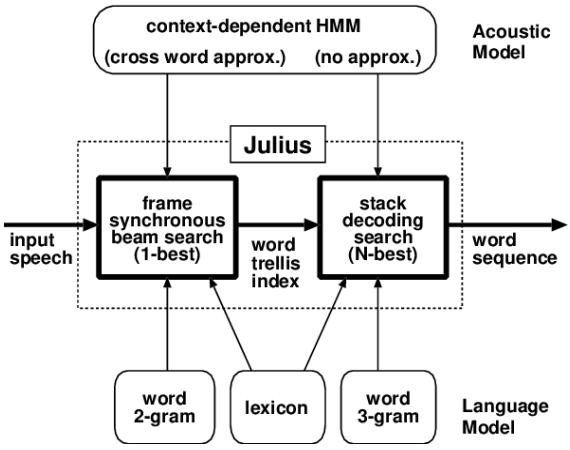
Slika 3.2.1. Izgled Audacity aplikacijskog okvira s dark temom [6]

3.3 Julia

Julia je od samog početka dizajnirana za visoke performanse. Radi se o programskome jeziku koji je dinamično pisan pa ostavlja dojam da se radi o skriptnome jeziku te posjeduje jako dobru sposobnost ponavljanja okruženja u kojem se radi duž više platformi [7]. Samo neke od značajki Julie su asinkroni input/output, metaprogramiranje, debugging, logiranje itd., a važno je napomenuti da se radi o open source projektu kojeg održava preko 1000 ljudi. Za potrebe ovog projekta pomoću Julie stvara se akustični model što će detaljnije biti opisano u sljedećim poglavljima. Juliu je moguće preuzeti sa službene web stranice Download Julia (julialang.org).

3.4 Julius

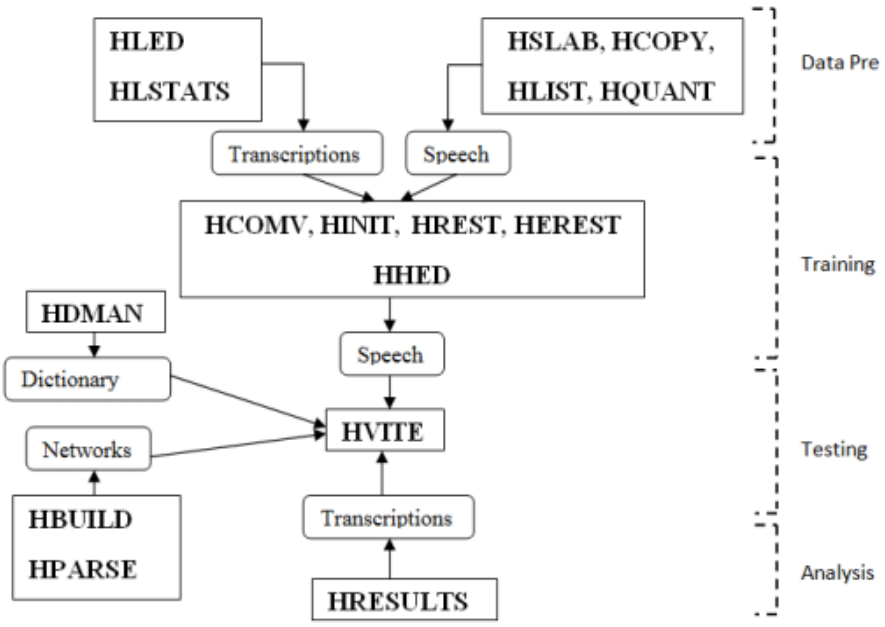
Julius je LVCSR (large vocabulary continuous speech recognition) dekoder visokih performansi kojeg koriste istraživači na području govora i razni developer [8]. Omogućava izvođenje dekodiranja u gotovo pravom vremenu na trenutnim računalima kod zadataka diktacije sa 60 000 riječi. Tipično se Julius koristi na Linuxu iako radi i na Windows operacijskom sustavu. Neke od značajki Julius-a su rad u pravom vremenu, open source kod, čitava dokumentacija dostupna na engleskom i japanskom jeziku, nije zahtjevan za pokretanje, jako visoke preciznosti itd. Iako za potrebe ovog rada Julius nije korišten on je jako koristan alat jer omogućuje provjeru uspješnosti izrade transkripcija što je vrijedna značajka te je on proučen u slučaju da se naknadno mora koristiti. Julius je moguće preuzeti sa službene web stranice Open-Source Large Vocabulary CSR Engine Julius (osdn.jp)



Slika 3.4.1. Princip rada Julius dekodera [8]

3.5 HTK Toolkit

Hidden Markov Model Toolkit poznatiji kao HTK Toolkit koristan je alat za izgradnju i manipulaciju Markovljevih modela. Iako je primarna upotreba ovog alata u istraživanjima vezanim uz prepoznavanje govora često se koristi i kod sinteze govora, prepoznavanja znakova te za DNA sekvenciranje. HTK Toolkit čini set modula knjižnica i alata dostupnih u C izvornom obliku. Ti alati omogućuju analizu govora, treniranje skrivenih Markovljevih modela, analizu testova i rezultata itd. Softver podržava skrivene Markovljeve modele dobivene s pomoću Gaussove mješavine kontinuirane gustoće te diskretne distribucije i može se koristiti za izgradnju kompleksnih skrivenih Markovljevih modela. Iako postoje određene limitacije vezane uz sam HTK Toolkit on je otvorenog koda i sve korisnike i istraživače se potiče na sudjelovanje u razvoju. Za preuzimanje HTK Toolkita potrebno je najprije obaviti registraciju putem službene web stranice, a uz preuzimanje samog Toolkita preporučuje se i preuzimanje HTK book-a koji predstavlja dokumentaciju koja se sastoji od detaljnih instrukcija i uputa kako koristiti sam alat. HTK Toolkit i HTK book moguće je preuzeti sa sljedeće hiperveze HTK Speech Recognition Toolkit (cam.ac.uk).



Slika 3.5.1. Faze izrade sustava prepoznavanja govora pomoću HTK Toolkit-a [8]

4. PRIPREMA PODATAKA I AUTOMATIZACIJA SNIMANJA

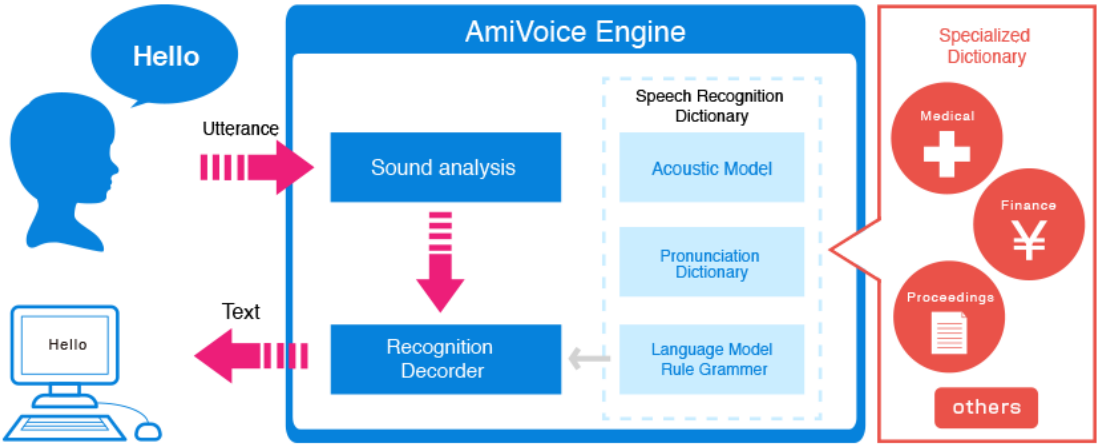
Izrada transkripcija uvijek započinje s prikupljanjem i pripremom podataka. Tri su osnovne komponente koje čine svaki Speech Recognition Engine (SRE): jezični model (ili gramatika), akustični model i dekoder.

Jezični model sastoji se od jako širokog popisa riječi i njihove vjerojatnosti pojavljivanja u danoj sekvenci, a koristi se uglavnom u diktacijskim aplikacijama. Gramatika je pak puno manja datoteka koja se sastoji od skupa unaprijed definiranih kombinacija riječi, a ona se koristi za tzv. Command and Control aplikacije na računalima. Svaka riječ jezičnog modela ili gramatike ima popis fonema s kojom je povezana, a fonem predstavlja specifične zvukove koji čine određene riječi. S pomoću gramatike mi definiramo ograničenja za naš SRE tj. dajemo mu do znanja koje riječi može očekivati kao input. Jednostavnije rečeno, gramatika predstavlja popis riječi ili rečenica koje SRE sluša i nastoji prepoznati. Kada se prepozna jedna od riječi ili fraza koje "očekujemo" SRE vraća tu riječ odnosno frazu pozivnom programu što može biti Dialog Manager ili neka skripta pisana u Python-u i slično. Bitno je zapamtiti da u gramatici možemo koristiti isključivo one riječi koje smo "trenirali" u našem akustičnom modelu pa se da zaključiti da je ovisnost između gramatike i akustičnog modela jako velika.

Akustični model sadrži statističku reprezentaciju specifičnih zvukova koji čine svaku pojedinu riječ u jezičnom modelu ili gramatici. Svaki pojedini zvuk odgovara jednom fonemu. Akustični model mora sadržati zvukove za svaku riječ korištenu u našoj gramatici. Riječi iz gramatike daju SRE sekvencu zvukova koje mora slušati, a zatim SRE po pronalasku specifične sekvence vraća istu sekvencu u tekstualnom obliku pozivnom programu. Stoga se zaključuje da kada SRE sluša riječi on zapravo sluša sekvence zvukova koji čine jednu od riječi koje smo definirali u gramatici i time se potvrđuje zajedničko djelovanje akustičnog modela i gramatike. SRE za komercijalnu uporabu koriste jako velike baze podataka kako bi izradili akustične modele i upravo zbog toga se najčešće korištene riječi unaprijed uključuju u akustični model. Pri izradi vlastitog akustičnog modela i gramatike važno je provjeriti da su svi fonemi koji čine riječi u našoj gramatici uključeni i u akustični model.

Posljednja komponenta koja čini SRE je dekoder. Dekoder je softverski program koji pretražuje akustični model kako bi pronašao ekvivalentni zvuk onome kojeg je upravo čuo. U trenutku kada se utvrdi par dekoder određuje foneme koji odgovaraju zadanom zvuku te ih pamti sve do prve kratke ili dugačke pauze u govoru. Nakon toga se pretražuju jezični model ili gramatika kako bi se došlo do para zvukova koji odgovara toj sekvenci fonema. Ako je par uspješno pronađen vraća se tekst odgovarajuće riječi ili fraze pozivnom programu.

Za potrebe ovog rada dovoljno je bilo proširiti bazu podataka s novim zapisima vremenskih prognoza tj. njihovim transkripcijama pa se ne izrađuje čitav SRE, ali prethodno opisane komponente su sve od velikog značaja. U nastavku će biti opisana faza pripreme podataka koja se sastoji od izrade samog rječnika, snimanja vremenskih prognoza izradom Python skripte za automatizaciju tog procesa, izrade inicijalnih transkripcija i kodiranja audio datoteka.

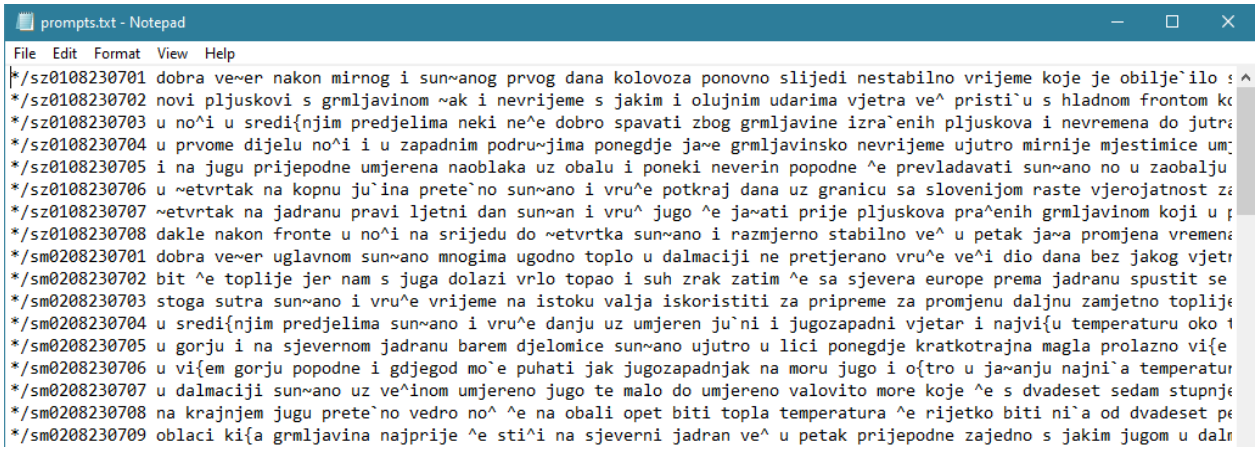


Slika 4.1. Princip rada osnovnog Speech Recognition Engine-a (SRE) [4]

4.1 Izrada fonetski balansiranog rječnika

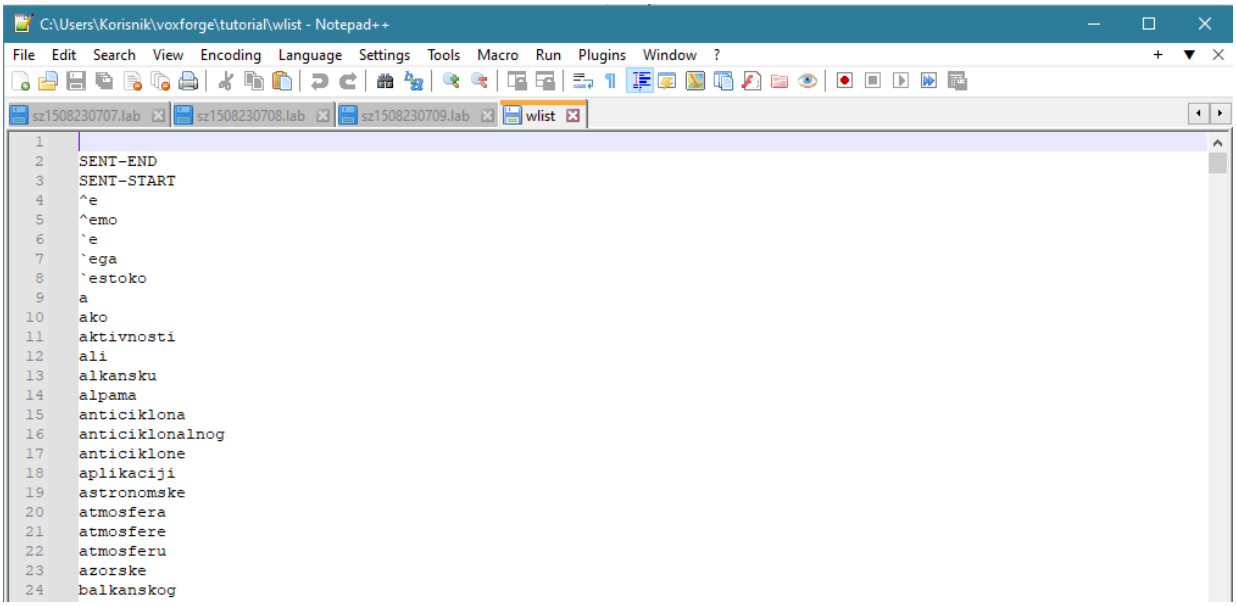
Tipično, prvi korak kod izrade rječnika izgovora predstavlja stvaranje popisa koji se sastoji od sortiranih riječi koje čine gramatiku tako da imamo jednu riječ po retku zajedno s izgovorom te riječi tj. fonemima koji ju čine. No, kako bi HTK bio u stanju izraditi audio datoteke našeg govora i njihove transkripcije u akustični model moramo imati fonetski balansirani rječnik. To znači da se rječnik mora sastojati od minimalno 30 do 40 rečenica tj. fraza koje se sastoje od 8 do 10 riječi. Ako naša gramatika ima manje riječi od definiranog minimuma ili se radi o gramatici koja nije fonetski balansirana tj. pojedini se fonemi javljaju samo jednom, potrebno je ručno dodati riječi kako bi se svaki fonem pojavio bar 3 do 5 puta u našemu rječniku. No, kako je za potrebe ovog rada korišteno gotovo 200 fraza za izradu rječnika nije bilo potrebno dodavati riječi kako bi ostvarili fonetski balansirani rječnik, ali više o korištenim podacima kasnije u radu.

