Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

Doktorska disertacija

AVTOMATSKO RAZPOZNAVANJE GOVORA ZA PREGIBNI JEZIK Z UPORABO MORFOLOŠKIH JEZIKOVNIH MODELOV S KONTEKSTNO ODVISNO STRUKTURO

Maribor, februar 2015

Gregor Donaj, univ. dipl. inž. el., univ. dipl. mat.
mentor: prof. dr. Zdravko Kačič

Avtor:	Gregor Donaj
Naslov disertacije:	Avtomatsko razpoznavanje govora za pregibni jezik z uporabo morfoloških jezikovnih modelov s kontekstno odvisno strukturo
Naslov v angleščini:	Automatic speech recognition in an inflective language using morphological language models with context dependent structure
UDK:	004.934: 81'366-047.58(043.3)
Ključne besede:	avtomatsko razpoznavanje govora z velikim slovarjem, jezikovno modeliranje, faktorizirani jezikovni modeli, perpleksnost, oblikoskladenjske oznake, dvoprehodni iskalni algoritmi
Mentor:	red. prof. dr. Zdravko Kačič
Obdelava besedila:	Gregor Donaj
Lektoriranje:	doc. dr. Darinka Verdonik
Število izvodov:	12
Kraj in datum:	Maribor, februar 2015

Univerza v Mariboru

Slomškov trg 15
2000 Maribor, Slovenija

Maribor, 17. 12. 2013
Številka: DR 114/2013/428-DM

Na osnovi 287., 140., 142. in 144. člena Statuta Univerze v Mariboru (Statut UM-UPB10, Ur. l. RS, št. 46/2012) ter sklepa 26. redne seje Senata Univerze v Mariboru, ki je potekala 17. 12. 2013 v zvezi z vlogo doktorskega kandidata Gregorja Donaja za sprejem odločitve o predlagani temi doktorske disertacije in mentorja

izdajam naslednji

SKLEP

Odobri se tema doktorske disertacije Gregorja Donaja s Fakultete za elektrotehniko, računalništvo in informatiko z naslovom »Avtomatsko razpoznavanje govora za pregibni jezik z uporabo morfoloških jezikovnih modelov s kontekstno odvisno strukturo«. Za mentorja se imenuje red. prof. dr. Zdravko Kačič. Kandidat mora članici predložiti izdelano doktorsko disertacijo v zadostnih izvodih najpozneje do 16. 12. 2017.

Obrazložitev:

Kandidat Gregor Donaj je 16. 8. 2013 na Fakulteti za elektrotehniko, računalništvo in informatiko vložil vlogo za potrditev teme doktorske disertacije z naslovom »Avtomatsko razpoznavanje govora za pregibni jezik z uporabo morfoloških jezikovnih modelov s kontekstno odvisno strukturo«. Za mentorja je bil predlagan red. prof. dr. Zdravko Kačič.

Senat Fakultete za elektrotehniko, računalništvo in informatiko je na osnovi pozitivnega mnenja komisije za oceno teme doktorske disertacije, ki je ugotovila, da kandidat izpolnjuje pogoje za pridobitev doktorata znanosti, in ocenila, da je predlagana tema ustrezna, sprejel pozitivno mnenje in poslal predlog teme doktorske disertacije s predlogom mentorja v odobritev Senatu univerze.

Senat Univerze v Mariboru je po proučitvi vloge in na osnovi določil Statuta Univerze v Mariboru sprejel svojo odločitev o predlagani temi doktorske disertacije in imenoval mentorja, kot izhaja iz izreka.

V skladu s 144. členom Statuta Univerze v Mariboru mora kandidat za pridobitev doktorata znanosti najpozneje v štirih letih od dneva izdaje tega sklepa, članici predložiti izdelano doktorsko disertacijo v zadostnih izvodih. Kandidatu je bil določen rok za oddajo izdelane doktorske disertacije glede na datum sprejetja teme na pristojnem organu.

Pouk o pravnem sredstvu:
Zoper ta sklep je možna pritožba na Senat Univerze v Mariboru v roku 8 dni od prejema tega sklepa.

Rektor:
Prof. dr. Danijel Rebolj

Obvestiti:
1. Kandidata.
2. Fakulteto.
3. Arhiv.

UNIVERZA V MARIBORU 17

ZAHVALA

Zahvaljujem se svojemu mentorju prof. dr. Zdravku Kačiču za pomoč in nasvete pri izdelavi doktorskega dela ter za vodenje raziskovalnega dela. Zahvaljujem se tudi sodelavcem v Laboratoriju za digitalno procesiranje signalov za prijetno delovno okolje in sodelovanje na raziskovalnem področju. Posebej za zahvaljujem dr. Mirjam Sepesy Maučec za vse nasvete in za približanje področja jezikovnega modeliranja.

Kazalo

Slike

Tabele

Algoritmi

Povzetek

Ključne besede: avtomatsko razpoznavanje govora z velikim slovarjem, jezikovno modeliranje, faktorizirani jezikovni modeli, perpleksnost, oblikoskladenjske oznake, dvoprehodni iskalni algoritmi

UKD: 004.934: 81'366-047.58(043.3)

V nalogi smo se posvetili jezikovnemu modeliranju za avtomatsko razpoznavanje govora z velikim slovarjem besed. Pri takšnem razpoznavanju je še vedno velika težava pravilnost razpoznavanja izgovorjenih besed. Ta je še posebej izrazita pri morfološko kompleksnejših jezikih, kot je slovenščina. Za delovanje sistema razpoznavanja tekočega govora potrebujemo jezikovne modele. Da lahko zgradimo primeren jezikovni model, potrebujemo ustrezno velike učne množice podatkov, ki morajo pri morfološko kompleksnejših jezikih biti še večje. Sodobni razpoznavalniki govora za slovenščino delajo več napak kot razpoznavalniki za druge jezike. Pogost problem so napačno razpoznane končnice besed. To kaže, da je smiselno razmišljati o vključevanju oblikoskladenjskih informacij v jezikovno modeliranje, če hočemo zmanjšati število napak. V doktorski nalogi predstavljamo zasnovo sistema, ki ob običajnih n-gramskih besednih jezikovnih modelih uporablja tudi modele, ki vključujejo informacije o besedni vrsti in slovničnih kategorijah prepoznanih besed. Imenujemo jih morfološki modeli. Razvili smo algoritem, ki na osnovi rezultatov perpleksnosti na razvojni množici določa najprimernejšo strukturo takšnih modelov glede na besedne vrste konteksta besede, ki jo ocenjujemo. Pravimo, da imajo modeli kontekstno odvisno strukturo. Implementirali smo jih kot faktorizirane jezikovne modele. V teh modelih se soočamo z veliko množico različnih možnih kontekstov besede in za vsak kontekst gradimo strukturo modelov ločeno. Pri tem lahko uporabimo le majhen del učne množice. Zato prihaja tudi tukaj do pomanjkanja učnih podatkov, kljub temu da imamo manjše zahteve po velikosti učne množice. Zato smo razvili pristope združevanja različnih kontekstov. Zaradi velikega števila možnih kontekstov in veliko različnih možnosti struktur modelov smo razvili tudi pristope za omejeno iskanje možnih struktur modelov na podlagi postopne gradnje njihovih struktur in sprotnega ocenjevanja. Sistem razpoznavanja je zasnovan v obliki dvoprehodnega algoritma, kjer v drugem prehodu uporabljamo v okviru doktorske disertacije razvite modele. Razvili smo tudi postopek za hitro optimizacijo uteži modelov in postopek dinamičnega uteževanja glede na kontekst besede. Uspešnost razpoznavanja z razvitimi modeli in brez njih smo testirali na slovenski govorni bazi Broadcast News.

Abstract

Key words: large vocabulary automatic speech recognition, language modelling, factored language models, perplexity, morphosyntactic description tags, two-pass search algorithms

UKD: 004.934: 81'366-047.58(043.3)

In this thesis, we are focused on language modelling for automatic speech recognition in large vocabulary applications, where we are still experiencing the problem of insufficient recognition accuracy. This problem is more present in morphologically complex languages, for example Slovene. For such a system to work properly we need language models. State of the art speech recognition systems for Slovene still produce a hidher number of recognition errors that recognizers for other langauges. We see many sentences that are still understandable, but which contain syntactical errors. Often errors are present in the word endings. Therefore it seems reasonable to include morphosyntactic information into language models to reduce syntactical errors. This thesis presents the development of a speech recognition system that uses not only the usual n-gram language models for words, but also models that include part-of-speech and morphosyntactic information. We call them morphological models. We developed an algorithm that determines the best structure for such models based on perplexities for with respect to the part-of-speech categories of a words context. We say that the models have a context dependent structure. We implemented them as factored language models. Although we do not need a very large training corpus we still experience data sparsity due to the large number of possible context of a word. We therefore also developed a method for merging different context. Because of a large number of possible models structures it was also necessary to develop an algorithm for limiting the search space by gradually determining a models structure. The system is designed as a two-pass recognition algorithm, where the morphological models are used in the second pass. We developed an algorithm for a fast optimization of the systems parameters and dynamic weighting of the models scores based on a words context. We tested speech accuracy on the Slovene Broadcast news speech database. We also added a more detailed analysis of the recognition results.

Seznam uporabljenih kratic

BO	sestopanje (angl. Back-off)
BN	Broadcast News
BNSI	slovenska baza Broadcast News
DFT	diskretna Fourierjeva transformacija (angl. Discrete Fourier Transformation)
FIR	filter s končnim impulznim odzivom (angl. Finite impulse response)
FFT	hitra Fourierjeva transformacija (angl. Fast Fourier Transformation)
HMM	prikriti modeli Markova (angl. Hidden Markov Model)
JOS	jezikoslovno označevanje slovenščine
LPC	koeficienti linearnega napovedovanja (angl. Linear Prediction Coefficientes)
MAP	največja a posteriori verjetnost (angl. Maximum a-Posteriori)
MBR	cenilka minimalnega Bayesovega tveganja (angl. Minimum Bayer Risk)
MFCC	mel-frekvenčni kepstralni koeficienti (angl. Mel-Frequency Cepstral Coefficientes)
MLE	cenilka največje verjetnosti (angl. Maximim Likelihood Estimate)
MSD	oblikoskladenjske oznake (angl. Morpho Syntactic Description)
OER	delež napak oraklja (angl. Oracle Error Rate)
OOV	izven slovarja (angl. Out Of Vocabulary)
PLP	koeficienti percepcijskega linearnega napovedovanja (angl. Perceptual Linear Prediction)
POS	besedna vrsta (angl. Part Of Speech)
PP	perpleksnost (angl. Perplexity)
RTF	faktor realnega časa (angl. Real Time Factor)
VAD	detektiranja govora (angl. Voice Activity Detection)
WER	delež napačno razpoznanih besed (angl. Word Error Rate)
WFST	uteženi končni pretvorniki (angl. Weighted Finite State Transducer)

Poglavje 1

Uvod

Govorna komunikacija je človeku najbolj naraven način sporazumevanja, ki se je razvijal in izpopolnjeval skupaj s celotnim človekovim razvojem. Govorna komunikacija z drugimi ljudmi je za nas tudi najbolj preprosta. V sodobnem svetu pa se veliko srečujemo s tehnologijo, ki je včasih težko razumljiva in težko uporabna. Z računalniškimi sistemi komuniciramo preko tipkovnice, miške, monitorja in drugih naprav. Razvoj govornih tehnologij omogoča, da lahko ljudje tudi z računalnikom komuniciramo na bolj naraven način. Za dvosmerno govorno sporazumevanje z računalnikom sta potrebni dve tehnologiji: avtomatsko razpoznavanje govora in sinteza govora. Sistemi za sintezo govora uporabljajo kot vhodne podatke besedilo, ki je pri tem natančno in enolično določeno v računalniku. Sistem to sporočilo pretvori v zvočni signal, ki ga nato posreduje v zvočnik.

V obratni smeri delujejo sistemi za avtomatsko razpoznavanje govora [29, 53]. Ti kot vhodni podatek dobijo zvočni signal iz mikrofona in kot izhodni podatek vrnejo besedo oz. zaporedje besed, ki so bile izgovorjene. Neka beseda ima znotraj računalnika vedno enako obliko – vedno je zapisana z istimi simboli. Zvočni signal neke besede pa je pri vsaki izgovorjavi drugačen. Razlike se lahko pojavijo že pri istem govorcu, pri večih govorcih pa se najpogosteje še povečajo. Dodatne razlike v zvočni signal prinašajo tudi zvoki iz ozadja in značilnosti prenosnega kanala. Medtem ko se nam zdi, da je za človeka razpoznavanje govora enostavna naloga, je implementacija tega procesa v računalniški sistem zelo zapletena. Eden izmed glavnih izzivov razvoja razpoznavalnikov govora je pravilno razpoznavanje besed.

Primer možne uporabe takih razpoznavalnikov je samodejno podnaslavljanje televizijskih oddaj, kar bi gluhim osebam omogočalo spremljanje televizijskega programa. Sisteme za razpoznavanje govora lahko uporabljamo tudi v sistemih govorno orientiranega strojnega prevajanja, ki so sestavljeni iz razpoznavalnika govora, strojnega prevajalnika in sistema za sintezo govora. Druga področja uporabe so še sistemi govornega dialoga, kot na primer zvočno izbiranje v telefonskih menijih, govorno krmiljenje računalnika in drugih naprav ter narekovanje sporočil računalniku. Različne aplikacije so različno zahtevne

in dajejo različno dobre rezultate. Ponekod ti rezultati še niso zadovoljivi za praktično uporabo. Sodobni sistemi razpoznavanja govora temeljijo na statističnih akustičnih in jezikovnih modelih. Tema doktorske disertacije sega v kompleksnejše metode jezikovnega modeliranja za izboljševanje rezultatov razpoznavanja.

1.1 Opis problema

Zahtevnost razpoznavanja govora je odvisna od karakteristik govora in okoliščin, v katerih ga uporabljamo. Način govora sega od izgovarjanja izoliranih besed do zveznega govora, stil govora pa ločimo na pripravljen oz. bran govor in spontan govor. Naslednja karakteristika je velikost slovarja. Poznamo majhne slovarje (do 20 besed), srednje slovarje (do nekaj tisoč besed) in velike slovarje (60.000 besed in več). Z velikostjo slovarja je povezana določitev primernih jezikovnih modelov. Ti se ločijo na modele s končnimi stanji, ki se lahko uporabljajo pri majhnih slovarjih in dopuščajo le točno določene prehode med stanji, in na statistične modele, ki so naučeni s statističnimi metodami in modelirajo verjetnosti zaporedij besed. Slednje uporabljamo v razpoznavalnikih z velikim slovarjem besed. Večji slovar pomeni tudi več možnosti za zamenjave besed pri razpoznavanju. Akustični modeli so prilagojeni na govorce ter tehnologijo zajemanja in prenosa signalov. Ločimo sisteme, ki so namenjeni določenemu govorcu, in sisteme za neodvisnega (poljubnega) govorca. Kvaliteta zvočnega signala lahko bistveno vpliva na uspešnost razpoznavanja. Tukaj so pomembni razmerje med signalom in šumom, morebitni hrup v ozadju govora in tehnologija izdelave mikrofona, s katerim zajemamo govor. Kadar se zvočni signal od mikrofona do razpoznavalnika prenaša preko telekomunikacijskih linij, je pomembno poznati tudi značilnosti prenosnega kanala in uporabljeno kodiranje signalov.

Med zahtevne modalitete razpoznavanja govora spada razpoznavanje tekočega govora z velikim slovarjem. Primer aplikacije je razpoznavanje tekočega govora v dnevnoinformativnih oddajah. Takšne sisteme razvijamo z uporabo govornih baz tipa *Broadcast News* (BN), ki so na voljo za razne jezike, med njimi tudi za slovenščino [73]. Tipično so sestavljene iz množice posnetkov televizijskih oddaj in pripadajočih transkripcij govora. Zahteve za razpoznavalnike, ki delujejo na teh bazah, so: razpoznavanje tekočega govora z velikim slovarjem, razpoznavanje za neodvisnega govorca, razpoznavanje deloma pripravljenega in deloma spontanega govora, različne akustične značilnosti kanala (zvok ali glasba v ozadju), različne kvalitete prenosnega medija, deloma tudi prenos zvoka preko telefonske linije.

Dodatni faktor, ki vpliva na razpoznavanje govora, je sam jezik govora. Akustične značilnosti jezika se ločijo po naboru fonemov tega jezika. Razlike so tudi na nivoju besedišča in načina tvorjenja stavkov, ki sta pomembna za izdelavo jezikovnih modelov. Znano je, da razpoznavalniki govora dobro delujejo za razpoznavanje angleščine, za katero sta značilna majhna pregibnost besed in dokaj fiksen vrstni red besed znotraj stavka. Večjo

zahtevnost predstavljajo morfološko kompleksnejši jeziki. Taki so nemški [45, 47], turški [59] in finski jezik [25, 26]. Za te jezike je značilno, da se v njih pojavlja veliko sestavljenih besed. Zato potrebujemo za dobro pokritost besedišča večji slovar, če uporabljamo razpoznavalnik v splošni domeni. Med morfološko kompleksnejše jezike spadajo tudi slovanski jeziki [31, 48, 49, 65, 70], vključno s slovenščino [62]. Značilnost slovenščine je visoka pregibnost besed, ki svojo obliko spreminjajo glede na pomen in vlogo v stavku. Značilna tipa pregibanja sta sklanjanje samostalniških in pridevniških besed ter spreganje glagolov. Pregibnost besed ima za posledico, da za določen pojem obstaja veliko število besednih oblik. Izrazit primer je glagol *biti*, za katerega obstaja 38 različnih oblik, ki se razlikujejo glede na čas, osebo, število in naklon glagola. Ocenjujemo, da potrebujemo za doseganje primerljivega deleža besed izven slovarja za slovenščino do 10-krat večji slovar kot pa za angleški jezik [58]. Podobne ocene lahko najdemo na primer za ruščino [68]. Podobno kot v jezikih, kjer so značilne sestavljanke besed (nemški, arabski, finski), pomeni pregibnost besed potrebo po večjem slovarju za dovolj dobro pokritost jezika. V slovenščini se pregibnost besed vidi v spreminjanju besedne končnice. Iz nje lahko razberemo vlogo pojma, ki ga beseda izraža, v stavku. Posledično za sporazumevanje v slovenščini ni potreben določen vrstni red besed v jeziku, da bi lahko prepoznali pomen stavka. Ker pa statistični jezikovni modeli temeljijo prav na modeliranju verjetnosti zaporedij besed, to pomeni še dodaten izziv za modeliranje pregibnih jezikov. Ker obstaja več različnih možnih zaporedij za izražanje neke misli, potrebujemo večje učne korpuse, v katerih se ta možna zaporedja pojavijo. Dodatno pa veliko število možnih zaporedij, ki so vključena v jezikovni model, pomeni večjo možnost njihovih zamenjav, če obstaja akustična podobnost.

Dobro znan pristop modeliranja morfološko kompleksnih jezikov je uporaba podbesednih enot [25, 57, 58, 65]. Pri tem pristopu besedo razdelimo na več manjših enot ter gradimo jezikovne modele na teh enotah. V tem primeru razumemo pod pojmom n-gram množico n zaporednih podbesednih enot v besedilu. Ta metoda večinoma ni dala bistvenih izboljšav. Dodatno pa se je izkazalo, da za uspešno modeliranje s podbesednimi enotami potrebujemo n-gramske modele visokega reda [27].

Še en primer jezika, ki je morfološko kompleksen, je arabščina [46]. V namen izboljšanja možnosti modeliranja arabščine so bili razviti faktorizirani jezikovni modeli [7, 35, 36]. Ti ponujajo pomembno posplošitev prej znanih jezikovnih modelov z veliko prostora za definiranje faktorjev in struktur modelov. Kljub temu da obstaja že nekaj raziskav o uporabi faktoriziranih jezikovnih modelov, jih še zmeraj lahko štejemo za slabo raziskane. Skupaj s faktoriziranimi jezikovnimi modeli je bil tudi vpeljan koncept posplošenega in vzporednega sestopanja jezikovnih modelov. Sestopanje pomeni umik na model nižjega reda, kadar v modelu ne najdemo celotnega konteksta in besede, ki jo ocenjujemo. V običajnih besednih modelih poteka sestopanje vedno z odstranitvijo najbolj oddaljene besede. V faktoriziranih jezikovnih modelih je to zaporedje lahko poljubno. Lahko ga fiksno določimo za vse vnose v modelu ali pa ga določamo dinamično glede na neke lastnosti besed v kontekstu.

Za uporabo faktoriziranih jezikovnih modelov in morfoloških informacij moramo imeti na voljo dovolj velike in dovolj kvalitetne jezikovne vire, ki vsebujejo morfološke informacije. V okviru projekta Jezikoslovno označevanje slovenščine (JOS) [15] so bile razvite oblikoskladenjske specifikacije za slovenščino. Zraven je bil razvit tudi korpus JOS100k, ki je vseboval 100.000 besed, ki so jim bile pripisane oblikoskladenjske oznake. Te so bile ročno pregledane. Ta korpus je bil kasneje razširjen v petkrat večji korpus ssj500k [23], ki ima enake značilnosti. Prav tako je na voljo oblikoskladenjski označevalnik Obeliks [23], ki označuje besedilo po specifikacijah JOS. Obstoj takšnih korpusov in orodij je pogoj za izdelavo kvalitetnih morfoloških jezikovnih modelov.

1.2 Cilji

V okviru doktorske disertacije hočemo pokazati, da lahko z uvedbo morfoloških jezikovnih modelov v algoritem razpoznavanja govora dosežemo izboljšanje rezultatov pri razpoznavanju govora z velikim slovarjem. Morfološki modeli pri tem temeljijo na oblikoskladenjskih oznakah hipotez razpoznavalnika, njihova struktura oz. potek sestopanja, način uporabe in utež v končni oceni hipoteze pa so odvisni od besednih vrst, ki se pojavijo v hipotezi. Predvidevamo, da lahko z ustrezno uporabo verjetnosti, ki jo dobimo iz jezikovnega modela, na vsakem koraku ocenjevanja hipoteze strukturo modela prilagodimo na kontekst besede, ki jo trenutno ocenjujemo. Pri tem poskušamo prilagoditi strukturo modela tako, da uporabimo tiste besede in njihove lastnosti v kontekstu, ki so pomembne za ocenjevanje trenutne besede.

1.3 Izvirni znanstveni prispevki

Glavni znanstveni prispevek disertacije je uvedba jezikovnih modelov s kontekstno odvisno strukturo, ki bodo predstavljali posplošitev n-gramskih modelov. V okvirju doktorske disertacije smo podali njihovo definicijo, potek učenja strukture modelov in način uporabe modelov. V okvirju disertacije smo tudi opozorili na razne probleme, ki nastopijo pri uvedbi takšnih modelov, ter podali rešitve za te probleme. Zato kot ostale izvirne znanstvene prispevke navajamo:

1. Določili smo velikost iskalnega prostora struktur modelov in nabor različnih kontekstov, od katerih bo odvisna struktura modela.

2. Definirali smo algoritem za učenje strukture morfoloških modelov, ki temelji na testiranju različnih modelov za različne kontekste. Testiranje temelji na kriterijski funkciji, izpeljani iz perpleksnosti jezikovnih modelov.

3. Definirali smo algoritem za postopno gradnjo struktur modelov, ki temelji na začetnem naboru kandidatov rešitev, dodajanju posameznih faktorjev v strukture v trenutnem naboru in izločanju posameznih struktur iz nabora rešitev na podlagi snopovnega omejevanja. S tem smo zmanjšali nabor modelov za testiranje.

4. Definirali smo algoritem za združevanje različnih kontekstov v razrede, ki temelji na podobnosti modelov. To podobnost določamo iz nabora rešitev za dva konteksta. Tako smo zmanjšali nabor razredov in se izognili pomanjkanju učnih podatkov.

5. Do sedaj navedene algoritme smo združili v končni algoritem in dodali kriterij za izbiro končne rešitve iz nabora rešitev za vsak razred kontekstov.

6. Sestavili smo dvoprehodni iskalni algoritem za preverjanje uspešnosti delovanja modelov s kontekstno odvisno strukturo. V tem algoritmu smo uvedli postopek za optimizacijo uteži posameznih modelov in dinamično spreminjanje uteži na podlagi ocenjevanega govornega segmenta.

7. Za namen podrobnejše analize rezultatov na jezikovnem nivoju smo uvedli tudi posplošitev Levenshteinove razdalje, ki daje boljšo poravnavo med pravilnimi in razpoznanimi besedami. S tem smo dosegli bolj zanesljivo analizo napak.

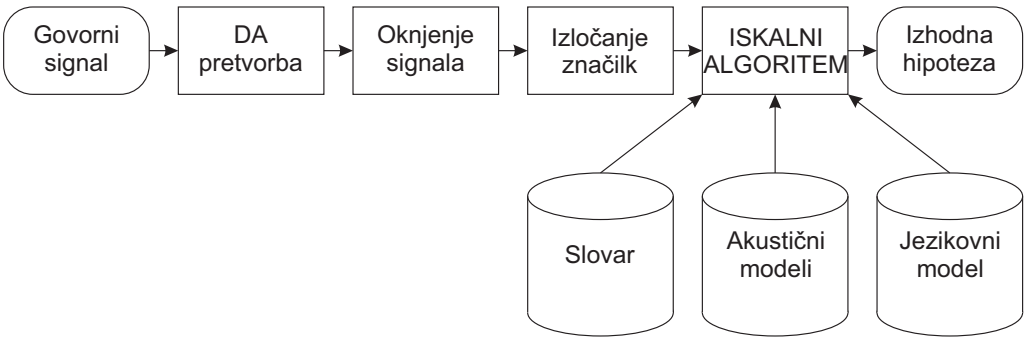
1.4 Opis poglavij

V drugem poglavju bomo predstavili pregled osnovnih tehnologij, ki se uporabljajo v razpoznavanju tekočega govora. Predstavili bomo osnovne postopke akustičnega procesiranja govora in osnovne gradnike razpoznavalnika. Ti so slovar, akustični modeli, jezikovni modeli in iskalni algoritem. Nadaljevali bomo z opisom faktoriziranih jezikovnih modelov, ki jih bomo uporabljali v okvirju disertacije. V poglavju bomo tudi predstavili način vrednotenja izhoda razpoznavalnika in primerjave več rezultatov. V tretjem poglavju bomo predstavili osnove oblikoskladenjskega označevanja. Predstavili bomo osnove raznih algoritmov, ki se uporabljajo pri označevanju. Bolj podrobno bomo pogledali označevalnik Obeliks, ki ga bomo v nadaljevanju uporabljali. V četrtem poglavju bomo vpeljali faktorizirane jezikovne modele s kontekstno odvisno strukturo. Predstavili bomo definicije faktorjev, zasnovo strukture modela, algoritem za določanje strukture ter postopek učenja modelov. V petem poglavju bomo vpeljali iskalni algoritem, ki uporablja jezikovne modele s kontekstno odvisno strukturo, ter algoritma za optimizacijo parametrov iskalnega algoritma in prilagajanje uteži modelov. V šestem poglavju bomo opisali eksperimentalni sistem, s katerim smo preverili učinkovitost uporabe predlaganih modelov in delovanje iskalnega algoritma. Sistem je zasnovan na govorni bazi s tekočim govorom v domeni *Broadcast News*. V sedmem poglavju bomo predstavili eksperimentalne rezultate. V osmem poglavju bo sledil zaključek.

Poglavje 2

Razpoznavanje govora

V tem poglavju bomo opisali osnovne korake v procesiranju govora, ki so potrebni za pripravo signala v primerno obliko za razpoznavanje, ter osnovne gradnike razpoznavalnika. Zvočni posnetki se najprej segmentirajo na govorne in negovorne segmente. Za ta namen uporabljamo na primer algoritme zaznavanja govora (Voice Activity Detection – VAD) [54]. Ti so namenjeni predvsem ločevanju posnetkov na govor in tišino oz. šum. Za ločevanje različnih vsebin zvočnega posnetka, na primer govora in glasbe, uporabljamo bolj zapletene algoritme [39]. Osnovna zgradba sistemov za razpoznavanje govora je prikazana na sliki 2.1.



Slika 2.1: Blokovna shema osnovnih gradnikov razpoznavalnika govora.

Sam postopek razpoznavanja govora poteka v več korakih. Prvi korak je akustično procesiranje, katerega namen je pretvoriti vhodni govorni signal v zaporedje vektorjev značilk. S temi vektorji tudi naučimo akustične modele govora. Najbolj razširjena tipa značilk sta mel-frekvenčni kepstralni koeficienti (MFCC) [6] in koeficienti percepcijskega linearnega napovedovanja (PLP) [24]. Oba temeljita na značilnostih človeškega sluha in sta se izkazala za učinkovita pri razpoznavanju govora. Za izračun značilk moramo določiti nekatere parametre: dolžino in obliko okna, zamik med okni, število filtrov in število koeficientov. K značilkam dodajamo še normirano energijo signala ter prve in druge odvode značilk. Značilke so predstavljene kot vektorji, njihova dimenzionalnost pa

je tipično do 39 [74] in ponekod še več. Po izračunu značilk ne uporabljamo več zvočnega signala.

Osrednji del razpoznavalnika govora je iskalni algoritem [4, 22, 64, 71]. Njegova naloga je na podlagi vhodnih podatkov – vektorjev značilk – poiskati najverjetnejše zaporedje besed, ki bi lahko tvorilo vhodno zaporedje značilk. Pri tem algoritem primerja različna možna zaporedja besed, ki jih imenujemo hipoteze. Poznamo različne iskalne algoritme. Ločimo jih lahko na časovno sinhrone in časovno asinhrone ter na algoritme s statičnim in algoritme z dinamičnim načinom razširjanja iskalnega prostora. Nekateri algoritmi uporabljajo še dodatne hevristične metode. Iskalne algoritme ločimo tudi na enoprehodne in večprehodne. Enoprehodni algoritmi izvedejo operacijo iskanja najboljše hipoteze le enkrat in pri tem uporabijo vse jezikovne vire, ki so na voljo. Večprehodni algoritmi izvajajo iskanje in ocenjevanje hipotez večkrat. Običajno v prvem prehodu ne uporabimo vseh razpoložljivih jezikovnih virov ali pa jih uporabljamo v poenostavljeni obliki. Rezultat prvega prehoda je nabor različnih hipotez. Te so lahko predstavljene v obliki preprostega seznama, t. i. seznama N najboljših hipotez, ali pa v obliki besedne mreže. Iskalni algoritem določi najboljšo hipotezo na podlagi utežene vsote logaritmov verjetnosti, ki jih dobi s strani akustičnega in jezikovnega modela, ter števila besed v hipotezi. Optimalne vrednosti uteži modelov so običajno fiksne in določene s pomočjo razvojne množice govornih baz, ki je ločena od učne in testne množice.

Iskalni algoritem za primerjavo hipotez uporablja tri vire informacij. Prvi je slovar besed s pripadajočimi fonetičnimi transkripcijami. Drugi vir je nabor akustičnih modelov. S pomočjo teh dveh virov lahko iskalni algoritem določa akustične verjetnosti, da vhodni signal ustreza neki izgovorjeni besedi. Tretji vir je jezikovni model. Ta je predvsem pomemben pri razpoznavanju tekočega govora. Jezikovni model vsebuje verjetnosti, da se v jeziku pojavi določeno zaporedje besed. Poznamo različne vrste jezikovnih modelov. Modeliranje jezika predstavlja pomemben izziv predvsem pri modeliranju pregibnih jezikov, kot so slovenščina in drugi slovanski jeziki. V zadnjem desetletju so se pojavili tudi t. i. faktorizirani jezikovni modeli [7], ki so primerni za modeliranje morfološko zahtevnejših jezikov.

V naslednjih podpoglavjih bomo podrobneje predstavili omenjene komponente razpoznavalnika govora in posebnosti pri razpoznavanju slovenščine in drugih pregibnih jezikov. V zadnjem podpoglavju pa bomo opisali tudi postopke za ocenjevanje uspešnosti razpoznavanja in primerjavo rezultatov s statističnimi testi.

2.1 Akustično procesiranje

2.1.1 Predprocesiranje

Prvi korak v obdelavi akustičnega signala govora je njegova analogno-digitalna pretvorba. Za govor lahko rečemo, da njegove značilnosti segajo vse do frekvence 16 kHz, sam govor pa lahko ljudje prepoznamo, tudi če imamo na voljo le spodnje 4 kHz signala. Toliko je tudi zgornja frekvenčna meja v telefoniji. Običajne hitrosti vzorčenja v razpoznavanju govora so od 8 do 16 tisoč vzorcev na sekundo. Druga lastnost analogno-digitalne pretvorbe je ločljivost posameznih vzorcev. Dinamični razpon človeškega sluha ustreza vse do približno 20-bitne ločljivosti. V razpoznavanju govora se je izkazalo, da je dovolj, če uporabimo 16-bitno ločljivost vzorcev.

Značilnosti govora lažje opazujemo v frekvenčnem prostoru kot pa v časovnem. Tako prepoznavamo foneme po formantih (vrhih v frekvenčnem spektru). Za govor je značilno, da njegova energija hitro pojema z večanjem frekvence. Ker se v višjih frekvencah skrivajo formanti, ki lahko vsebujejo pomembno informacijo o govoru, pred samim izločanjem značilk signal predobdelamo z ustreznim digitalnim filtrom. Uporabimo kar FIR-filter prvega reda, ki ga opišemo z enačbo:

$$y[n] = x[n] - \alpha \cdot x[n-1], \tag{2.1}$$

kjer je α parameter filtra. Tipična vrednost zanj je 0,95. Takšen filter ojača komponente višjih frekvenc za več kot 20 dB.

Signale analiziramo v frekvenčnem prostoru tako, da jih pretvorimo z diskretno Fourierjevo transformacijo (DFT). Značilnosti govornega signala se med govorom nenehno spreminjajo. Da jih sploh lahko analiziramo, predpostavimo, da so v nekem kratkem časovnem okviru stacionarne in jih analiziramo samo znotraj tega okvira. Običajne dolžine okvirov se gibljejo okoli 25 ms, nizamo pa jih na vsakih 10 ms. Na takšnem časovnem okviru uporabimo kratkočasovno Fourierjevo transformacijo. Še pred tem pa signal znotraj okvira oblikujemo s primernim oknom. Največkrat uporabljeno okno v razpoznavanju govora je Hammingovo okno, določeno z enačbo:

$$w[n] = \begin{cases} 0,54 - 0,46 \cdot \cos\left(\frac{2\pi n}{N-1}\right) & ; \quad n = 0, ..., N-1 \\ 0 & ; \quad \text{sicer} \end{cases} . \tag{2.2}$$

2.1.2 Spektralna in kepstralna analiza

Prvi korak analize govornega signala je njegova pretvorba v spektralno obliko. Naj bo $x[n]$ vhodni signal. Signal razdelimo na okvire. Tako dobimo za okvir m kratkočasovni signal $x_m[n] = x[n]w_m[n]$, kjer je $w_m[n] = w[m-n]$ na ustrezno mesto postavljeno okno.

Kratkočasovna Fourierjeva transformacija tega okvira je potem

$$X_m(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_m[n]e^{-j\omega n} = \sum_{n=-\infty}^{\infty} w[m-n]x[n]e^{-j\omega n}. \tag{2.3}$$

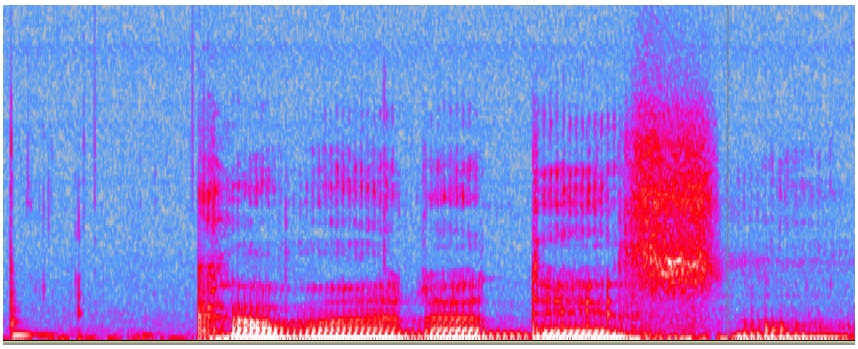
Ko vrednosti ω enakomerno razporedimo, lahko zapišemo

$$X_m[k] = X_m(e^{2\pi k/N}) \qquad k = 0, ..., N-1. \tag{2.4}$$

Ker je človeško uho neobčutljivo na fazne razlike med frekvenčnimi komponentami zvoka, nas v spektrogramu zanimajo le energije komponent, ki jih izračunamo iz realnega in kompleksnega dela:

$$|X[k]|^2 = X_r^2[k] + X_i^2[k]. \tag{2.5}$$

Primer spektrograma je na sliki 2.2. Vodoravna os v njem je čas, navpična pa frekvenca. Različne vrednosti energij so predstavljene z različnimi barvami.



Slika 2.2: Spektrogram posnetka besed *kamera desno*.

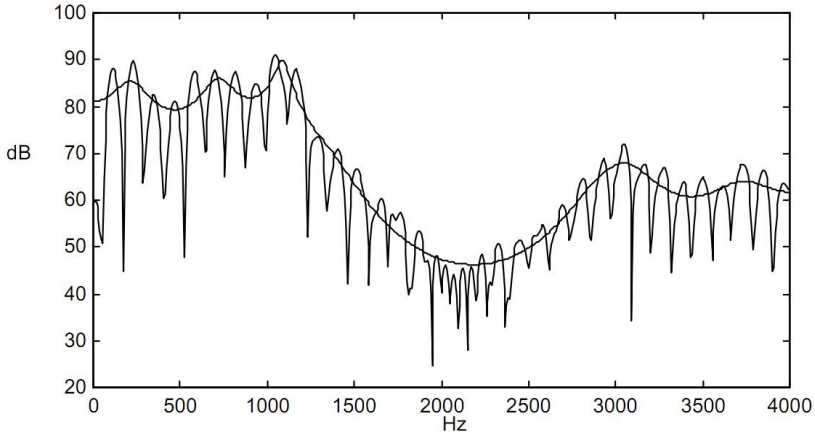
Naslednji korak je izračun kepstra. Realni kepster diskretnega signala je definiran z

$$c[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln|X(e^{j\omega})|e^{j\omega n}d\omega, \tag{2.6}$$

kompleksni kepster pa z

$$\hat{x}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln X(e^{j\omega})e^{j\omega n}d\omega. \tag{2.7}$$

V obeh enačbah smo uporabili kompleksni logaritem. Tako izračun realnega kot izračun kompleksnega kepstra sta homomorfni preslikavi. To pomeni, da konvolucijo dveh signalov pretvorita v vsoto. Na tak način lahko ločujemo izvor signala in filter.



Slika 2.3: LPC-spekter fonema /*ah*/ v besedi *lifes* in kratkočasovni spekter [29].

2.1.3 Koeficienti percepcijskega linearnega napovedovanja

Najprej bomo opisali linearno napovedovanje (LPC) ali avtoregresivno modeliranje. Pokazati je mogoče, da je vsepolni filter lahko dober približek za modeliranje govora. Tak filter definiramo z

$$H(z) = \frac{X(z)}{E(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)}, \tag{2.8}$$

kjer je p red analize LPC. Inverzna z-transformacija zgornje enačbe je

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + e[n]. \tag{2.9}$$

V linearnem napovedovanju napovemo trenutni otipek kot linearno kombinacijo preteklih p otipkov:

$$\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k]. \tag{2.10}$$

Napaka napovedovanja je pri tem: $e[n] = x[n] - \tilde{x}[n]$.

Za ocenjevanje koeficientov linearnega napovedovanja $a_1, ..., a_p$ uporabljamo kratkočasovne metode. Za vsak otipek definiramo kratek segment govora. Za ta segment pa definiramo napako kot vsoto kvadratov napak vseh otipkov v segmentu. Kriterij, s katerim izračunamo koeficiente, je ravno minimizacija te napake. Ta metoda vodi do t. i. Yule-Walkerjeve enačbe. To je sicer matrična enačba za sistem linearnih enačb in jo zato lahko rešujemo s katerokoli metodo za reševanje takih sistemov. Ker ima ta enačba pri izračunu koeficientov LPC posebno strukturo, lahko uporabljamo bolj učinkovite metode: kovariančno metodo, avtokorelacijsko metodo in mrežno formulacijo. Natančen postopek izračuna koeficientov in omenjenih metod najdemo v [29]. Iz izračunanih koeficientov potem sestavimo vektor značilk, ki ga uporabljamo v nadaljnjem procesiranju govora.

Na sliki 2.3 je primerjava med običajnim kratkočasovnim spektrom, dobljenim z diskretno Fourierjevo transformacijo, in spektrom, dobljenim na podlagi analize LPC s 14 koeficienti. Na sliki vidimo, da je spekter LPC bolj gladek od običajnega in da lepo prikaže formante.

Klasično linearno napovedovanje, ki smo ga opisali zgoraj, ne upošteva značilnosti človeškega sluha. Model namreč aproksimira signal enako dobro v vseh frekvenčnih področjih. Človeški sluh pa je v različnih območjih različno občutljiv.

Hermansky [24] je predlagal modificiran postopek računanja koeficientov napovedovanja, ki upošteva tri značilnosti človeškega sluha:

- spektralno razločljivost kritičnih pasov,
- krivulje enakih glasnosti,
- zvezo med intenziteto in glasnostjo.

V nadaljevanju bomo opisali postopek percepcijskega linearnega napovedovanja (PLP). Okvirna shema postopka je prikazana na sliki 2.4.

Slika 2.4: Osnovni koraki v izračunu koeficientov percepcijskega linearnega napovedovanja.

Spektralna analiza

Prvi korak PLP je spektralna analiza. Uporabljamo Hammingonovo okno in hitro Fourierjevo transformacijo (FFT). Najprej izračunamo spektralne energije $P(\omega)$. Nato upoštevamo lastnost človeškega ušesa, da izkazuje različno občutljivost na različnih frekvenčnih

področjih in da pri visokih frekvencah frekvenčna ločljivost ušesa pada. Spekter raztegnemo po frekvenčni osi ω v Barkovo frekvenco Ω:

$$\Omega(\omega) = 6 \ln \left\{ \frac{\omega}{1200\pi} + \sqrt{\left(\frac{\omega}{1200\pi}\right)^2 + 1} \right\}. \tag{2.11}$$

Nato uporabimo kritične pasove, definirane s predpisom

$$\Psi(\Omega) = \begin{cases} 0 & ; \quad \Omega < -1,3 \\ 10^{1,5(\Omega+0,5)} & ; \quad -1,3 \leq \Omega \leq -0,5 \\ 1 & ; \quad -0,5 < \Omega < 0,5 \\ 10^{-1,0(\Omega-0,5)} & ; \quad 0,5 \leq \Omega \leq 2,5 \\ 0 & ; \quad \Omega \end{cases}, \tag{2.12}$$

da dobimo energijske spektre posameznih pasov:

$$\Theta(\Omega_i) = \sum_{\Omega=-1,3}^{2,5} P(\Omega - \Omega_i)\Psi(\Omega_i). \tag{2.13}$$

Enake glasnosti

Energijske spektre pasov ojačimo s predpisom

$$\Xi[\Omega(\omega)] = E(\omega)\Theta[\Omega(\omega)], \tag{2.14}$$

kjer je $E(\omega)$ aproksimacija občutljivosti človeškega ušesa na različne frekvence:

$$E(\omega) = \frac{(\omega^2 + 56,8 \cdot 10^6)\omega^4}{(\omega^2 + 6,3 \cdot 10^6)^2(\omega^2 + 0,38 \cdot 10^9)}. \tag{2.15}$$

Intenziteta – glasnost

Človeško uho tudi ne izkazuje linearnega odnosa med močjo signala in glasnostjo, ki jo zaznamo. Zato uporabimo naslednjo pretvorbo:

$$\Phi(\Omega) = \sqrt[3]{\Xi(\Omega)}. \tag{2.16}$$

Inverzna DFT in avtoregresivni koeficienti

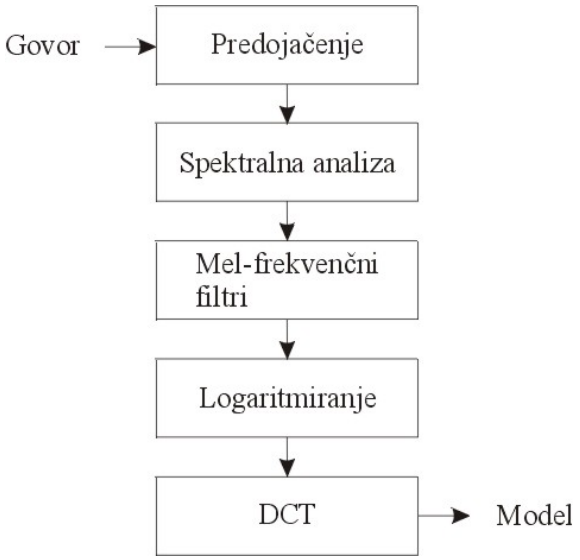
Na tem mestu smo že končali s posebnostmi PLP. Preostane nam še izračun koeficientov. Najprej končni spekter $\Phi(\Omega)$ obdelamo z inverzno diskretno Fourierjevo transformacijo in na dobljenem signalu uporabimo postopek linearne predikcije tako, kot smo ga opisali v začetku tega podpoglavja.

Uporaba koeficientov PLP v razpoznavanju govora se je večkrat izkazala uspešnejša od uporabe klasične metode LPC. Značilno pa je tudi, da dosega PLP najboljše rezultate pri manjšem številu uporabljenih koeficientov oz. manjšem redu linearne analize.

2.1.4 Mel-frekvenčni kepstralni koeficienti

Idejo mel-frekvenčnih kepstralnih koeficientov (MFCC) so vpeljali Biem in drugi [6]. Prav tako kot pri PLP je ideja v vključevanju znanja o delovanju človeškega sluha. Koraki izračuna koeficientov MFCC so predstavljeni na sliki 2.5. V nadaljevanju podpoglavja bomo opisali posamezne korake izračuna MFCC in jih pri tem primerjali s PLP. Metoda je dobila ime po mel-skali za frekvenco, ki je definirana z

$$B(f) = 1125 \ln\left(\frac{f}{700} + 1\right). \tag{2.17}$$



Slika 2.5: Osnovni koraki v izračunu mel-frekvenčnih kepstralnih koeficientov.

Predojačenje in spektralna analiza

Prvi korak je predojačanje višjefrekvenčnih signalov s filtrom, ki je opisan z enačbo 2.1. Tudi spektralna analiza pri izračunu MFCC je običajna, kot smo jo opisali v tretjem podpoglavju.

Predojačanje je namenjeno upoštevanju različnih frekvenčnih občutljivosti ušesa, ki jo v PLP upoštevamo s predojačenjem za enake glasnosti.

Mel-frekvenčni filtri

Za signal, dobljen po diskretni Fourierjevi transformaciji

$$X_a[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N} \qquad 0 \leq k < N, \tag{2.18}$$

definiramo nabor M filtrov, v katerem je m-ti filter definiran z

$$H_m[k] = \begin{cases} 0 & ; \quad k < f[m-1] \\ \frac{2(k-f[m-1])}{(f[m+1]-f[m-1])(f[m]-f[m-1])} & ; \quad f[m-1] \leq k \leq f[m] \\ \frac{2(f[m+1]-k)}{(f[m+1]-f[m-1])(f[m+1]-f[m])} & ; \quad f[m] < k \leq f[m+1] \\ 0 & ; \quad k > f[m+1] \end{cases}. \tag{2.19}$$

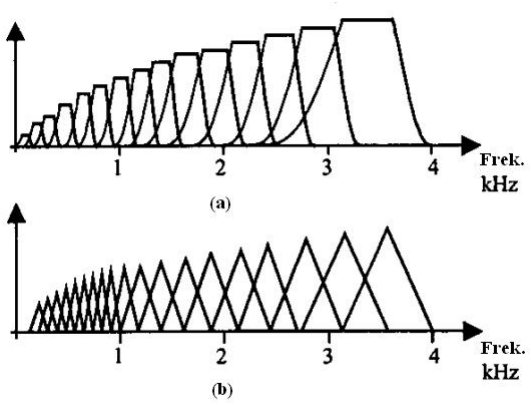
Da lahko zgornjo enačbo uporabimo, moramo definirati frekvence $f[m]$, ki imajo vlogo osrednjih frekvenc posameznih filtrov. Naj bosta f_l in f_h najmanjša in največja frekvenca v naboru filtrov, F_s frekvenca vzorčenja, M število filtrov in N velikost FFT. Frekvence $f[m]$ so enakomerno porazdeljene na mel-skali:

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_1) + m\frac{B(f_h) - B(f_l)}{M+1}\right), \tag{2.20}$$

kjer je uporabljen inverz mel-skale, ki je podan z

$$B^{-1}(b) = 700(e^{b/1125} - 1). \tag{2.21}$$

Ta korak ustreza uporabi analize kritičnih pasov v PLP. Razlike so v natančni obliki uporabe logaritemske skale za frekvenco in v obliki uporabljenih filtrov. Mel-frekvenčni filtri so trikotne oblike, medtem ko so filtri pri kritičnih pasovih nekoliko podobni trapezu. Primerjava med oblikami filtrov je na sliki 2.6.



Slika 2.6: Primerjava med oblikami (a) filtrov kritičnih pasov in (b) mel-frekvenčnih filtrov [43].

Logaritmiranje

V naslednjem koraku dobljene izhode filtrov logaritmiramo in tako dobimo za m-ti filter:

$$S[m] = \ln \left[\sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k] \right] \qquad 0 \leq m < M. \tag{2.22}$$

S tem upoštevamo nelinearno občutljivost ušesa na intenziteto zvoka podobno kot v PLP s pretvorbo med intenziteto in glasnostjo.

Diskretna kosinusna transformacija

Mel-frekvenčni kepster dobimo s kosinusno transformacijo v spektru više ležečih logaritmiranih energij:

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos \left(\pi n \frac{m + \frac{1}{2}}{M} \right) \qquad 0 \leq n < M, \tag{2.23}$$

kjer za M izbiramo vrednosti med 24 in 40. Pri razpoznavanju govora običajno uporabimo prvih 13 koeficientov.

2.2 Akustično modeliranje

Prevladujoča tipa akustičnih modelov sta prikriti modeli Markova (HMM) [17, 52] in končni pretvorniki (WFST) [44]. Z akustičnimi modeli modeliramo akustične značilnosti fonemov jezika. V govoru pa se isti fonemi akustično razlikujejo glede na njihov kontekst – sosednja fonema. Zato v modeliranju govora uporabljamo t. i. trifonske modele, ki upoštevajo kontekst fonema. Trifonski modeli so v osnovi ločeni za vse možne trojice fonemov. Med samim učenjem pa zaradi podobnosti določene trifonske modele združujemo. Poznamo tudi sisteme, kjer kot osnovno akustično enoto uporabljamo cele besede, vendar so takšni sistemi primerni le za razpoznavanje z majhnim slovarjem [40].

2.2.1 Prikriti modeli Markova

Prikriti modeli Markova (HMM) so definirani z množico stanj, verjetnostmi prehodov med stanji in verjetnostmi porazdelitvami izhodnih simbolov.

Najprej poglejmo osnovno obliko prikritega modela Markova. Naj bo N neko naravno število in $S = S_1, S_2, ..., S_N$ množica, ki jo imenujemo stanja HMM. S $t = 1, 2, 3, ...$ označujemo časovne točke, v katerih HMM prehaja med stanji. Zaporedje $q_1, q_2, ..., q_t, ...$ naj označuje zaporedje stanj, skozi katera prehaja HMM, tako da q_i pomeni stanje, v

katerem je HMM ob času i. Definiramo verjetnost prehoda iz stanja i v stranje j: $a_{ij}(t) = P[q_t = S_j|q_{t-1} = S_i]$. Stanjem priredimo tudi začetno verjetnostno porazdelitev v obliki vektorja $\pi = (\pi_1, \pi_2, ..., \pi_N)$, kjer je $\pi_i = P[q_1 = S_i]$. Strukturo, sestavljeno iz množice stanj, verjetnosti prehodov med njimi in vektorjem začetnih verjetnosti, imenujemo markovska veriga.

V homogeni markovski verigi in homogenem HMM velja, da so te verjetnosti neodvisne od časa t, zato lahko pišemo le preprosto a_{ij}. Te verjetnosti zapišemo v obliki matrike $A = \{a_{ij}\}$.

Markovsko verigo razširimo do prikritega modela Markova tako, da vsakemu stanju priredimo tudi neko verjetnostno funkcijo p_i za zvezno naključno spremenljivko z zalogo vrednosti M. Ta naključna spremenljivka predstavlja neka opazovanja, medtem ko prehajanj med stanji ne moremo neposredno opazovati. Proces tvorjenja opazovanj imenujemo tudi vidni proces, medtem ko je proces prehajanja stanj prikrit (od tod izhaja ime prikriti modeli Markova). Z $B = \{p_1, p_2, ..., p_N\}$ označimo množico verjetnostih porazdelitev za vsa stanja. Tako je prikriti model Markova podan z naslednjimi parametri:

- številom stanj N,
- verjetnostnim prostorom vidnega procesa M,
- matriko verjetnosti prehajanja stanj A,
- množico verjetnostnih porazdelitev vidnega procesa B in
- vektorjem začetnih verjetnosti π.

Glede na to, ali je prostor M zvezen oz. diskreten (in s tem tudi glede na to, ali so verjetnostne porazdelitve v množici B zvezne oz. diskretne), imenujemo tak model zvezni oz. diskretni prikriti model Markova. V razpoznavanju govora se uporabljajo zvezni prikriti modeli Markova, kjer je M večdimenzionalni realni vektorski prostor, porazdelitve v B pa so večdimenzionalne Gaussove.

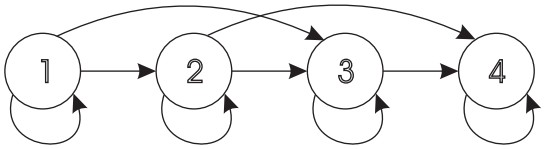
V vseh zgoraj omenjenih časovnih točkah zaseda HMM neko stanje, v katerem odda eno realizacijo naključne spremenljivke iz množice M, ki jo lahko opazujemo. Potem preide HMM v neko novo (morda isto) stanje glede na matriko prehodov A.

V razpoznavanju govora uporabljamo levo-desne HMM, za katere velja

$$a_{ij} = 0, \ \forall i > j. \tag{2.24}$$

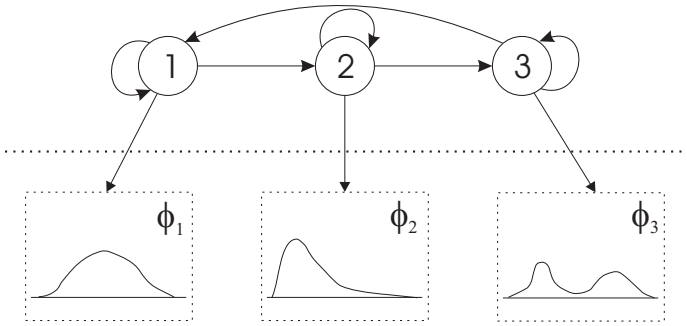
To pomeni, da lahko model prehaja med stanji le v eno smer. Ko model neko stanje zapusti, se ne more več v njega vrniti. Primer takega modela je narisan na sliki 2.7. Dovoljeni prehodi so označeni s puščicami, vsi ostali prehodi pa imajo verjetnost enako

0. Na sliki vidimo, da so dovoljeni prehodi le iz nekega stanja spet vase ali pa v eno od naslednjih stanj. Če v takšnem modelu prepovemo tudi prehode iz stanja v samega sebe, imenujemo ta model strogo levo-desni.



Slika 2.7: Možni prehodi med stanji v levo-desnem HMM.

Če ima v splošnem HMM N stanj, lahko prehodne verjetnosti opišemo z $N^2 - N$ parametri. Število parametrov, ki jih potrebujemo za izhodne verjetnosti, je odvisno od vrste porazdelitve izhodne spremenljivke. Če ima ta na primer enodimenzionalno Gaussovo porazdelitev, potrebujemo 2 parametra (srednja vrednost in varianca). Primer HMM, ki vključuje tudi ponazoritev izhodnih porazdelitev, je na sliki 2.8. Da ga natančno definiramo, moramo podati 6 prehodnih verjetnosti in vse parametre, ki so potrebni za definicijo treh izhodnih porazdelitev.



Slika 2.8: Primer HMM s tremi stanji.

HMM tvori v vsakem stanju eno vrednost izhodne naključne spremenljivke glede na verjetnostno porazdelitev, ki pripada temu stanju. Kadar uporabljamo HMM-e v akustičnem modeliranju govora, te izhodne spremenljivke ustrezajo vektorjem značilk. Običajno se uporabljajo večdimenzionalne mešane Gaussove porazdelitve značilk. To so porazdelitve, ki so definirane kot utežene vsote (navadnih) večdimenzionalnih Gaussovih porazdelitev. Za opis take porazdelitve moramo podati za vsako komponento porazdelitve utež, srednji vektor in kovariančno matriko.

V razpoznavanju tekočega govora z velikim slovarjem se uporabljajo trifonski akustični modeli. Ti so sestavljeni iz treh različnih stanj. Za vsak fonem jezika v katerem koli kontekstu (predhodni in naslednji fonem) definiramo en akustični model. Za jezik z N fonemi pomeni to N^3 trifonskih modelov. K temu se kasneje prištejejo še modeli za šum oz. tišino in kratko pavzo. Zaradi podobnosti med modeli glede na različen kontekst se nekateri modeli kasneje združijo. Kljub temu običajno ostane število različnih trifonskih modelov veliko.

Vse te modele lahko opišemo, če v vsakem modelu za vsako stanje podamo prehodne verjetnosti do ostalih stanj in parametre za opis večdimenzionalne verjetnostne porazdelitve vektorja značilk. Zaradi velikega števila posameznih trifonskih modelov in velike dimenzionalnosti značilk to pomeni ogromno število parametrov, ki so potrebni za opis celotnega akustičnega modela. Zato za dobro ocenjevanje teh parametrov potrebujemo čim večjo učno množico.