Tri su osnovna koraka koja moramo izvršiti kako bi s pomoću HTK izradili rječnik, a to su izrada prompts.txt datoteke, izvođenje wlist datoteke iz dobivene prompts.txt datoteke te izrada samog rječnika dodavanjem informacija o izgovoru riječima iz wlist-a. Prompts.txt datoteka sadrži popis riječi koje su izgovorene u vremenskim prognozama. Svaki redak datoteke sastoji se od oznake jedne od snimaka vremenskih prognoza (npr. sm1008230701) te njezine ručno izrađene transkripcije u nastavku. Dakle možemo zaključiti da se prompts.txt datoteka mora sastojati od svih riječi koje čine gramatiku te po potrebi i od dodatnih riječi kako bi naš rječnik bio fonetski balansiran. Kod ručno izrađenih transkripcija važno je zamijeniti hrvatske dijakritičke znakove s njihovim zamjenskim parom kako bi ih mogli raspoznati npr. znak “š” zamjenjuje se sa “{“ dok umjesto znaka “ć” pišemo znak “^” itd.



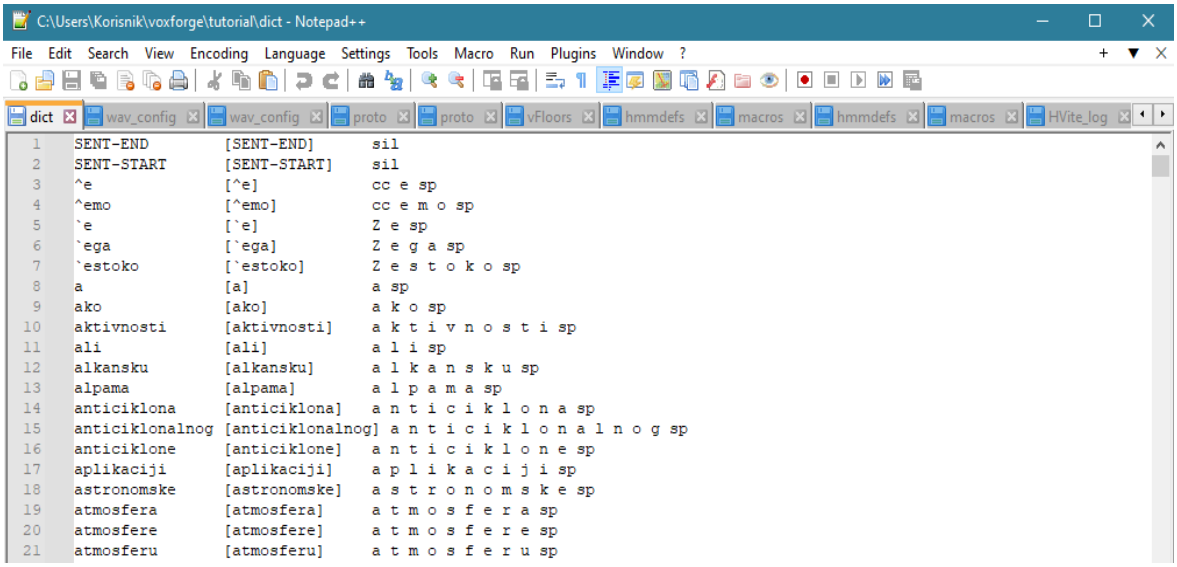
Slika 4.1.1. Isječak prompts.txt datoteke

Nakon izrade prompts.txt datoteke potrebno je ukloniti nazive datoteka iz nje te ispisati svaku riječ u zasebnom redu, a ta datoteka naziva se wlist. Wlist datoteku moguće je poput prompts.txt-a ručno izraditi, ali puno brže je to učiniti s pomoću Julia skriptu te je tako u sklopu ovoga rada korištena skripta naziva prompts2wlist.jl (skriptu je moguće preuzeti sa raw.githubusercontent.com/VoxForge/develop/master/bin/prompts2wlist.jl). Rezultat izvođenja ove skripte je wlist datoteka u prethodno opisanom formatu, a datoteci su automatski dodane dvije stavke tj. SENT-END i SENT-START koje predstavljaju interni unos HTK Toolkit-u za izradu akustičnog modela i njegovo procesiranje s pomoću Julius-a.



Slika 4.1.2. Isječak wlist datoteke

Posljednji korak kako bi konačno dobili rječnik je dodavanje informacija o izgovoru svakoj riječi koja se nalazi u wlist datoteci. To se može postići koristeći HDMan naredbu. HDMan je alat kojim se priprema rječnik izgovora iz jednog ili više izvora. Alat čita iz popisa naredbi za uređivanje koja se nalazi u nekoj skripti, a zatim kao izlaz vraća uređenu i spojenu kopiju jednog ili više rječnika. U slučaju da nemamo definirane datoteke za uređivanje HDMan spaja sve rječnike na ulazu u novi, sortirani rječnik na izlazu. Konkretna naredba koja se u sklopu ovoga rada koristi je "HDMan -A -D -T 1 -m -w wlist -n monophones1 -i -l dlog dict ../lexicon/VoxForgeDict.txt" koja daje dva izlaza, a to su dict i monophones1 datoteke. Dict predstavlja rječnik izgovora naše gramatike dok je monophones1 običan popis fonema korištenih u rječniku. Također, dlog datoteka koja se generira pomoću –i zastavice sadrži popis svih fonema i frekvenciju njihova pojavljivanja pa služi kao dobra provjera fonetske balansiranosti rječnika.

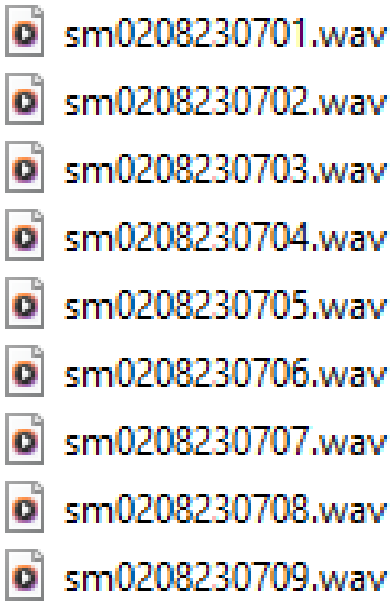


Slika 4.1.3. Dobivena dict datoteka (riječi i izgovor)

Prije prelaska na idući korak potrebno je kopirati monophones1 datoteku te ju preimenovati u monophones0 i ukloniti “sp” iz popisa monofona što će nam biti važno u jednom od koraka koji slijedi.

4.2 Snimanje vremenskih prognoza

Kao što je to već ranije spomenuto u sklopu ovoga rada snimljene su vremenske prognoze Hrvatske radiotelevizije te je cilj generirati njihove transkripcije. Snimke su dobavljene s prvog programa hrvatskog radija (Radio uživo: Prvi program :: Hrvatski radio (hrt.hr)) gdje se vremenska prognoza emitira više puta na dnevnoj bazi, a za potrebe ovog rada korištene su isključivo vremenske prognoze emitirane u 19 sati prosječnog trajanja između 150 s i 180 s. Te vremenske prognoze je zatim potrebno izrezati na manje audio datoteke koje se sastoje od nekoliko riječi ili fraza, a pritom to moraju biti smislene cjeline govora koje uključuju kratke pauze, uzdahe i ostale pojave koje su uobičajene tijekom ljudskog govora. Audio datoteke moraju biti imenovane prema nomenklaturi datoteka iz VEPRAD baze podataka tj. datoteka započinje oznakom spola govornika (sm ili sz) zatim slijedi datum emitiranja vremenske prognoze i sat tijekom kojeg je ta prognoza emitirana (ddmmggr) te na kraju se nalazi redni broj tog isječka unutar same prognoze. Prilikom snimanja i rezanja audio datoteka potrebno je obratiti pozornost na razinu zvuka te je poželjno da ona ne izlazi iz raspona -1 do 1 jer tada dolazi do nastajanja distorzije, a isto tako se preporučuje ostavljanje kratke stanke prije i nakon izgovorenih fraza kako bi zadnje izgovorene riječi bile jasne za prepoznavanje.



Slika 4.2.1. Primjer naziva audio datoteka

Kako bi se pojednostavio proces snimanja tih vremenskih prognoza trebalo je izraditi skriptu koja će automatski pokrenuti snimalicu u određenom intervalu. Skripta je izrađena s pomoću Python programskog jezika te se jako veliki dio skripte realizira s pomoću PyAutoGUI knjižnice funkcija. PyAutoGUI je knjižnica funkcija koja omogućuje Python skriptama kontrolu nad mišem i tipkovnicom računala kako bi se automatizirale interakcije s drugim aplikacijama. Samo neke od značajki ove knjižnice funkcija su mogućnost pomicanja kursora miša i klikanje unutar prozora drugih aplikacija, automatski unos s tipkovnice npr. za ispunjenje login forme, stvaranje snimke zaslona te pronalaženje sadržaja te fotografije na zaslonu, upravljanje veličinom prozora aplikacija itd.

Skripta obavlja snimanje vremenskih prognoza s pomoću Audacity audio editor-a te je prije izrade i pokretanja skripte bilo potrebno konfigurirati samu "snimalicu". Datoteke koje dobijemo snimanjem vremenskih prognoza moraju biti .wav formata s frekvencijom uzorkovanja od 16 000 Hz u mono formatu tj. snimljene jednim kanalom. Osnovna ideja ove skripte za snimanje vremenskih prognoza je da se jednom dnevno malo prije 19:00 sati pokrene web browser i pristupi prvome programu hrvatske radiotelevizije te započinje s njegovim slušanjem. Zatim se uključuje sam Audacity audio editor i stvara nova datoteka te se snima ukupno 3 minute sadržaja nakon čega se ta datoteka sprema pod nekim nazivom i ovaj bi se postupak trebao ponavljati svaka 24 sata.

Tako se korisnika oslobađa od praćenja vremenskih prognoza i može se skupiti jako velik broj istih bez gubitka vremena na taj dio rada. Nakon dodavanja potrebnih knjižnica funkcija skripti prije bilo kakve izrade funkcija bilo je potrebno definirati dvije globalne varijable koje će biti korištene u ostatku skripte, a mijenjane prema potrebama korisnika. Prva je polje start_times u kojem se definira vrijeme početka snimanja te za potrebe ovog rada koristimo isključivo jednu vrijednost (u obliku stringa) npr. “19:00” dok ako želimo snimiti više vremenskih prognoza možemo polje ispuniti s više vrijednosti. Druga globalna varijabla je recording_duration u kojoj se definira trajanje snimanja te kad ovo vrijeme istekne zaustavlja se snimanje i pohranjuje se snimljeno u obliku .wav datoteke.

Dvije su osnovne funkcije koje se koriste za implementaciju funkcionalnosti skripte, a to su get_and_update_number() te start_audacity_recording(). Kada se pohranjuje .wav datoteka nakon snimanja potrebno je svakog puta promijeniti naziv datoteke kako bi se izbjegla situacija gdje nova datoteka ima isti naziv kao i stara pa dolazi do “overwrite-anja” tj. nova se datoteka “piše” preko stare i gubi se stara. Kako bi se to izbjeglo izrađena je funkcija get_and_update_number() koja koristi datoteku number.txt u koju se upisuje cijeli broj npr. 7 te ona provjerava ako datoteka naziva “7” postoji. U slučaju da takva datoteka ne postoji nova audio datoteka će imati broj “7” kao svoj naziv tj. “7.wav”, a u slučaju da takva datoteka već postoji broj iz number.txt datoteke se inkrementira za 1 i nastaje datoteka “8.wav”.

```
def get_and_update_number():
    number_file = "number.txt"

    # read the number written in the first line of the file
    if os.path.exists(number_file):
        with open(number_file, "r") as f:   # open file using read flag
            current_number = int(f.read())
    else:
        current_number = 1 # if number file is empty set current number to default value (1)

    # increment the current_number value and re-write it inside the file
    updated_number = current_number + 1
    with open(number_file, "w") as f:
        f.write(str(updated_number))

    return updated_number
```

Slika 4.2.2. Funckija get_and_update_number()

Unutar funkcije start_audacity_recording() poziva se funkcija get_and_update_number() te je sadržana čitava logika vezana uz snimanje vremenskih prognoza. Ovdje se koristi prethodno spomenuta PyAutoGUI knjižnica funkcija kako bi mogli micati kursor miša i unositi vrijednosti

tipkovnicom bez fizičkog upravljanja računalom. Nakon poziva funkcije get_and_update_number() i postavljanja vremena čekanja između pojedinih aktivnosti (default postavljen na 5 s) započinje proces snimanja. S pomoću funkcije subprocess.Popen otvara se web browser, u ovom konkretnom slučaju je to Safari, a zatim se pomoću PyAutoGUI funkcija preko tipkovnice automatski upisuje hiperveza prvog radija Hrvatske radio televizije i pokreće stream. Nakon što stream krene ovoga se puta s pomoću subprocess funkcije pokreće Audacity te se stvara nova datoteka unutar programa. Snimanje se pokreće s pomoću pyautogui.hotkey funkcije odgovarajućom kombinacijom unosa te zaustavlja nakon definiranog vremena, a zatim se novonastala snimka sprema u .wav formatu i zatvaraju se kako Audacity tako i sam web browser.

```
# export the file

pyautogui.hotkey("shift", "command", "e")  # save the project
time.sleep(2)  # wait for the export dialog to appear
pyautogui.typewrite(number_string)
time.sleep(2)
pyautogui.press("enter")  # confirm the export
time.sleep(2)  # wait for the next dialog to appear
pyautogui.press("enter") # confirm save data
time.sleep(2)
pyautogui.hotkey("command", "q")  # close audacity
time.sleep(2)
pyautogui.click(720, 370)
#text_image_path3 = "/Users/pierobattelli/Desktop/no.png"
#locate_and_click_text(text_image_path3)

# close web browser

time.sleep(2)
pyautogui.click(180, 575)
time.sleep(2)
pyautogui.hotkey("command", "q")
```

Slika 4.2.3. Isječak start_audacity_recording() funkcije

Navedene funkcije pozivaju se unutar jedne beskonačne while petlje u kojoj se najprije računa trenutno vrijeme u sekundama, a zatim se gleda da li postoji još neko zadano vrijeme toga dana. Ako postoji u prozoru terminala ispisuje se poruka koja ukazuje na broj sekundi preostalih do snimanja, a u suprotnom se čeka do prvog vremena idućeg dana tj. konkretno u ovom radu snimanje se odvijalo u 19:00 sati i nakon snimanja se čekalo 24 sata do ponovnog pokretanja snimalice.

```
while True:

    # fetch the current time in seconds from midnight

    current_time = time.localtime(time.time())
    current_time_seconds = current_time.tm_hour * 3600 + current_time.tm_min * 60 + current_time.tm_sec

    # find the next starting time of recording

    next_start_time_seconds = None
    for start_time_seconds in start_times_seconds:
        if start_time_seconds > current_time_seconds:
            next_start_time_seconds = start_time_seconds
            break

    if next_start_time_seconds is None:

        # if there are no upcoming start times for today, wait 24 hours until the next start time

        wait_time = start_times_seconds[0] + (24 * 3600) - current_time_seconds

    else:

        # calculate the number of seconds until the next start time

        wait_time = next_start_time_seconds - current_time_seconds

    print(f"Waiting for {wait_time} seconds until the next recording...")
    time.sleep(wait_time)

    # start the recording process

    start_audacity_recording()
```

Slika 4.2.4. While petlja koja pokreće skriptu

Nakon realizacije automatske snimalice za potrebe proširenja baze snimaka vremenskih prognoza VEPRAD snimljeno je ukupno 20 vremenskih prognoza. Ukupno je bilo 9 muških govornika te 11 ženskih, a neki se govornici ponavljaju tj. nisu svi jedinstveni. U tablici u nastavku prikazana je statistika vezana uz snimke.

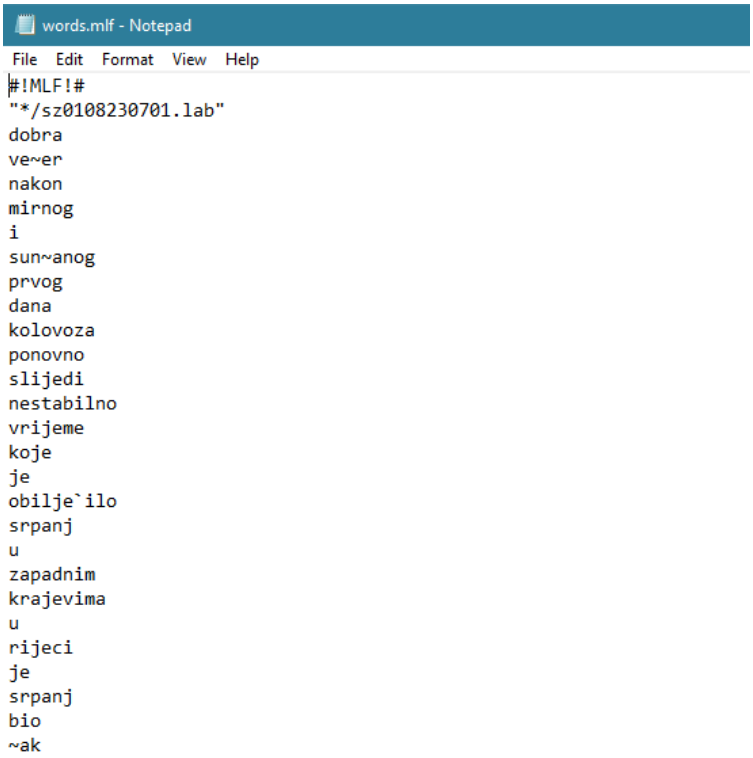
Tablica 4.2.1. Opis snimljenih podataka

broj prognoza	broj izrezanih snimki	vremensko trajanje snimki (sekunde)	vremensko trajanje snimki (sati)	ukupan broj jedinstvenih riječi
20	196	3676 s	1.0211 h	1286

Osim snimaka potrebno je bilo izraditi i ručne transkripcije na temelju izrezanih audio datoteka koju su nužne uz .wav i .lab datoteke za prepoznavanje govora, a te transkripcije bile su korištene i u sklopu prompts.txt datoteke za izradu rječnika. Transkripcije su izrađene preslušavanjem pojedinih snimaka te su formata .txt, a imenovane su jednako kao i njihov .wav par.

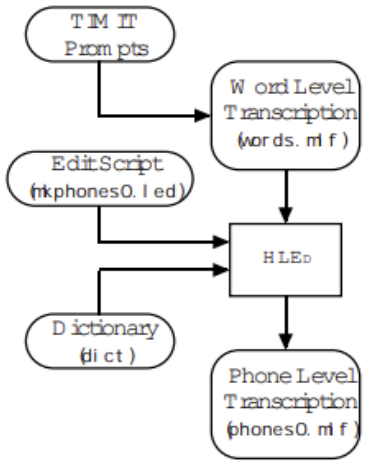
4.3 Izrada transkripcija na razini fonema i kodiranje audio podataka

Prompts.txt datoteku nije moguće procesirati direktno preko HTK Toolkit-a stoga je moguće odlučiti se za jednu od dvije opcije. Sporiji pristup je ručna izrada pojedinačnih "label" datoteka za svaku liniju prompts.txt datoteke, a brži i jednostavniji pristup koji je korišten u sklopu ovoga rada je izrada Master Label File-a tj. MLF datoteke. MLF je datoteka koja sadrži labelu za svaku liniju naše prompts.txt datoteke. Generiranje MLF datoteke radi se pomoću Julia skripte prompts2mlf.jl (skriptu je moguće preuzeti sa raw.githubusercontent.com/VoxForge/develop/master/bin/prompts2mlf.jl) koja s pomoću prompts.txt datoteke stvara words.mlf datoteku.

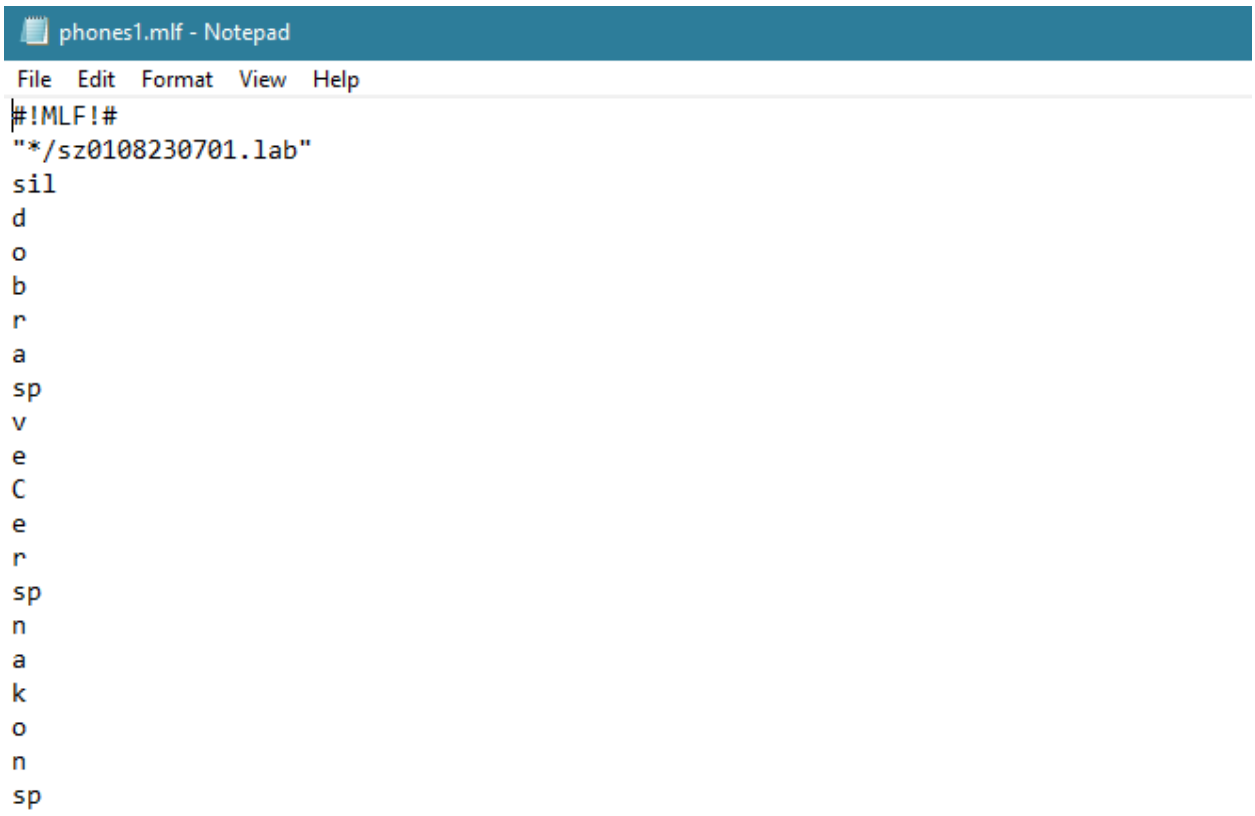


Slika 4.3.1. Isječak words.mlf datoteke

Sljedeće što je potrebno napraviti je s pomoću HLEd naredbe proširiti transkripcije na razini riječi u transkripcije na razini fonema. HLEd je jako jednostavni uređivač koji omogućuje manipulaciju label datotekama. Tipični primjeri uporabe ovog alata su kod spajanja niza labela u jednu kompozitnu labelu ili za proširenje skupa labela u skup osjetljiv na kontekst. HLEd radi tako da čita popis naredbi za uređivanje iz skripte namjenjenoj uređivanju i zatim stvara uređenu kopiju jedne ili više label datoteka. Proširenje transkripcije na razini riječi u transkripcije na razini fonema znači da se svaka riječ zamjenjuje njezinim fonemom, a rezultat se sprema u novi Master Label File na razini fonema. To se postiže tako da se pregleda svaka riječ MLF datoteke te se pretražuju fonemi koji čine tu riječ u prethodno izrađenoj dict datoteci te se izlaz tj. rezultat sprema u datoteku naziva phones0.mlf te je važno napomenuti da ta datoteka neće imati kratke stanke nakon svake grupe fonema riječi. Najprije je potrebno napraviti skriptu mkphones0.led (skriptu je moguće preuzeti s raw.githubusercontent.com/VoxForge/develop/master/tutorial/mkphones0.led), a zatim se izvršava naredba "C:>HLEd -A -D -T 1 -l * -d dict -i phones0.mlf mkphones0.led words.mlf" koja stvara phones0.mlf datoteku. Sljedeće što je potrebno napraviti je kopiju dobivene datoteke s transkripcijama na razini fonema, ali ovoga puta uključiti kratke pauze nakon svake grupe fonema riječi. Sada je potrebno napraviti skriptu mkphones1.led (skriptu moguće preuzeti sa raw.githubusercontent.com/VoxForge/develop/master/tutorial/mkphones1.led), a zatim izvršiti naredbu "C:>HLEd -A -D -T 1 -l * -d dict -i phones1.mlf mkphones1.led words.mlf" koja stvara phones1.mlf datoteku.



Slika 4.3.2. Ilustracija HLEd naredbe [10]



Slika 4.3.3. Isječak phones1.mlf datoteke koja uključuje kratke pauze sp

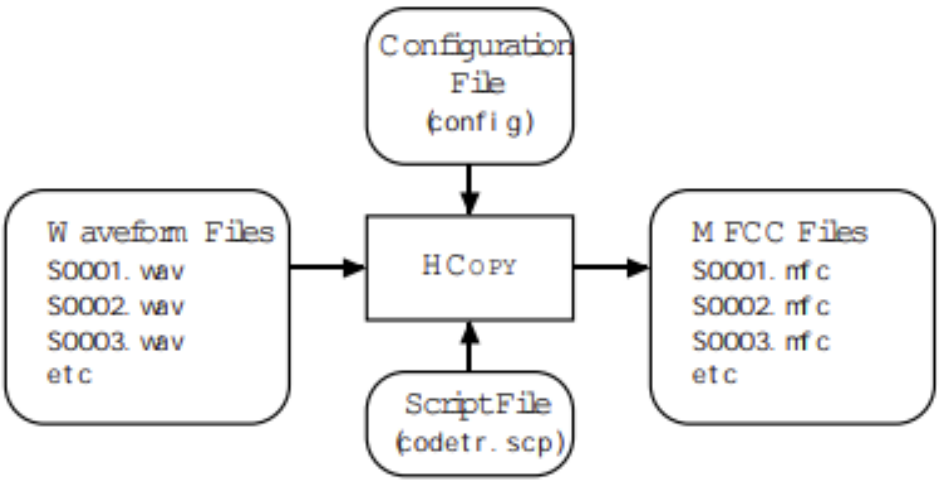
Kodiranje audio podataka posljednji je korak pripreme podataka kojeg je moguće definirati kao parametriziranje valnih oblika "sirovog" govora u sekvence vektora značajki. Jednostavnije rečeno, to znači da je HTK puno efikasniji kod procesiranja internih formata nego što je kod procesiranja .wav datoteka. Stoga se radi konverzija audio datoteka iz .wav formata u format poznat kao MFCC tj. Mel Frequency Cepstral Coefficients format što se često naziva vektorima značajki. U obradi zvuka mel-frequency cepstrum (MFC) je reprezentacija kratkoročnog spektra snage zvuka temeljena na linearnoj kosinusnoj transformaciji logaritamskog spektra snage na nelinearnoj mel skali frekvencije. Mel-frequency cepstral koeficijenti (MFCC) su koeficijenti koji zajedno čine MFC. Ti koeficijenti tipično se koriste kao značajke kod prepoznavanja govora poput sustava za automatsko prepoznavanje izgovorenih riječi ili brojeva u mikrofon. Za konverziju datoteka iz .wav formata u MFCC format koristi se HCopy alati iz HTK Toolkit-a. Kao što to i sam naziv kaže ovaj se alat koristi za manipulaciju i kopiranje datoteka govora. Konverziju je moguće napraviti na 2 načina. Moguće je izvršiti HCopy naredbu ručno za svaku audio datoteku koju želimo konvertirati ili jednostavniji pristup koji se koristi u sklopu ovoga rada je izrada datoteke koja sadrži popis svake izvorne audio datoteke i naziv MFCC datoteke u koju će se ona pretvoriti i ta se datoteka koristi kao parametar kod izvršavanja HCopy naredbe. Popis izvornih audio datoteka zajedno s nazivom MFCC datoteka koje stvaramo nalazit će se unutar HTK skripte

naziva codetrain.scp, a da bi mogli obaviti konverziju iz .wav u MFCC format potrebno je postaviti parametre konverzije. Stoga se stvara datoteka naziva wav_config u kojoj se definira format izvornih datoteka, format izlaznih datoteka, veličina prozora itd.

```
SOURCEFORMAT = WAV
TARGETKIND = MFCC_0_D
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
```

Slika 4.3.4. Konfiguracijska datoteka wav_config

Kada je konfiguracijska datoteka spremna izvršava se naredba "HCopy -A -D -T 1 -C wav_config -S codetrain.scp" te kao rezultat nastaje niz MFCC datoteka koje će biti važne za poboljšanje transkripcija u nastavku.

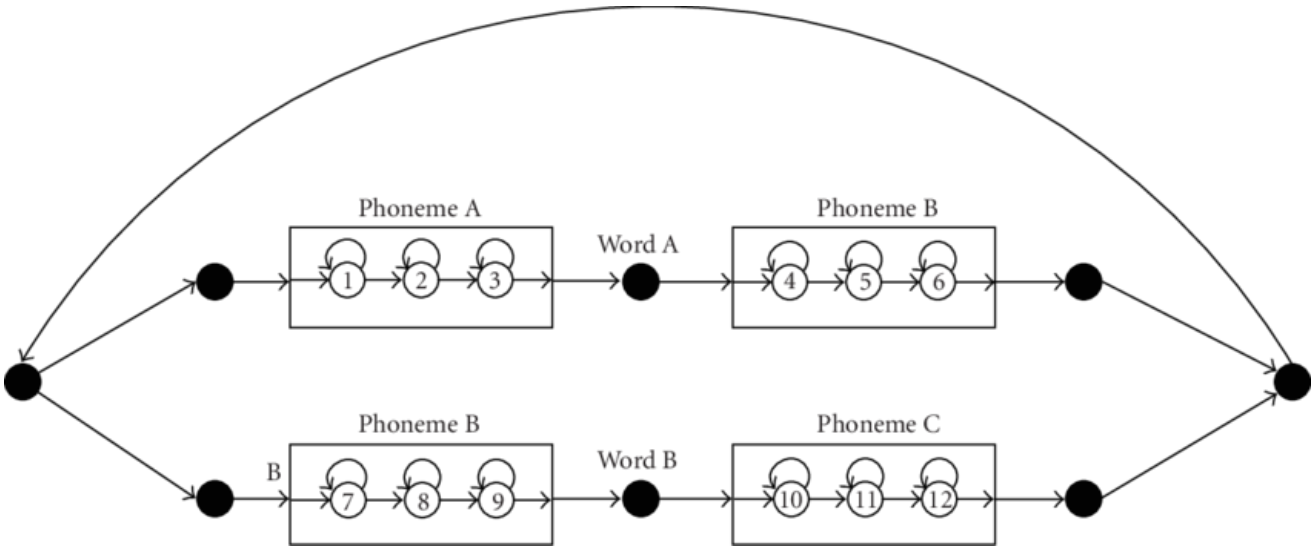


Slika 4.3.5. Ilustracija postupka kodiranja (.wav u MFCC) [10]

5. IZRADA MONOFONIH SKRIVENIH MARKOVLJEVIH MODELA (HMM)

HMM tj. skriveni Markovljevi modeli jako su moćan statistički alat za modeliranje koji se koriste u prepoznavanju govora, rukopisa i slično. Skriveni Markovljevi modeli sastoje se od nekoliko osnovnih komponenti, a to su stanja koja promatramo, skrivena stanja, prijelazna stanja (početno i stanje terminala), raspodjele vjerojatnosti prijelaza stanja i raspodjele vjerojatnosti emisije stanja. Skriven Markovljev model ima dva osnovna dijela: skriveni i promatrani. Skriveni dio sastoji se od skrivenih stanja koja se ne promatraju direktno tj. njihova se prisutnost promatra simbolima promatranja (vidljivi simboli) koje emitiraju skrivena stanja. Za primjer je moguće uzeti situaciju kada ne znamo kakvog je raspoloženja (skriveno stanje) naš prijatelj, ali moguće je promatrati njegovo ponašanje i djelovanje (vidljivi simboli) pa na temelju tih akcija koje se promatraju moguće je pogoditi u kojem se on skrivenom stanju nalazi. Tipično se za skrivena stanja uzimaju pojave koje se ne mogu direktno promatrati, a kao vidljive simbole uzimaju se oni koje je moguće promatrati čitavo vrijeme. Raspodjela vjerojatnosti prijelaza stanja, kao što i sam naziv kaže, objašnjava prijelaze između skrivenih stanja. U većini slučajeva moguće je prijeći iz bilo kojeg stanja u bilo koje drugo stanje pa isto tako vratiti se u isto stanje. Suma vjerojatnosti tranzicija mora biti vrijednosti 1. Raspodjela vjerojatnosti emisije stanja omogućuje emisije vidljivih simbola kod svake tranzicije u skriveno stanje. Suma svih vjerojatnosti emisije također iznosi 1. Važno je napomenuti kako kod prisustva skrivenih stanja postoje još dvoje stanja koja nisu direktno vezana uz model, ali se koriste za izračune, a to su početno stanje i stanje "terminala".