2.2.2 Učenje prikritih modelov Markova

Naloga učenja modelov je določiti njihove parametre λ glede na podane učne podatke $(\mathcal{X}, \mathcal{L})$. Tukaj smo z $\mathcal{X}$ označili množico zvočnih posnetkov, z $\mathcal{L}$ pa množico pripadajočih transkripcij govora. Najpogostejša cenilka za učenje modelov je cenilka največje verjetnosti (MLE). Kriterijska funkcija za MLE je

$$F_{MLE}(\Lambda) = \frac{1}{T}\sum_{t=1}^{T}\log(p(X_t|S_t,\Lambda)). \tag{2.25}$$

Glede na njo so optimalni parametri takšnega modela določeni z

$$\tilde{\Lambda}_{MLE} = \arg\max_{\lambda} p(\mathcal{X}|\mathcal{L},\Lambda) = \arg\max_{\lambda} \prod_{t=1}^{T} p(X_t|S_t,\Lambda). \tag{2.26}$$

MLE je dobro znan in velikokrat uporabljen kriterij za učenje modelov. Parametre ocenjujemo z Baum-Welchevim algoritmom [17, 33]. MLE modele nauči tako, da se maksimirajo pogojne verjetnosti opazovanj glede na razrede. Pri učenju parametrov nekega modela se tako upoštevajo le akustične značilnosti vzorcev iz pripadajočega razreda, ne pa tudi značilnosti iz drugih razredov.

To pomeni, da kadar ocenjujemo parametre nekega trifona, upoštevamo le njegove akustične značilnosti. Kadar pa uporabljamo diskriminativno učenje, pa ne le poskušamo maksimirati verjetnost glede na pravilno govorno enoto, ampak istočasno tudi minimizirati verjetnosti glede na napačne govorne enote. Prednost diskriminativnega učenja je tudi ta, da za njegovo delovanje ni potrebna predpostavka o pravilnosti modelov [60, 67].

2.2.3 Končni pretvorniki

Pristop, ki se je v zadnjih letih uveljavil pri razpoznavanju govora s statičnim razširjanjem, je uporaba končnih pretvornikov. Pretvorniki so končne mreže, ki povezujejo vhodne in izhodne simbole na povezavah, ki so lahko utežene. Z njimi vključujemo vse modele, ki so nam na razpolago. Lahko jih združujemo z operacijo kompozituma. Tako lahko iz pretvornikov, ki predstavljajo fonetične modele, slovar in jezikovni model, konstruiramo skupni pretvornik za razpoznavanje govora.

Končni pretvorniki preslikajo zaporedje vhodnih simbolov v zaporedje izhodnih simbolov in mu pri tem dodelijo uteži. Razpoznavanje govora poteka tako, da poiščemo izhodno zaporedje, ki ima najmanjšo utež [11].

Glavne značilnosti končnih pretvornikov so, da lahko uporabljamo modele fleksibilno in neodvisno od posebnosti razpoznavalnika, da je končna optimizirana mreža nekajkrat večja od prvotnega jezikovnega modela in da modeliranje medbesednega konteksta poveča velikost mreže le malo. Prednost končnih pretvornikov je tudi ta, da so vsi modeli kombinirani v eno mrežo, kar pomeni, da med samim razpoznavanjem ni potrebno kombinirati različnih modelov.

2.3 Jezikovno modeliranje

V splošnem poznamo dve vrsti jezikovnih modelov [29]. Prvi vsebujejo formalna pravila, besede in fraze, ki jih je potrebno ročno vnesti v sistem. Takšni modeli ocenjujejo niz besed le kot napačen ali pa pravilen. V razpoznavanju tekočega govora z velikim slovarjem takšni modeli niso uporabni, saj ni možno ročno vnesti dovolj pravil in možnih stavkov, s katerimi bi dosegli zadovoljivo pokritost vseh možnih stavkov, ki se lahko v nekem jeziku pojavijo. Druga vrsta jezikovnih modelov so statistični.

S pomočjo statističnih jezikovnih modelov [32, 68, 72] računamo verjetnost, da se določeno zaporedje besed pojavi v danem jeziku. Rečemo, da je model naučen na tem jeziku. Najpogosteje uporabljeni jezikovni modeli so besedni n-gramski modeli. Ti modelirajo verjetnost neke besede na podlagi predhodnih $n-1$ besed. Za njihovo učenje so potrebni učni korpusi – elektronske zbirke besedil v jeziku, ki ga modeliramo. V jezikovnih modelih uporabljamo postopke glajenja [10, 21] in sestopanja [34].

2.3.1 Besedni n-gramski modeli

Statistični jezikovni model priredi nizu besed $W = (w_1, w_2, ..., w_N)$ verjetnost, da se ta niz besed pojavi v danem jeziku

$$P(W) = P(w_1, w_2, ..., w_N). \tag{2.27}$$

Kot vse statistične metode temelji ta model na opazovanju vzorca jezika, ki ga imenujemo jezikovni korpus. Enostavna metoda, s katero bi lahko tak model zgradili, je preštevanje vseh možnih različnih stavkov v korpusu. Zaradi velikega števila besed in možnosti njihovega sestavljanja v stavke ter omejenih velikosti jezikovnih korpusov taka metoda ne bi delovala. Vzrok za to je premajhno število učnih podatkov.

Da lahko sploh uporabljamo statistične modele, sprejmemo dve predpostavki. Prva je Markovova predpostavka, ki pravi, da je verjetnost, da se na določenem delu stavka pojavi neka beseda, odvisna samo od zadnjih nekaj besed pred to besedo.

S pomočjo verižnega pravila računanja verjetnosti lahko verjetnost iz prejšnje enačbe zapišemo v faktorizirani obliki pogojnih verjetnosti

$$P(w_1, w_2, ..., w_N) = \prod_{i=1}^{N} P(w_i|w_1, ..., w_{i-1}). \tag{2.28}$$

Na tem mestu uporabimo Markovovo predpostavko in rečemo, naj bo pogojna verjetnost besed odvisna samo od predhodnih $n-1$ besed. Formalno to izrazimo kot

$$P(w_i|w_1, ..., w_{i-1}) = P(w_i|w_{i-n+1}, ..., w_{i-1}). \tag{2.29}$$

Zgornjo verjetnost lahko sedaj zapišemo kot

$$P(W) = \prod_{i=1}^{N} P(w_i|w_{i-n+1}, ..., w_{i-1}). \tag{2.30}$$

Te nize n besed imenujemo n-grami, jezikovne modele, ki jih na takšen način sestavimo, pa n-gramski modeli.

Druga predpostavka je predpostavka o stacionarnosti, ki pravi, da naj bodo verjetnosti $P(w_i|w_{i-n+1}, ..., w_{i-1})$ neodvisne od števila i. Enostavno povedano to pomeni, da je verjetnost, da neka beseda sledi danemu nizu besed, neodvisna od tega, ali se ta niz besed pojavi na začetku stavka, na koncu stavka ali kje vmes.

Verjetnosti v modelu sedaj lahko računamo z enačbo ocenjevanja po statistični cenilki MLE:

$$P_{MLE}(w_i|w_{i-n+1}, ..., w_{i-1}) = \frac{C(w_{i-n+1}, ..., w_i)}{C(w_{i-n+1}, ..., w_{i-1})}, \tag{2.31}$$

kjer je $C(W)$ funkcija, ki pove, kolikokrat se niz besed W pojavi v učnem korpusu.

S tema dvema predpostavkama omogočimo boljše ocenjevanje parametrov modela, saj je vseh možnih zaporedij nizov n besed veliko manj kot pa vseh možnih stavkov. Kljub temu se lahko pojavijo v jeziku zaporedja besed, ki jih ni v učnem korpusu. Da preprečimo, da bi se tem zaporedjem besed kasneje pripisala verjetnost 0, uporabimo metode glajenja in algoritme sestopanja.

2.3.2 Glajenje in sestopanje

Najpogosteje se v razpoznavanju govora uporabljajo bigramski in trigramski modeli. Če vzamemo slovar s 100.000 besedami, bi za popoln opis bigramskega modela morali oceniti

10 milijard parametrov, za trigramski model pa celo 10^{15} parametrov. V učnih korpusih pa se velika večina vseh možnih n-gramov sploh ne pojavi, veliko ostalih pa se pojavi le enkrat ali pa zelo malokrat, kar pa je premalo za dobro statistično ocenjevanje verjetnosti.

Z glajenjem spremenimo verjetnosti n-gramov v modelu. Najenostavnejše glajenje dosežemo tako, da predpostavimo, da se je vsak n-gram pojavil v učni množici enkrat več, kot pa smo prešteli. Zgornja enačba dobi tako obliko

$$P_{glajena}(w_i|w_{i-n+1},...,w_{i-1}) = \frac{C(w_{i-n+1},...,w_i)+1}{C(w_{i-n+1},...,w_{i-1})+1}. \tag{2.32}$$

S tem smo dosegli osnovni cilj glajenja. Vsem n-gramom se tako priredi verjetnost, večja od 0.

Takšna preprosta tehnika glajenja slabo deluje, zato so bili s časom razviti različni drugi algoritmi glajenja. Nekateri med njimi so: aditivno glajenje, linearna interpolacija, Good-Turingovo, Jelinek-Mercerjevo, Witten-Bellovo, Kneser-Neyevo in modificirano Kneser-Neyevo glajenje ter druge [10].

Drugi osnovni pristop je uporaba algoritma sestopanja [34]. Kadar v hipotezi opazimo neki n-gram, ki se je premalokrat pojavil v učnem korpusu in zanj posledično nimamo dobre ocene verjetnosti, se pri tem pristopu umaknemo na bolj preprost n-gramski model. Na primer: za ocenjevanje hipoteze uporabljamo trigramski model. V hipotezi najdemo zaporedje treh besed, za katero v modelu nimamo ocene verjetnosti. Na tem mestu zavržemo prvo besedo in v modelu iščemo bigram, ki ustreza preostalima dvema besedama.

V splošnem lahko tako postopek zapišemo kot

$$\begin{aligned} &P_{BO}(w_i|w_{i-n+1},...,w_{i-1}) = \\ &= \begin{cases} d_{N(w_i,...,w_{i-1})}P(w_i|w_{i-n+1},...,w_{i-1}) & ; \quad C(w_{i-n+1},...,w_{i-1},w_i) > N_n \\ \alpha(w_{i-n+1},...,w_{i-1})P_{BO}(w_i|w_{i-n+2},...,w_{i-1}) & ; \quad \text{sicer} \end{cases}, \end{aligned} \tag{2.33}$$

kjer je N_n mejno število n-gramov za glajenje.

Sestopanje lahko poteka v več korakih. V primeru štirigramskega modela tako preidemo najprej na trigramskega, nato na bigramskega in v zadnjem koraku lahko še unigramski model, v katerem gledamo le besede kot neodvisne od predhodnih besed. V tem klasičnem postopku sestopanja smo na vsakem koraku zavrgli besedo, ki je bila najdlje oddaljena od trenutne besede.

2.3.3 Perpleksnost

Ko izdelamo jezikovni model, nas zanima, koliko le-ta ustreza jeziku. Najpogostejša mera, ki se pri tem uporablja, je perpleksnost. Njena definicija izhaja iz teorije informacij. Na jezik lahko gledamo kot na informacijski vir, ki daje kot izhode besede iz slovarja jezika [29]. Za izračun perpleksnosti potrebujemo testno množico stavkov iz jezika, za katerega smo izdelali model. Verjetnost nekega stavka W iz testne množice lahko izračunamo po enačbi 2.30. Potrebne podatke o verjetnosti n-gramov dobimo iz jezikovnega modela. Verjetnost celotne testne množice stavkov $\mathcal{W}$ dobimo kot produkt

$$P(\mathcal{W}) = \prod_{i=1}^{T} P(W_i). \tag{2.34}$$

Križno entropijo modela zapišemo kot

$$H_P(\mathcal{W}) = -\frac{1}{N_T} \log_2 P(\mathcal{W}), \tag{2.35}$$

kjer je N_T število vseh besed v testni množici. Perpleksnost modela je obratna vrednost geometrijske sredine verjetnosti, ki jih je model priredil besedam v testni množici. Izračunamo jo s pomočjo križne entropije:

$$PPL_P(\mathcal{W}) = 2^{H_P(\mathcal{W})}. \tag{2.36}$$

Nižje vrednosti perpleksnosti pomenijo, da je model na testni množici besedam prirejal večje verjetnosti, kar pomeni, da se je tak model bolje prilegal testnim podatkom, kot pa bi se model z večjo perpleksnostjo. Večkrat je bilo pokazano tudi, da so jezikovni modeli z manjšo perpleksnostjo tudi uspešnejši pri razpoznavanju govora [10]. Pri izračunu perpleksnosti običajno velja, da dajejo modeli višjih redov manjše vrednosti perpleksnosti kot pa modeli nižjih redov.

2.4 Iskalni algoritmi

2.4.1 Naloge iskalnega algoritma

Naloga iskalnega algoritma je poiskati niz besed $\hat{W} = w_1, ..., w_n$ neznane dolžine n, za katerega velja:

$$\hat{W} = \arg\max_{W} \{P(O|W)P(W)\}, \tag{2.37}$$

kjer je $O = o_1, ..., o_T$ zaporedje vektorjev značilk in kjer W poteka po vseh nizih besed različnih dolžin iz slovarja z N_W besedami. Faktor $P(O|W)$ predstavlja verjetnost, da se ob izgovorjavi niza besed W pojavi zaporedje vektorjev značilk O. Ta faktor določimo s

pomočjo akustičnih modelov in fonetičnih transkripcij. Drugi faktor $P(W)$ pa predstavlja verjetnost, da se v govoru pojavi niz besed W. Določimo jo s pomočjo jezikovnega modela.

Vpeljimo oznako o_1^T za zaporedje vektorjev značilk dolžine T. Naj s_1^T označuje zaporedje stanj v HMM, ki je prav tako dolžine T, niz besed pa označujmo z w_1^n. V enačbi 2.37 lahko faktor $P(O|W) = P(o_1^T|w_1^n)$ zapišemo kot vsoto verjetnosti, ki jih prispevajo vsa možna zaporedja stanj HMM. Tako dobimo enačbo

$$\hat{w}_1^n = \arg\max_{w_1^n} \left\{ P(w_1^n) \cdot \sum_{s_1^T} P(o_1^T, s_1^T|w_1^n) \right\}. \tag{2.38}$$

Na tem mestu lahko uporabimo Viterbijev kriterij, ki nam pove, da lahko vsoto vseh verjetnosti zamenjamo z največjo med njimi. Dodamo tudi hevristično konstanto α, s katero utežimo doprinos jezikovnega modela. Končno dobimo enačbo

$$\hat{w}_1^n = \arg\max_{w_1^n} \left\{ P(w_1^n)^\alpha \cdot \max_{s_1^T} \left\{ P(o_1^T, s_1^T|w_1^n) \right\} \right\}. \tag{2.39}$$

Naloga iskalnega algoritma je sedaj poiskati to zaporedje besed $\hat{w}_1^n$.

Verjetnost v enačbi 2.37 imenujemo tudi *maksimalna a posteriori* (MAP) verjetnost in jo izpeljemo s pomočjo Bayesovega obrazca iz pogojne verjetnosti $P(W|O)$. Tukaj poskušamo določiti zaporedje besed tako, da minimiziramo pričakovano število napačno razpoznanih nizov besed. Glede na aplikacijo, kjer uporabljamo razpoznavalnik govora, pa je včasih bolj primeren drugi pristop, ki ga imenujemo klasifikator minimalnega Bayesovega tveganja (MBR) [20]. Pri tem pristopu razlikujemo med napakami, ki jih lahko storimo pri razpoznavanju niza besed, in jih različno utežimo. Ta ideja izhaja iz narave določenih aplikacij razpoznavanja govora, kjer imajo različne napake lahko različne posledice.

Iz modelov trifonov lahko sestavimo besedne modele, iz katerih lahko dalje sestavimo modele za nize besed. Na ta način bi lahko za vsak možen niz besed sestavili model in zanj izračunali verjetnost, da je niz teh besed tvoril neko podano zaporedje vektorjev značilk. Vendar je ta metoda neuporabna, saj je število možnih besed in možnih nizov besed preveliko, da bi lahko v doglednem času za njih izračunali verjetnosti. Zato je v razpoznavanju tekočega govora pomembno, da uporabimo iskalne algoritme, katerih časovna in prostorska zahtevnost sta primerni za implementacijo in uporabo.

Zato so se razvili številni algoritmi, s katerimi iščemo najverjetnejši niz besed, ne da bi pri tem preverili vse možne kombinacije. Ti algoritmi tvorijo omejeno število delnih hipotez o izgovorjenem nizu besed, ki jih potem postopoma razširjajo. Na koncu podajo najverjetnejše zaporedje besed. Za te algoritme je pomembna hitrost, da lahko delujejo v realnem času. Pomembno je tudi, da se pojavi malo iskalnih napak, pri katerih algoritem vrne zaporedje besed, ki ni najverjetnejše, ker je prej zavrnil pravilno hipotezo kot malo verjetno.

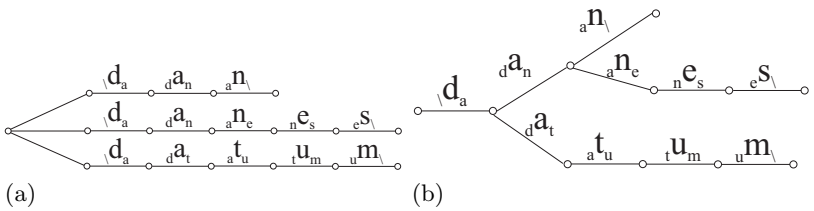
Tako akustični kot tudi jezikovni model uporabljata omejeno število informacij za določanje verjetnosti tvorjenja vektorjev akustičnih značilk oz. zaporedja besed. Lahko rečemo, da imata "kratek spomin". Ta lastnost modelov omogoča uporabo principa rekombinacije. Ta pravi, da kadar med iskanjem naletimo na več delnih poti, ki bi v nadaljevanju imele enak doprinos pri verjetnosti, lahko zanemarimo vse razen najverjetnejše poti, saj druge poti ne morejo v nadaljevanju spet postati bolj verjetne. Na ta način zmanjšamo število hipotez, ki jih preverjamo med iskanjem najverjetnejše poti.

Osnovne naloge iskalnega algoritma so torej:

- tvorjenje hipotez nizov besed,
- ocenjevanje vseh aktivnih hipotez,
- rekombinacija delnih poti,
- zavračanje malo verjetnih hipotez,
- tvorjenje končnega niza besed.

2.4.2 Slovarska drevesa

Slovar besed lahko organiziramo na dva načina: v obliki linearnega slovarja in v obliki drevesnega slovarja. Oba načina sta prikazana na sliki 2.9a. V obeh načinih predstavimo iskalni prostor kot vozlišča in povezave, ki predstavljajo trifonske modele.



Slika 2.9: Ureditev iskalnega prostora z slovar z besedami: *danes, datum* in *dan* (a) linearni iskalni prostor, (b) drevesni iskalni prostor.

V linearnem slovarju so besedni modeli prestavljeni ločeno eden od drugega. Tako so v iskalni mreži povezane vstopne in izstopne točke besed. Ta način organizacije je primeren za aplikacije razpoznavanja govora izoliranih besed z majhnim slovarjem. Pri tem načinu iskanja lahko vključimo prispevek jezikovnega modela že na začetku besede, saj je že tukaj le-ta enolično določen.

V razpoznavanju govora z velikim slovarjem uporabljamo organizacijo slovarja v obliki drevesa. V njem so besede organizirane tako, da imajo skupna vozlišča za skupne foneme v svojih izgovorjavah (glej sliko 2.9b). Uporaba take strukture omogoča zmanjšanje redundance na začetku iskanja ter s tem zmanjšanje števila aktivnih iskalnih hipotez in

potrebnega časa za iskanje. Pri drevesni strukturi iskalnega prostora so besede določene šele v končnih vozliščih in šele takrat lahko uporabimo jezikovni model. Možna je tudi uporaba kombiniranega linearnega in drevesnega slovarja [38].

2.4.3 Razdelitev iskalnih algoritmov

Iskalne algoritme lahko delimo glede na različne lastnosti [4]. Ena izmed njih je razširitev iskalnega prostora. Tako poznamo algoritme, ki uporabljajo statično razširjen iskalni prostor, in algoritme, ki uporabljajo dinamično razširjen iskalni prostor. Druga lastnost je časovna sinhronost iskanja, po kateri algoritme delimo na časovno sinhrone in časovno asinhrone. Značilnosti algoritmov glede na ti delitvi si bomo ogledali v naslednjih dveh podpoglavjih.

V primerih statičnega razširjanja iskalnega prostora lahko uporabimo različne sinhrone ali pa asinhrone algoritme, v praksi pa se običajno uporablja sinhroni Viterbijev razpoznavalnik. V primeru dinamičnega razširjanja pa je izbira sinhronega ali asinhronega algoritma odločilna. Sinhrone algoritme delimo dalje na besedno pogojene in časovno pogojene. Asinhrone algoritme pa delimo na enoskladovne in večskladovne.

2.4.4 Statično razširjanje iskalnega prostora

Pod pojmom razširjanja iskalnega prostora razumemo način, kako iz osnovnih HMM gradimo večje modele, v katerih ocenjujemo hipoteze nizov razpoznanih besed. Osnovna pristopa sta statično razširjanje, kjer iskalni prostor razširimo že pred iskanjem, in dinamično razširjanje, kjer ga razširjamo šele med samim iskanjem.

Statično razširjanje iskalnega prostora je bil dokaj naraven način dela, preden so zaradi vedno večjih slovarjev postale pomembne prostorske omejitve. Če se hočemo izogniti dinamičnemu razširjanju, imamo na voljo nekaj optimizacijskih metod, s katerimi lahko iskalni prostor predstavimo bolj kompaktno.

Izkoristimo lahko pičlost (angl. sparsity) uporabljenih modelov. Ta se pojavi zaradi pomanjkljivih učnih podatkov. Pičlost se pozna posebej pri n-gramskih jezikovnih modelih, kjer je n večji od 2, in medbesednih kontekstih. Kadar imamo neki model z velikim slovarjem, se pojavi ogromna množica n-gramov, od katerih se večina ne bo pojavila v učni množici.

Redundance se pojavijo zaradi gradnje večjih modelov iz podobnih manjših enot, kot so trifonski HMM, oblikovanja iskalnega prostora v drevesni obliki in podvojitev modelov zaradi vezanja parametrov.

Prednost statičnega razširjanja iskalnega prostora je tudi enostavna faktorizacija verjetnosti n-gramov, ki jih lahko razdelimo po zaporedju fonemov. S tem dobimo veliko

redundantnih poti, ki jih lahko združimo in skupaj obravnavamo za vse kontekste, v katerih se beseda nahaja.

2.4.5 Dinamično razširjanje iskalnega prostora

Ideja uporabe dinamično razširjenega iskalnega prostora je postala pomembna ob hitrem večanju slovarjev v aplikacijah razpoznavanja govora in ob potrebi po procesiranju v realnem času. Prav tako je dinamično razširjanje motivirano z uporabo snopovnega omejevanja aktivnih hipotez iskanja. Glavno vodilo dinamičnega razširjanja je uporaba vseh razpoložljivih informacij za čim večjo zožitev aktivnega iskalnega prostora. Tudi tukaj uporabljamo drevesno strukturiran iskalni prostor, ki pa je shranjen samo enkrat. Algoritem ves čas upravlja le z virtualnimi vozlišči ter delnimi in dinamičnimi kopijami drevesa, s katerimi zajame aktivne hipoteze.

Pomembna lastnost pri dinamičnem razširjanju je strukturiranje iskalnega prostora glede na jezikovni model. Tukaj lahko damo poudarek jezikovnemu kontekstu neke besede ali pa času začetka besede. S tem ločimo dva pristopa uporabe drevesne strukture z n-gramskimi modeli:

Besedno pogojeno iskanje, kjer pregledamo virtualno kopijo slovarskega drevesa za vsak aktivni jezikovni kontekst. V tem pristopu delamo rekombinacije na virtualnih korenih dreves. Te so odvisne od prejšnjih besed in jezikovnega modela. Časovno pogojeno iskanje, kjer se virtualna kopija drevesa doda vedno, ko se zahteva hipoteza naslednje besede. Vse poti v aktivnih hipotezah, ki se končajo v istem trenutku, so razširjene s kopijo drevesa, ki jo v tem trenutku dodamo.

2.4.6 Sinhroni iskalni algoritmi

Pri časovno sinhronih algoritmih procesiramo niz akustičnih vektorjev od začetka do konca in pri tem vzporedno obravnavamo vse delne hipoteze. V tem primeru obravnavamo hipoteze na podlagi iste količine akustičnih informacij, zato so njihove verjetnosti med sabo primerljive. Časovno sinhrono iskanje temelji na dinamičnem programiranju in Viterbijevem algoritmu.

V vsakem časovnem okviru lahko izključimo nekatere hipoteze, ki se zdijo premalo verjetne. Verjetnostno območje preostalih hipotez imenujemo snop, tak postopek izključevanja hipotez pa snopovno omejevanje. Njegov namen je zmanjšati iskalni prostor in s tem pohitriti algoritem. Nivo omejevanja lahko določimo na različne načine. Ker s snopovnim omejevanjem izključujemo hipoteze iz iskanja, ne da bi pri tem zagotovili, da so napačne, se lahko med razpoznavanjem pojavijo iskalne napake. Z ustrezno nastavitvijo nivoja omejevanja lahko število iskalnih napak močno znižamo.

Besedno pogojeno sinhrono iskanje je eno najpogostejših v razpoznavanju tekočega govora z velikim slovarjem. Da lahko v tak algoritem pravilno vključimo n-gramski jezikovni model, moramo za vsako aktivno hipotezo beležiti preteklih $n-1$ besed vse do rekombinacije. Raziskana sta bila dva pristopa, kako zadostiti optimalnosti n-gramov. V prvem pristopu, imenovanem besedno pogojena organizacija z dinamičnim programiranjem, so aktivne hipoteze zabeležene v trinivojskih seznamih. V prvem je vodilna spremenljivka, to je zgodovina besed ali stanje jezikovnega modela, v drugem je zabeležen trenutni lok v slovarskem drevesu, v tretjem pa trenutno stanje HMM.

Drugi pristop, imenovan skladovna organizacija po stanjih, se od prvega razlikuje v glavnem po zaporedju, v katerem je zgrajena hierarhija beleženja hipotez. Tukaj je prva spremenljivka lok v slovarskem drevesu, druga je stanje HMM, tretja pa stanje jezikovnega modela. Ta pristop omogoča večjo fleksibilnost pri omejevanju na nivoju stanj HMM.

V časovno sinhronem iskanju pridemo do modeliranja besed v medbesednem kontekstu. To pomeni, da moramo za vsako besedo upoštevati različne kontekste, v katerih se lahko pojavi zadnji fonem glede na naslednjo besedo. Tako damo zadnjemu fonemskemu loku več instanc. Kadar preidemo v hipotezi na naslednjo besedo, moramo glede na njo izbrati ustrezni lok iz prejšnje besede.

V časovno pogojenem sinhronem iskanju [4, 51] uporabljamo razširitev z naslednjo besedo na vseh poteh, ki se končajo ob nekem času, vendar z različnimi zgodovinami besednega modela. S tem je iskalni prostor strukturiran na času začetka besede, optimizacija mej med besedami pa se opravi na vsakem koncu besed. Rekombinacija koncev besed je v tem primeru zahtevnejša, vendar to kompenziramo z velikostjo aktivnega iskalnega prostora. Ta metoda je manj primerna za vključevanje medbesednega konteksta in omejevanja z vpogledom naprej v jezikovnem modelu.

2.4.7 Asinhroni iskalni algoritmi

Med časovno asinhrone algoritme v razpoznavanju govora spadajo različni skladovni algoritmi, ki predstavljajo iskanje po principu prvi najboljši. Sklad je struktura, ki vsebuje delne hipoteze, urejene po njihovi verjetnosti. Različne hipoteze, ki se lahko končajo ob različnih časih, razširjamo besedo po besedo, tako da iz sklada vzamemo najverjetnejšo hipotezo in jo razširimo z najverjetnejšo naslednjo besedo. Hipoteze so pri tem shranjene v enem ali več skladih. Tako ločimo enoskladovne in večskladovne algoritme.

Prvo vprašanje je, kako določiti, katera hipoteza je najverjetnejša in s tem katero bomo razširjali. Določanje verjetnosti je v tem primeru težavno, ker niso za vse hipoteze uporabljene iste količine akustične informacije. Uporabljamo hevristične metode, ki so odvisne predvsem od informacij o še nerazpoznanem delu govora. Z uporabo večprehodnih metod lahko s prvim iskanjem dobimo približek za verjetnost preostalega dela, v enoprehodnem razpoznavalniku pa lahko ocenimo verjetnost z vpogledom naprej. Potem moramo

določiti, s katero besedo bomo hipotezo razširili. To lahko storimo s pomočjo časovno pogojenega sinhronega iskanja. Drugi način pa je uporaba algoritma hitrega ujemanja (angl. fast-match), s katerim dobimo kratek seznam kandidatov, ki jih posebej ocenimo.

Drugi problem pri uporabi asinhronega iskanja je določitev meje omejevanja. Tukaj je težava spet ta, da hipoteze med seboj niso neposredno primerljive zaradi različnih količin akustične informacije. Tukaj postopoma osvežujemo najboljše možne ocene s potjo, ki ima celotne besede za razširitve. To pripelje do principa ovojnice, ki je definirana kot najnižja zgornja meja ocen posameznih poti do sedaj. Glede na to ovojnico lahko določimo, katere hipoteze bomo označili kot aktivne in katere ne. To odločitev pa lahko kasneje spremenimo.

Medbesedni fonetični kontekst vključimo v dveh korakih. Najprej upoštevamo le leve kontekste, po razširitvi z naslednjo besedo pa tudi desne kontekste.

2.4.8 Hevristične metode

Hevristično pomeni, da iščemo približne metode, ki pa ne zagotavljajo več, da najdemo pravilno rešitev glede na uporabljene modele. Te metode dajejo pomembno pohitritev algoritmov z majhnim poslabšanjem uspešnosti [22].

Omenili smo že eno hevristično metodo, to je snopovno omejevanje, s katero izključujemo malo verjetne hipoteze. Z njo dosežemo zmanjšanje iskalnega prostora in posledično tudi pohitritev algoritma. Čeprav je snopovno omejevanje običajno res hevristična metoda, obstajajo metode, s katerimi lahko omejevanje optimiziramo. Tako lahko vključimo funkcijo, ki določa omejevanje na podlagi značilnosti hipotez [28].

Kot pravilno rešitev smo poimenovali tisto, ki maksimira skupno verjetnost od vseh uporabljenih modelov. Odcepitev prispevka jezikovnega modela se zdi uporabna, saj je le-ta po svoji naravi drugačen od akustičnih in fonetičnih.

Kadar uporabljamo večgramske jezikovne modele, moramo hipoteze, ki vsebujejo različne pretekle besede, ločiti, vse dokler ne uporabimo verjetnosti iz jezikovnega modela, tudi če imamo v hipotezah isto besedo, ki se konča ob nekem času. Tako lahko šele po uporabi jezikovnega modela določimo optimalni začetni čas besede. V splošnem namreč v n-gramskem modelu velja, da je optimalni začetni čas besede odvisen od te besede, njenega končnega časa in $n-1$ predhodnih besed. Tukaj pa je verjetnost večja, da bodo ti časi različni le v primeru, ko so konci predhodnih besed fonetično različni. Z upoštevanjem te lastnosti lahko zmanjšamo število redundantnih izračunov.

Zakasnjena uporaba jezikovnega modela s hevristično optimizacijo mej besed temelji na ideji, da so meje med besedami odvisne od zmanjšanega konteksta – manj kot n besed. Zakasnjena uporaba tako omogoča zmanjšanje aktivnega iskalnega prostora, saj odstrani redundantne kopije n-gramov. Primer uporabe je besedno parna aproksimacija, kjer privzamemo, da je meja med besedama odvisna samo od teh dveh besed. Ocenjevanje

z jezikovnim modelom lahko opravimo pozneje z malo računske zahtevnosti, če besedne hipoteze shranimo v obliki mreže.

Ideja omejevanja z vpogledom naprej je, da s pogledom na akustični signal poskušamo izločiti čim več malo verjetnih poti še pred korakom razširitve. Če lahko to opravimo z malo računske zahtevnosti, je možno zmanjšati število hipotez, ki jih preverjamo z natančnejšim in zahtevnejšim ocenjevanjem. Tako opravljamo razpoznavanje v dveh korakih, zato rečemo tudi, da ta metoda spada med večprehodne algoritme.

Omejevanje z vpogledom naprej se že nekaj časa uporablja v časovno sinhronih algoritmih. Pri tem pogledamo akustične informacije naprej od trenutnega časovnega indeksa in jih ocenimo, da določimo možne nadaljevalne foneme. Tako ugotovimo, katere povezave v slovarskem drevesu je smiselno vključiti v naslednjo razširitev.

2.4.9 Večprehodni algoritmi

Algoritem imenujemo večprehodni, če deluje v več fazah. Običajno šele v zadnji fazi uporabimo vse razpoložljive modele za razpoznavanje govora, v prejšnjih fazah pa uporabljamo hitre in nenatančne fast-match algoritme, s katerimi poiščemo seznam možnih besed, ki so bile izgovorjene, ali pa z njim izločamo najmanj verjetne. S temi fazami omogočamo bolj ozko omejevanje hipotez v zadnji fazi in s tem pohitritev algoritmov. Ti algoritmi lahko delujejo v dveh [71] ali več [2] fazah. Pomanjkljivost takega algoritma je možnost, da se v eni izmed faz izloči najverjetnejša pot in potem pride do iskalne napake.

Poznamo dve strategiji omejevanja iskalnega prehoda pri prvih prehodih. Prva je tvorjenje seznama N najboljših hipotez. Tukaj zabeležimo N najboljših hipotez, ki smo jih našli v prvem prehodu, v drugem oziroma končnem prehodu pa jih ovrednotimo. Druga možnost je besedni graf, kjer so besedne hipoteze predstavljene kot povezave v acikličnem usmerjenem grafu. Prednost besednega grafa je kompaktnejša predstavitev hipotez.

Eden izmed teh algoritmov je omejevanje z akustičnim vpogledom naprej, ki smo ga opisali v prejšnjem razdelku. Drugi primer večprehodnega algoritma je tak, ki postopoma uporablja vedno več informacij. Alleva, Huang in Hwang [2] so predlagali algoritem, ki v treh fazah uporablja različne algoritme. V prvi fazi se uporabi Viterbijev algoritem v smeri od leve proti desni. Tako dobimo možne čase koncev besed. S pomočjo rezultatov iz prve faze v drugi fazi uporabimo Viterbijev algoritem v nasprotni smeri – od desne proti levi – in tako dobimo čase začetkov besed. V obeh fazah uporabimo tudi snopovno omejevanje. V tretji fazi delamo s skladovnim algoritmom in večgramskim jezikovnim modelom.

2.4.10 Paralelno procesiranje

Veliko procesorskega časa porabimo za izračun verjetnosti iz Gaussovih porazdelitev modelov [57]. Ti izračuni so primerni za paralelno procesiranje, saj dobimo verjetnosti neodvisno eno od druge. Tako lahko za izračun uporabljamo inštrukcije SIMD (single instruction multiple data), ki hkrati izvajajo iste računske operacije na več podatkih. Druga možnost je uporaba večjedrnih procesorjev, kjer se izračuni porazdelijo na posamezna jedra.

Seward [64] je predlagal sinhroni algoritem, v katerem je izvedel akustično razpoznavanje s HMM, ki je odcepljeno od glavnega iskalnega algoritma in primerno za paralelizacijo. V njegovi metodi je vsak HMM primerjan samo enkrat z vsakim časovnim intervalom. Rezultat primerjave je potem nenehno na voljo glavnemu iskalnemu algoritmu.

2.5 Razpoznavanje pregibnih jezikov

Iz vidika razpoznavanja govora je glavna značilnost pregibnih jezikov v primerjavi z nepregibnimi njihova velika morfološka raznolikost besed. To pomeni, da imajo ti jeziki veliko več besednih oblik nekega pojma kot pa nepregibni. Kako zapišemo neki glagol, je odvisno od slovničnih dejavnikov. Tehnike razpoznavanja govora so bile v večini prvotno razvite za angleški jezik, ki je nepregiben.

Besede v pregibnih jezikih lahko delimo na pregibne besedne vrste in nepregibne besedne vrste. Nepregibne besedne vrste so vezniki, predlogi, (delno) prislovi, členki in medmeti. Pregibne besedne vrste pa so samostalniška beseda, pridevniška beseda in glagol. Pregibni jeziki poznajo kompleksne mehanizme tvorjenja in pregibanja besed, s katerimi spreminjamo končnice besed. Tako je končnica pregibne besede odvisna od spola, sklona, števila, osebe, časa in drugih slovničnih lastnosti.

Med pregibne jezike sodijo tudi slovanski jeziki in med njimi slovenščina. Kadar hočemo s slovarjem za slovanski jezik pokriti isti delež jezikovnega korpusa kot v angleščini, rabimo približno 10-krat večji slovar [58, 70]. Ruski slovnični priročnik Zaliznyakov na primer vsebuje 160.000 besed, ki se pojavijo v 3,7 milijona oblikah, kar presega zmogljivosti današnjih razpoznavalnikov govora.

Če tehnike razpoznavanja poskušamo neposredno prenesti na pregibne jezike, se soočimo z manjšo učinkovitostjo. Če uporabimo slovar enake velikosti kot za angleški jezik, se pojavi veliko t. i. besed izven slovarja (angl. out of vocabulary – OOV). Ker se te besede ne pojavijo v slovarju, ki ga uporablja razpoznavalnik govora, pride do večjega števila nerazpoznanih ali pa napačno razpoznanih besed.

Druga lastnost pregibnih jezikov je sproščen vrstni red besed v stavku. V angleščini je običajen red stavčnih členov osebek-povedek-predmet. Ker angleške besede niso pregibne, je vrstni red besed velikokrat pomemben, da lahko sploh prepoznamo, kaj je v stavku

osebek in kaj predmet. Pregibnost besed v pregibnih jezikih pa omogoča, da osebek in povedek prepoznamo po obliki besede oziroma po njeni končnici, saj besede pregibamo tudi po sklonu. Ta lastnost pregibnih jezikov omogoča, da lahko spremenimo vrstni red besed v stavku, ne da bi spremenili njegov pomen. Ker ni več jasno definiranega zaporedja besed, se s tem zmanjša tudi učinkovitost n-gramskih jezikovnih modelov. Ti namreč temeljijo na statističnem modeliranju krajših zaporedij besed v jeziku. Ker se pojavijo ta zaporedja v več oblikah, se poveča perpleksnost jezikovnega modela in s tem se zmanjša učinkovitost napovedovanja naslednje besede.

2.5.1 Značilnosti slovenskega jezika

Slovanske jezike delimo na vzhodne, zahodne in južne slovanske jezike, med slednje spada slovenščina. Ta si večino značilnosti deli z drugimi slovanskimi jeziki. Obstajajo pa tudi posebnosti slovenskega jezika, ki jih v drugih jezikih le redko ali pa nikoli ne najdemo. Spodaj so naštete nekatere značilnosti slovenskega jezika in primerjava z angleščino.

3 spoli – moški, ženski in srednji: Angleščina pozna tudi 3 spole, vendar se ženski in moški spol le redko uporabljata. V angleščini je spol naravno določen, zato so predmeti običajno vedno srednjega spola, medtem ko je v slovenščini spol nekega pojma slovnično določen. 3 števila: Slovenski jezik pozna ednino, dvojino in množino, medtem ko angleščina pozna samo ednino in množino. Slovenski jezik pozna 4 čase – sedanjik, prihodnjik, preteklik in predpreteklik. Angleščina pozna več časov, vendar se ti tvorijo s kombinacijami glagola in drugih besed. V slovenščini pa tvorimo čase z različnimi končnicami. Podobno kot angleščina ima tudi slovenščina 3 osebe, 3 oblike stopnjevanja, dva načina glagolov in 3 glagolske naklone. Slovenski jezik pozna tudi glagolski vid, po katerem ločimo glagole na dovršne in nedovršne.

Slovenščina pozna jasno morfološko strukturo, po kateri se besede tvorijo v obliki osnova – končnica [58]. Osnova predstavlja splošni pomen besede. Po njej prepoznamo, ali gre za samostalnik, pridevnik, glagol itd. Končnica pa pove slovnične značilnosti besede (spol, število ...) in njeno vlogo v stavku (sklon).

Posebnost slovenskega jezika so tudi glasovne premene, pri katerih se pri pregibanju besede zamenja tudi črka v osnovi besede, kar še dodatno poveča število besednih oblik. Premene se pojavijo na primer pri pomanjševalnicah. Slovenščina pozna skupno do 1000 kombinacij različnih morfoloških kategorij, medtem ko jih angleščina pozna le približno 30.

Pozitivna lastnost slovenskega jezika v smislu zahtevnosti za razpoznavanje govora pa je jasna in tesna povezanost med zapisano in govorjeno besedo. Črke v zapisani besedi ustrezajo velikokrat istemu glasu v govorjeni besedi. Le redko se ista črka v besedah različno izgovarja. Večinoma so v teh primerih izgovorjave odvisne le od predhodne in

naslednje črke. V angleščini moramo velikokrat poznati celo besedo, preden lahko rečemo, kateremu glasu ustreza neka črka v njej.

2.5.2 Drugi zahtevni jeziki

Ob pregibnih jezikih so za razpoznavanje govora težavni tudi algutinativni jeziki. Glavna razlika med pregibnimi in aglutinativnimi jeziki v smislu morfološke strukture je ta, da imajo pregibni jeziki običajno dva morfema, osnovo in pregibno končnico, medtem ko imajo alglutinativni jeziki večje število morfemov v besedi. V alglutinativnih jezikih lahko besede razdelimo na osnovo in več predpon in pripon. Običajno lahko v aglutinativnih jezikih besede lažje razbijemo na posamezne morfeme kot pa v pregibnih jezikih. Primeri takih jezikov so japonski, finski, madžarski, korejski in turški.

Dodatne izzive za razpoznavanje govora predstavlja tudi arabščina. Zraven kompleksne morfologije je za njo značilna tudi velika narečna raznolikost in razlike med zapisanimi in govorjenimi besedami [36].

2.6 Faktorizirani jezikovni modeli

Faktorizirani jezikovni modeli so bili prvič uvedeni za namen izboljšanja modeliranja arabskega jezika [7, 35, 37]. Kljub temu so uporabni tudi v drugih jezikih. Arabski jezik ima kompleksno morfološko strukturo, zaradi česar obstaja veliko različnih besednih oblik. To veliko število besed povzroča velik delež besed izven slovarja. Prav tako pri takšnih jezikih težje sestavimo dober jezikovni model, saj potrebujemo zaradi večjega števila besed tudi večji jezikovni korpus, da lahko dobro ocenimo parametre modela.

2.6.1 Definicija faktoriziranih jezikovnih modelov

Da lahko definiramo faktorizirani jezikovni model [5, 7, 35, 37], najprej povejmo, kaj razumemo pod pojmom faktorji. Lahko jih razumemo kot lastnosti, prirejene besedam v korpusu. Uporabimo lahko vrsto različnih faktorjev, kot so besedna vrsta, slovnične kategorije in besedna lema. Faktor je lahko tudi beseda sama, ki ji prirejamo faktorje. Vsako besedo w_i lahko tako zapišemo kot množico K faktorjev v obliki

$$w_i = \left\{ f_i^1, f_i^2, ..., f_i^K \right\} = \{ f_i^{1:K} \}. \tag{2.40}$$

V klasičnih jezikovnih modelih ocenjujemo verjetnost besede v stavku pogojno glede na nekaj preteklih besed. V faktoriziranih modelih ocenjujemo verjetnosti glede na faktorje preteklih $n-1$ besed. To lahko zapišemo v obliki

$$P(f_i^{1:K} | f_{i-n+1}^{1:K}, ..., f_{i-1}^{1:K}). \tag{2.41}$$

Tukaj računamo verjetnost več faktorjev. Ta izraz lahko z verižnim pravilom preoblikujemo v obliko

$$P(f_i^{1:K}) = \prod_{k=1}^{K} P(f_i^k | f_i^{1:k-1}, f_{i-n+1}^{1:K}, ..., f_{i-1}^{1:K}), \tag{2.42}$$

kjer smo najprej izračunali verjetnost prvega faktorja pogojno glede na faktorje prejšnjih besed, verjetnosti naslednjih faktorjev pa pogojno glede na vse faktorje prejšnjih besed in dosedanje faktorje trenutne besede.

Faktoriziran jezikovni model je torej statistični model, ki ocenjuje verjetnost besede v nizu in verjetnosti njenih faktorjev glede na faktorje besed, ki so se v nizu pojavile pred njo.

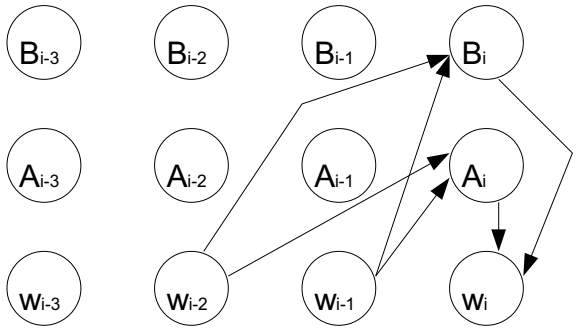
Pri gradnji faktoriziranega modela se srečamo z dvema osnovnima problemoma. Prvi je izbira primernih faktorjev za predstavitev besed v jeziku. Cilj je izbrati faktorje, ki so statistično najbolj značilni za sledeče besede in za faktorje teh besed. Faktorje lahko določimo s podatkovno vodenimi metodami ali pa ročno, z uporabo znanja o jeziku. Drugi problem je najti ustrezen jezikovni model (njegovo strukturo oz. pot sestopanja) na množici izbranih faktorjev.

Primer faktoriziranega modela je ponazorjen na sliki 2.10. Besede w_i so razporejene v vodoravni vrsti, navpično nad njimi pa so njim prirejeni faktorji. Vsaki besedi smo priredili dva faktorja A in B. Tudi beseda sama služi kot faktor. V zgornji enačbi smo zapisali najbolj splošen primer, kjer je vsak faktor, za katerega računamo verjetnost, odvisen od vseh prejšnjih faktorjev. V našem primeru pa dopuščamo le majhno število odvisnosti. Vsaka takšna odvisnost je na sliki ponazorjena s puščico. Tako je na primer faktor w_i neposredno odvisen od faktorja B_i, ne pa od faktorja w_{i-1}. Primer na sliki lahko opišemo z enačbo

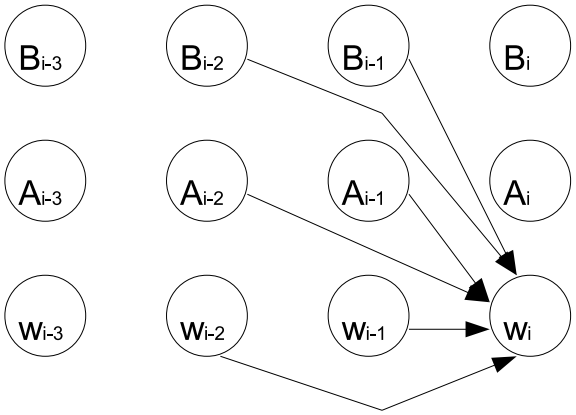
$$P(w_i) = P(w_i|A_i, B_i)P(A_i|w_{i-1}w_{i-2})P(B_i|w_{i-1}w_{i-2}). \tag{2.43}$$

Nekoliko enostavnejši primer je prikazan na sliki 2.11 in ga opišemo z enačbo

$$P(w_i) = P(w_i|w_{i-1}, w_{i-2}, A_{i-1}, A_{i-2}, B_{i-1}, B_{i-2}). \tag{2.44}$$



Slika 2.10: Primer faktoriziranega jezikovnega modela.



Slika 2.11: Enostavnejši primer faktoriziranega jezikovnega modela.

Na faktorizirane jezikovne modele lahko gledamo tudi kot na posplošitev že znanih jezikovnih modelov. Prvi primer je dejansko posplošitev razrednega jezikovnega modela na model z dvema kategorijama razredov. Če iz njega odmislimo faktorje B, lahko na faktorje A gledamo kot na besedne razrede. Tudi običajne n-gramske jezikovne modele lahko zapišemo v obliki faktoriziranega jezikovnega modela, tako da uporabimo le en faktor (besedo) in odvisnost od zadnjih $n-1$ besed.

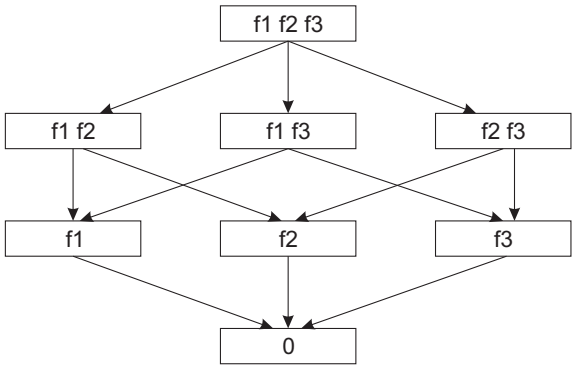
2.6.2 Posplošeni postopek sestopanja

Tako kot pri klasičnih n-gramih se lahko tudi pri faktoriziranih modelih v testni množici pojavijo nizi faktorjev, ki jih v učni množici nismo dovolj velikokrat opazili. Zato tudi pri teh modelih potrebujemo tehniki glajenja in sestopanja.

V klasičnih modelih predvidevamo, da imajo besede, ki so bližje trenutni besedi, večji vpliv nanjo. Zato med algoritmom sestopanja vedno zavržemo najbolj oddaljeno besedo. V faktoriziranih modelih pa lahko imamo več faktorjev, ki so enako oddaljeni od trenutne besede. Tukaj izbira vrstnega reda, po katerem bomo zavrgli faktorje, ni več tako enostavna. V faktoriziranih jezikovnih modelih tako uvedemo pojem posplošenega postopka sestopanja. Na sliki 2.12 imamo prikazano mrežo, katere elementi so podmnožice faktorjev. Najvišji element je množica, ki vsebuje vse faktorje. V primeru na sliki so to faktorji f_1, f_2 in f_3. Najnižji element pa je prazna množica faktorjev. Puščice prikazujejo možne korake in smeri, v katerih lahko poteka postopek sestopanja. Prikazana mreža je graf sestopanja za jezikovni model

$$P(w_i|f_1, f_2, f_3). \tag{2.45}$$

Če v tem modelu kot faktorje izberemo $f_1 = w_{i-1}$, $f_2 = w_{i-2}$ in $f_3 = w_{i-3}$ in v grafu izberemo fiksno pot sestopanja, ki na sliki poteka najbolj levo, dobimo klasični besedni štirigramski model.



Slika 2.12: Graf sestopanja za model s tremi začetnimi faktorji.

Faktorizirani modeli dopuščajo možnost, da lahko vnaprej izberemo neko fiksno pot sestopanja skozi graf. V trigramskem modelu bi tako lahko izbrali pot, v kateri izmed besed w_{i-1} in w_{i-2} najprej zavržemo besede w_{i-2}. Faktorizirani modeli dopuščajo tudi, da zavrnemo več faktorjev v enem koraku. Fiksno pot sestopanja bi lahko izbirali na podlagi znanj in predpostavk o jeziku. Če bi v modelu imeli faktorje nekaj preteklih besed, bi se verjetno odločili, da jih najprej zavračamo podobno kot v n-gramskih modelih na podlagi oddaljenosti od trenutne besede. Med faktorji, ki so enako oddaljeni, pa bi določili vrstni red morda na podlagi formalnih znanj o jeziku. Besedo (če je med faktorji), bi zavrgli prej kot pa neko slovnično lastnost (na primer sklon ali število), ki ima manj možnih vrednosti in je zato večja verjetnost, da smo v učnem korpusu našli dovolj takih nizov faktorjev.

Druga možnost, ki jo omogočajo faktorizirani modeli, pa je posplošeni postopek sestopanja, kjer pot skozi graf ni fiksna in vnaprej predpisana. Tako se pot sestopanja določa dinamično med samim ocenjevanjem stavkov. Tukaj imamo spet dve možnosti. Lahko izberemo na vsaki točki v stavku eno pot na podlagi podanih vrednosti faktorjev ali pa izberemo več poti skozi graf in na koncu dobimo več različnih verjetnosti glede na izbrane poti. Verjetnosti na koncu združimo v končno verjetnost. Tudi tukaj je izbira te množice poti lahko odvisna od konkretno podanih vrednosti faktorjev.

Posplošen postopek sestopanja za model $P(f, f_1, f_2, f_3)$ na prvem koraku (ko še imamo vse faktorje) izrazimo z enačbo

$$P_{BO}(w_i|w_{i-n+1}, ..., w_{i-1}) = \\ = \begin{cases} d_{N(w_i,...,w_{i-1})} P(w_i|w_{i-n+1}, ..., w_{i-1}) & ; \quad C(w_{i-n+1}, ..., w_{i-1}, w_i) > N_3 \\ \alpha(w_{i-n+1}, ..., w_{i-1}) P_{BO}(w_i|w_{i-n+2}, ..., w_{i-1}) & ; \quad \text{sicer} \end{cases} . \tag{2.46}$$

Tukaj je tako kot v n-gramskih modelih d_N koeficient sestopanja. Funkcija $\alpha(\cdot)$ poskrbi, da bo vsota vseh verjetnosti enaka 1. Število N_3 predstavlja mejo v številu pojavitev iskane skupine faktorjev v učni množici. Ima enako vlogo kot v n-gramskih

modelih, vendar je tukaj lahko odvisno od konkretnega nabora faktorjev.

Zgoraj naštete značilnosti posplošenega postopka sestopanja so zajete v funkciji $g(\cdot)$. V n-gramskem jezikovnem modelu ima ta funkcija obliko $g(w_i, w_{i-1}, w_{i-2}, w_{i-3}) = P(w|w_{i-1}, w_{i-2})$. V primeru fiksne poti sestopanja za faktorizirani model pa ima ta obliko $g(f, f_1, f_2, f_3) = P(f|f_x, f_y)$, kjer sta f_x in f_y dva poljubna faktorja izmed vseh faktorjev.

Obstaja več metod, kako lahko določimo izbor faktorjev in s tem izbiro funkcije g. Dve sta opisani v nadaljevanju.

- Največje število nizov faktorjev. Izberemo tisto podmnožico faktorjev, ki se je v učni množici največkrat pojavila:

$$g(f, f_1, f_2, f_3) = P(f|f_{l_1}, f_{l_2}), \tag{2.47}$$

  kjer sta

$$(l_1, l_2) = \arg \max_{(m_1,m_2)} N(f, f_{m_1}, f_{m_2}). \tag{2.48}$$

- Največje normalizirano število nizov faktorjev. V tem primeru uporabimo število pojavov niza faktorjev (vključno s faktorjem f, za katerega računamo verjetnost), normalizirano s številom pojavov pogojnih faktorjev. Tako dobimo

$$(l_1, l_2) = \arg \max_{(m_1,m_2)} \frac{N(f, f_{m_1}, f_{m_2})}{N(f_{m_1}, f_{m_2})}. \tag{2.49}$$

- Največja verjetnost vozlišča. Tukaj izberemo tisto množico faktorjev, ki dajo največjo verjetnost:

$$g(f, f_1, f_2, f_3) = P(f|f_{l_1}, f_{l_2}), \tag{2.50}$$

  kjer sta

$$(l_1, l_2) = \arg \max_{(m_1,m_2)} P(f|f_{m_1}, f_{m_2}). \tag{2.51}$$

Nadaljnje možnosti za izbiro te funkcije lahko najdemo v [37]. Tukaj najdemo tudi nekaj primerov s paralelnimi potmi sestopanja. Tukaj velja še opomniti, da so funkcije g vedno pozitivne, niso pa vedno verjetnostne (vsota ni vedno enaka 1).

Izbira začetnih faktorjev in poti sestopanja je lahko zelo zahtevna. Za ocenjevanje nekega faktorja imamo na volje vse faktorje besed, ki jih upoštevamo pri ocenjevanju verjetnosti. Če je teh faktorjev k, potem imamo na voljo $\sum_{n=1}^{k} \binom{n}{k}$ možnih podmnožic začetnih faktorjev modela. Za m začetnih faktorjev pa obstaja m! možnih poti sestopanja. Tukaj smo upoštevali, da zavržemo na vsakem koraku le en faktor. Če še upoštevamo, da imamo na vsakem koraku na voljo več različnih tehnik glajenja, vidimo, da s številom začetnih faktorjev število možnih izbir za postopek sestopanja hitro narašča. Za izbiro primernih faktorjev in poti sestopanja se lahko uporablja algoritem in programska oprema, opisana v [37].

2.6.3 Uporaba faktoriziranih jezikovnih modelov

Faktorizirani modeli so se sprva uporabljali na nivoju računanja perpleksnosti. V [37] so opisani poskusi s korpusom Callhome v arabskem jeziku. Korpus je bil pripravljen s faktorji: beseda, morfološki razred, osnova, koren besede in besedni vzorec. Z običajnim bigramskim oz. trigramskim modelom je bila dobljena perpleksnost 175 oz. 173. Z različnimi faktoriziranimi modeli je bila najnižja dosežena perpleksnost 167 za bigramski model (v tem primeru je to model, ki je upošteval le faktorje ene besede) in 166 za trigramskega.

V [7] zasledimo poskuse s perpleksnostjo še na bazi Wall Street Journal (WSJ). Čeprav so bile perpleksnosti bigramskih faktoriziranih modelov boljše od klasičnega bigramskega (izboljšanje s 320 na 266), so te bile še vedno slabše od klasičnega trigramskega (258).

Faktorizirani jezikovni modeli so se uporabljali tudi na turškem jeziku. V [59] je bilo opaženo, da dajejo faktorizirani modeli boljše rezultate predvsem pri omejeni velikosti korpusa in da se njihova prednost zmanjšuje pri velikih korpusih.

V [66] so bili faktorizirani modeli prvič uporabljeni na amharščini (uradni jezik Etiopije). Žal tukaj ni bilo opaženo izboljšanje uspešnosti razpoznavanja.