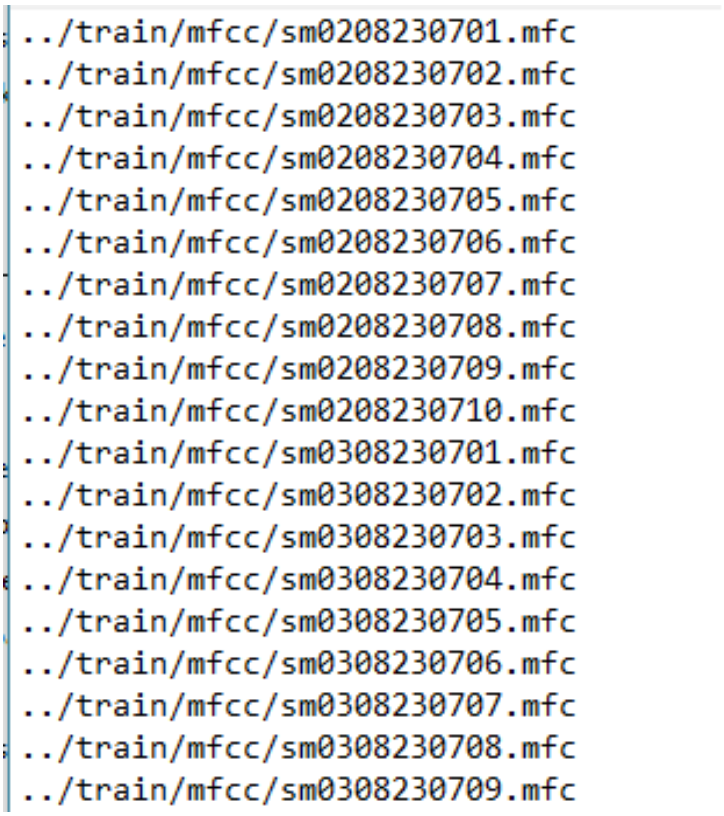
Skriveni Markovljevi modeli jako se često koriste za potrebe prepoznavanja i sinteze govora jer su jako jednostavni za implementaciju te se bez većih poteškoća može automatizirati njihovo treniranje. Omogućuju promatranje signala govora u kratkom razdoblju npr. 5 ms kao stacionarni signal ili kratkotrajni stacionarni signal pa je tako moguće govor aproksimirati kao stacionarni proces. U nastavku rada slijedi opis izrade tzv. flat start monofona, ispravljanje modela tišine te restrukturiranje treniranih podataka.



Slika 5.1. Primjer HMM-a za prepoznavanje govora

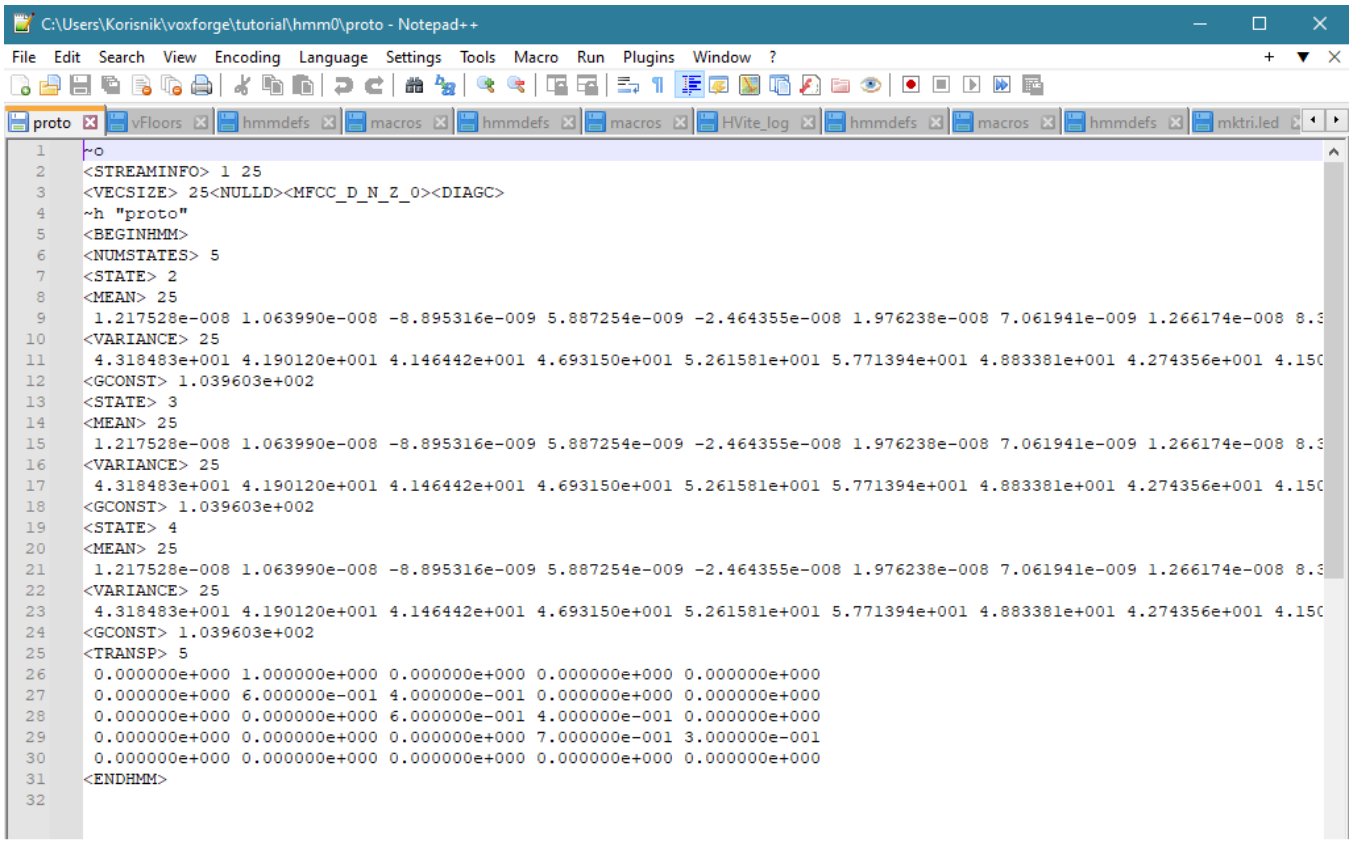
5.1 „Flat start" monofoni

Treniranje skrivenih Markovljevih modela započinje definiranjem modela prototipa pod nazivom "proto" (definiciju modela moguće preuzeti sa raw.githubusercontent.com/VoxForge/develop/master/tutorial/hmm0/proto). Cilj ovog koraka je izrada strukture modela, a parametri u ovome trenutku nisu važni. Za sustave bazirane na fonemima dobra topologija je "left-to-right" s 3 stanja bez preskoka. Potrebna nam je i konfiguracijska datoteka koja će biti naziva config, a sadržavat će parametre slične konfiguracijskoj datoteci korištenoj kod kodiranja audio datoteka (konfiguracijsku datoteku moguće preuzeti sa raw.githubusercontent.com/VoxForge/develop/master/tutorial/config). Nakon izrade modela prototipa proto i konfiguracijske datoteke config potrebno je još dati do znanja HTK Toolkit-u gdje se nalaze datoteke vektora značajki tj. MFCC datoteke izrađene u prethodnom koraku. Kako bi to realizirali potrebno je napraviti novu HTK skriptu naziva train.scp koja će sadržavati putanje do svake MFCC datoteke koju koristimo u sklopu ovoga rada. Potrebno je definirati apsolutnu putanju do MFCC datoteka.



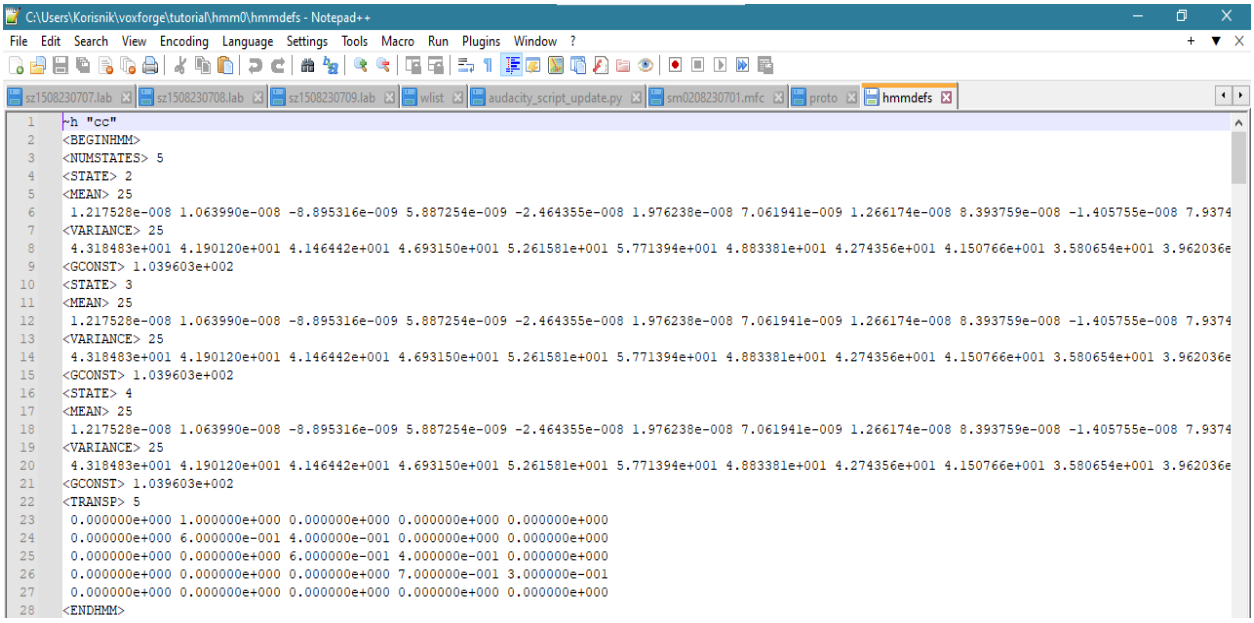
Slika 5.1.1. Isječak train.scp skripte

Nakon pripreme skripte potrebno je još napraviti novu mapu naziva hmm0 te s pomoću HCompV alata stvoriti novu verziju proto datoteke. HCompV je alat HTK Toolkit-a s pomoću kojega je moguće računati globalnu sredinu i kovarijancu skupine podataka za treniranje. Primarna uporaba ovog alata je kod inicijalizacije parametara skrivenih Markovljevih modela tako da se sve sredine i kovarijance komponenti postavljaju na vrijednosti jednake sredinama i kovarijancama globalnih podataka. Ovime je moguće realizirati prvu fazu treniranja flat start monofona gdje su u početku svim modelima dani isti parametri. Kada se treniraju skupine širokih modela iz ograničenih podataka postavljanje "floor-a" tj. donje granice je često potrebno kako bi se izbjegle loše procjene varijance zbog manjka podataka. Jedan od načina postizanja toga je definiranje macro datoteke varijance naziva varFloorN gdje N predstavlja index stream-a. Način koji je korišten u ovome radu je da se direktno iz HCompV alata generira ažurirana verzija prototipa modela sa vFloors datotekom koja predstavlja macro donje granice varijance s vrijednosti jednakim određenom udjelu globalne varijance. Naredba koja će biti korištena za to je "HCompV -A -D -T 1 -C config -f 0.01 -m -S train.scp -M hmm0 proto" te su njezin izlaz dvije datoteke, a to su proto i vFloors. Novonastala proto datoteka imat će početne srednje vrijednosti i varijance zamjenjene s globalnim vrijednostima.



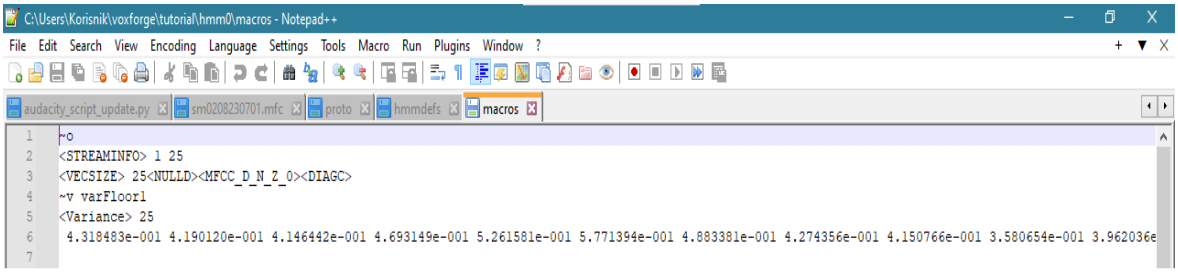
Slika 5.1.2. Prikaz proto datoteke

Nakon stvaranja proto i vFloors datoteka moguće je započeti s izradom hmmdefs datoteke, a unutar nje bit će zapisani “flat start” monofoni. Prvo što je potrebno napraviti je u hmm0 folder kopirati monophones0 datoteku iz prethodnih koraka te preimenovati ju u hmmdefs te sada je potrebno za svaki fonem unutar te datoteke napraviti sljedeće korake. Najprije se svaki fonem okružuje dvostrukim navodnicima “” te se dodaje oznaka “~h” prije fonema npr. ~h “ae”. Zatim je potrebno kopirati sadržaj iz proto datoteke koja predstavlja model prototipa, a uzima se kopija od 5. linije na dalje tj. od “<BEGINHMM>” oznake do “<ENDHMM>” oznake i taj postupak ponavljamo za svaki fonem. Potrebno je ostaviti jedan prazan red na završetku datoteke i time završava izrada hmmdefs datoteke koja sadrži “flat start” monofone.



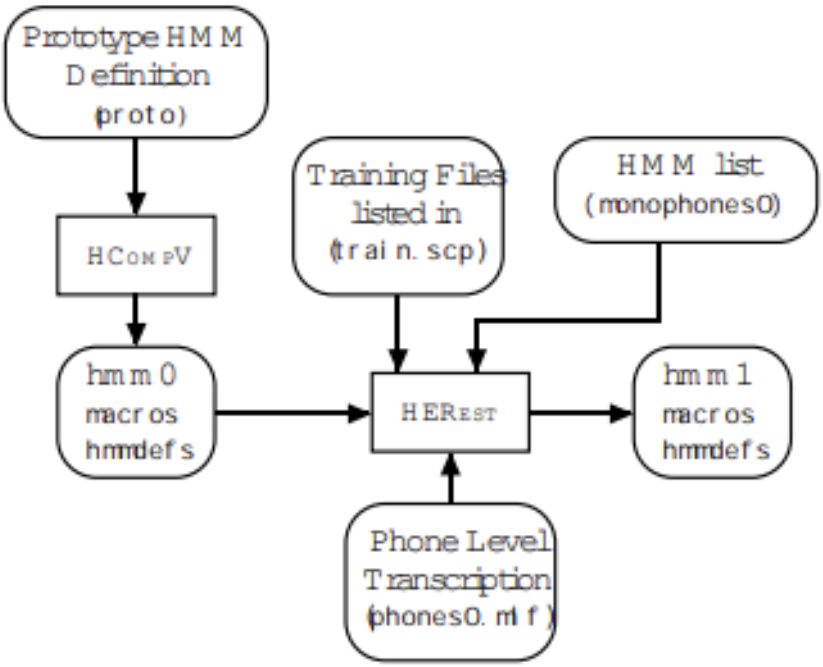
Slika 5.1.3. Prikaz definicije fonema ae u hmmdefs datoteci

Zadnji korak izrade "flat start" monofona predstavlja stvaranje "macros" datoteke. Stvara se nova datoteka naziva macros unutar hmm0 direktorija gdje se nalaze proto, vFloors i hmmdefs datoteke. Zatim se sadržaj iz vFloors kopira u macros datoteku te se prve 3 linije iz proto datoteke dodaju na vrh macros datoteke tj. dodaje se sadržaj od ~o do <DIAGC>.