V [47] je bila uporabljena kombinirana metoda faktoriziranih jezikovnih modelov in uporabe morfemskih enot za modeliranje jezika. S ponovnim ocenjevanjem N najboljših seznamov je bila dosežena izboljšava razpoznavanja tekočega govora za $0,3\%$ absolutno. Podobne rezultate zasledimo tudi v [46].

2.7 Ocenjevanje uspešnosti razpoznavanja

Za vrednotenje uspešnosti razpoznavanja običajno uporabljamo delež napačno razpoznanih besed (WER) – razmerje med številom napak in številom vseh besed. Običajno je število napak definirano kot Levenshteinova razdalja med referenčno transkripcijo in hipotezo razpoznavalnika. Delež napačno razpoznanih besed izračunamo z enačbo

$$WER = \frac{I + D + S}{N}, \tag{2.52}$$

kjer je I število vrinjenih, D število izbrisanih in S število zamenjanih besed med Levenshteinovo poravnavo. N pa predstavlja število besed v referenčni transkripciji.

Večina objav, ki so usmerjene v izboljšanje rezultatov razpoznavanja, podaja uspešnost v absolutnem ali pa relativnem izboljšanju deleža WER. Večkrat se zgodi, da so izboljšanja dokaj majhna. Da lahko preverimo, ali je neko izboljšanje res rezultat boljšega pristopa k reševanju problema in ne le naključna značilnost uporabljene testne množice, uporabljamo teste statistične signifikance (statistične značilnosti). Rezultat teh testov je

najpogosteje t. i. p-vrednost. Ta vrednost se izračuna na uporabljeni testni množici pri primerjavi dveh sistemov razpoznavanja. Izračunana p-vrednost nam pove verjetnost, da bo v naključno izbranem testnem vzorcu ob predpostavljeni ničelni hipotezi razlika med dobljenima rezultatoma enaka ali večja od razlike, ki smo jo dobili pri primerjavi naših dveh sistemov. Ničelna hipoteza v teh primerih pravi, da oba sistema delujeta enako dobro. Kadar je ta verjetnost zelo majhna, lahko sklepamo, da ničelna hipoteza ne drži.

Poenostavljeno povedano to pomeni, da kadar bo izračunana p-vrednost pod nekim pragom, lahko rečemo, da je razlika v uspešnosti razpoznavanja primerjanih sistemov statistično signifikantna. Zgoraj omenjen prag imenujemo stopnja signifikance in ima običajno vrednost 0,05.

Na področju razpoznavanja govora so bili uporabljeni različni testi statistične signifikance. Izbira testa je odvisna od aplikacije razpoznavanja. Pri razpoznavanju izoliranih besed lahko uporabljamo McNemarjev test [18]. Pri razpoznavanju tekočega govora je bila predlagana metoda bootstrap [8]. Na drugih področjih procesiranja naravnega jezika, natančneje pri strojnem prevajanju, pa tudi metoda random approximation [56].

Poglavje 3

Oblikoskladenjsko označevanje

Pod pojmom oblikoskladenjsko (MSD) označevanje razumemo pripisovanje oznak k besedam v nekem besedilu. Te oznake opisujejo slovnične lastnosti besed, kot so: besedna vrsta, spol, sklon, število, glagolska oseba in druge. Tako kot razpoznavanje govora je tudi MSD-označevanje zahtevnejša naloga v morfološko zahtevnih jezikih, pri katerih obstaja večje število možnih oznak.

MSD-označevanje se uporablja pri različnih aplikacijah na področju procesiranja naravnega jezika in mora biti zato kar se da natančno. Primeri uporabe oznak MSD so strojno prevajanje [63], lematizacija [23] in tudi razpoznavanje govora [30]. Druga področja uporabe so še razčlenjevanje pomena besed in razumevanje, sinteza govora in iskanje informacij (angl. information retrieval) [41].

Eden izmed problemov na področju MSD-označevanja je pičlost podatkov. Modeli za označevanje besedila lahko vsebujejo zelo veliko množico parametrov, ki jih moramo naučiti iz učne množice. To število je posebej veliko, kadar označujemo besedila v morfološko kompleksnem jeziku, kjer imamo za besede na voljo veliko število različnih možnih oznak. Za učinkovito učenje parametrov modela potrebujemo velike učne množice. Te so v tem primeru korpusi besedil, ki so ročno označeni. Izdelava takih korpusov pa je zelo zamudna in draga.

Drugi problem so neznane besede. Pri označevanju besedil so to besede, ki se ne pojavijo v učni množici ali pa v slovarju. Takšne besede lahko na primer označujemo na podlagi končnic, ki v pregibnih besedah dajejo informacijo o možnih oznakah za to besedo. Posebni primer so lahko tudi lastna imena. Če je neka beseda v slovenščini pisana z veliko začetnico in ni na začetku povedi, lahko sklepamo, da gre za lastno ime.

V nadaljevanju bomo predstavili dva izmed osnovnih pristopov za označevanje besedil in označevalnik za slovenščino, imenovan Obeliks.

3.1 Algoritmi označevanja

Ena izmed pogosteje uporabljenih metod v MSD-označevanju so modeli Markova [9]. Naj bodo $w_1, ..., w_n$ besede v stavku. Naloga MSD-označevalnika je tem besedam prirediti zaporedje oznak $t_1, ..., t_n$. Na voljo imamo slovar besed. Vsaki besedi je prirejen nabor možnih oznak MSD. Neka beseda ima lahko le eno možno oznako ali pa več možnih oznak. Kadar označujemo besedilo in naletimo na besedo, ki ima več možnih oznak, lahko uporabimo statistični pristop [50]. To pomeni, da iščemo takšno zaporedje oznak, za katero bo veljalo [1]:

$$\widehat{\{t_1, ..., t_n\}} = \arg\max_{\{t_1,...,t_n\}} P(\{t_1, ..., t_n\}|\{w_1, ..., w_n\}), \tag{3.1}$$

kjer je P verjetnost, ki jo dobimo na podlogi modela za označevanje. Z uporabo Bayesovega izreka o pogojni verjetnosti lahko problem preoblikujemo v obliko:

$$\widehat{\{t_1, ..., t_n\}} = \arg\max_{\{t_1,...,t_n\}} P(\{t_1, ..., t_n\})P(\{w_1, ..., w_n\}|\{t_1, ..., t_n\}). \tag{3.2}$$

Za ocenjevanje verjetnosti $P(\{t_1, ..., t_n\})$ lahko uporabimo n-gramski model, na podoben način kot z njim ocenjujemo zaporedja besed. Za ocenjevanje verjetnosti besed $P(\{w_1, ..., w_n\}|\{t_1, ..., t_n\})$ lahko predpostavimo, da je verjetnost neke besede odvisna samo od trenutne oznake. Tako dobimo

$$P(\{w_1, ..., w_n\}|\{t_1, ..., t_n\}) = \prod_{i=1}^{n} P(w_i|t_i). \tag{3.3}$$

Uporabimo preprosto cenilko največje verjetnosti in dobimo

$$P_{MLE}(w_i|t) = \frac{C(w_i, t_i)}{C(t_i)}, \tag{3.4}$$

kjer je $C(\cdot)$ funkcija preštevanja.

Analiza vseh možnih oznak MSD za podano zaporedje besed bi bila zelo obsežna naloga, zato uporabimo Viterbijev algoritem, ki učinkovito poišče najbolj verjetno zaporedje oznak. Tudi Viterbijev algoritem lahko dodatno pohitrimo, tako da med njegovim izvajanjem zavržemo vse hipoteze oznak, katerih verjetnost se dovolj razlikuje od najboljše hipoteze do tiste točke. Ta pristop imenujemo snopovno omejevanje in je dobro znan tudi na področju razpoznavanja govora s HMM.

Druga pogosta metoda je uporaba algoritmov na principu največje entropije [55, 23]. Naj bo t trenutna oznaka, ki jo iščemo, h pa nabor podatkov, ki nam pri tem pomaga. To so lahko sosednje besede, pretekle oznake MSD in drugo. V takšnem modelu je verjetnost

zaporedja oznak MSD izražena z

$$P(t|h) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(t|h)}, \tag{3.5}$$

kjer je π normalizacijska konstanta, μ in α_i pa so pozitivni parametri modela. Funkcije f_i imenujemo značilke in imajo dve možni vrednosti: 0 in 1. Vsak parameter α_j ustreza značilki f_j. Parametre modela izberemo tako, da maksimirajo verjetnost na učni množici z uporabo enačbe 3.5. Tako dobimo:

$$L(p) = \prod_{i=1}^{n} p(t_{1,\ldots,i}) = \prod_{i=1}^{n} \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(t|h)}. \tag{3.6}$$

Verjetnost nekega zaporedja oznak je odvisna od tistih parametrov modela, katerih pripadajoča značilka ima vrednost 1.

Poznamo še druge metode MSD-označevanja, na primer nevronske mreže [61], podporne vektorje [19] in odločitvena drevesa [42].

3.2 Slovenski označevalnik Obeliks

Za slovenski jezik imamo na voljo označevalnik Obeliks, ki je bil razvit v okviru projekta Sporazumevanje v slovenskem jeziku [23]. Označevalnik je prosto dostopen na spletni strani projekta. Vanj je implementiran tudi modul za lematizacijo.

Označevalnik temelji na principu nadzorovanega učenja. Njegove osnovne komponente so: drevo končnic, algoritem za tvorjenje vektorjev značilk, algoritem za učenje in algoritem za označevanje.

Vektorji značilk opisujejo vsako besedo in so sestavljeni iz: besede, kontekstnih besed, predhodne oblikoslovne oznake, oznake iz drevesa končnic za trenutno in naslednje besede, predpone in končnice ter črkovne in druge značilke. Učenje temelji na principu največje entropije. V označevalnik so vključena tudi nekatera ekspertna pravila, ki se upoštevajo pri označevanju.

Označevalnik Obeliks tvori oznake po sistemu JOS [16]. Nabor besednih vrst v sistemu JOS in vse slovnične kategorije, ki jih določamo, so podani v tabeli 3.1. Sistem oznak pozna 12 besednih vrst (skupaj s kategorijo neuvrščeno, ki zajema tuje besede, tipkarske napake in programe), 37 kategorij za različne besedne vrste in skupno 1902 različni možni vrednosti celotnih . Natančnejše podatke, vključno s celotnimi oblikoskladenjskimi specifikacijami, lahko najdemo na spletni strani projekta.

Tabela 3.1: Besedne vrste po sistemu JOS in pripadajoče slovnične kategorije.

Besedna vrsta	Kategorije
Samostalnik	vrsta, spol, število, sklon, živost
Glagol	vrsta, vid, oblika, oseba, število, spol, nikalnost
Pridevnik	vrsta, stopnja, spol, število, sklon, določnost
Prislov	vrsta, stopnja
Zaimek	vrsta, oseba, spol, število, sklon, število in spol svojine, naslonskost
Števnik	zapis, vrsta, spol, število, sklon, določnost
Predlog	sklon
Veznik	vrsta
Členek	/
Medmet	/
Okrajšava	/
Neuvrščeno	vrsta

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
    <text>
        <body>
            <p>
                <s>
                    <w msd="Ppnzei" lemma="miren">mirna</w>
                    <w msd="Sozei" lemma="sobota">sobota</w>
                    <w msd="Vd" lemma="ki">ki</w>
                    <w msd="Gp-stm-n" lemma="biti">so</w>
                    <w msd="Zotzet--k" lemma="on">jo</w>
                    <w msd="Zn-mmi" lemma="mnog">mnogi</w>
                    <w msd="Ggdd-mm" lemma="preživeti">preživeli</w>
                    <w msd="Dm" lemma="na">na</w>
                    <w msd="Sosmm" lemma="smučišče">smučiščih</w>
                    <w msd="Zn-mmi" lemma="nekateri">nekateri</w>
                    <w msd="Vp" lemma="pa">pa</w>
                    <w msd="Gp-stm-n" lemma="biti">so</w>
                    <w msd="Do" lemma="pred">pred</w>
                    <w msd="Sommo" lemma="televizor">televizorji</w>
                    <w msd="Ggnd-mm" lemma="spodbujati">spodbujali</w>
                    <w msd="Zspmmtm" lemma="naš">naše</w>
                    <w msd="Sommt" lemma="športnik">športnike</w>
```

Slika 3.1: Primer oblikoskladenjsko označenega in lematiziranega besedila kot izhod iz označevalnika Obeliks.

Za učno množico lahko uporabljamo korpus ssj500k, ki je prav tako dostopen na spletni strani projekta. V [23] so predstavljeni rezultati analize označevanja pri uporabi tega korpusa. Natančnost označevalnika je 91,34.

Obeliks označeno besedilo izpiše v obliki XML-TEI, ki je namenjena nadaljnji obdelavi z računalniškimi orodji. Primer označenega besedila je prikazan na sliki 3.1.

Poglavje 4

Zasnova morfoloških modelov s kontekstno odvisno strukturo

V tem poglavju bomo definirali strukturo predlaganih modelov, algoritem za določanje poti sestopanja v danih n-gramih in postopek učenja modelov. Modeli bodo izhajali iz faktoriziranih jezikovnih modelov. Osnovna problema teh modelov sta definiranje faktorjev in določanje poti sestopanja. Na začetku bomo definirali faktorje. Ti bodo izbrani ročno, glede na nekatere predhodne rezultate in jezikoslovno znanje.

Določanje poti sestopanja bo odvisno od besednih vrst v danem n-gramu. Predstavili bomo funkcijo za določanje poti sestopanja, ki bo temeljila na besednih vrstah besed, ki se nahajajo v n-gramu, ki ga trenutno ocenjujemo. Ker bomo pri tem sprva ločili vse možne kombinacije besednih vrst, bomo s tem ponovno naleteli na problem pičlosti (pomanjkanja) podatkov. Tega se bomo izogibali z združevanjem različnih kombinacij zaporednih besednih vrst, ki dajejo podobne rezultate, v razrede. V nadaljevanju bomo poti sestopanja določali za posamezne razrede, ne bomo pa različnih kombinacij razdruževali na bolj podrobne. Za tak pristop je smiselno, da na začetku besedne vrste bolj podrobno razdelimo.

Glede na število faktorjev in dolžino zgodovine besede, ki jo bomo upoštevali, bomo naleteli na veliko število možnih zaporedij faktorjev v poti sestopanja. Ta "iskalni prostor" kombinacij je prevelik, da bi v njem pregledali vse možne kombinacije. Zato bomo poti sestopanja gradili postopoma od faktorja, ki se nazadnje zavrže, do faktorja, ki se najprej zavrže.

Tudi ko izberemo fiksno pot sestopanja, dobimo kot rezultat zelo velik model, v katerem faktorji, ki jih najprej zavržemo, morda sploh niso uporabni ali pa celo zmanjšujejo uspešnost modela. Iz tega vidika je potrebno določiti tudi mejo, ob kateri več ne dodajamo faktorjev v pot sestopanja in gradnjo strukture modela zaključimo.

V tem poglavju bomo naslovili vse zgoraj opisane probleme in predstavili njihove rešitve, ki smo jih uporabili pri učenju predlaganih modelov. Poglavje bomo zaključili s predstavitvijo celotnega algoritma za določanje strukture modelov in postopka učenja modelov.

4.1 Definiranje faktorjev

Pri izbiri faktorjev bomo upoštevali, katere podatke potrebujemo za osnovno zamisel modela s kontekstno odvisno strukturo. Omejevali se bomo na možnosti, ki jih ponuja označevalnik za slovenski jezik Obeliks, ter deloma tudi na slovnično znanje o podatkih v njegovih oznakah. Izhod označevalnika je besedilo, označeno z oznakami MSD, ki predstavljajo skupno 14 različnih slovničnih lastnosti besed. Za vsako posamezno besedno vrsto pa je na voljo manj podatkov.

Faktorje bomo določali ročno. Kot osnovno vodilo bomo uporabili naslednje štiri točke:

1. Kot faktor besede želimo določiti besedo sámo.
2. Kot faktor želimo določiti podatek o besedni vrsti, ki bo koristil pri določanju strukture modelov.
3. Kot en faktor ali več faktorjev želimo določiti podatke iz oznake MSD, ki lahko pomagajo pri izboljšanju rezultata razpoznavanja.
4. Ker lahko število faktorjev močno poveča število vseh možnih poti sestopanja, hočemo slovnične podatke iz oznake MSD združiti v en faktor.

Besedo smo izbrali kot faktor zato, ker je najbolj natančen faktor in se že uspešno uporablja v obliki običajnih n-gramskim modelov za razpoznavanje govora. Velikost slovarja besed bo enaka velikosti osnovnega slovarja v prvem prehodu. Velikosti slovarjev v domeni *Broadcast News* so lahko od več deset tisoč do nekaj sto tisoč besed. Besedo bomo označevali z W.

Algoritem za definiranje strukture modelov, ki ga bomo predstavili kasneje v tem poglavju, temelji med drugim na združevanju različnih kombinacij oznak, če za njih ne bo na voljo dovolj velika učna množica. Zato lahko za faktor, ki bo opisoval besedno vrsto, definiramo bolj podroben nabor možnih vrednosti, kot pa je besedna vrsta, definirana v oznakah MSD. Kot prvi faktor smo zato izbrali kombinacijo besedne vrste in tipa. Vse možne kombinacije besedne vrste in tipa, ki jih predvidevajo oznake MSD, so predstavljene v tabeli 4.1. Faktor smo poimenovali razširjena besedna vrsta ali krajše kar besedna vrsta, označevali pa ga bomo s P. Zadnjo vrednost – *neznano* – dobimo, kadar označevalnik ne

prepozna besede in ji ne zna prirediti oznake MSD. Ko še dodamo oznaki za začetek in konec povedi oz. govornega segmenta, ima pravkar definiran faktor 34 različnih možnih vrednosti.

Tabela 4.1: Vse možne vrednosti faktorja *razširjena besedna vrsta* in njihovi pomeni.

Oznaka	Pomen	Oznaka	Pomen
so	Samostalnik – občno ime	zn	Nedoločni zaimek
sl	Samostalnik – lastno ime	zl	Nikalni zaimek
gg	Glavni glagol	kg	Glavni števnik
gp	Pomožni glagol	kv	Vrstilni števnik
pp	Splošni pridevnik	kz	Zaimkovni števnik
ps	Svojilni pridevnik	kd	Drugi števnik
pd	Deležniški pridevnik	d	Predlog
rs	Splošni prislov	vp	Priredni veznik
rd	Prislov – deležje	vd	Podredni veznik
zo	Osebni zaimek	l	Členek
zs	Svojilni zaimek	m	Medmed
zk	Kazalni zaimek	o	Okrajšava
zz	Oziralni zaimek	nj	Neuvrščeno – tujejezično
zp	Povratni zaimek	nt	Neuvrščeno – tipkarska
zc	Celostni zaimek	np	Neuvrščeno – program
zv	Vprašalni zaimek	00	Neznano

Tretji faktor, ki ga bomo definirali, bo poenostavljena oznaka MSD. Izbirali bomo med slovničnimi kategorijami besed, ki bi lahko pomagale pri izboljšanju rezultatov. Izvzeli bomo oznako za tip besede, ki smo jo že uporabili pri definiciji prvega faktorja. Na preostalih oznakah smo naredili predhodne teste perpleksnosti na razvojni množici BNSI. Izbrali smo lastnosti, ki se zdijo smiselne iz slovničnega stališča in ki kažejo zmanjšanje perpleksnosti pri prehodu na modele višjega reda. Na koncu smo izbrali: spol, sklon, število in glagolsko osebo. Izbira lastnosti je tudi iz slovničnega vidika smiselna, za slovenščino je namreč značilno ujemanje v spolu, sklonu in številu med samostalniki in pridevniki. Definirano oznako MSD smo tudi oblikovno nekoliko spremenili, tako da ima vsaka besedna vrsta oznako MSD dolžine 4, v kateri so navedene vse štiri zgoraj omenjene lastnosti. V primeru, da besedni vrsti ni mogoče prirediti katere izmed lastnosti (npr. samostalniku ne moremo prirediti osebe), bo na ustreznem mestu v vrednosti faktorja napisana ničla. Faktor poenostavljene oznake MSD bomo označevali z M. Za vsak podatek znotraj oznake vemo, koliko možnih vrednosti lahko zavzame. Spol lahko zavzame 5 vrednosti: moški, ženski, srednji, ne obstaja, neznano. Vrednost *ne obstaja* nastopi pri besednih vrstah, kjer spola ni možno določati, vrednost *neznano* pa pomeni, da je bil označevalnik neuspešen pri označevanju te besede. Podobno imata tudi lastnosti število in oseba 5 možnih vrednosti, sklon pa 8 možnih vrednosti. Niso pa možne vse kombinacije teh vrednosti pri posameznih besednih vrstah. Po specifikacijah sistema JOS [16] obstaja za različne besedne vrste od 1 do 255 različnih možnih kombinacij.

Povzamemo lahko, da smo izbrali tri faktorje:

1. beseda (W),
2. razširjena besedna vrsta (P),
3. oznaka MSD (M).

Primer stavka, zapisanega v faktorizirani obliki, ki je ustrezna za orodje SRILM, je prikazan na sliki 4.1. Faktorji so med sabo ločeni z dvopičjem. Oznake faktorjev so od njihovih vrednosti ločene z vezajem. Oznake P, M in W označujejo zgoraj definirane tri faktorje.

```
P-kv:M-mim0:W-prvi P-so:M-mim0:W-ukrepi
P-d0:M-0t00:W-zoper P-sl:M-mie0:W-triglav
```

Slika 4.1: Primer faktoriziranega govornega segmenta "Prvi ukrepi zoper Triglav" iz razvojne množice BNSI ob uporabi definiranih faktorjev. Zaradi načina transkribiranja posnetkov v bazi ne uporabljamo velikih začetnic besed.

4.2 Preprosti modeli z morfološkimi informacijami

Preden bomo definirali kontekstno odvisne modele, bomo predstavili preproste modele, ki temeljijo na faktorjih razširjene besedne vrste in poenostavljene oznake MSD, predstavljene v prejšnjem podpoglavju. Predstavili bomo dva ločena modela – enega za besedne vrste in enega za oznake MSD. Te modele bomo kasneje uporabili pri razpoznavanju govora. Z njihovo pomočjo hočemo ugotoviti, kolikšen je doprinos vpeljave definiranih faktorjev v razpoznavanje govora in kolikšen je naknadni doprinos izdelave kontekstno odvisnih struktur teh modelov.

Definiramo lahko osnovna n-gramska modela za razširjeno besedno vrsto in poenostavljeno oznako MSD:

$$\begin{aligned} &P(P_0|P_{-1},...,P_{-n}),\\ &P(M_0|M_{-1},...,M_{-n}). \end{aligned} \tag{4.1}$$

V obeh modelih uporabimo pot sestopanja, ki najprej odstranjuje najbolj oddaljene faktorje. Oba modela sta po strukturi enaka običajnim besednim n-gramskim modelom. Razlika je ta, da modeliramo zaporedja besednih vrst oz. oznak MSD. Nekoliko zahtevnejše lahko postavimo modela, ki bosta upoštevala oba faktorja:

$$\begin{aligned} &P(P_0|P_{-1},...,P_{-n},M_0,...,M_{-n}),\\ &P(M_0|M_{-1},...,M_{-n},P_0,...,P_{-n}). \end{aligned} \tag{4.2}$$

V teh modelih modeliramo vsak faktor glede na drug faktor iz trenutne besede in vse faktorje iz prejšnjih besed. Za pot sestopanja bomo izbrali takšno pot, ki zajema največ 7 faktorjev, kjer ne štejemo faktorja, ki ga modeliramo. Faktorje pa izbiramo iz nabora faktorjev trenutne in predhodnih 5 besed. Tako dobimo 6-gramske faktorizirane modele s fiksnimi potmi sestopanja z največ osmimi vozlišči. Zadnje vozlišče je (nepogojna) verjetnost modeliranega faktorja brez zgodovine ali drugih faktorjev iste besede.

Na število faktorjev največ 7 smo se omejili zaradi velikosti končnega modela, na (največ) 6-gramske modele pa zaradi možne izgube povezave med besedami in faktorji na večjih razdaljah v stavku.

Dodatno postavimo omejitev, da če se v poti sestopanja pojavi poljuben faktor, se v tej poti pojavijo tudi vsi faktorji iste vrste (besedna vrsta oz. oznaka MSD, ki so bližje modeliranemu faktorju. Npr. če se v poti pojavi faktor P_{-3}, se pojavijo tudi faktorji P_{-2} in P_{-1}, če modeliramo M_0, pa se pojavi tudi P_0. Naslednja omejitev je ta, da se bo faktor, ki je bližje modeliranemu faktorju, v poti sestopanja pojavil na takšnem mestu, da ga med sestopanjem kasneje zavržemo. Primer dveh poti sestopanja, ki ustrezata tema pogojema, definirata predpisa

$$\begin{aligned} M_0 &: \{P_0, P_{-1}, M_{-1}, P_{-2}, M_{-2}, P_{-3}, M_{-3}\}, \\ M_0 &: \{M_{-1}, M_{-2}, M_{-3}, P_0, P_{-1}, P_{-2}, P_{-3}\}, \end{aligned} \tag{4.3}$$

kjer so faktorji, ki se kasneje zavržejo, napisani bolj levo.

Vsako pot sestopanja lahko preprosto gradimo od leve proti desni. Pri tem na vsakem koraku odločamo, ali bomo kot naslednji faktor uporabili P ali M, če ga imamo še na voljo. Izračunamo lahko, da tako za vsakega izmed modeliranih faktorjev P_0 in M_0 dobimo skupno 244 možnih poti sestopanja dolžin od 1 do 7.

Ker ti modeli ne uporabljajo besede kot faktorja, bomo v modeliranju imeli le majhen slovar – 34 za razširjene besedne vrste in nekaj 100 za oznake MSD. Ker imamo majhen slovar, ne potrebujemo tako velikih učnih množic. Za učenje teh modelov lahko zato uporabimo relativno majhen korpus z oznakami MSD.

4.3 Osnovna struktura kontekstno odvisnih modelov

Osnovna naloga klasičnega n-gramskega modela je modeliranje verjetnosti besede glede na prejšnje besede. V faktoriziranih jezikovnih modelih besede nadomestimo s faktorji. Sedaj modeliramo verjetnosti faktorjev glede na faktorje prejšnjih besed in glede na prejšnje faktorje trenutne besede, kot je to izraženo v enačbi 2.42. V našem primeru imamo 3 faktorje. Za zgodovino bomo upoštevali do 5 besed. To pomeni, da bomo v osnovi izhajali

iz 6-gramskega modela. Enačbo 2.42 lahko sedaj zapišemo kot

$$P(f_i^{1:3}) = \prod_{k=1}^{3} P(f_i^k | f_i^{1:k-1}, f_{i-1}^{1:3}, ..., f_{i-5}^{1:3}). \tag{4.4}$$

Sedaj lahko še vstavimo faktorje in dodamo oznako za nabor faktorjev ene besede

$$\begin{aligned} f_i^1 &= P_i \\ f_i^2 &= M_i \\ f_i^3 &= W_i \\ F_i &= \{P_i, M_i, W_i\} \end{aligned} \quad , \tag{4.5}$$

ter dobimo

$$\begin{aligned} P(F_i) = & P(P_i | F_{i-1}, ..., F_{i-5}) \cdot \\ & P(M_i | P_i, F_{i-1}, ..., F_{i-5}) \cdot \\ & P(W_i | M_i, P_i, F_{i-1}, ..., F_{i-5}) \end{aligned} \quad . \tag{4.6}$$

Tako smo osnovni model razdelili na 3 modele, ki vsebujejo 15, 16 oz. 17 faktorjev. Modele lahko obravnavamo ločeno in za vsakega izmed njih iščemo najprimernejšo pot sestopanja. Le-to lahko izrazimo kot fiksno zaporedje faktorjev, ki jih bomo odstranjevali v postopku sestopanja od zadnjega do prvega. Formalno lahko to izrazimo kot funkcijo, ki jo bomo označevali z BO. Vrednosti funkcije bodo zaporedja faktorjev:

$$(f_1, ..., f_l)\,, \tag{4.7}$$

kjer je vsak faktor f_i element množice 15, 16 oz. 17 faktorjev, ki so na voljo v modelih v enačbi 4.6. Dolžina zaporedja je l, ki je lahko manjši ali enak številu vseh možnih faktorjev v poti sestopanja m. Modele smo zasnovali tako, da je pot sestopanja funkcija oznak razširjenih besednih vrst:

$$BO : (P_{-5}, ..., P_0) \mapsto (f_1, ..., f_l). \tag{4.8}$$

Še enkrat lahko poudarimo, da je definicijsko območje funkcije BO zaporedje podanih vrednosti faktorja, medtem ko je funkcijska vrednost zaporedje faktorjev, tj. le njihovih oznak, in ne konkretnih vrednosti. Vrednosti funkcije BO – poti sestopanja – bomo določali s pomočjo učnih množic, učenja "delnih" modelov in njihovega testiranja na razvojni množici BNSI.

Tabela 4.2: Število vseh možnih zaporedij faktorja P, kjer upoštevamo število možnih vrednosti za P.

Red modela n	Število zaporedij
1	32
2	1.155
3	36.992
4	1.183.744
5	37.897.808
6	1.212.153.865
Skupaj	1.251.255.587

4.4 Velikost iskalnega prostora za pot sestopanja

Vrednosti funkcije BO bomo določali za vsak element njene domene – zaporedje možnih razširjenih besednih vrst. Med uporabo modelov naletimo na različne možne dolžine teh zaporedij. Medtem ko bomo v osnovi izhajali iz 6-gramskega modela, moramo na primer na začetku stavka oz. govornega segmenta uporabiti model nižjega reda. Takrat ocenjujemo verjetnost za besedo, za katero nimamo tako dolge zgodovine.

Označimo s $|P|$ velikost množice vseh možnih oznak razširjene besedne vrste. Teh je 34, pri tem sta šteti tudi oznaki za začetek in konec stavka. Vseh možnih kombinacij za n-gramski primer je

$$(|P|)^2 \cdot (|P| - 2)^{n-2}\,. \tag{4.9}$$

Ta enačba velja za vrednosti n, ki so večje od 2. Kadar je n enak 2, lahko odštejemo primer, kjer se pojavi zaporedje oznak začetek stavka in nato konec stavka:

$$(|P|)^2 - 1. \tag{4.10}$$

Kadar pa je n enak 1, lahko izpustimo oznaki za začetek in konec stavka. Primer za začetek stavka lahko izpustimo, ker v praksi jezikovnega modela ne uporabljamo za ocenjevanje začetka stavka oz. za verjetnost začetka stavka vedno vstavimo 1. Oznako za konec stavka lahko izpustimo, ker ni zanimiva, saj v tem primeru poznamo vrednosti vseh faktorjev.

V tabeli 4.2 je predstavljeno število vseh možnih zaporedij v našem primeru, kjer smo upoštevali nabor možnih vrednosti za razširjeno besedno vrsto. Kot vidimo, število vseh zaporedij presega eno milijardo. To predstavlja prevelik iskalni prostor, da bi lahko preverili vse možne kombinacije. Rešitve za ta problem bomo predstavili v nadaljevanju.

Največja možna dolžina poti sestopanja je odvisna od trenutnega faktorja in dolžine n-grama, iz katerega izbiramo faktorje. Kadar ocenjujemo prvi faktor v besedi, je največja

Tabela 4.3: Največje možne dolžine zaporedij faktorjev glede na trenutno ocenjevan faktor v trenutni besedi in število besed v zgodovini, ki jih upoštevamo.

Število besed v zgodovini	Faktor v trenutni besedi		
	1	2	3
0	0	1	4
1	3	4	5
2	6	7	8
3	9	10	11
4	12	13	14
5	15	16	17

dolžina enaka produktu med številom faktorjev v besedi in številom besed v zgodovini $(n-1)$. Za vsak naslednji faktor v trenutni besedi se največja možna dolžina poveča za 1.

Kot funkcijske vrednosti BO dobimo zaporedja, ki lahko imajo največjo možno dolžino ali pa so krajša. Naj bo K število faktorjev na besedo. Kadar ocenjujemo i-ti faktor v n-gramskem modelu, lahko največjo možno dolžino m izračunamo po enačbi

$$m = (n-1) \cdot K + i - 1. \tag{4.11}$$

Število možnih zaporedij m elementov je enako $m!$. Zgoraj smo že opisali, da je dolžina zaporedja, ki ga definira funkcija BO, lahko tudi krajša. Označimo z l dolžino zaporedja. Velja, da je l manjši ali enak m. Število možnih zaporedij z l elementi je:

$$\frac{m!}{(m-l)!}. \tag{4.12}$$

Število vseh možnih zaporedij dolžin od 1 do največje možne dolžine m pa je

$$\sum_{l=1}^{m} \left(\frac{m!}{(m-l)!} \right). \tag{4.13}$$

V našem primeru uporabljamo tri faktorje na besedo in zgodovino do 5 besed. Največje možne dolžine zaporedij faktorjev so podane v tabeli 4.3.

Rešitve enačbe 4.12 za te primere pa lahko najdemo v tabeli 4.4. Kot vidimo iz podatkov v tabeli 4.4, je tudi množica možnih funkcijskih vrednosti funkcije BO zelo velika. Tudi za ta problem bomo rešitve predstavili v nadaljevanju.

Tabela 4.4: Število možnih zaporedij, katerih dolžina je enaka ali krajša od največje možne dolžine glede na trenutno ocenjevan faktor v besedi in število besed v zgodovini, ki jih upoštevamo.

Besed v zgodovini	Faktor v trenutni besedi		
	1	2	3
0	0	2	5
1	16	65	326
2	1.957	13.700	109.601
3	986.410	$9{,}8641 \cdot 10^{6}$	$1{,}08505 \cdot 10^{8}$
4	$1{,}30206 \cdot 10^{9}$	$1{,}69268 \cdot 10^{10}$	$2{,}36975 \cdot 10^{11}$
5	$3{,}55463 \cdot 10^{12}$	$5{,}6874 \cdot 10^{13}$	$6{,}66859 \cdot 10^{14}$

4.5 Zasnova algoritma za določanje poti sestopanja

4.5.1 Izhodiščni algoritem

Predpostavimo najprej, da imamo na voljo neomejeno veliko učno množico ter dovolj časa in procesorske zmogljivosti, da lahko preiščemo ves iskalni prostor. Predstavili bomo osnovno zamisel algoritma. Predstavljena je v algoritmu 4.1.

Algoritem 4.1: Izhodiščni algoritem za določanje poti sestopanja.

Input: N
Input: K
Input: $\mathcal{P}_n \quad \forall n \in \{1, ..., N\}$
Output: $\mathcal{S}$

```
1  begin
2      S = ∅
3      for n ∈ 1, ..., N do
4          for k ∈ 1, ..., K do
5              for P ∈ P_n do
6                  M̂ ⟵ Ni rešitve
7                  for M ∈ M do
8                      Testiraj model M
9                      if T(M) > T(M̂) then
10                         M̂ = M
11                     S ⟵ S ∪ (n, k, P, M̂)
```

Vhodni podatki v algoritem so:

- N – največja dolžina n-grama
- K – število faktorjev na besedo
- P_n – množica vseh možnih zaporedij razširjenih oznak POS dolžine n

Izhodni podatek iz algoritma je množica , ki za vsako dolžino n-grama, za vsak faktor v besedi in vsako zaporedje vrednosti oznak POS vsebuje ustrezno zaporedje faktorjev v poti sestopanja. To množico lahko imenujemo tudi množica rešitev algoritma.

Algoritem je nastavljen tako, da izhaja iz prazne množice rešitev. Nato za vse možne dolžine n-grama, vse faktorje trenutne besede in zaporedja vrednosti oznak POS začnemo iskati pot sestopanja. Najprej določimo vse možne poti sestopanja M in določimo, da je najboljša pot prazna. Nato testiramo vse možne poti M iz množice $\mathcal{M}$. Kadar najdemo pot sestopanja, ki daje boljše rezultate od trenutno najboljše poti, jo določimo kot novo najboljšo pot. Poti primerjamo preko kriterijske funkcije T. Ko preverimo vse možne poti, dodamo najboljšo pot v množico rešitev. Za popolnost algoritma moramo še definirati kriterijsko funkcijo T, kar bomo storili v nadaljevanju poglavja.

Tako nastavljen algoritem bo predstavljal izhodišče za postavitev končnega algoritma. Da bo uporaben tudi v praksi, ga moramo prilagoditi tako, da bomo naslovili naslednje probleme:

- Definirati moramo vrstni red preizkušanja možnih poti sestopanja, s katerim bomo najlažje in najhitreje dosegli pot sestopanja z zadovoljivim rezultatom.
- Definirati moramo kriterij, kdaj bo rezultat preizkušane poti sestopanja zadovoljiv in kdaj bomo ustavili dodajanje faktorjev v pot.
- Definirati moramo način združevanja testiranja pri različnih kombinacijah POS, ki dajejo podobne rezultate in za katere ne obstaja dovolj učnega materiala v korpusih.

4.5.2 Predvidena uporaba jezikovnih modelov

Da se bomo lahko smiselno lotili reševanja zgoraj omenjenih problemov, si bomo najprej ogledali končni namen modelov. Uporabljali jih bomo v drugem prehodu dvoprehodnega algoritma razpoznavanja tekočega govora z velikim slovarjem. V drugem prehodu bomo imeli kot izhodišče seznam N najboljših hipotez, ki bodo označene z oznakami MSD.

Algoritem za uporabo morfoloških jezikovnih modelov bo ocenjeval jezikovne verjetnosti hipotez. Pri tem bo najprej preveril oznake POS ocenjevane hipoteze in njene zgodovine. Na podlagi te informacije bo izbral model z ustrezno strukturo – z ustrezno potjo sestopanja. Model bomo nato uporabili za ocenjevanje verjetnosti trenutne besede. Vsako naslednjo besedo ocenjujemo neodvisno od prejšnje.

4.5.3 Kriterijska funkcija

Kriterijska funkcija je osrednji del algoritma. Z njo določamo, kateri model lahko štejemo kot boljši. Za ocenjevanje jezikovnih modelov običajno uporabljamo dva pristopa. Prvi je neposredna uporaba modela. V našem primeru bi to pomenilo, da bi vsak model, ki ga preizkušamo, uporabili v razpoznavanju govora na izbrani razvojni množici. To lahko storimo neposredno v iskalnem algoritmu razpoznavalnika. Toda razpoznavanje govora je časovno zelo zahteven algoritem, zato je primerjanje različnih jezikovnih modelov z razpoznavanjem zelo zamudno. Druga možnost je uporaba jezikovnih modelov v dvoprehodnem algoritmu, kjer preizkušamo modele le v drugem prehodu na besedni mreži ali pa na seznamu N najboljših hipotez. Ker je samo ocenjevanje teksta z jezikovnimi modeli veliko hitrejše od celotnega postopka razpoznavanja, je ta postopek veliko hitrejši. Kljub temu še vedno zahteva veliko časa.

Velikokrat ocenjujemo jezikovne modele na podlagi njihove perpleksnosti – enačba 2.36. V tem primeru računamo verjetnosti na izbranem besedilu. V primerjavi z razpoznavanjem na primerljivo dolgem besedilu je ta postopek veliko hitrejši, saj moramo pri razpoznavanju ocenjevati veliko množico različnih hipotez, medtem ko pri računanju perpleksnosti besedilo ocenjujemo le enkrat. Razne raziskave so tudi pokazale korelacijo med perpleksnostjo in deležem napak v razpoznavalniku govora [10]. Zato smo se odločili, da bomo kriterijsko funkcijo tvorili na podlagi perpleksnosti.

V algoritmu za določanje poti sestopanja želimo ocenjevati modele z različnimi potmi sestopanja, vendar le pri tistih kombinacijah besed, ki se ujemajo v zaporedju oznak POS. Uspešnost modelov s perpleksnostjo pa običajno merimo s perpleksnostjo na celotni množici. Ker bo v končnem ocenjevanju izbira strukture modela za dano besedo neodvisna od izbire za sosednjo besedo, bo tudi računanje perpleksnosti neodvisno. Tako bo sprememba strukture modela za dano zaporedje oznak POS vplivala le na ustrezne člene v računanju perpleksnosti, medtem ko bodo členi, ki ustrezajo drugačnim zaporedjem oznak POS, nespremenjeni. Zato lahko kot izhodišče za kriterijsko funkcijo uporabimo perpleksnost.

Perpleksnost modela P na besedilu $\mathcal{W}$ računamo po enačbi

$$PPL_P(\mathcal{W}) = 2^{H_P(\mathcal{W})}, \tag{4.14}$$

kjer je $H_P(W)$ križna entropija modela:

$$\begin{aligned} H_P(\mathcal{W}) &= \frac{1}{-T} \cdot \log_2 P(\mathcal{W}) \\ &= \frac{1}{-T} \cdot \log_2 \prod_{i=1}^{T} P(W_i) \end{aligned}. \tag{4.15}$$

pri tem je T dolžina besedila, W_i pa predstavlja i-to besedo v njem. Sedaj želimo definicijo

perpleksnosti prilagoditi tako, da bo zajemala samo določene besede v besedilu. Naj bo sedaj P^* oznaka za zaporedje oznak POS

$$P^* = (P_{-n}, ..., P_0). \tag{4.16}$$

Za vsak P^* definirajmo takšno množico števil

$$\mathcal{I}(P^*) \subset \{1, ..., T\}, \tag{4.17}$$

za katero velja

$$i \in \mathcal{I}(P^*) \implies (P_{i-n}, ..., P_i) = P^*. \tag{4.18}$$

Poenostavljeno to pomeni, da imamo v množici $\mathcal{I}(\mathcal{P}^*)$ tiste indekse besed iz besedila, pri katerih nastopi podano zaporedje oznak POS P^*. Najprej prilagodimo enačbo za križno entropijo na delno besedilo:

$$H^*(\mathcal{W}, P^*) = \frac{1}{-|\mathcal{I}(P^*)|} \log_2 \prod_{i \in \mathcal{I}(P^*)} P(W_i). \tag{4.19}$$

Na tem mestu lahko izrazimo tudi prilagojeno perpleksnost kot

$$PP^*(\mathcal{W}, P^*) = 2^{H^*(\mathcal{W}, P^*)}. \tag{4.20}$$

Model velja za boljšega, če ima nižjo perpleksnost. Ker je eksponentna funkcija v enačbi 4.19 strogo naraščajoča, lahko kot kriterijsko funkcijo uporabimo tudi križno entropijo, saj nas zanima le, kateri model daje manjšo vrednost. Pri podanem zaporedju oznak POS imamo fiksno število besed v besedilu, ki jim ustrezajo, zato tudi število besed v enačbi 4.19 nima vpliva na končno izbiro med modeli. Odstranimo lahko tudi negativni predznak, vendar moramo zato od sedaj naprej obravnavati večjo vrednost kot boljši rezultat. Kot končni rezultat dobimo

$$\log_2 \prod_{i \in \mathcal{I}(P^*)} P(W_i) = \sum_{i \in \mathcal{I}(P^*)} \log_2 P(W_i). \tag{4.21}$$

Vsota logaritmov v zadnji enačbi je najprimernejša izbira iz vidika enostavne uporabe in hitrosti ocenjevanja, saj jezikovni modeli običajno vsebujejo logaritme verjetnosti in tudi orodja za uporabo modelov dajejo rezultate v tej obliki.

4.5.4 Dodajanje faktorjev v pot sestopanja

Kot smo videli v tabeli 4.4, obstaja zelo veliko število možnih zaporedij faktorjev v poti sestopanja. Ker vseh možnih stanj ni mogoče preveriti, bomo pri izbiri možnih kandidatov upoštevali naslednja vodila:

- Začeli bomo s potmi sestopanja, ki vsebujejo le majhno število faktorjev. Daljše poti sestopanja bomo gradili iz krajših.
- Zgledovali se bomo po iskalnih algoritmih iz razpoznavanja govora in tehniki snopovnega omejevanja. Pri tem bomo vodili seznam potencialnih kandidatov za končni rezultat in bomo zavračali kandidate, katerih rezultat je dovolj slabši od trenutno najboljšega kandidata. Nove kandidate bomo dobili iz trenutnih kandidatov z dodajanjem faktorjev.
- Predpostavili bomo, da imajo faktorji, ki so od ocenjevane besede manj oddaljeni, večji doprinos k izboljšanju modela kot pa tisti, ki so bolj oddaljeni.

Na začetku postavimo množico možnih poti sestopanja, v kateri imamo najprej le prazno pot sestopanja. Nato v njo dodajamo možne poti. To storimo s pomočjo algoritma za dodajanje faktorjev 4.2, ki ga ponovimo m-krat, kjer je m največje možno število faktorjev.

Algoritem 4.2: Algoritem za gradnjo množice možnih poti sestopanja.