Slika 5.1.4. Sadržaj macros datoteke

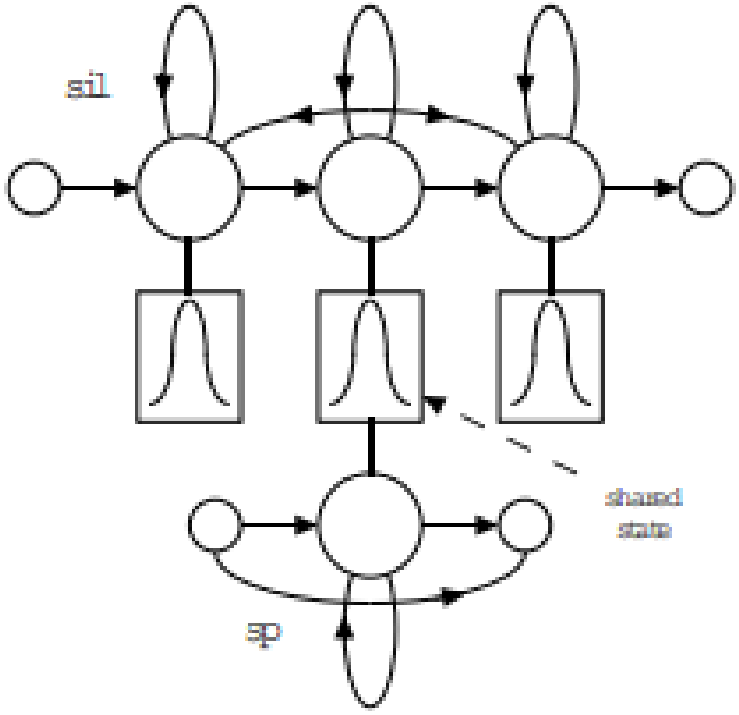
Prije ispravljanja modela tišine potrebno je napraviti ponovnu procjenu monofona. Monofone je moguće ponovno procijeniti koristeći HERest alat iz HTK Toolkit-a. HERest je program koji omogućuje jednostruku ponovnu procjenu parametara skupa skrivenih Markovljevih modela ili linearne transformacije koristeći ugrađenu trening verziju Baum-Welch algoritma. Podaci za treniranje sastoje se od jednog ili više iskaza od kojih svaki ima transkripciju u obliku standardne oznake tj. label datoteke. Za svaki iskaz trening podataka učinkovito se sintetizira kompozitni model tako da se spajaju fonemski modeli dobiveni od transkripcija. Svaki fonemski model ima jednak skup sakupljača dodijeljenih kao i u HERest-u, ali u HERest-u oni se ažuriraju simultano izvođenjem standardnog Baum-Welch prijelaza preko svakog iskaza trening podataka koristeći kompozitni model. Poput ostalih alata za ponovnu procjenu HERest također omogućuje postavljanje floor vrijednosti tj. donje granice za svaku pojedinu varijancu. Za ponovnu procjenu monofona potrebno je pripremiti 9 novih direktorija naziva hmm1 do hmm9. Ponovna estimacija vrši se koristeći MFCC datoteke navedene u train.scp skripti, a novi skup modela stvara se u hmm1 direktoriju. Naredba koju je potrebno izvršiti je "HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm0/macros -H hmm0/hmmdefs -M hmm1 monophones0", a rezultat njezinog izvođenja su ponovno procijenjena hmmdefs i macros datoteka. Od silnih parametara i zastavica vrijedi istaknuti –D kojom se ažurira donja granica varijance kod njihove ponovne procjene te –t 250.0 150.0 1000.0 što označava minimalnu statistiku korištenu kod ponovne procjene tj. ovime se kontrolira kada će se parametri ažurirati. Ako Gaussova raspodjela ima manje od 250 promatranja njezina srednja vrijednost neće biti ažurirana, ako ima manje od 150 promatranja njezina varijanca neće biti ažurirana, a ako ima manje od 1000 promatranja njezina težina smjese neće biti ažurirana. Ovaj postupak se ponavlja dva puta s istim parametrima te jedino što se mijenja je direktorij iz hmm1 u hmm2 te kasnije u hmm3, a rezultat su uvijek ponovno procijenjene hmmdefs i macros datoteke.



Slika 5.1.5. Proces stvaranja početnog skupa "flat start" monofona pomoću HERest naredbe [10]

5.2 Ispravak modela tišine te prestrojenje trening podataka

Rezultat prethodnog koraka je generiranje skrivenog Markovljevog modela s 3 stanja "left-to-right" za svaki fonem te skriveni Markovljev model za model tišine tj. silence model "sil". Silence model odnosi se na stanke dužeg trajanje koje susrećemo na kraju rečenica i slično. Sada je potrebno dodati dodatne prijelaze iz stanja 2 u stanje 4 te iz stanja 4 u stanje 2 unutar modela tišine. Ideja je napraviti robusniji model tako što se dopušta pojedinim stanjima apsorpcija raznih impulsnih šumova u trening podacima. Preskakanje unatrag omogućuje da se to dogodi bez obveze modela da prijeđe na sljedeću riječ. Sada je potrebno napraviti short pause "sp" model tišine s jednim stanjem. Short pause model odnosi se na tip jako kratke stanke koja nastaje između riječi u normalnom govoru. To bi trebao biti tzv. "tee-model" s direktnim prijelazom iz ulaznog u izlazni čvor. Short pause model mora imati stanje odašiljanja vezano na središnje stanje modela tišine što znači da je potrebno napraviti novi "sp" model unutar hmmdefs koji će koristiti središnje stanje "sil" modela i zatim je potrebno oba povezati.



Slika 5.2.1. Topologija potrebna dva modela tišine (sp i sil) [10]

Povezivanje modela tišine moguće je ostvariti tako da se kopira središnje stanje “sil” modela unutar hmmdefs datotete te se to dodaje “sp” modelu, a zatim se izvršava HTK Toolkit alat HHEd kako bi se povezao “sp” model sa “sil” modelom tako da mogu dijeliti isto središnje stanje. HHEd je uređivač baziran na skripti korišten za manipulaciju skupa definicija skrivenih Markovljevih modela. Njegova osnovna operacija je učitati skup skrivenih Markovljevih modela, primijeniti niz operacija uređivanja te kao izlaz dati transformirani skup. Primarno se HHEd koristi za primjenu vezanja duž odabrane parametre skrivenih Markovljevih modela. Također ima primjenu kod kloniranja skrivenih Markovljevih modela, nagomilavanja stanja te uređivanja struktura. Brojne HHEd komande rade nad skupovima sličnih predmeta odabranih iz skupa trenutno učitanih skrivenih Markovljevih modela. Na primjer, moguće je definirati skup konačnih stanja od svih modela samoglasnika ili vektore srednjih vrijednosti svih komponenti mješavine unutar X modela itd.

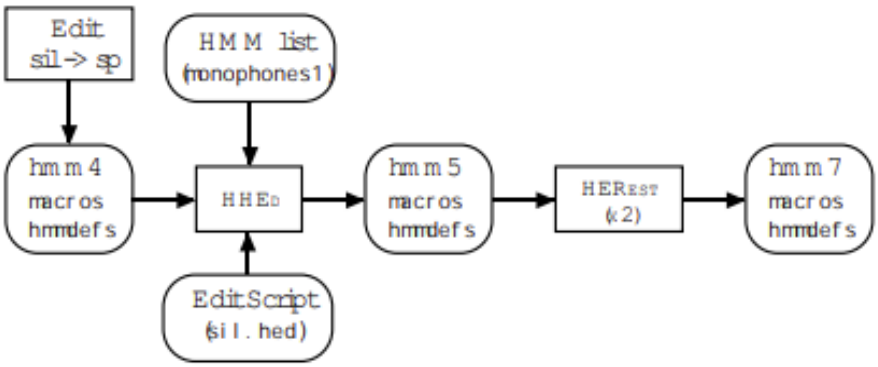
Prvi korak povezivanja modela tišine je kopiranje sadržaja iz direktorija hmm3 u direktorij hmm4 te zatim koristeći neki od uređivača teksta kao npr. Notepad++ potrebno je stvoriti novi "sp" model unutar hmmdefs datoteke u direktoriju hmm4. Najprije je potrebno kopirati i zalijepiti "sil" model te ga preimenovati u "sp". Ovdje je važno napomenuti da se ne briše stari "sil" model jer će on biti potreban u nastavku stoga se radi njegova kopija. Zatim se brišu stanje 2 i stanje 4 iz novog sp model tj. zadržava se isključivo središnje stanje starog "sil" modela u novom "sp" modelu. Sljedeće je potrebno promijeniti vrijednosti pojedinih varijabli odnosno postavlja se <NUMSTATES> na vrijednost 3, <STATE> na vrijednost 2, a <TRANSP> na vrijednost 3 te matrica koja je dio <TRANSP> pretvara se u polje dimenzija 3x3 koje se sastoji od vrijednost: [0.0 1.0 0.0, 0.0 0.9 0.1, 0.0 0.0 0.0].

```
~h "sp"
<BEGINHMM>
<NUMSTATES> 3
<STATE> 2
<MEAN> 25
 -3.829519e+000 -9.587511e-001 -7.366201e-001 4.577263e+000 4.843282e+000 4.855254e+000 5.478207e+000 2.986999e+000 1.110230e+
<VARIANCE> 25
 1.702318e+001 1.727787e+001 1.748722e+001 2.253789e+001 2.449597e+001 2.941601e+001 3.407864e+001 2.381423e+001 2.098525e+001
<GCONST> 8.941380e+001
<TRANSP> 3
0.0 1.0 0.0
 0.0 0.9 0.1
 0.0 0.0 0.0
<ENDHMM>
```

Slika 5.2.2. Prikaz "sp" modela

Sada je potrebno pokrenuti uređivač skrivenih Markovljevih modela HHEd kako bi se dodale potrebne dodatne tranzicije i time spojilo tj. povezalo sp stanje sa središtem sil stanja. Povezivanje znači da jedan ili više skrivenih Markovljevih modela dijele skup parametara. HHEd alat radi na način sličan HLEd-u odnosno primjenjuje skup naredbi iz skripte kako bi mogao modificirati skup skrivenih Markovljevih modela. Prije korištenja HHEd alata potrebno je stvoriti skriptu naziva sil.hed čiji je sadržaj moguće preuzeti s raw.githubusercontent.com/VoxForge/develop/master/tutorial/sil.hed. Skripta se sastoji od AT naredbi koje dodaju tranzicije danim tranzicijskim matricama, a konačna TI naredba stvara povezano stanje naziva silst. Parametri tog povezanog stanja pohranjeni su u hmmdefs datoteci unutar svakog modela tišine dok se originalni parametri stanja zamjenjuju imenom ove makronaredbe. Na makronaredbe moguće je gledati kao mehanizam kojim HTK implementira dijeljenje parametara. Važno je napomenuti da je popis fonema koji se ovdje koristi promijenjena zato što je originalni popis monophones0 proširen s novim "sp" modelom te se ta nova datoteka

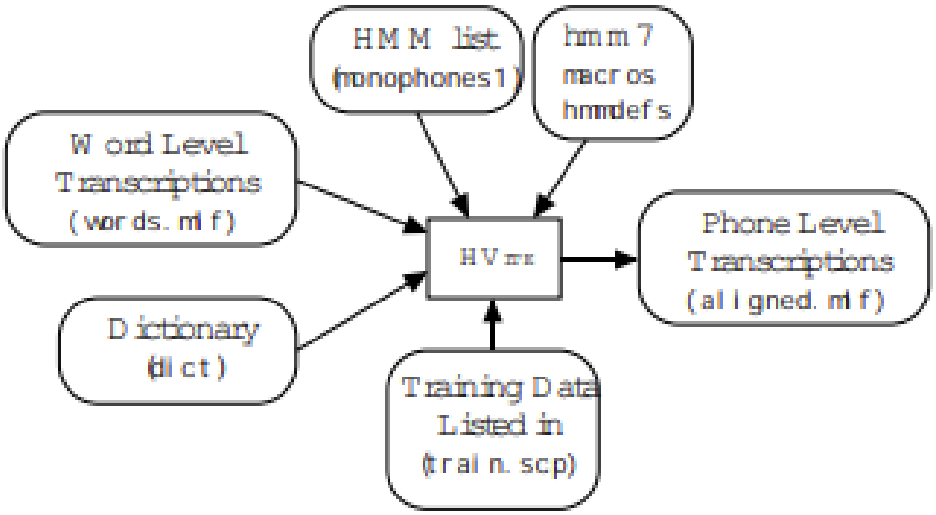
naziva monophones1 koristi kao dio HHEd komande koja će se izvršiti. Naredba koju je potrebno izvršiti je "HHEd -A -D -T 1 -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1" čime nastaju dvije nove (ponovno procijenjene) datoteke hmmdefs i macros u direktoriju hmm5, a nakon uspješnog izvođenja HHEd naredbe potrebno je još dva puta primijeniti HERest alat koristeći iste parametre kao u prošlom potpoglavlju, ali ovoga puta se koriste transkripcije na razini fonema sa "sp" modelom između riječi. Time se stvaraju ažurirane hmmdefs i macros datoteke u direktorijima hmm6 i hmm7.



Slika 5.2.3. Ilustracija postupka ispravljanja modela tišine [10]

Rječnik sadržava višestruke izgovore pojedinih riječi osobito funkcijskih riječi stoga se do sada izrađen fonemski model može koristiti za "prestrojenje" trening podataka i stvaranje novih transkripcija. Ovaj postupak sličan je HLEd operaciji mapiranja riječi u foneme iz koraka izrade početnih transkripcija, no u ovome slučaju HVite komanda koja se koristi može uzeti u obzir sve moguće izgovore svake riječi i zatim kao izlaz dati izgovor koji najbolje odgovara akustičnim podacima. HVite je Viterbi prepoznavač riječi opće namjene. Uspoređuje govornu datoteku s mrežom skrivenih Markovljevih modela i izbacuje transkripciju za svaku. Prilikom izvođenja N-best prepoznavanja na razini riječi može se proizvesti rešetka koja sadrži više hipoteza. Može se čitati rešetka na razini riječi ili label datoteka, a zatim se to proširuje koristeći dani rječnik kako bi se stvorila mreža bazirana na modelu te ovo omogućuje proizvoljne mreže riječi s konačnim

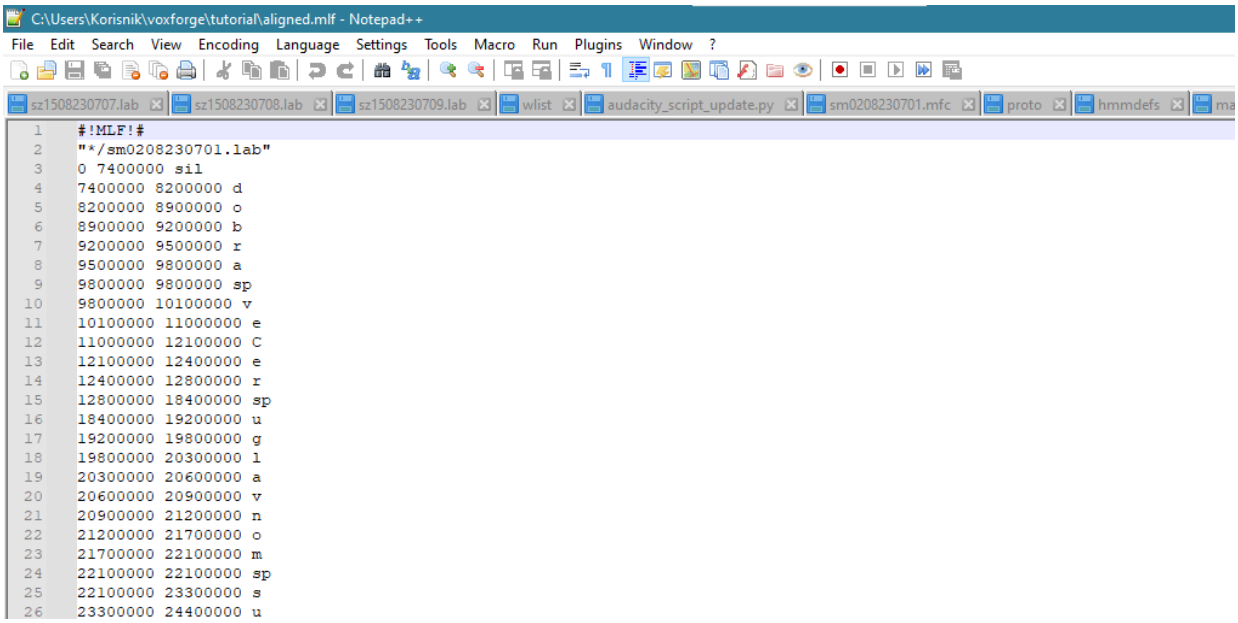
stanjem i jednostavno prisilno poravnavanje koje treba specificirati. HVite također podržava dijeljene parametre i na odgovarajući način izračunava izlazne vjerojatnosti. Naredba koju je potrebno koristiti je "HVite -A -D -T 1 -l * -o SWT -b SENT-END -C config -H hmm7/macros -H hmm7/hmmdefs -i aligned.mlf -m -t 250.0 150.0 1000.0 -y lab -a -I words.mlf -S train.scp dict monophones1> HVite_log". Ova naredba koristi skrivene Markovljeve modele pohranjene unutar hmm7 direktorija za transformaciju transkripcije na razini ulazne riječi words.mlf na novu fonemsku razinu transkripcije aligned.mlf koristeći izgovore pohranjene u rječniku dict.