```
Input: F
Input: M
Input: M̂
Output: M*
Output: M̂*
1  begin
2      M* = M
3      for ∀M ∈ M do
4          Določi F*
5          for ∀F ∈ F* do
6              M* = M* ∪ {(M + F)}
7      M̂* = arg max_{M*∈M*} T(M*)
8      Določi ϵ
9      for ∀M* ∈ M* do
10         if T(M*) ≤ T(M̂) − ϵ then
11             M* = M* \ {(M)}
```

Vhod algoritma so nabor možnih faktorjev, dosedanja množica možnih poti sestopanja in pot sestopanja iz te množice, ki je bila do sedaj najbolje ocenjena. Izhod algoritma sta nova množica možnih poti sestopanja in nova najbolje ocenjena pot sestopanja iz te množice.

Namen algoritma je določiti novo množico možnih poti $\mathcal{M}^*$. Pri tem najprej izhajamo iz vhodne množice $\mathcal{M}$. Nato za vsako pot v njej določimo podmnožico možnih

faktorjev $\mathcal{F}^*$, ki jih lahko dodamo tej poti na konec, in vsako tako pot dodamo v množico. Nato poiščemo pot v novi množici, ki daje najboljši rezultat kriterijske funkcije. Na koncu iz množice odstranimo vse poti, ki dajejo rezultat za več kot ϵ slabši od najboljše poti. Parameter ϵ smo poimenovali širina iskalnega pasu in predstavlja dovoljeno razliko med najboljšo potjo (glede na kriterijsko funkcijo) in vsako ostalo potjo, ki jo obdržimo v množici možnih rešitev. Kadar ima neka pot v množici preveliko razliko, sklepamo, da z dodajanjem faktorjev v to pot ne bomo več dosegli boljšega rezultata od trenutno najboljše poti. Zato to pot zavržemo (vrstica 11 v algoritmu 4.2). Širina iskalnega pasu tako deluje podobno kot širina pasu pri snopovnem omejevanju v iskalnih algoritmih za razpoznavanje govora.

Širino iskalnega pasu določimo v vsakem izvajanju algoritma na novo glede na trenutni nabor poti v množici in na rezultat najboljše poti. Tukaj upoštevamo, da se pri krajših poteh sestopanja lahko ob dodajanju novega faktorja rezultat kriterijske funkcije bolj spremeni kot pa pri dodajanju faktorja pri daljših poteh. Podoben vzorec opazimo pri običajnih n-gramskih jezikovnih modelih, kjer je na primer razlika v perpleksnosti med bigramskim in trigramskim modelom večja kot pa razlika med trigramskim in štirigramskim. Upoštevamo lahko tudi število poti v trenutni množici in lastnost kriterijske funkcije, da predstavlja logaritem verjetnosti.

4.5.5 Združevanje ločenih primerov za različne kombinacije zaporedij POS

Do sedaj smo opisali postopke, s katerimi lahko določimo strukturo modela za ocenjevanje n-gramov z določenim zaporedjem oznak POS. Zaradi velikega števila možnih zaporedij oznak POS ne moremo zgraditi za vsako kombinacijo ločenega modela. Drugi problem predstavlja omejena velikost učne množice, kar pomeni, da imamo lahko za nekatere kombinacije na voljo le zelo majhno število pojavnic. To potem pomeni, da dobimo le zelo nezanesljive modele za te kombinacije.

Zato smo se odločili, da med samim določanjem poti sestopanja združujemo kombinacije oznak. Združujemo kombinacije z majhnim številom pojavnic v učni množici s kombinacijami, ki so si ali slovnično podobne ali pa imajo podobno do sedaj določeno strukturo.

Naslednji problem predstavljajo kombinacije oznak, ki se v učni množici nikoli ne pojavijo. Kombinacije bomo združevali v vsak red n-grama ločeno, potem ko smo za vsako kombinacijo že dobili strukture modelov. Nabor združenih kombinacij poimenujemo razred. Pri tem izvedemo določeno število iteracij določanja strukture in združevanja poti, dokler ne dosežemo dovolj majhnega števila razredov. Pri tem pa mora veljati, da bomo za vsak razred imeli dovolj velik nabor pojavnic iz učne množice, da lahko zgradimo zanesljiv model. Postopek je predstavljen v algoritmu za združevanje zaporedij oznak POS 4.3.

Algoritem 4.3: Algoritem za združevanje kombinacij POS.

Input: $\mathcal{P}$
Output: $\mathcal{P}^*$
Output: $\mathcal{M}(\bar{P}) \quad \forall \bar{P} \in \mathcal{P}^*$
begin
  $\mathcal{P}^* = \mathcal{P}$
  Določi α
  repeat
    for $\forall \bar{P} \in \mathcal{P}^*$ **do**
      Določi $\mathcal{M}(\bar{P})$
      Določi $\hat{M}(\bar{P})$
    Določi β
    repeat
      $\bar{P}' = \arg\min_{\bar{P} \in \mathcal{P}} (C(\bar{P}))$
      $\bar{P}'' = \arg\min_{\bar{P} \in \mathcal{P} \setminus \{\bar{P}'\}} D(\bar{P}', \bar{P}'')$
      $\mathcal{P}^* \leftarrow$ Združi $\bar{P}'$ in $\bar{P}''$
    until *Število razredov* $< \beta$
  until *Število razredov* $< \alpha$

Vhod algoritma je nabor možnih zaporedij oznak POS, izhod pa množica razredov zaporedij oznak POS, kjer je v vsakem razredu eno ali več zaporedij oznak POS. Izhod je tudi nabor možnih poti sestopanja za vsak razred zaporedij oznak POS. V algoritmu najprej določimo množico razredov kombinacij POS, tako da vsak razred vsebuje eno možno kombinacijo in za vsako kombinacijo obstaja en razred. Nato iterativno združujemo razrede, dokler število razredov na pade pod mejo α. V vsaki iteraciji najprej določimo nabor možnih poti sestopanja in najboljšo pot sestopanja. To storimo z algoritmom za dodajanje faktorjev 4.2. Nato iterativno združujemo razred z najmanjšim številom pojavnic (P') z razredom, ki mu je najbolj podoben (P''). Podobnost določimo s pomočjo metrike D. Podobnost dveh razredov določamo na podlagi preseka med množicama možnih poti sestopanja. Pri tem je mišljeno, da iščemo poti takih razredov, v katerih nastopa kaka pot sestopanja, ki da v obeh razredih dobra rezultata, ki pa nista nujno tudi najboljša. To smo lahko dosegli s tem, da kot izhod iz algoritma za dodajanje faktorjev 4.2 nismo podali le najboljše poti, ampak tudi množico možnih kandidatov poti. Kadar imamo na voljo le zelo majhno število pojavnic, uporabimo tudi slovnične informacije za določanje metrike D. Tako lahko na primer damo manjšo vrednost metrike, če opazimo, da se zaporedji razlikujeta po vrsti besedne vrste, kot pa če se razlikujeta po sami besedni vrsti. Natančno obliko metrike D bomo kasneje eksperimentalno določili.

4.5.6 Kriteriji za končno izbiro poti sestopanja

Šele potem, ko smo združili različna zaporedja oznak POS v razrede, se lahko odločamo za eno samo pot sestopanja za dani razred. Podobno kot algoritem 4.2, ki je dal rezultate za določeno zaporedje oznak POS, tudi algoritem 4.3 kot rešitev vrne množico možnih kandidatov. Končno rešitev izberemo tako, da primerjamo perpleksnosti modelov na razvojni množici, velikost modelov (velikost datoteke) in čas ocenjevanja verjetnosti zaporedja besed.

Končna izbira je kompromis med pričakovano uspešnostjo modela pri razpoznavanju govora ter med časovnimi in prostorskimi zahtevami njegove uporabe. Postavimo lahko mejo, za koliko se lahko poveča velikost modela, če pri tem dosežemo določeno zmanjšanje perpleksnosti. Ker predstavlja ocenjevanje hipotez z jezikovnimi modeli v drugem prehodu glede na celoten postopek razpoznavanja govora le majhen del časove zahteve, smo se odločili postaviti mejo v prid zmanjšanju perpleksnosti.

Algoritem 4.4: Algoritem za izbiro končne rešitve iz množice kandidatov.

Input: $\mathcal{M}$
Output: $\hat{M}$
begin
  Sortiraj $\mathcal{M}$ po velikosti
  $\hat{M} \longleftarrow$ Poišči najmanjši model
  for $\forall M \in \mathcal{M}$ **do**
    Določi γ in δ
    if $(|M| > |\hat{M}|\ \ \&\ \ T(M) > T(\hat{M})$ **then**
      Preskoči
    if $T(M) > T(\hat{M}) + \gamma$ **then**
      $\hat{M} \longleftarrow M$
    if $T(M) > T(\hat{M})\ \ \&\ \ |M| < |\hat{M}| + \delta$ **then**
      $\hat{M} \longleftarrow M$

Postopek izbire končne poti je predstavljen v algoritmu za izbiro poti sestopanja 4.4. Izbiro začnemo pri najmanjšem modelu v naboru. Nato pregledamo vse modele v vrstnem redu od najmanjšega do največjega. Če je kateri drugi model večji in daje slabše rezultate perpleksnosti, ga preskočimo. Če je zmanjšanje perpleksnosti večje od parametra γ, preidemo na ta model. Tudi kadar je zmanjšanje perpleksnosti manjše, preidemo na model, če je istočasno povečanje velikosti modela manjše od parametra δ. Algoritem je nastavljen tako, da parametra γ in δ ponovno izračunamo pri vsakem prehodu na novo rešitev. Ko s postopkom preverimo vse kandidate v množici, algoritem končamo. Končna rešitev je $\hat{M}$.

4.5.7 Implementacija rešitev v končni algoritem

Do sedaj opisane algoritme smo definirali za sistematični pregled iskalnega prostora. Ker bi pregled celotnega iskalnega prostora zahteval preveč časa, smo že pri algoritmu za dodajanje faktorjev 4.2 močno zmanjšali iskalni prostor, s tem ko smo izločali kandidate, ki so dali rezultat, ki se je od najboljšega razlikoval za določeno mejo. S spreminjanjem te meje lahko vplivamo na število možnih poti sestopanja, ki jih pregledamo, in s tem tudi na čas določanja poti sestopanja.

V algoritmu za združevanje zaporedij oznak POS 4.3 imamo parametra α in β, s katerima določamo število iteracij združevanja poti in iskanja poti sestopanja. Z ustrezno izbiro parametrov lahko tudi vplivamo na čas izvajanja algoritmov. Parametra lahko določimo eksperimentalno.

Med samim testiranjem poti sestopanja se lahko ponavljajo že pregledane kombinacije. V tem primeru je pametno zabeležiti že dobljene rezultate, čeprav jih v trenutni iteraciji algoritma ne potrebujemo več.

Če združimo do sedaj opisane algoritme ter pravkar opisane metode za pohitritev delovanja, dobimo algoritem 4.5, ki predstavlja končni postopek za določanje strukture kontekstno odvisnih modelov. V algoritmu se izraz "Išči rešitev" nanaša na algoritem za dodajanje faktorjev 4.2, izraz "Izberi končno pot" pa na algoritem za izbiro poti sestopanja 4.4.

Končni rezultat algoritma je množica rešitev oblike $(n, k, \bar{P}, \hat{M})$, kjer n označuje red n-grama, k označuje indeks faktorja v besedi, $\bar{P}$ je razred možnih oznak POS, $\hat{M}$ pa struktura modela oz. pot sestopanja. Algoritem je zasnovan tako, da vsako možno zaporedje oznak POS zapade v enega izmed razredov oznak. Na začetku poglavja smo z BO označili funkcijo, ki danemu zaporedju oznak POS priredi strukturo modela. Vrednosti funkcije lahko sedaj definiramo s pomočjo množice rešitev $\mathcal{M}$ na sledeči način: vrednost funkcije BO za dano zaporedje oznak POS je tista struktura modela iz množice $\mathcal{M}$ v rešitvi, ki vsebuje razred zaporedij oznak POS z danim zaporedjem; pri tem upoštevamo red modela n in indeks faktorja k, za katerega iščemo model.

4.6 Postopek učenja modelov s kontekstno strukturo

Najpogosteje izvajan korak v zgoraj opisanem postopku za določanje poti sestopanja je učenje modelov na izbrani učni množici in testiranje na razvojni množici ciljne baze. Modeli, predstavljeni v podpoglavju 4.2, temeljijo izključno na oblikoskladenjskih informacijah. Predstavljeni modeli s kontekstno odvisno strukturo pa vsebujejo kot faktorje tudi besede. Za učenje modelov z besedami so primerni veliki korpusi besedil, ki lahko vsebujejo po več sto milijonov besed. Za učenje modelov s slovničnimi informacijami pa so

Algoritem 4.5: Sestavljen algoritem za določanje poti sestopanja kontekstno odvisnih modelov.

Input: N
Input: K
Input: $\mathcal{P}_n \quad \forall n \in \{1, ..., N\}$
Output: $\mathcal{S}$
Output: $\mathcal{P}_n^* \quad \forall n \in \{1, ..., N\}$

begin
  $\mathcal{S} = \emptyset$
  for $n \in 1, ..., N$ **do**
    for $k \in 1, ..., K$ **do**
      $\mathcal{P}^* \longleftarrow \mathcal{P}_N$
      Določi α
      repeat
        for $\forall \bar{P} \in \mathcal{P}^*$ **do**
          $\mathcal{F}(\bar{P}) \leftarrow$ Nabor možnih faktorjev za kombinacijo $\bar{P}$
          $\mathcal{M}(\bar{P}) = \emptyset$
          $\hat{M}(\bar{P}) \longleftarrow$ Ni rešitve
          $(\mathcal{M}(\bar{P}), \hat{M}(\bar{P})) \longleftarrow$ **Išči rešitve** $(\mathcal{F}(\bar{P}), \mathcal{M}(\bar{P}), \hat{M}(\bar{P}))$
        Določi β
        repeat
          $\bar{P}' = \arg\min_{\bar{P}' \in \mathcal{P}}(C(\bar{P}))$
          $\bar{P}'' = \arg\min_{\bar{P}'' \in \mathcal{P} \setminus \{\bar{P}'\}} D(P', P'')$
          $\mathcal{P}^* \leftarrow$ Združi $\bar{P}'$ in $\bar{P}''$
        until *Število razredov* $< \beta$
      until *Število razredov* $< \alpha$
      for $\forall \bar{P} \in \mathcal{P}^*$ **do**
        $(\mathcal{M}(\bar{P}), \hat{M}(\bar{P})) \longleftarrow$ **Išči rešitve** $(\mathcal{F}(\bar{P}), \mathcal{M}(\bar{P}), \hat{M}(\bar{P}))$
        $\hat{M}(\bar{P}) \longleftarrow$ **Izberi končno pot** $(\mathcal{M}(\bar{P}))$
        $\mathcal{S} \longleftarrow \mathcal{S} \cup (n, k, \bar{P}, \hat{M})$

primerni korpusi, označeni z MSD. Zaradi dragega in zamudnega označevanja so ti korpusi precej manjši.

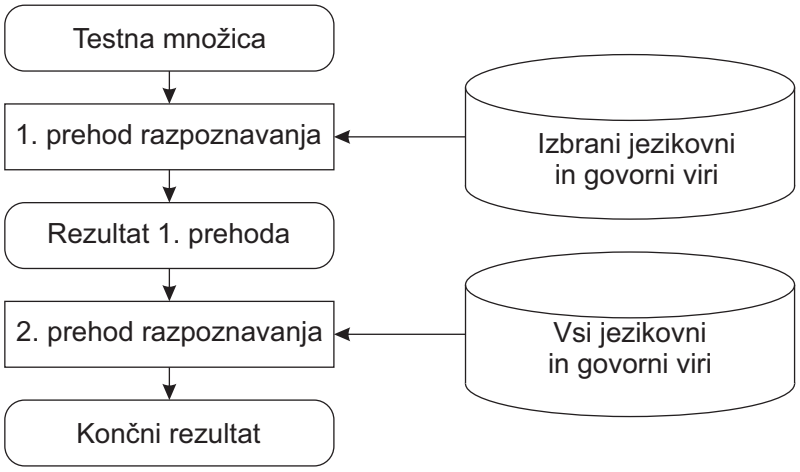
Manjši korpus pa ima prednost, da lahko na njem modele hitreje naučimo. Pri učenju modelov v postopku določanja poti sestopanja bomo predpostavili, da bo rezultat z večjo učno množico dajal boljše rezultate, vendar pa pri tem ne bo potrebno spremeniti poti sestopanja. Zato smo se odločili, da bomo kot učni korpus v tem postopku uporabili majhen korpus, označen z MSD.

Ko smo določili pot sestopanja, naučimo končni model. V okviru te naloge smo se pri tem tudi pri faktorju za besede in modelih, ki vsebujejo neko besedo v poti sestopanja, omejili le na majhne korpuse, ker s tem pridobimo pri hitrosti učenja.

Poglavje 5

Zasnova iskalnega algoritma za uporabo jezikovnih modelov

V tem poglavju bomo definirali iskalni algoritem, v katerem smo uporabili jezikovne modele s kontekstno odvisno strukturo. Osnova predlaganega iskalnega algoritma je običajni dvoprehodni algoritem za razpoznavanje govora. Njegova struktura je predstavljena na sliki 5.1.



Slika 5.1: Splošna oblika dvoprehodnega sistema za razpoznavanje govora.

5.1 Algoritem razpoznavanja v prvem prehodu in označevanje hipotez

V prvem prehodu razpoznavanja uporabimo tako akustične modele kot osnovni besedni jezikovni model. Rešitev prvega prehoda bo sprva besedna mreža, ki predstavlja iskalni prostor hipotez razpoznavalnika ob zaključku razpoznavanja segmenta. Te besedne mreže

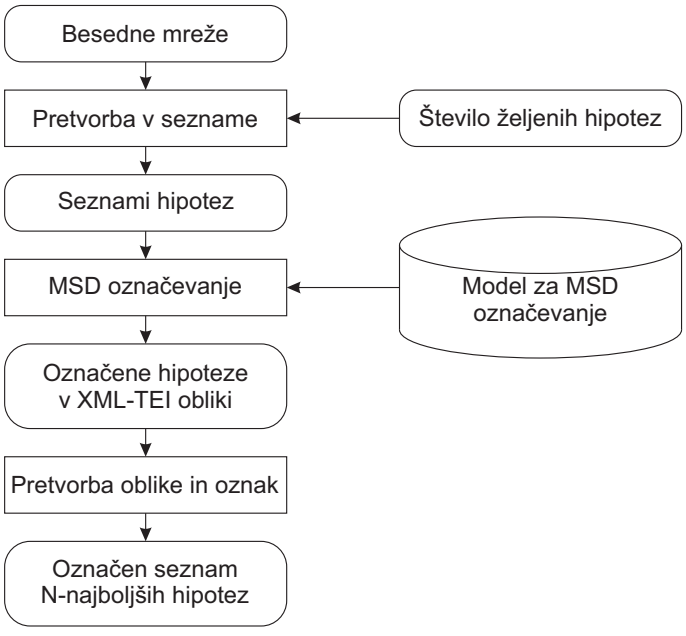
kasneje pretvori v sezname N najboljših hipotez. Pri sami pretvorbi definiramo število N, tj. največje število hipotez, ki jih izluščimo iz besedne mreže in jih zapišemo na seznam.

Za vsak segment imamo sedaj na voljo množico hipotez, h katerim so pripisane še akustična in jezikovna verjetnost ter število besed v hipotezi. Te tri vrednosti bomo s skupno besedo imenovali ocene. Hipoteze so v prvem prehodu ocenjene po enačbi

$$P(W) = \alpha P_A(W) + \beta P_L(W) + \gamma C(W), \tag{5.1}$$

kjer so P_A[1], P_L, $C(W)$ oznake za logaritem akustične verjetnosti, logaritem jezikovne verjetnosti in število besed v hipotezi; z oznakami α in β sta označeni uteži akustičnega in jezikovnega modela; koeficient γ pa predstavlja utež vrinjene besede (angl. word insertion penalty).

Naslednji korak v dvoprehodnem algoritmu je označevanje MSD vseh hipotez. Za označevanje slovenskega jezika smo pri izvedbi sistema uporabili označevalnik Obeliks. Sezname obdelamo tako, da jih pretvorimo iz oblike XML-TEI v običajno obliko zapisa hipotez, tj. po ena hipoteza na vrstico v datoteki. Nato oznake MSD pretvorimo v obliko, ki smo jo definirali v okviru te naloge. Ta oblika vsebuje le 5 podatkov za vsako besedo. Ti seznami služijo kot izhodiščni podatek za drugi prehod razpoznavanja. Postopek pretvorbe je prikazan na sliki 5.2.



Slika 5.2: Pretvorba izhoda razpoznavalnika v prvem prehodu v sezname N najboljših hipotez za ponovno ocenjevanje v drugem prehodu.

[1]Za jezikovno modeliranje in razpoznavanje govora se običajno uporabljajo logaritmi verjetnosti zaradi lažje implementacije v orodja in natančnosti v računanju. Zaradi preglednosti v tej nalogi uporabljamo oznako P za logaritem verjetnosti, in ne za verjetnost, kot je to običajno v matematičnem označevanju.

5.2 Ocenjevanje hipotez v drugem prehodu

Za izbiro najboljše hipoteze potrebujemo tudi v drugem prehodu nabor ocen za vse hipoteze. Uporabili bomo ocene iz prvega prehoda in dodali ocene, ki jih bomo dobili v drugem prehodu. Končni nabor ocen je:

- P_A: akustična verjetnost v prvem prehodu,
- P_L: jezikovna verjetnost klasičnega n-gramskega modela v prvem prehodu,
- P_f1: verjetnost, dobljena iz kontekstno odvisnega modela za besede v drugem prehodu,
- P_f2: verjetnost, dobljena iz kontekstno odvisnega modela za razširjene besedne vrste v drugem prehodu,
- P_f3: verjetnost, dobljena iz kontekstno odvisnega modela za poenostavljene oznake MSD v drugem prehodu, in
- $C(W)$: število besed v hipotezi.

Prvi dve in zadnjo oceno prevzamemo iz prvega prehoda. Za ocenjevanja z ostalimi tremi modeli uporabimo algoritem za določanje ocene 5.1.

Algoritem 5.1: Algoritem za ocenjevanje hipotez z jezikovnimi modeli s kontekstno odvisno strukturo.

Input: $W = (w_1, w_2, ..., w_{C(W)})$
Output: P_{f1}
Output: P_{f2}
Output: P_{f3}
1 **begin**
2 **for** $k \in \{1, 2, 3\}$ **do**
3 $P = 0$
4 **for** $i \in 1, ..., C(W)$ **do**
5 $P^* \longleftarrow$ Izberi model glede na: i, k in $(POS_{i-5}, ..., POS_i)$
   $P = P + P^*(w_{i-6}, ..., w_i)$
6 $P_{fk} = P$

Vhod algoritma je hipoteza W, izhod pa logaritem verjetnosti P. Izbira modela se nanaša na iskanje ustreznega modela v množici rešitev, kot smo jo opisali v prejšnjem poglavju, glede na faktor, ki ga ocenjujemo, in zaporedje oznak POS. Upoštevati moramo tudi, kateri model izbrati glede na mesto besede v hipotezi, saj pri prvih besedah v vsaki hipotezi ne moremo uporabiti modelov visokih redov. Kočno verjetnost, po kateri razvrstimo rešitve, pa dobimo po enačbi

$$P(W) = \alpha P_A(W) + \beta P_L(W) + \delta_1 P_{f1}(W) + \delta_2 P_{f2}(W) + \delta_3 P_{f3}(W) + \gamma C(W). \quad (5.2)$$

5.3 Optimizacija uteži in drugih parametrov drugega prehoda

Ocene v enačbah 5.1 in 5.2 dobimo s pomočjo modelov. K temu pa moramo določiti še vrednosti vseh uteži. V prvem prehodu to storimo s testiranjem razpoznavanja razvojne množice pri različnih vrednostih uteži. Na koncu izberemo uteži, ki dajo najboljši rezultat. Postopek imenujemo optimizacija uteži. Uspešnost razpoznavalnika merimo z razpoznavanjem na testni množici.

V prvem prehodu imamo tri uteži: α, β in γ. Iskalni algoritmi razpoznavalnikov govora iščejo hipotezo, ki daje najboljši rezultat. Ker se ta izbira ne spremeni, če vse ocene vseh hipotez množimo z določenim faktorjem (različnim od 0 seveda), lahko enačbi 5.1 in 5.2 poenostavimo tako, da ju delimo z α. Utež akustičnega modela s tem fiksno postavimo na vrednost 1. Sedaj ostaneta še dva prosta parametra za optimizacijo: β in γ.

Podobno lahko storimo v drugem prehodu, vendar v tem primeru dobimo 5 prostih parametrov. Sistematično preizkušanje vseh možnih kombinacij bi bilo precej zahtevno opravilo. Če bi na primer za vsak parameter hoteli preizkusiti 10 možnih vrednosti, bi skupno to pomenilo 100.000 različnih kombinacij vrednosti parametrov. Pri ocenjevanju optimalnih parametrov imamo opravka le z že izračunanimi verjetnostmi, ki jih množimo in seštevamo. Ta postopek je veliko hitrejši kot pa optimizacija v prvem prehodu, kjer moramo za vsako novo utež ponoviti postopek razpoznavanja. Kljub temu pa vsako preverjanje rezultata pri določenih utežeh traja nekaj sekund, kar je še vedno preveč, da bi preverili vse možne kombinacije.

Za namen hitrega iskanja optimalnih uteži smo v okviru naloge razvili namensko orodje. V njem smo implementirali algoritem optimizacije uteži 5.2.

Vhodni podatki algoritma so vse ocene in število napak za vsako hipotezo v razvojni množici. Vhod algoritma so tudi začetne vrednosti parametrov, izhod pa so optimirane vrednosti parametrov. Algoritem lahko ponovimo večkrat pri različnih začetnih vrednostih.

Algoritem optimizacije uteži 5.2 deluje tako, da najprej izračuna število napak pri vhodnih vrednostih parametrov. Nato za vsak parameter določi spodnjo in zgornjo mejo ter poišče optimalno vrednost tega parametra. Pri tem se vrednosti ostalih parametrov ne spreminjajo. Ta postopek ponovimo za vse parametre. Ko zaključimo optimizacijo vseh parametrov, postopek večkrat ponovimo. Število ponovitev določimo s parametrom i_{max}. V vsaki ponovitvi zmanjšamo meje za spodnje in zgornje vrednosti parametrov, pri čemer se število korakov med mejama ne spreminja. Tako v vsaki ponovitvi algoritma iščemo optimalne vrednosti na vedno manjših razmikih med vrednostmi, ki jih preizkušamo.

Predhodne raziskave, ki smo jih opravili v okviru doktorske disertacije, so potrdile povezavo med utežjo jezikovnih modelov in utežjo vrinjene besede. Eksperimentalno je

Algoritem 5.2: Algoritem za optimizacijo parametrov.

Input: $P_A(W), P_L(W), C(W), P_{f1}(W), P_{f1}(W), P_{f2}(W), P_{f3}(W), E(W) \quad \forall W$
Input: $\Lambda = \{\alpha, \beta, \gamma, \delta_1, \delta_2, \delta_3\}$
Output: $\Lambda = \{\alpha, \beta, \gamma, \delta_1, \delta_2, \delta_3\}$