Slika 5.2.4. Ilustracija postupka dobivanja aligned.mlf datoteke [10]

Što se korištenih zastavica i parametara naredbe tiče važno je napomenuti kako je –b parametar korišten za unos modela tišine na početak i kraj svakog iskaza, a specifično je riječ o silence sil modelu tj. dužim stankama. Parametar –t postavlja razinu rezanja na 250.0, a parametar –o koristi se za suzbijanje ispisa rezultata, naziva riječi i vremenskih granica u izlaznom MLF-u tj. Master Label File-u. Važno je napomenuti kako se s pomoću –l zastavice stvaraju label datoteke za svaku MLF datoteku iz aligned.mlf datoteke, a te label datoteke ključni su dio procesa proširenja baze govornih snimaka. U slučaju da zastavica ne generira label datoteke kao izlaz moguće ih je ručno izraditi prateći aligned.mlf datoteku, ali ih je potrebno imati u oba slučaja. Nakon izvođenja dane HVite naredbe potrebno je jako pažljivo pregledati izlaz naredbe jer uočavanje grešaka u

ovome trenutku puno će pomoći u posljednjim koracima rada, a izlaz se sprema u definiranu datoteku naziva hvite_log. Ta datoteka sadrži zapis svih postavljenih parametara nakon izvršavanja HVite naredbe te prikaz čitanja i manipuliranja svih skrivenih Markovljevih modela kako bi se mogli uočiti potencijalne pogreške i ispraviti ih na vrijeme.



Slika 5.2.5. Isječak aligned.mlf datoteke (s vremenskim oznakama)

Nakon uspješnog "prestrojenja" trening podataka preostalo je još dva puta primijeniti HERest naredbu kao što je to učinjeno u prethodnom koraku kako bi se ponovno procijenio skup parametara skrivenih Markovljevih modela. Time najprije nastaju ažurirane verzije hmmdefs i macros datoteka u direktoriju hmm8, a zatim i u direktoriju hmm9. Važno je napomenuti kako su modeli monofona dobiveni u hmm9 direktoriju već dovoljno dobri za prepoznavanje govora koristeći Julius, ali kako bi se značajno poboljšala preciznost prepoznavanja govora potrebno je stvoriti trifone povezanih stanja što je ujedno i posljednji korak ovog rada.

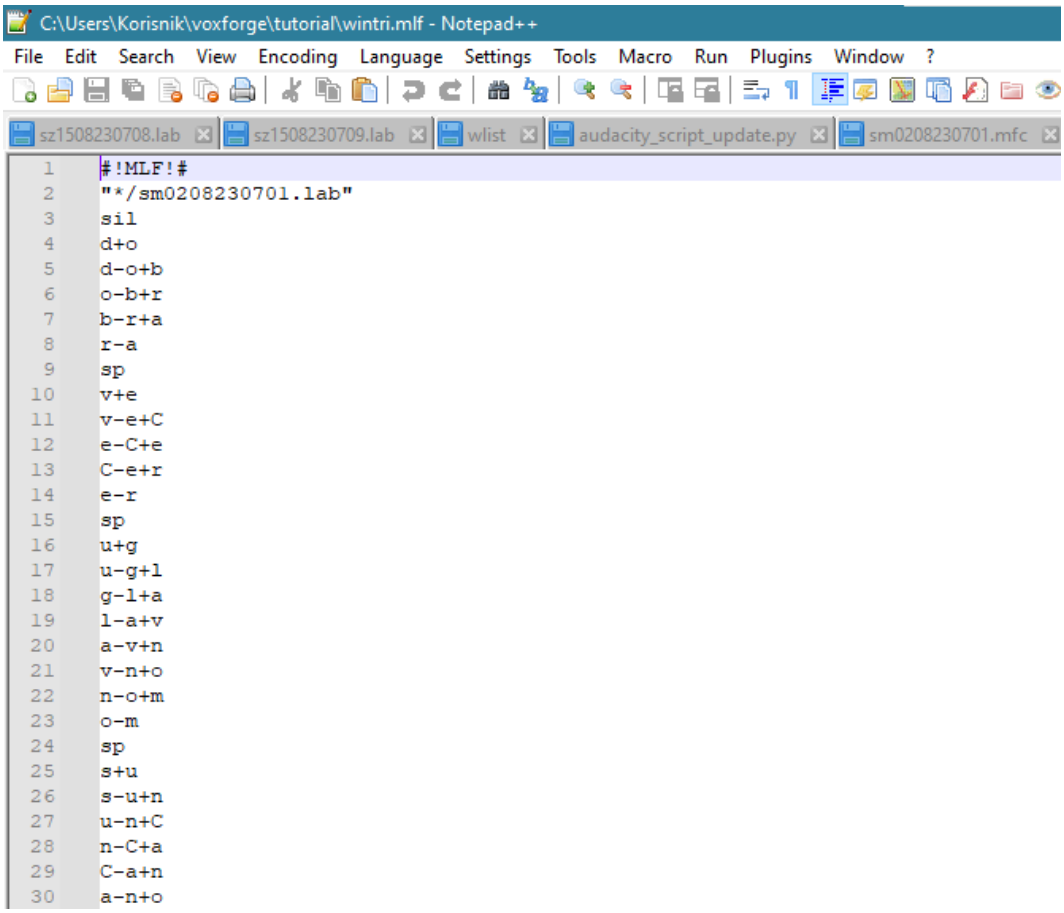
6. IZRADA MODELA TRIFONA POVEZANIH STANJA

Za dani skup monofona skrivenih Markovljevih modela konačni je korak kod izrade modela stvaranje trifona skrivenih Markovljevih modela ovisnih o kontekstu (context-dependent). Trifoni su oblika "L-X+R" stoga je iz monofona potrebno deklarirati "L" tj. lijevi fonem koji prethodi "X" fonemu, a nakon njega dolazi "R" tj. desni fonem. Na primjer, iz fonema ae, z i t pretvorbom se stvara trifon oblika ae-z+t. Razlog zbog kojega se koriste trifoni su njihova puno veća preciznost kod prepoznavanja govora jer se sada osim samih fonema uzima u obzir i njihov kontekst unutar sekvence riječi tj. rečenice, a samim time se smanjuje i vjerojatnost pogreške da će dva slična zvuka biti međusobno zamijenjena. Na to se može gledati kao na povećanje preciznosti Google pretrage tako što umjesto pretrage jednom riječju npr. "vrijeme" upišu se 3 riječi u tražilicu npr. "vremenska prognoza danas" što će dati puno preciznije rezultate zbog kontekstualne informacije i detaljnijeg unosa.

Prije rada s trifonima važno je shvatiti struktura skrivenih Markovljevih modela. Svaki skriveni Markovljev model sastoji se od puno stanja, a ta stanja mogu biti dijeljenja na isti način kao što sada sp i sil fonemi dijele središnje stanje. Kada se radi s monofonima nema smisla dijeliti stanja jer su oni međusobno jako različiti. Cijeli smisao monofona je zasebno modeliranje različitih zvukova kako bi ih SRE mogao prepoznati. No, kada se krene u rad s trifonima od kojih svaki ima vlastitu definiciju skrivenog Markovljevog modela dolazi do više instanci trifona koji imaju dovoljno slična stanja pa se ti podaci mogu dijeliti među njima. Taj proces dijeljenja je ono što se čitavo ovo vrijeme naziva povezivanjem. Stoga se da zaključiti da kod ponovne procjene novih povezanih parametara podaci originalnih nepovezanih parametara se skupljaju što u konačnici daje bolju procjenu. Najjednostavnije rečeno, ne postoji dovoljno podataka govora kako bi se modelirala svaka moguća kombinacija trifona koje sačinjavaju postojeći skup trening podataka pa se dijeli dio podataka između sličnih trifona kako bi se poboljšala preciznost.

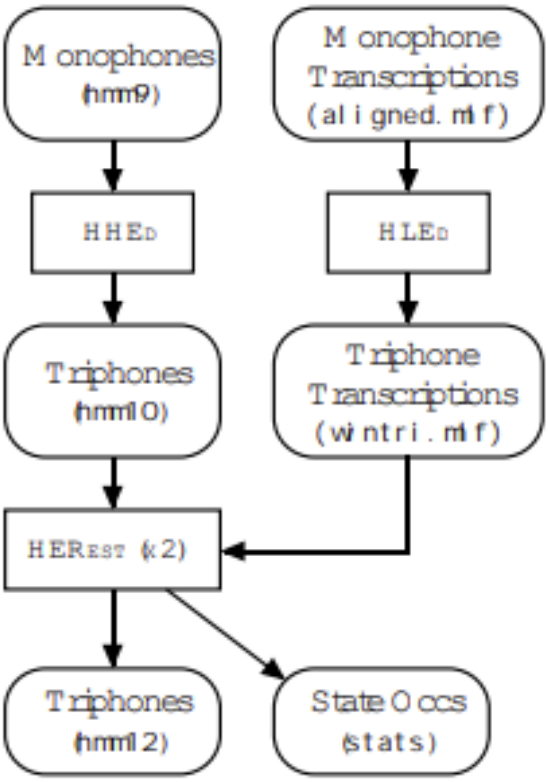
6.1 Izrada trifona

Trifone koji su ovisni o kontekstu moguće je stvoriti kloniranjem monofona, a zatim se te iste monofone ponovno procjenjuje koristeći transkripcije trifona. Najprije je potrebno napraviti transkripcije trifona koristeći HLEd alat iz HTK Toolkit-a i mktri.led skriptu za uređivanje. Skriptu je moguće preuzeti s raw.githubusercontent.com/VoxForge/develop/master/tutorial/mktri.led. Zatim se pokreće naredba "HLEd -A -D -T 1 -n triphones1 -l * -i wintri.mlf mktri.led aligned.mlf" koja kao izlaz stvara dvije datoteke, a to su wintri.mlf te triphones1. Ovom HLEd naredbom se pretvaraju transkripcije monofona iz aligned.mlf datoteke u ekvivalentni skup transkripcija trifona u wintri.mlf datoteci. Također, popis trifona zapisan je u novonastaloj datoteci triphones1.



Slika 6.1.1. Isječak wintri.mlf datoteke

Nakon stvaranja wintri.mlf i triphones1 datoteke potrebno je klonirati modele za što se koristi uređivač skrivenih Markovljevih modela HHEd. Potrebno je izvršiti Julia skriptu mktrihed.jl (moguće preuzeti s raw.githubusercontent.com/VoxForge/develop/master/bin/mktrihed.jl) koja stvara mktri.hed datoteku. Mktri.hed je skripta uređivanja koja sadrži naredbu za kloniranje CL nakon koje slijede TI naredbe za povezivanje svih tranzicijskih matrica u svakom skupu trifona. Kod izvođenja HHEd naredbe pojavit će se brojna upozorenja zbog pokušaja spajanja tranzicijskih matrica za sil i sp modele, a kako ni jedan od ta dva modela nije ovisan o kontekstu ne postoje matrice koje bi se mogle povezati. Naredba za kloniranje CL uzima kao argument naziv datoteke koja sadrži popis trifona i bifona (datoteka triphones1). Za svaki model oblika a-b+c iz popisa traži monofon b i stvara njegovu kopiju, a zatim svaka TI naredba kao svoj argument uzima ime makronaredbe i popis komponenti skrivenih Markovljevih modela. Prije izvršavanja HHEd naredbe potrebno je još napraviti 3 direktorija naziva hmm10 do hmm12. Naredba koja se izvršava za kloniranje, a zatim povezivanje skrivenih Markovljevih modela je 'HHEd -A -D -T 1 -H hmm9/macros -H hmm9/hmmdefs -M hmm10 mktri.hed monophones1' što stvara ažurirane verzije hmmdefs i macros datoteka u direktoriju hmm10. Kao i svaki puta do sada, nakon stvaranja kopija modela ovisnih o kontekstu moguće je ponovno procijeniti novi skup trifona koristeći HERest alat iz HTK Toolkit-a. Naredba i korišteni argumenti su uvijek isti samo što se sada umjesto popisa monofona koristi popis trifona i transkripcije trifona zamjenjuju transkripcije monofona. Za konačno izvođenje HERest naredbe dodaje se još i –s zastavica kako bi se stvorila datoteka stats koja osim zapisa srednjih vrijednosti i varijanca omogućuje izračun vjerojatnosti za gomile stanja i potrebna je tijekom procesa grupiranja stanja. Rade se dvije ponovne procjene s pomoću naredbe 'HERest -A -D -T 1 -C config -I wintri.mlf -t 250.0 150.0 3000.0 -s stats -S train.scp -H hmm11/macros -H hmm11/hmmdefs -M hmm12 triphones1', a rezultantni model bit će pohranjen u hmm12 direktoriju.



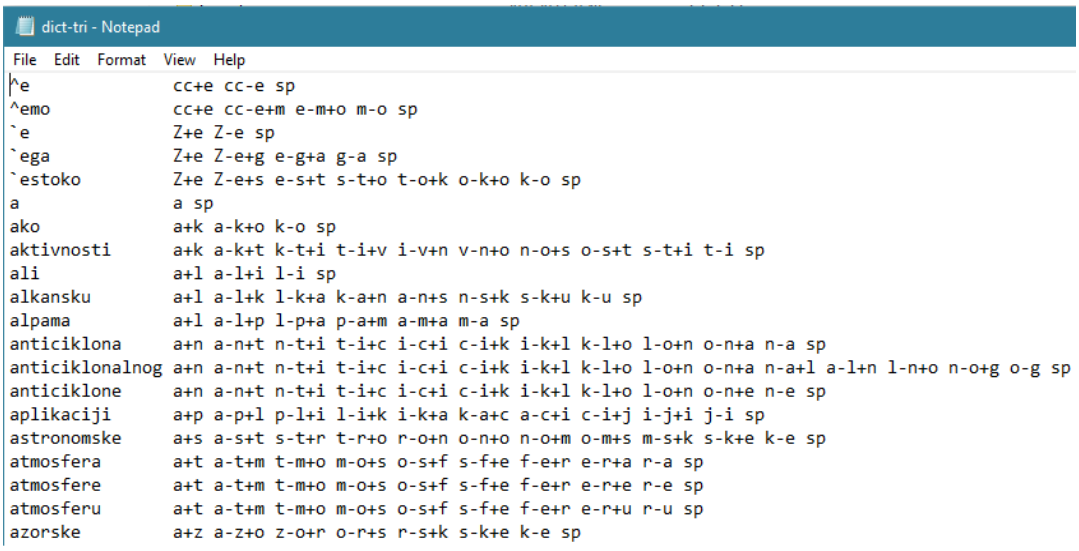
Slika 6.1.2. Ilustracija postupka stvaranja trifona [10]

6.2 Povezivanje stanja trifona

Rezultat prethodne faze je skup skrivenih Markovljevih modela trifona sa svim trifonima u skupu fonema koji dijele istu tranzicijsku matricu. Kod procjene tih modela brojne varijance u izlaznoj distribuciji bit će ograničene jer neće biti dovoljno podataka povezano s velikim brojem stanja tj. prethodno izrađeni akustični model baziran na trifonima ne može djelovati nad trifonima koji nisu trenirani.. Stoga je posljednji korak u procesu izrade modela povezivanje stanja unutar skupa trifona kako bi se dijelili podaci i samim time stvorile robusnije procjene parametara. U prethodnom koraku izrade trifona TI naredba korištena je za povezivanje članova skupa tranzicijskih matrica, ali odabir stanja koje će biti povezana zahtjeva malo više razmišljanja jer performanse prepoznavača uvelike ovise o preciznosti izlaznih distribucija stanja po pitanju ‘dohvaćanja’ statistike govornih podataka.

Rješenje ovog problema nalazi se u stablima odluke. Nagomilavanje stabala odluke omogućuje sintezu prethodno neviđenih tj. 'netreniranih' trifona tako da se koristi fonetsko stablo odluke gdje su modeli organizirani u obliku stabla, a parametri koji se prosljeđuju tom stablu nazivaju se pitanjima. Dekoder u ovom slučaju postavlja pitanje vezano uz kontekst fonema, a zatim odlučuje koji model je potrebno koristiti. Fonetsko stablo je binarno stablo u kojemu je da/ne fonetsko pitanje vezano uz svaki čvor, a ta pitanja su odabrana tako da se maksimizira vjerojatnost trening podataka s obzir na konačni skup povezanih stanja. Stabala su definirana s pomoću naredbe TB, a sva moguća fonetska pitanja učitavaju se u HHED alat koristeći QS naredbe. Svako pitanje nastoji definirati je li lijevi ili desni kontekst prisutan u imenovanom skupu, a kontekst je u tom slučaju definiran logičnim imenom.

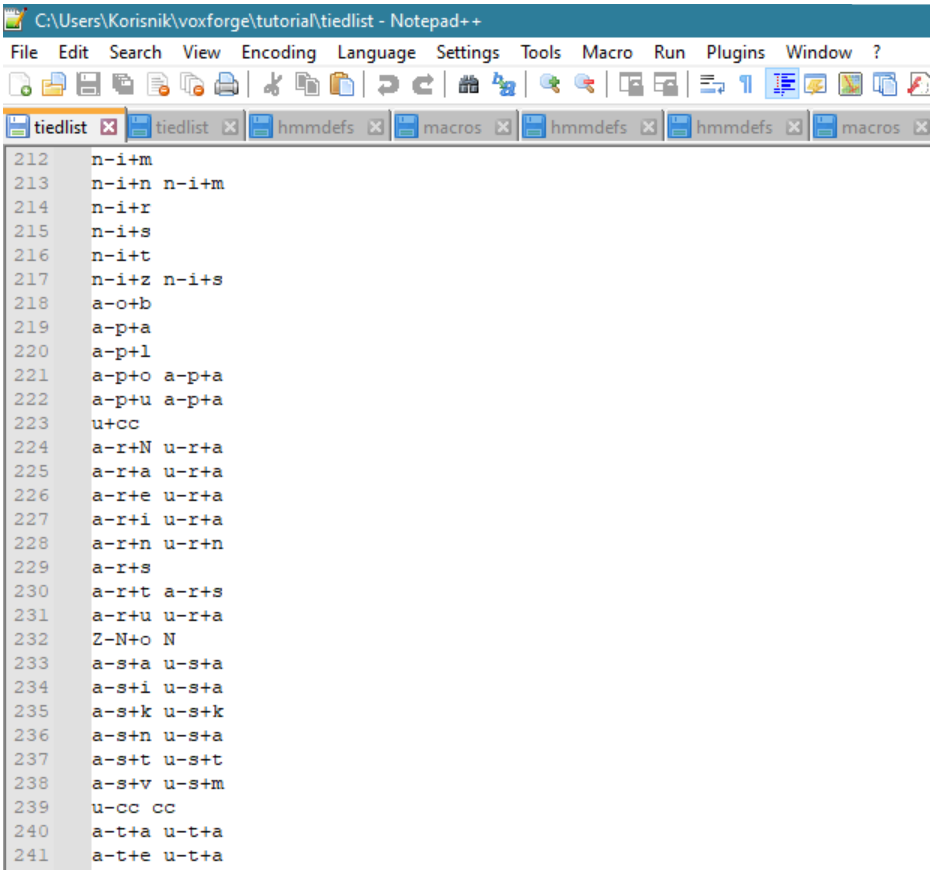
Prvi korak kod povezivanja stanja trifona je izrada skripte maketriphones.ded koju je moguće preuzeti sa raw.githubusercontent.com/VoxForge/develop/master/tutorial/maketriphones.ded. Ova skripta daje do znanja HDMan alatu da kao izlaz mora generirati trifone. Razlika u odnosu na izradu monofona je što sada izvršavamo HDMan naredbu koristeći čitav leksikon, a ne samo istrenirani rječnik. Konkretno se koristi naredba 'HDMan -A -D -T 1 -b sp -n fulllist0 -g maketriphones.ded -l flog dict-tri ../lexicon/VoxForgeDict.txt' koja kao rezultat generira dvije datoteke, a to su dict-tri i fulllist0. Dict-tri se može smatrati ažuriranom verzijom dict datoteke samo što su sada izgovori svih riječi zapisani u obliku trifona, a fulllist0 datoteka sadrži zapis svih postojećih trifona. Prije prelaska na sljedeći korak potrebno je još dodati sadržaj monophones0 datoteke na početak fulllist0 datoteke i rezultat spremiti u novu fulllist datoteku. To je moguće automatizirati s pomoću Julia skripte fixfulllist.jl koju je moguće preuzeti s raw.githubusercontent.com/VoxForge/develop/master/bin/fixfulllist.jl.



Slika 6.2.1. Isječak dict-tri datoteke

Sljedeće je potrebno izraditi novu HTK skriptu naziva tree.hed. Radi se o skripti uređivanja koja sadrži naredbe vezane uz određivanje konteksta koje je potrebno proučiti za potencijalno nagomilavanje tj. clustering. Ova skripta je u stanju isključivo generirati TB naredbe, a pitanja tj. QS korisnik i dalje mora sam definirati. No, postoji lista primjera pitanja koja je primjerena za neke zadatke. Skripta je jako velika i opširna, a moguće ju je preuzeti s raw.githubusercontent.com/VoxForge/develop/master/tutorial/tree1.hed. Neke od naredbi sadržane u skripti su QS i TB što već znamo da se odnose na pitanja i definicije stabala, ST koja sprema stabala odluke u datoteku, AU za sintetizaciju prethodno neviđenih trifona itd.

Sada je potrebno preuzeti mcklscript.jl Julia skriptu kako bi se gomile stanja dodale na već stvorenu tree.hed datoteku. Julia skriptu mcklscript.jl moguće je preuzeti s raw.githubusercontent.com/VoxForge/develop/master/bin/mkclscript.jl i nakon njezina izvršavanja dobije se ažurirana verzija tree.hed datoteke. Zadnje što preostaje učiniti je napraviti 3 nova direktorija od hmm13 do hmm15 te kao i ranije najprije s HHEd naredbom nastaju ažurirane verzije hmmdefs i macros datoteke unutar hmm13 direktorija, a nastaje i nova datoteka tiedlist koja sadrži listu povezanih trifona, a nakon HHEd naredbe potrebno je još dva puta izvršiti HERest naredbu kako bi se izvršile ponovne procjene te su konačne hmmdefs i macros datoteke sada pohranjene u hmm15 direktoriju.



Slika 6.2.2. Isječak tiedlist datoteke

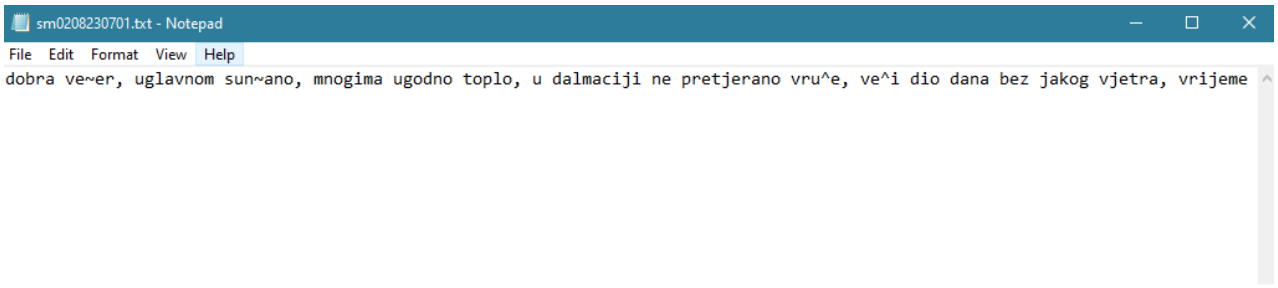
7. REZULTATI

Rezultat prethodno opisanih koraka su Python skripta za automatizaciju snimanja vremenskih prognoza tj.snimalica i tri osnovne vrste datoteka kojima će biti proširena baza govornih snimaka VEPRAD. Snimalica je jako koristan alat jer će se ona moći koristiti u budućnosti za prikupljanje jako velikog broja snimaka vremenskih prognoza i tako korisnika koji radi na proširenju baze VEPRAD osloboditi od tog zadatka pa tako jedino što on mora napraviti je modificirati skriptu prema svojim potrebama i pokrenuti ju. Ako za to postoji potreba skripta može višestruko puta snimati tijekom jednog dana što je jako velika prednost u odnosu na ručno snimanje gdje bi korisnik svakoga puta morao biti prisutan za računalom. U budućnosti bi korisno bilo i automatizirati postupak rezanja snimaka na manje audio datoteke koje sadrže nekoliko riječi ili fraza pa bi zapravo čitav postupak prikupljanja i pripreme audio datoteka bio automatiziran, no kako rezane snimke moraju biti smislene ovaj postupak vrlo vjerojatno zahtjeva implementaciju kompleksnih algoritama umjetne inteligencije što povećava složenost zadatka. Što se dobivenih datoteka tiče, rezultat izrade akustičnog modela zaključno s povezivanjem stanja trifona su .txt, .wav i .lab datoteke koji će biti iskorištene za proširenje baze govornih snimaka VEPRAD. Kao što je to već napomenuto najprije su prikupljene .wav datoteke s pomoću Python skripte tj. snimalice. Ukupno je za potrebe ovog rada korišteno 20 snimaka vremenskih prognoza, a omjer između muških i ženskih govornika je gotovo 50/50 tj. 9 snimaka sadrži muškog govornika dok preostalih 11 sadrži ženskog. Kao što je to već više puta spomenuto, nakon prikupljanja snimaka bilo je potrebno razrezati ih na manje smislene snimke. Tablica koja slijedi sadrži najvažnije podatke o prikupljenim podacima tj. snimkama.

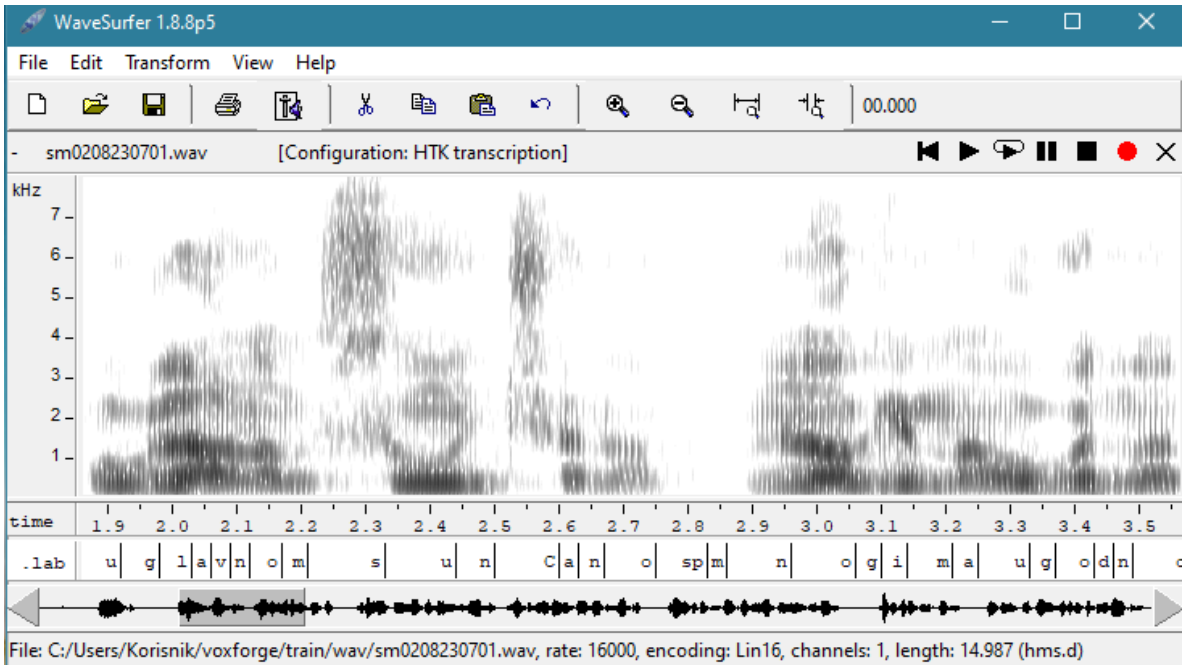
7.1 Statistika prikupljenih riječi

Ukupan broj vremenskih prognoza	Ukupan broj izrezanih snimaka	Trajanje snimaka s muškim govornikom	Trajanje snimaka s ženskim govornikom	Ukupno vremensko trajanje snimaka	Vrijeme prikupljanja snimaka	Ukupan broj jedinstvenih riječi
20	196	1677 s	1999 s	3676 s	19:00 h	1286

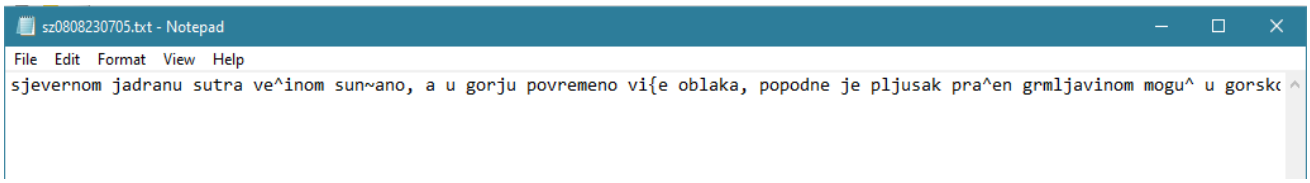
Zatim su izrađene .txt datoteke koje predstavljaju ručno izrađene transkripcije na temelju preslušavanja .wav datoteka što nije jako praktično zbog brzine pričanja govornika, prisutnog šuma u snimkama itd. stoga se još generiraju .lab datoteke. Datoteke s .lab ekstenzijom (tzv. label datoteke) predstavljaju automatski generirane transkripcije .wav datoteka, a svaka .lab datoteka sadrži vremenske oznake početka i kraja svakog pojedinog glasa tj. fonema zapisanih svaki u svome redu te su ovdje uključena oba tipa stanki tj. "sil" model koji predstavlja dužu stanku i "sp" model koji predstavlja kraću stanku u govoru. Te tri datoteke zajedno omogućuju uspješno prepoznavanje govora na temelju istreniranih riječi, a njima će biti proširena i sama baza govornih snimaka VEPRAD. Važno je napomenuti kako je za potrebe ovog rada korišteno 20 snimaka vremenskih prognoza ukupnog trajanja oko sat vremena što u kombinaciji sa svim prethodnim koracima rezultira dovoljno dobrom preciznošću prepoznavanja govora, no to ne znači da je prepoznavanje 100% točno i uvijek je moguće poboljšati preciznost prepoznavanja dodavanjem većeg broja snimaka vremenskih prognoza i povećanjem broja istreniranih riječi. Razni su razlozi zbog kojih dolazi do grešaka kod prepoznavanja govora, a u ovom konkretnom slučaju pretpostavka je da do grešaka dolazi zbog brzine govora koja otežava razumijevanje izgovorenih riječi te loša kvaliteta samih .wav datoteka zbog korištenja mikrofona ugrađenog u laptopu te šuma prisutnog u pozadini tijekom snimanja. Na sljedećim fotografijama prikazana su najprije dva primjera ručno izrađenih transkripcija i njihovih ispravnih automatski generiranih transkripcija za jednog muškog i ženskog govornika, a zatim slijede dva primjera gdje je došlo do grešaka kod generiranja transkripcija.



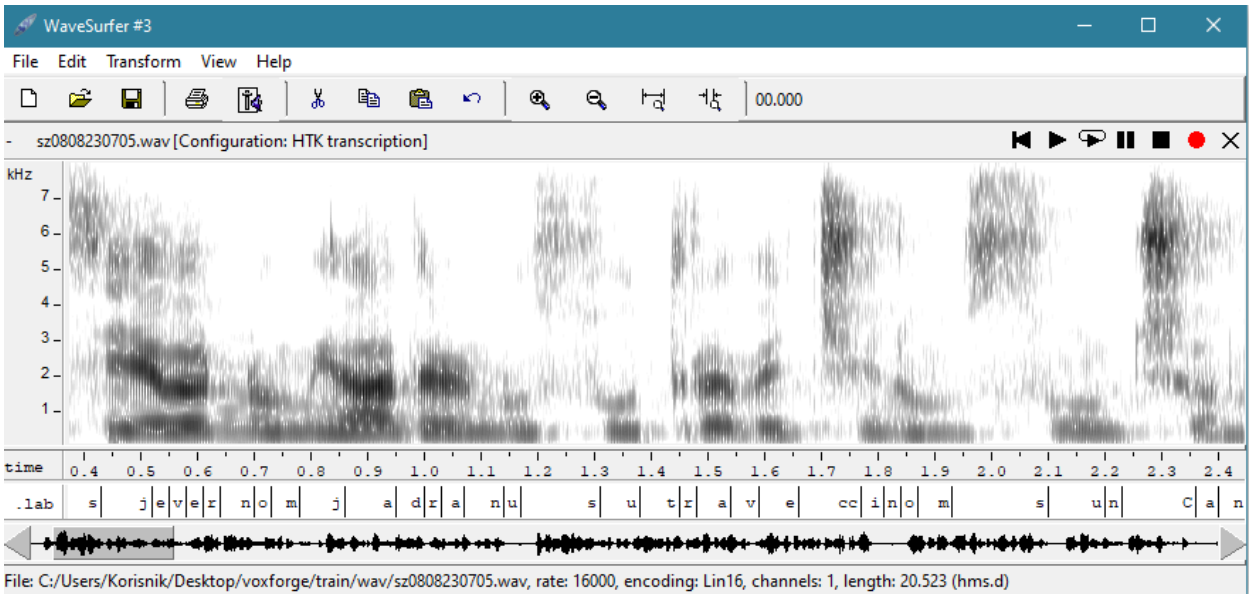
Slika 7.1. Ručno izrađena transkripcija snimke sm0208230701



Slika 7.2. Automatski generirana transkripcija snimke sm0208230701

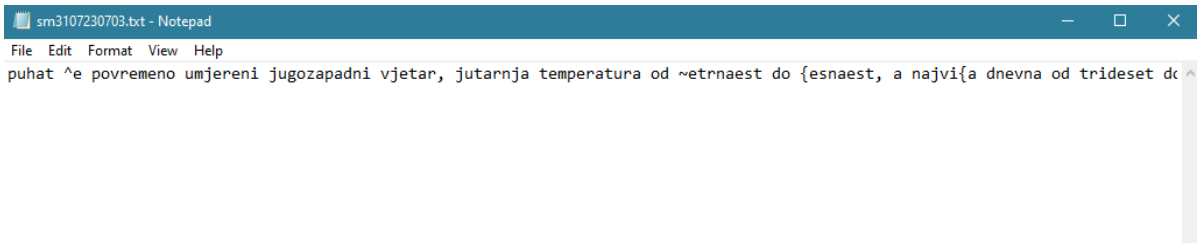


Slika 7.3. Ručno izrađena transkripcija snimke sz0808230705

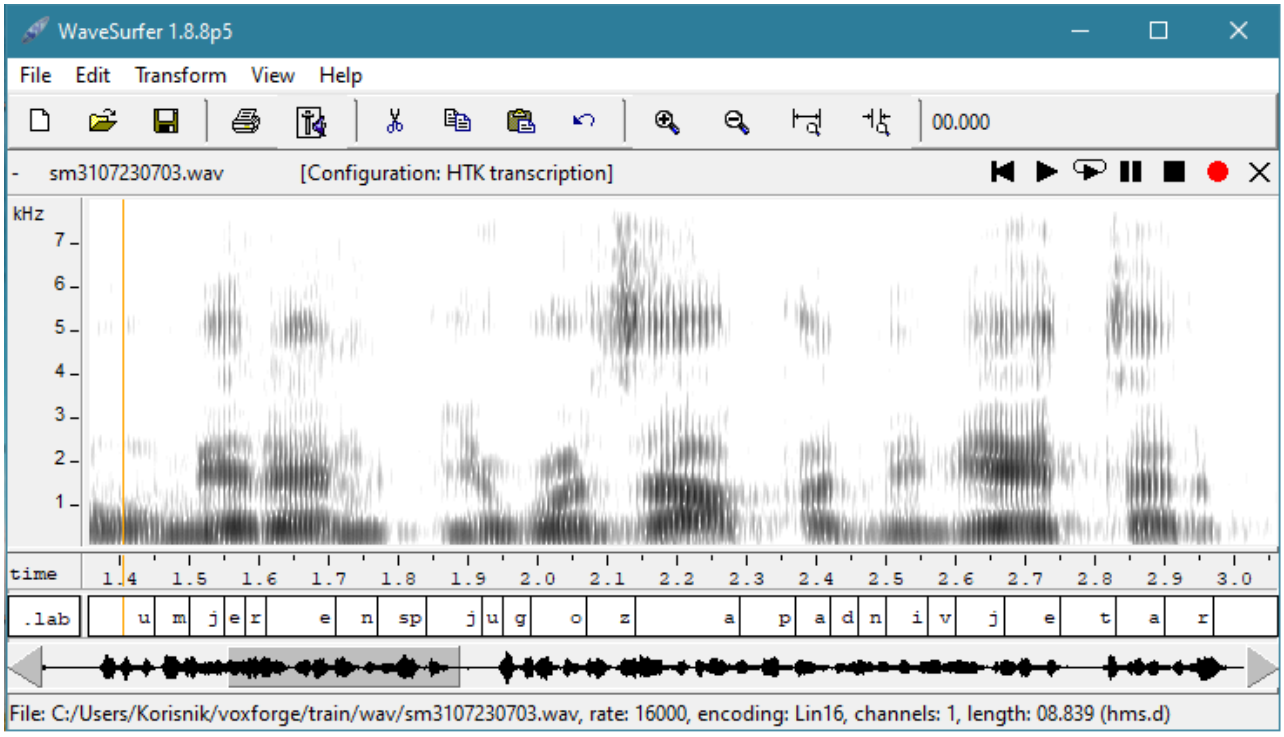


Slika 7.4. Automatski generirana transkripcija snimke sz0808230705

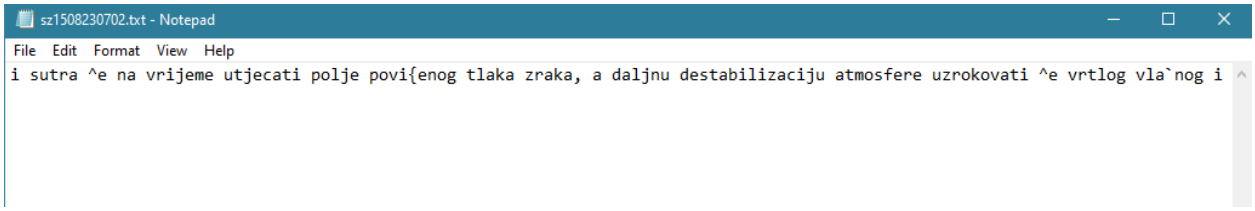
No, kao što je to već ranije spomenuto prepoznavanje govora nije 100% precizno pa je došlo i do nekoliko grešaka, a na slikama u nastavku bit će prikazano pogrešno prepoznavanje riječi za jednu snimku s muškim te jednu snimku sa ženskim govornikom. Grešaka nema puno te nije lako pronaći ih, ali kako se radi o relativno malenom broju uzoraka (oko 200 snimaka) greške su neizbježne.



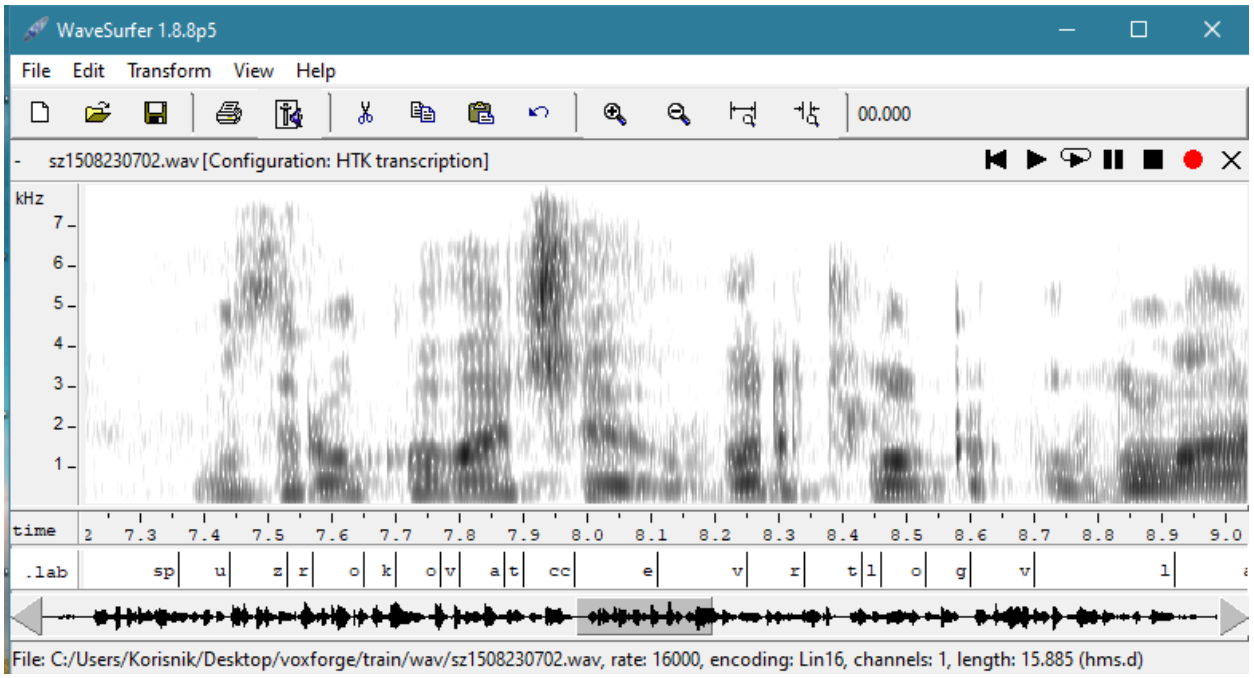
Slika 7.5. Ručno izrađena transkripcija datoteke sm3107230703 sadrži riječ umjereni



Slika 7.6. Automatski generirana transkripcija datoteke sm3107230703 sadrži riječ umjeren



Slika 7.7. Ručno izrađena transkripcija snimke sz1508230702 sadrži riječ uzrokovati



Slika 7.8. Automatski generirana transkripcija snimke sz1508230702 sadrži riječ uzrokovat

Kao što se da vidjeti sa slikama datoteka koje sadrže grešku, obje greške događaju se na kraju riječi u obliku izostajanja jednog slova tj. fonema, a pretpostavka je da do toga dolazi zbog jako brzog izgovora riječi od strane govornika i šuma prisutnog u snimkama koji često otežava raspoznavanje krajnjih glasova riječju.

8. ZAKLJUČAK

Tema ovog rada bila je vezana uz prepoznavanje govora, a kao cilj potrebno je bilo proširiti bazu govornih snimaka VEPRAD. Cilj je uspješno ostvaren te je baza govornih snimaka VEPRAD proširena novim audio datotekama i pripadajućim transkripcijama istih, a uz to uspješno je izrađena snimalica koja će se moći u budućnosti koristiti za prikupljanje snimaka u željenim vremenskim intervalima što nas oslobađa od jednog jako vremenski zahtjevnog zadatka. Kako bi se postupak proširenja baze govornih snimaka VEPRAD dodatno ubrzao dobro je razmisliti o načinu automatizacije samog rezanja audio datoteka na manje smislene cjeline te u budućnosti implementirati i tu funkcionalnost kako bi u potpunosti bili oslobođeni od faze pripreme podataka. Kako se rječnik u ovome radu sastoji od samo 20 vremenskih prognoza ne može se očekivati preciznost raspoznavanja govora od 100% te u slučaju da se želi dodatno popraviti trenutna preciznost moguće je dodati nove snimke i koristiti bolju opremu za njihovo snimanje kako bi što više uklonili šum i ostale smetnje. Ovim radom stečena su nova znanja vezana uz korištenje Audacity audio uređivača, osvježeno je i unaprijeđeno znanje o Python skriptiranju te je izrađena funkcionalna skripta koja ima ulogu snimalice vremenskih prognoza, a isto tako je proučen i obavljen čitav proces izrade akustičnog modela koji uključuje izradu monofona i trifona što je jako dobro temelje za daljnji rad u području prepoznavanja govora. Ovo područje je u konstantnom razvoju i taj trend će nastaviti još dugo godina stoga smatram kako je jako korisno upoznati se s osnovama prepoznavanja i sinteze govora.

LITERATURA

[1] Definicija i primjena prepoznavanja govora, s Interneta, What is Speech Recognition? (techtarget.com)

[2] Speech recognition, Wikipedia, s Interneta, Speech recognition - Wikipedia

[3] Speech Recognition Software: Past, Present, and Future, Summa Linguae, Speech Recognition Software: History, Present, and Future (summalinguae.com)

[4] History of voice search and voice recognition, s Interneta, History of voice search and voice recognition - Adido Digital (adido-digital.co.uk)

[5] Python 3, programski jezik, s Interneta, Welcome to Python.org

[6] Audacity, audio editor, s Interneta, Audacity ® | Free, open source, cross-platform audio software for multi-track recording and editing. (audacityteam.org)

[7] Julia, programski jezik, s Interneta, The Julia Programming Language (julialang.org)

[8] Julius, Open-Source Large Vocabulary CSR Engine Julius, s Interneta, Open-Source Large Vocabulary CSR Engine Julius (osdn.jp)

[9] HTK Toolkit, The Hidden Markov Model Toolkit, s Interneta, HTK Speech Recognition Toolkit (cam.ac.uk)

[10] HTKBook, HTK Toolkit dokumentacija, s Interneta, HTK Speech Recognition Toolkit (cam.ac.uk)

[11] HRTi, stare vremenske prognoze, s Interneta, HRTi

[12] HRT, hrvatski radio, prvi program, dnevne vremenske prognoze uživo, s Interneta, Radio uživo: Prvi program :: Hrvatski radio (hrt.hr)

[13] Python automatizacija, Automating with Python, s Interneta, Python Automation Benefits (linkedin.com)

[14] Python Automation: Scripts to Automate Critical Workflows, s Interneta, What to Know about Automating Tasks with Python Scripts (turing.com)

[15] PyAutoGUI, skripta za upravljanje mišem i tipkovnicom, s Interneta, Welcome to PyAutoGUI's documentation! — PyAutoGUI documentation

[16] Mel-frequency cepstrum, definicija s Wikipedia-e, s Interneta, Mel-frequency cepstrum - Wikipedia

[17] HMM, HMM definicija s Wikipedia-e, s Interneta, Hidden Markov model - Wikipedia

[18] Hidden Markov model – simple explanation in high level, s Interneta, Hidden Markov Model (HMM) — simple explanation in high level | by Darius Sabaliauskas | Towards Data Science

[19] VoxForge, Tutorial: Create Acoustic Model – Manually, s Interneta, Tutorial: Create Acoustic Model - Manually (copy) - voxforge.org

[20] Speech Recognition Challenges and How to Solve Them, s Interneta, Speech Recognition Challenges and How to Solve Them | Rev

[21] WaveSurfer, open source tool for sound visualization and manipulation, s Interneta, WaveSurfer download | SourceForge.net

POPIS SLIKA

SAŽETAK

U ovome se radu opisuje postupak izrade akustičnog modela koji će omogućiti prepoznavanje govora i proširenje baze govornih snimaka VEPRAD. Rad započinje opisom znanstvene discipline prepoznavanja govora te njezinim povijesnim razvojem. Zatim se opisuju ključni alati koji će biti korišteni za izradu skripte za automatizaciju snimanja vremenskih prognoza tj. snimalice i izradu samog akustičnog modela. Ostatak rada čini kronološki opis koraka potrebnih za prikupljanje i pripremu potrebnih podataka, izradu i povezivanje monofonih skrivenih Markovljevih modela te izradu i povezivanje trifona kako bi se dodatno povećala robusnost i preciznost prepoznavanja govora. Rezultat rada su automatska snimalica vremenskih prognoza te tri vrste datoteka (.wav, .txt i .lab datoteke) kojima se proširuje baza govornih snimaka VEPRAD.

Ključne riječi **– prepoznavanje govora, VEPRAD, snimalica, skriven Markovljev model, monofon, trifon**

ABSTRACT

This paper describes the process of creating an acoustic model which allows for speech recognition and expansion of the VEPRAD speech recordings database. The paper begins with a description of the scientific discipline of speech recognition and its historical development. Next, there is a description of all the key tools that will be used to create a script for automating the recording of weather forecasts i.e the recorder and the creation of the acoustic model itself. The rest of the paper consists of a chronological description of all the steps required to collect and prepare the necessary data, create and connect monophonic hidden Markov models and create and connect triphones in order to further increase the robustness and precision of speech recognition. The result is an automatic recorder of weather forecasts and three types of files (.wav, .txt and .lab files) which expand the database of VEPRAD voice recordings.

Keywords **– speech recognition, VEPRAD, the recorder, hidden Markov model, monophone, triphone**