```
1  begin
2      for i ∈ 1, ..., i_max do
3          E ⟵ Določi število napak pri parametrih Λ
4          for λ ∈ Λ do
5              λ_min ⟵ Določi spodnjo mejo za parameter
6              λ_max ⟵ Določi zgornjo mejo za parameter
7              for λ̂ ∈ λ_min, ..., λ_max do
8                  E* ⟵ Določi število napak, tokrat z λ̂
9                  if E* < E then
10                     Λ ⟵ Vstavi λ̂ za novo vrednost od λ
11                     E = E*
```

bilo potrjeno, da se ob spreminjanju uteži za verjetnosti, ki so dobljene iz jezikovnih modelov, spremeni tudi optimalna vrednost uteži vrinjene besede [13]. Zato pri algoritmu za ocenjevanje 5.1 upoštevamo tudi ta parameter.

Algoritem za optimizacijo uteži 5.2 lahko ponovimo večkrat, pri čemer spreminjamo število upoštevanih hipotez. Tudi pri tem so naše predhodne raziskave pokazale, da dobimo najboljše rezultate pri določenem številu upoštevanih hipotez. Na koncu optimizacijskega postopka tako dobimo skupno 6 parametrov, ki jih uporabimo v razpoznavanju na testni množici.

V algoritmu za ocenjevanje uporabljamo izračun števila napak. To storimo tako, da za vsak segment pregledamo vse hipoteze in jim določimo skupno oceno po enačbi 5.2. Nato poiščemo tisto hipotezo, ki ima najboljšo oceno, in na njej izračunamo število napak.

5.4 Zasnova algoritma za prilagajanje uteži

Vpeljali smo tri jezikovne modele, ki ocenjujejo besede, oznake POS in oznake MSD ter pri tem uporabljajo informacije iz zgodovine besede. Predpostavili smo, da bo model, ki ocenjuje oznake POS, lahko pomagal zmanjšati napake, kjer je bila napačno razpoznana besedna vrsta. Podobno smo predpostavili, da bo model z oznakami MSD pomagal pri napakah, kjer je napačno razpoznana besedna oblika oziroma končnica besede.

V optimizacijskem postopku, kot smo ga opisali podpoglavju 5.3, dobimo kot rezultat fiksne uteži, ki jih uporabljamo pri vsaki hipotezi, ki jo ocenjujemo. V nadaljevanju bomo optimizacijski postopek posplošili. Pri tem želimo določiti različne vrednosti opti-

mizacijskih uteži za različne hipoteze. Celotni algoritem razpoznavanja govora bo tako med samim razpoznavanjem prilagajal uteži modelov.

Da lahko definiramo tak algoritem, moramo najprej definirati kriterije, na podlagi katerih se določi utež. Imenujmo jih slovnične značilke hipotez. Ker takšen sistem doslej še ni bil zasnovan, ga bomo definirali v osnovni obliki.

Kot slovnične značilke bomo uporabili deleže pregibnih besed v hipotezah. Ker se delež pregibnih besed razlikuje med hipotezami, ki jih dobimo pri nekem govornem segmentu, bomo uporabili razmerje iz najboljše hipoteze po prvem prehodu. Izkušnje kažejo, da ima najboljša hipoteza po prvem prehodu večkrat tudi najmanjše število napak. Ostale hipoteze pa se običajno le malo razlikujejo od prve hipoteze.

Za namen uporabe različnih uteži moramo tudi prilagoditi postopek za iskanje optimalnih uteži v enačbi 5.2. Začnemo enako, kot smo opisali v algoritmu za ocenjevanje 5.1. Ta optimizacija poteka na celotni razvojni množici. Nato analiziramo besedne vrste v prvih hipotezah segmentov iz razvojne množice. Glede na dobljene rezultate razdelimo vse segmente v razrede. Poskušamo lahko različna števila razredov. Za vsak razred ponovno optimiramo vrednosti parametrov. Pri tem kot izhodiščno točko izberemo končni rezultat pri optimizaciji na celotni razvojni množici. Prilagojene vrednosti uteži nato preizkusimo na testni množici.

Poglavje 6

Eksperimentalni sistem

V tem poglavju bomo opisali vse uporabljene pisne in govorne vire, s katerimi smo izvajali eksperimente za preverjanje učinkovitosti kontekstno odvisnih modelov. Predstavili bomo razpoznavalnik tekočega govora, razvit na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, ki smo ga nadgradili z dodanim drugim prehodom razpoznavanja in novimi jezikovnimi modeli. Opisali bomo posamezne komponente razpoznavalnika in parametre, ki smo jih nastavili za delovanje razpoznavalnika.

Nato bomo podrobneje opisali implementacijo algoritmov za določanje strukture jezikovnih modelov, algoritem za vključitev modelov v postopek razpoznavanja in postopek optimizacije parametrov uporabe jezikovnih modelov v drugem prehodu. Na koncu bomo opisali še postopek ocenjevanja uspešnosti razpoznavanja.

6.1 Uporabljeni jezikovni viri

6.1.1 Korpus FidaPLUS

Za gradnjo besednih jezikovnih modelov za razpoznavalnik z velikim slovarjem besed potrebujemo čim večji učni korpus jezika. Največji nam dosegljiv jezikovni vir je slovenski referenčni korpus FidaPLUS [3]. Sestavljen je iz knjižnega, časopisnega, revijalnega, internetnega in drugega gradiva, ki je v večjem delu lektorirano. Skupaj vsebuje približno 621 milijonov besed. Korpus je nastal kot kvalitativna in kvantitativna nadgradnja korpusa FIDA [14].

Korpus FidaPLUS je zapisan v obliki XML-TEI, v kateri so označene vse pojavnice (besede in ločila) s pripadajočimi lastnostmi. Tako je pri vsaki besedi pripisana lema in oznaka MSD po sistemu MULTEXT-EAST. Ker smo imeli na voljo tudi druge jezikovne vire, ki so označeni z novejšimi orodji za lematizacijo in označevanje, smo se odločili, da

bomo iz vsebine korpusa FidaPLUS uporabljali le neoznačene besede. Oznake MSD so bile kasneje v pomoč pri iskanju številnikov in okrajšav, kot bo opisano v nadaljevanju.

Končni cilj uporabe korpusa je izdelava jezikovnega modela. Ker bo ta namenjen uporabi v razpoznavalniku govora, smo korpus najprej preoblikovali v ustrezno obliko. Iz njega smo izločili vsa ločila, ker se ta v postopku razpoznavanja govora ne pojavljajo. Da bomo jezikovne modele, ki jih bomo izdelali kasneje, lažje vključili v razpoznavalnik govora, smo celoten korpus pretvorili v format, ki ustreza uporabljenim transkripcijam iz baze. To pomeni, da smo vse velike črke pretvorili v male črke, namesto šumnikov pa smo pisali ustrezne velike sičnike (npr. š $\rightarrow$ S). Odstranili smo tudi vsa naglasna znamenja ter druge simbole ob črkah, ki se uporabljajo predvsem v tujih besedah (npr. preglas).

6.1.2 Korpus in oblikoskladenjske specifikacije JOS

Namen projekta JOS je bil izdelati standardiziran in prosto dostopen označen korpus in prenova nabora oblikoskladenjskih oznak za slovenski jezik. V projektu sta bila izdelana dva korpusa: jos100k in jos1M, ki vsebujeta 100.000 oz. 1 milijon besed. Besedila so bila izbrana z vzorčenjem iz korpusa FidaPLUS, ob upoštevanju uravnoteženosti in reprezentativnosti, kvalitete besedil in avtorskih pravic.

Namen projekta JOS je bila tudi standardizacija nabora oznak za slovenščino. Pri zasnovni nabora oznak so bili upoštevani različni jezikovni viri za slovenščino. Končne specifikacije vsebujejo 1.908 različnih možnih oznak. Za označevanje korpusa jos100k so bile oznake v uporabljenih besedilih v korpusu FidaPLUS pretvorjene v obliko JOS, nato so bile strokovno pregledane in popravljene. Zaradi obsega korpusa jos1M so bile v njem preverjene in popravljene le sumljive oznake.

6.1.3 Označevalnik Obeliks in korpus ssj500k

Obeliks je statistični oblikoskladenjski označevalnik, ki je bil razvit v projektu Sporazumevanje v slovenskem jeziku. Deluje tako, da vhodnemu besedilu pripiše oznake MSD. Pri tem pa oznake izbira iz nabora oznak po specifikacijah JOS. Obeliks vključuje tudi lematizator, ki na podlagi besede in oznake MSD besedam priredi leme.

Ena temeljnih komponent za označevanje besedil je model za označevanje. Ta je pri Obeliksu naučen na korpusu ssj500k, ki je sestavljen iz korpusa jos100k in dodatnih 400.000 besed iz korpusa jos1M. Vse oznake in leme v sestavljenem korpusu so bile še enkrat pregledane. Tako predstavlja korpus ssj500k največjo množico ročno označenih besedil v slovenskem jeziku.

Tako označevalnik kot učni korpus sta prosto dostopna na spletni strani projekta, zato smo se odločili za njuno uporabo v okviru doktorske naloge.

6.2 Govorna baza BNSI

Razvoj jezikovnih modelov, opisanih v tej nalogi, je namenjen izboljšanju razpoznavanja tekočega govora z velikim slovarjem besed. Da lahko modele testiramo, potrebujemo ustrezno govorno bazo slovenskega jezika. Edina primerna baza, ki je na voljo v ta namen, je slovenska baza Broadcast News (BNSI), ki je nastala v sodelovanju med Fakulteto za elektrotehniko, računalništvo in informatiko Univerze v Mariboru in RTV Slovenija.

Baza BNSI vsebuje avdio posnetke 42 oddaj slovenske državne televizije. Ti vsebujejo skupno 36 ur govornega materiala, ki je razdeljen na tri dele. Prvi in največji del so učni podatki, ki se uporabljajo za učenje HMM-ov. Učni podatki vsebujejo približno 29 ur govornega materiala. Drugi del so razvojni podatki, s katerimi prilagajamo parametre razpoznavalnika z namenom doseganja večje uspešnosti razpoznavanja govora. Zadnji del so testni podatki, s katerimi ocenjujemo uspešnost razpoznavalnika. Razvojni in testni podatki vsebujejo po približno 3 ure in pol govornih segmentov. Vsi segmenti v posnetkih, ki vsebujejo slovenski govor, imajo pripadajočo transkripcijo in so še dodatno označeni glede na govorca, njegov spol, narečje, kvaliteto posnetka in F-kategorijo. Segmenti brez govora (npr. glasba), segmenti s tujim govorom in segmenti s prikrivajočim se govorom več oseb so izločeni.

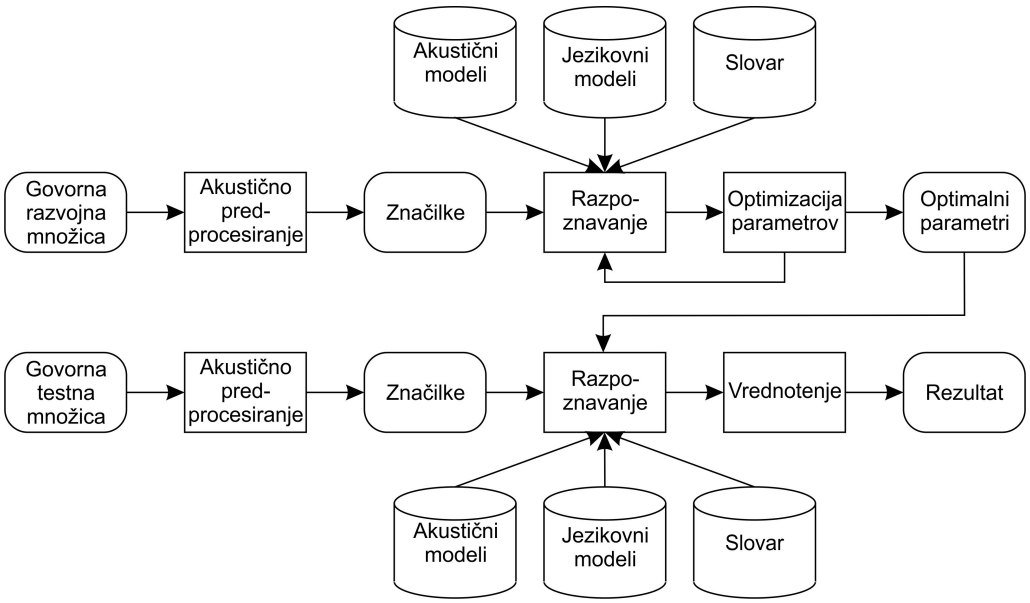
Posnetki v BNSI so segmentirani, tako da vsak segment predstavlja govor v enakih okoliščinah. Glede na glavne značilnosti govora je vsak segment v posnetkih označen z eno izmed F-kategorij. Te so [10]:

- F0 – pripravljeni (brani) govor,
- F1 – spontani govor,
- F2 – govor s slabšo zvočno kvaliteto (vključuje prenos preko telefonske linije),
- F3 – govor z glasbo v ozadju,
- F4 – govor s hrupnim ozadjem,
- F5 – tuji govorec,
- FX – ostalo.

Ob govornem delu baze BNSI imamo še dodaten tekstovni korpus, ki vsebuje 11 milijonov besed. Sestavljen je iz scenarijev različnih oddaj iz obdobja štirih let. Besedilni korpus uporabljamo za sestavljanje slovarjev in učenje jezikovnih modelov. Skupno vsebuje 175.000 različnih besed. Tekstovni korpus je lahko v pomoč pri izdelavi jezikovnih modelov, vendar jih v okviru te naloge nismo uporabljali. Več podrobnosti o bazi BNSI lahko najdemo v [73].

6.3 Osnovna struktura razpoznavalnika govora UMB Broadcast News

V izvedenih eksperimentih smo izhajali iz osnovne strukture razpoznavalnika UMB Broadcast News, ki je predstavljena na sliki 6.1



Slika 6.1: Razpoznavalnik govora z velikim slovarjem besed UMB BN.

Razpoznavalnik v tej obliki izvaja prvi prehod razpoznavanja. Sistem smo nadgradili s tvorjenjem besednih mrež in seznamov N najboljših hipotez ter drugim prehodom razpoznavanja. Podrobnejša razlaga posameznih komponent sistema je podana v naslednjih podpoglavjih.

6.4 Prvi prehod

V prvem prehodu razpoznavanja uporabljamo za iskanje najboljše hipoteze za posamezen govorni segment tako akustične kot jezikovne vire. V tem prehodu izhajamo iz avdio posnetkov. Te najprej pretvorimo v datoteke, ki vsebujejo pripadajoče vektorje značilk. Vektorji značilk nato služijo kot vhodni podatek za razpoznavalnik govora, ki pri razpoznavanju uporablja slovar ter jezikovni in akustični model. V prvem prehodu razpoznavanja kot tudi v postopku učenja akustičnih modelov smo uporabljali orodja iz paketa HTK [69], vključno z dodanim razpoznavalnikom HDecode.

6.4.1 Izločanje značilk

Najpogosteje uporabljane značilke pri razpoznavanju govora so koeficienti MFCC in PLP. Pri izbiri značilk v našem sistemu smo upoštevali rezultate predhodnih eksperimentov, ki

so bili izvedeni na bazi izgovarjav BNSI. V [74] je bilo pokazano, da lahko z ustrezno izbiro značilk in parametrov za njihov izračun nekoliko izboljšamo uspešnost razpoznavanja.

Parameter vhodnega FIR-filtra (glej enačbo 2.1) je 0,97. Uporabljene značilke so koeficienti MFCC, izračunani na Hammingovih oknih dolžine 32 ms in v razmiku 10 ms. Izračunali smo 12 značilk in pri tem uporabili 26 kanalov in 22 kepstralnih filtrov. Značilkam MFCC smo dodali še energijo. Same datoteke, v katere smo shranjevali značilke, ne vsebujejo prvih in drugih odvodov značilk, ki se običajno tudi uporabljajo med razpoznavanjem, pač pa odvode med samo uporabo računata orodje za učenje akustičnih modelov in orodje za razpoznavanje.

6.4.2 Akustični modeli

Ker nimamo na voljo natančnih pravil za grafemsko-fonemsko pretvorbo za slovenski jezik, je bila uporabljena grafemska transkripcija besed v učni množici, kar pomeni, da smo na mestu fonemov, ki naj bi predstavljali izgovorjave besede, zapisali kar njene črke. Modele smo naučili na učni množici baze BNSI.

Med učenjem smo najprej generirali monofonske modele za vseh 25 slovenskih grafemov in temu dodali modele za tišino. V naslednjih iteracijah učenja smo modele pretvorili v modele z več Gaussovimi porazdelitvami. Monofonske modele smo uporabili kot izhodišče za učenje trifonskih modelov. Pri učenju trifonskih modelov smo uporabili tudi vezanje stanj. Tudi trifonske modele smo na koncu pretvorili v modele z več Gaussovimi porazdelitvami, ki smo jih iterativno učili. Med postopkom učenja smo izvajali tudi časovno poravnavo med transkripcijami in posnetki ter v primeru, da je bila na določenem segmentu poravnava neuspešna, ta segment izločili iz učne množice.

Končni uporabljeni modeli so medbesedni trigrafemski modeli s 16 Gaussovimi porazdelitvami in vezanimi stanji. Skupno smo generirali 8327 vezanih modelov.

6.4.3 Slovarji besed

Besede, ki smo jih vključili v slovar, smo iskali v korpusu FidaPLUS [3]. Slovar smo zgradili tako, da smo iz korpusa izpisali vse besede in njihove frekvence. Izločili smo številke in besede, v katerih so se pojavile številke, različna ločila in drugi znaki, preostale besede pa smo razvrstili po frekvenci. Ob začetku gradnje slovarja smo določili njegovo želeno velikost in dodajali besede v vrstnem redu, kot smo jih razvrščali. Ko smo prišli do želene velikosti, smo dodali še vse besede, ki so se pojavile z enako pogostostjo kot beseda, s katero smo dosegli želeno velikost. Zgradili smo 4 slovarje velikosti 60k, 100k, 200k in 300k besed. Velikost slovarja 60k smo izbrali zato, ker se ta pojavlja v veliko različnih poskusih in smo tako olajšali primerjavo z drugimi jeziki in eksperimenti. Tudi slovarji ostalih velikosti, ki so izbrane v enakomernih presledkih, so primerljivi z raziskavami drugih

Tabela 6.1: Velikosti uporabljenih slovarjev za razpoznavanje govora v prvem prehodu in pripadajoče vrednosti OOV na testni množici BNSI

Slovar	Velikost	OOV [%]
60k	60.022	6,94
100k	100.189	3,44
200k	201.034	1,64
300k	301.357	1,02

raziskovalcev, kar bo omogočilo primerjavo z izvedenim sistemom. Natančne velikosti slovarjev so podane v tabeli 6.1. Od velikosti slovarja je odvisen delež besed izven slovarja (OOV), ki se pojavijo v testni množici. Ker so takšne besede vzrok nekaterih napak v sistemih za razpoznavanje govora, smo za vse uporabljene slovarje izračunali ta delež na testni množici BNSI. Rezultati so prav tako podani v tabeli 6.1

Vsaki besedi v slovarju smo morali dodati tudi fonetični zapis, ki pa je bil v našem primeru enak grafemskemu zapisu, le pri besedah, ki so vsebovale tuje črke, smo naredili pretvorbe, in sicer:

- x → ks
- w → v
- y → j
- q → k
- đ → dž

Te črke prihajajo iz drugih jezikov, predvsem angleščine. Pretvorbe smo izbrali glede na to, kateremu slovenskemu grafemu najbolje ustreza črka v izgovorjavi tuje besede. Težavna izbira je bila pretvorba črke y. Ta se v nekaterih besedah izgovarja podobno kot slovenski ‘i’, v drugih pa kot ‘j’, kar je posledica tega, da ‘y’ zavzema v angleščini včasih vlogo samoglasnika, včasih pa soglasnika. Odločili smo se za avtomatsko pretvorbo v grafem ‘j’. Ker se ‘y’ v slovarju pojavi več kot 1000-krat, nismo delali ročne pretvorbe, saj bi bila preveč časovno zahtevna.

6.4.4 Osnovni jezikovni modeli

Jezikovni modeli v prvem prehodu razpoznavanja so običajni n-gramski jezikovni modeli. Gradili smo jih na korpusu FidaPLUS, pri tem pa smo uporabljali zgoraj opisane slovarje velikosti 60k, 100k, 200k in 300k. Pred samo gradnjo jezikovnih modelov smo zamenjali nekatere nebesedne pojavnice, ki so bile v korpusu prisotne tudi po obdelavi, opisani v podpoglavju 6.1. Vse številke, zapisane s števili (arabskimi ali rimskimi), smo nadomestili

z oznako <number>. Tako se v jezikovnem modelu pojavijo le števniki, zapisani z besedo. Vse okrajšave smo nadomestili z oznako <okrajsava>, ker se te v govorjenem jeziku ne pojavljajo in zato niso uporabne v modelu za razpoznavalnik. Števnike in okrajšave smo prepoznali s pomočjo oznak MSD. Vse ostale pojavnice, ki očitno niso bile besede, ustrezne za jezikovni model, ker so vsebovale še kakšne druge simbole razen črk, pa smo nadomestili z oznako <alfanum>. Končna oblika korpusa, primerna za orodje, s katerim smo izdelali jezikovne modele, je preprosta tekstovna datoteka, v kateri meje med povedmi določa prehod v novo vrstico.

Razpoznavanje je kasneje potekalo z orodjem HDecode, ki pa deluje le z bigramskimi in trigramskimi jezikovnimi modeli. V nekaterih testih smo uporabili tudi štirigramske modele v dvoprehodnem algoritmu, vendar so rezultati pokazali, da ni bistvenih razlik v uspešnosti razpoznavanja v primerjavi z uporabo trigramskih modelov [12], zato smo se v nadaljevanju odločili, da bomo izdelali le bigramske in trigramske jezikovne modele. Pri tem smo preizkušali tako Good-Turingovo kot Knesser-Neyevo [10] glajenje in sestopanje po Katzu [34].

6.4.5 Razpoznavanje

Osnovni programski paket HTK vsebuje iskalnik HVite, ki pa ni primeren za zelo velike slovarje. V primeru velikih slovarjev in medbesednih trifonov postane manj učinkovit. K celotnemu paketu HTK je bil dodan iskalnik HDecode, ki je načrtovan posebej za razpoznavanje z velikim slovarjem. Obstajajo pa določene omejitve v njegovi uporabi – najpomembnejši sta, da deluje le z medbesednimi trifonskimi modeli (v našem primeru trigrafemskimi) in da podpira le bigramske in trigramske jezikovne modele [69].

HDecode je v osnovni konfiguraciji, ki je razpoložljiva na internetu (različica 3.4.1), omejen na slovarje velikosti do 2^{16} besed (≈ 65.000). Da smo lahko uporabili iskalnik HDecode za večje slovarje, je bil potreben poseg v izvorno kodo programa. Razlog za to omejitev iskalnika sta dve spremenljivki v programski kodi: `PronId`, ki enolično označuje par besede in neke pripadajoče fonemske transkripcije v slovarju med razpoznavanjem, in `LMId`, ki označuje mesto v jezikovnem modelu. Najdemo ju v datoteki `HTKLVRec/config.h`, gledano iz korenskega imenika orodja HTK. V originalni programski kodi sta ti spremenljivki definirani kot spremenljivki tipa `unsigned short` (16-bitno nepredznačeno celo število). S tem so njune vrednosti omejene na vrednosti od 0 do 65535. V programski kodi je že pripravljen del kode, ki definira ti spremenljivki s tipom `unsigned int` (32-bitno nepredznačeno celo število). Kateri del kode se prebere med njenim prevajanjem, je krmiljeno z logično konstanto v predprocesorski direktivi. S spremembo konstante smo dosegli prevajanje programa z drugačnim tipom spremenljivk in na ta način omogočili, da deluje iskalnik HDecode tudi z večjimi slovarji.

6.5 Izhodišče za drugi prehod

Rezultat razpoznavanja z orodjem HDecode je seznam najboljših najdenih hipotez za vsak govorni segment v množici, ki jo razpoznavamo. Če na teh hipotezah izvedemo ocenjevanje uspešnosti razpoznavanja, dobimo rezultat prvega prehoda.

Dodatno k najboljši hipotezi lahko razpoznavalnik vrne še besedno mrežo, ki predstavlja več možnih hipotez, ki jih je razpoznavalni algoritem ocenjeval za dani segment. Besedne mreže so opisane v obliki točk in povezav (graf). Povezave predstavljajo možne besede, ki jih je razpoznavalnik obravnaval. Vsem povezavam so pripisane tudi akustične in jezikovne verjetnosti.

Obstajajo orodja, ki so zmožna nadaljnje prehode razpoznavanja izvajati neposredno na besednih mrežah, vendar takšno izvajanje drugega prehoda za nas ni primerno, zato smo besedne mreže pretvorili v sezname najboljših hipotez. Pri tem se iz množice vseh različnih hipotez, ki jih je mogoče tvoriti v besedni mreži, v preprost seznam izpiše določeno število hipotez, ki dajejo najboljše rezultate. V nalogi smo se omejili na najboljših 1000 hipotez. Seznami N najboljših hipotez vsebujejo akustično in jezikovno verjetnost za vsako hipotezo.

Naslednji korak je bilo označevanje hipotez z oznakami MSD. V ta namen smo iz seznamov izluščili le gole hipoteze ter jih podali kot vhod označevalniku Obeliks. Označene hipoteze smo nato oblikovali v nov seznam N najboljših hipotez, vključno s pripadajočimi verjetnostmi.

Z orodji, ki so nam na voljo kot del paketa SRILM, smo lahko izvedli ponovno ocenjevanje jezikovnih verjetnosti na seznamih N najboljših hipotez s poljubnimi faktoriziranimi jezikovnimi modeli.

6.6 Uporaba preprostih morfoloških modelov v drugem prehodu

Več jezikovnih modelov (običajnih ali oblikoslovnih) v drugem prehodu uporabljamo tako, da z njimi ocenjujemo hipoteze v seznamu N najboljših hipotez. V ta namen za vsak segment sestavimo vrsto datotek.

Različne datoteke vsebujejo verjetnosti akustičnega modela, verjetnosti jezikovnega modela, števila besed, hipoteze v besedni obliki, hipoteze v faktorizirani obliki, verjetnosti ostalih jezikovnih modelov, števila napak posameznih hipotez. Prve tri vrednosti in hipoteze v besedni obliki dobimo iz prvega prehoda. Hipoteze v faktorizirani obliki dobimo z označevanjem hipotez v besedni obliki z oblikoskladenjskim označevalnikom. Število napak definiramo kot vsoto števil zamenjav besed, vrinjenih besed in izbrisanih

besed. Verjetnosti ostalih modelov dobimo s ponovnim ocenjevanjem hipotez v faktorizirani obliki s faktorizirami oblikoslovnimi jezikovnimi modeli. Vse verjetnosti so zapisane v logaritemski obliki.

Ko imamo na voljo vse verjetnosti v logaritemski obliki, sledi njihovo združevanje v obliki utežene vsote. Optimalne vrednosti uteži – parametre – določamo s pomočjo razvojne množice in algoritma za optimizacijo parametrov 5.2. Po izvedeni optimizaciji parametrov uporabimo optimalne vrednosti še enkrat na testni množici, kjer dobimo končni rezultat uspešnosti razpoznavanja z izbranimi modeli.

6.7 Učenje morfoloških modelov s kontekstno odvisno strukturo

Osnovni potek učenja modelov z morfološko odvisno strukturo smo že opisali v poglavju 4. Pri praktični izvedbi postopka učenja pa se je pokazalo, da je celoten postopek zelo dolgotrajen. Zato smo morali poiskati primerne vrednosti parametrov, ki pomembno vplivajo na hitrost učenja.

V tabeli 4.4 so navedena števila različnih možnih zaporedij faktorjev P. Vsako tako zaporedje je v začetku izvajanja algoritma učenja uporabljeno kot definicija razreda. Ker učenje izvajamo za vsak razred posebej, to pomeni, da čas trajanja učenja narašča s številom teh zaporedij. Za večino teh zaporedij ne pričakujemo, da jih bomo v praktični uporabi algoritma srečali. Zato se želimo omejiti le na tista zaporedja faktorjev, ki jih dejansko opažamo v domeni, v kateri bomo modele kasneje uporabljali. V ta namen smo uporabili razvojno množico baze BNSI in v njej poiskali vsa zaporedja faktorjev P. V tabeli 6.2 so zapisana števila različnih zaporedij glede na njihovo dolžino.

Tabela 6.2: Število zaporedij faktorjev P glede na dolžino, ki smo jih uporabili kot izhodiščno množico razredov.

N	Število zaporedij
1	30
2	475
3	3.003
4	8.666
5	13.682
6	15.478

Druga pomembna lastnost postopka učenja modelov, ki močno vpliva na časovno zahtevnost postopka, je nabor možnih poti sestopanja. Nabor možnih poti sestopanja smo najprej omejili tako, da bo prvi faktor, ki ga dodamo v pot sestopanja pripadal trenutni ali pa neposredno prejšnji besedi. Ta faktor je tisti, ki ga v postopku sestopanja zavržemo nazadnje. To omejitev smo izbrali zato, ker izkušnje kažejo, da imajo faktorji, ki so blizu

Tabela 6.3: Izbrane vrednosti parametra δ glede na število faktorjev v poti sestopanja, za katero določamo, ali jo bomo obdržali ali izločili.

Število faktorjev	δ
1	2,0
2	1,5
3	1,3
4	1,2
≥ 5	1,1

modeliranemu faktorju, največji vpliv na njegovo pogojno verjetnost. Pri izbiri nadaljnjih faktorjev je omejitev le dolžina n-grama. Dodatno pa smo omejili še največje dovoljeno število faktorjev v poti sestopanja oz. dolžino poti, in sicer na 8.

Najpomembnejši vidik pri izboljšanju časovne zahtevnosti učenja je izločanje poti s slabšimi rezultati. V algoritmu 4.2 to izločanje uravnavamo s parametrom δ, ki pove, za koliko lahko neka pot daje slabši rezultat od trenutno najboljše poti. Ker se vrednosti perpleksnosti manj spreminjajo pri dodajanju faktorjev pri daljših poteh, smo ta faktor določali odvisno od dolžine poti. V tabeli 6.3 so zapisane izbrane vrednosti parametra glede na število faktorjev.

Pri preizkušanju učenja se je pokazalo, da se kljub tej omejitvi pri sestavljanju daljših poti v postopku učenja pojavi velika množica možnih poti, kar ne le da predstavlja večjo časovno zahtevnost pri učenju in preverjanju uspešnosti teh poti, ampak pomeni tudi večjo zahtevo po prostoru na disku za shranjevanje izdelanih modelov. Zato smo v postopek izločanja manj obetavnih poti dodali še dodatne omejitve.

Najprej se omejujemo na določeno število poti. Mejo smo postavili pri 10 poteh, ki jih ohranjamo. Izberemo 10 poti z najmanjšo perpleksnostjo. Če najdemo še več poti, ki imajo enako perpleksnost kot najslabša pot izmed prvih desetih, potem tudi te poti ohranimo. Tako lahko ohranimo tudi nekaj več kot 10 poti med postopkom iskanja.

Nato vsako pot, ki jo dodajamo, primerjamo z vsemi ostalimi potencialnimi potmi, ki jih imamo v naboru. Če najdemo katero pot, ki predstavlja delno pot tiste poti, ki jo trenutno dodajamo, in perpleksnost trenutne poti ni za vsaj 5 % manjša, potem to pot izključimo. Drugače povedano: če neki poti dodamo faktor in se perpleksnost ne zmanjša za vsaj 5 %, potem štejemo, da ni smiselno dodajati tega faktorja.

Na koncu izločanja neobetavnih poti primerjamo med seboj še enako dolge poti. Če ugotovimo, da sta dve poti sestavljeni iz enakih faktorjev (v drugačnem vrstnem redu) in imata enako perpleksnost, ohranimo le eno, in sicer prvo izmed teh poti glede na sortiranje po ASCII vrednost zapisa poti.

Pri vsakem razredu, za katerega iščemo najboljše poti sestopanja, zmeraj uporabljamo isti način razvrščanja, tako da dosežemo, da se iste poti ohranjajo. To pospeši delovanje algoritma, ker v novih razredih ne bo potrebno izdelovati modelov, ki smo jih

definirali že pri prejšnjih modelih. Na splošno so že izdelani modeli in perpleksnosti, izračunani na vseh besedah razvojne množice, ponovno uporabni v postopku učenja.

Manjši vpliv na hitrost delovanja algoritma ima parameter α. Njegovo vrednost smo določili na 50. Izjema je le pri unigramskih modelih, kjer obstaja le 30 možnih zaporedij faktorjev P. Tukaj ima α vrednost 10. Parameter β določamo v vsakem izvajanju zanke α. Določamo ga v mejah med parametrom α in številom vseh možnih zaporedij faktorjev P, ki se pojavijo vsaj enkrat v razvojni množici. Parameter β najprej določimo kot trenutno število razredov, zmanjšano za petino razlike med omenjenima mejama. Vendar dodamo še omejitev, da lahko β znaša najmanj polovico trenutnega števila razredov (to pomeni, da v vsakem postopku združevanja razredov ohranimo vsaj polovico razredov). Nazadnje dodamo še omejitev, da β ne more biti manjši od α.

Delovanje algoritma smo pospešili še s tem, da smo v vsakem učenju na novem naboru razredov preverili, ali je neki razred že obstajal v prejšnjem naboru razredov, in smo v tem primeru uporabili že dobljene rezultate.

V algoritmu za določanje končne poti 4.4 moramo določiti še parametra γ in δ. Za parameter γ smo izbrali vrednost 5 % perpleksnosti trenutnega najboljšega modela, kar pomeni, da bomo neki večji model izbrali le, če bo njegova perpleksnost manjša za vsaj 5 %. Za parameter δ pa smo izbrali vrednost 25 % velikosti trenutnega najboljšega modela, kar pomeni, da bomo neki večji model z boljšo perpleksnostjo izbrali, če njegova velikost ne bo presegala 125 % velikosti trenutno izbranega modela.

Uporaba morfoloških modelov s kontekstno odvisno strukturo v drugem prehodu razpoznavanja je v svoji osnovni obliki enaka uporabi preprostih morfoloških modelov.

6.8 Ocenjevanje uspešnosti

Običajni način ocenjevanja uspešnosti razpoznavanja je izračun deleža napak besed (angl. Word Error Rate – WER). Pri tem načinu ocenjevanja napak na podlagi Levenshteinove razdalje različnih vrst napak ne razlikujemo po njihovi pomembnosti. Tudi sami smo izbrali delež napak kot osnovno oceno za uspešnost naših modelov, vendar smo dodali še nekoliko bolj podroben postopek analize rezultatov razpoznavanja.

Uvedli smo nekoliko posplošen postopek poravnave med hipotezo in referenčno transkripcijo, ki temelji na posplošitvi Levenshteinove razdalje na dva nivoja.

Levenshteinova razdalja je definirana kot najmanjše število zamenjav simbolov, vrinjenih simbolov in izbrisanih simbolov pri poravnavi dveh nizov simbolov. V primeru računanja WER so ti simboli besede.

V primeru utežene Levenshteinove razdalje število vrinjenih, brisanih in zamenjanih besed ne seštevamo več preprosto, ampak posameznim vrstam operacij (vrivanje, brisanje,

zamenjava) dodamo izbrane uteži. Tako je utežena Levenshteinova razdalja definirana kot:

$$WLD = W_i \cdot I + W_s \cdot S + W_d \cdot D, \tag{6.1}$$

kjer so I, S in D množice vrinjenih, zamenjanih in izbrisanih besed v poravnavi, W_i, W_s in W_d pa so uteži. Mi smo to enačbo še dodatno posplošili tako, da smo uvedli uteži za besede in črke. Tako smo prišli do enačbe

$$D = \sum_{i \in \mathcal{I}} [W_i + C_i \cdot L(i)] + \sum_{d \in \mathcal{D}} [W_d + C_d \cdot L(d)] + \sum_{(s_1, s_2) \in \mathcal{S}} [W_s + C_s \cdot CLD(s_1, s_2)], \tag{6.2}$$

kjer so I, S in D ponovno množice vrinjenih, zamenjanih in izbrisanih besed v poravnavi, Funkciji $L(i)$ in $L(d)$ povesta dolžino (število črk) vrinjene oz. zamenjane besede, CLD pa je Levenshteinova razdalja v številu črk med besedama v paru zamenjave besed. Enačba ima šest uteži. Tri uteži (W_i, W_d in W_s) so določene na besednem nivoju. Za njih smo izbrali vrednost 100. Ostale tri (C_i, C_d in C_s) pa so določene na črkovnem nivoju. Ob upoštevanju rezultatov predhodnih raziskav vpliva vrednosti uteži na poravnavo smo za te uteži izbrali vrednost 20.

Učinek uporabe poravnave na podlagi enačbe 6.2 lahko vidimo, če primerjamo sliko 6.2, kjer je prikazana poravnava z običajno Levenshteinovo razdaljo, in sliko 6.3, kjer imamo enaka stavka poravnana na podlagi enačbe 6.2. Čeprav nam da poravnava na drugi sliki večje število napak, je ta poravnava bolj smiselna, če hočemo analizirati vrste napak, ki se pojavijo med razpoznavanjem.

```
LAB: tonček s     svojimi   vragolijami občinstvo navdušuje
REC: tonček svoje vragolije mi          občinstvo navdušuje
```

Slika 6.2: Poravnava s tremi napakami.

```
LAB: tonček s svojimi vragolijami    občinstvo navdušuje
REC: tonček   svoje   vragolije   mi občinstvo navdušuje
```

Slika 6.3: Poravnava s štirimi napakami.

Takšno poravnavo bomo uporabili v naslednjem poglavju, kjer bomo napake analizirali podrobneje glede na njihove morfološke značilnosti.

Poglavje 7

Rezultati

V tem poglavju bomo predstavili eksperimentalne rezultate, ki smo jih dobili v tej nalogi. Ker so osrednja tema naše naloge jezikovni modeli, smo tukaj dodali tudi osnovne podatke in analize jezikovnih modelov, ki smo jih izdelali.

Na začetku podajamo podatke o besednih jezikovnih modelih, ki smo jih uporabili v prvem prehodu razpoznavanja, nato pa sledijo rezultati samega razpoznavanja v prvem prehodu. Ker so osnova za drugi prehod seznami N najboljših hipotez, podajamo tudi analizo teh seznamov glede na najmanjše število napak, ki ga lahko dosežemo z njihovo uporabo.

V naslednjih podpoglavjih sledijo rezultati razpoznavanja z različnimi morfološkimi jezikovnimi modeli v drugem prehodu razpoznavanja in opis značilnosti teh modelov. Nadaljujemo s statistično analizo sprememb rezultatov v drugem prehodu razpoznavanja.

V zadnjem delu poglavja z jezikovnega stališča podrobneje analiziramo izbran primer rezultatov razpoznavanja po prvem in po drugem prehodu glede na napake, ki se pojavijo v razpoznavanju.

7.1 Besedni jezikovni modeli

Pred samo uporabo jezikovnih modelov smo le-te analizirali glede na perpleksnost na testni množici in preverili njihovo velikost. Rezultati perpleksnosti so podani v tabeli 7.1. Razdeljeni so glede na velikost uporabljenega slovarja (60k do 300k), red jezikovnega modela (bigramski in trigramski) in glajenje (Good-Turingovo in Knesser-Neyevo[1]).

Rezultati perpleksnosti ustrezajo pričakovanjem. Perpleksnost se povečuje s povečevanjem slovarja, pada pa z redom jezikovnega modela. Opazimo tudi, da je perpleksnost

[1]Tukaj ne gre za izvirno Knesser-Neyevo glajenje, ampak za modificirano Knesser-Neyevo glajenje, predstavljeno v [10]

Tabela 7.1: Perpleksnosti besednih jezikovnih modelov na testni množici BNSI.

Velikost slovarja	Perpleksnost			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	369	258	382	231
100k	454	317	429	299
200k	501	349	494	341
300k	525	365	527	361

Tabela 7.2: Število n-gramov najvišjega reda v besednih jezikovnih modelih.

Velikost slovarja	Število n-gramov najvišjega reda	
	Bigramski modeli	Trigramski modeli
60k	39.958.426	41.379.910
100k	49.548.399	45.959.100
200k	60.701.460	50.151.181
300k	65.633.037	51.599.996

pri modelih s Knesser-Neyevim glajenjem z eno izjemo vedno manjša kot pa pri ustreznem modelu z Good-Turingovim glajenjem. Tudi to ustreza pričakovanjem, saj je bilo na splošno ugotovljeno, da z uporabo Knesser-Neyevega glajenja dobivamo boljše rezultate [10]. Edini primer, kjer je perpleksnost boljša z uporabo Good-Turingovega glajenja, so bigramski modeli pri slovarju s 300.000 besedami, vendar je razlika zelo majhna.

Rezultati v tabeli 7.2 prikazujejo število vseh n-gramov najvišjega reda, ki so vključeni v modele z Good-Turingovim glajenjem. V primeru bigramskega modela so to bigrami, v primeru trigramskega modela pa trigrami. Število unigramov v vseh modelih je enako velikosti slovarja, z izjemo modelov s Knesser-Neyevim glajenjem in velikostjo slovarjev 200k in 300k, kjer je število unigramov za ena večje. V teh modelih je algoritem za učenje modelov v seznam unigramov dodal oznako za neznano besedo: <unk>. Prav tako v teh modelih posledično prihaja do majhnih odstopanj v številu bigramov in trigramov. Števila bigramov v vseh trigramskih modelih pa so enaka številom bigramov v ustreznih bigramskih modelih.

Enake velikosti modelov glede na način glajenja so posledica uporabe enakih parametrov glajenja, ki določajo minimalno in maksimalno število pojavnic nekega n-grama, pri katerih se uporabi glajenje. S tem se določi tudi minimalno število pojavnic nekega n-grama, ki je potrebno, da se ta sploh vključi v model.

Ker so modeli ne glede na tip glajenja enako veliki, pričakujemo, da bodo imeli podobno časovno in prostorsko zahtevnost za algoritem razpoznavanja. To bomo preverili v naslednjem podpoglavju.

Celoten postopek izdelave modelov s Knesser-Neyevim glajenjem je bolj dolgotrajen. Z uporabo orodja SRILM smo za izdelavo teh modelov porabili približno dvakrat več časa

kot pa za izdelavo modelov z Good-Turingovim glajenjem. Ta podatek se nanaša na čas, ki je potreben, da iz datotek s številom pojavnic vseh n-gramov v učnem korpusu izdelamo jezikovni model. Postopek za izdelavo datotek s številom pojavnic je v obeh primerih enak.

7.2 Prvi prehod razpoznavanja

V prvem prehodu razpoznavanja smo uporabili izdelane akustične modele ter bigramske in trigramske jezikovne modele. Rezultati razpoznavanja so predstavljeni v tabeli 7.3. Podani so v deležu napačno razpoznanih besed, ki temelji na Levenshteinovi razdalji na nivoju besed. Iz rezultatov lahko vidimo izboljšanja razpoznavanja pri povečanju slovarja in povečanju reda jezikovnega modela.

Tabela 7.3: Rezultati razpoznavanja v prvem prehodu z različnimi jezikovnimi modeli na testni množici BNSI.

Velikost slovarja	Delež napak [%]			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	33,89	30,73	33,84	30,95
100k	31,27	28,08	31,24	28,36
200k	29,70	26,27	29,78	26,53
300k	29,22	25,64	29,28	25,87

Zmanjšanje deleža napačno razpoznanih besed pri povečevanju slovarja je dobro primerljivo z zmanjšanjem števila besed izven slovarja, podanim v tabeli 6.1. Pri prehodu z najmanjšega na največji slovar se delež besed izven slovarja zmanjša za 5,92 %. Izboljšanje rezultata razpoznavanja pa se giblje med 4,56 % in 5,09 %, odvisno od uporabljenega tipa glajenja in reda jezikovnega modela.

Pomembno izboljšanje rezultatov prinese prehod z bigramskega na trigramski model. Izboljšanja se gibljejo med 2,88 % in 3,57 %. Opazimo lahko, da so izboljšanja večja pri uporabi Good-Turingovega glajenja in pri večjih slovarjih.

Zanimiva je primerjava rezultatov pri različnih metodah glajenja. Medtem ko so drugi raziskovalci ugotavljali, da dajejo modeli, v katerih je bilo uporabljano Knesser-Neyevo glajenje, boljše rezultate, pa smo mi opazili izboljšanje rezultatov z uporabo Knesser-Neyevega glajenja le pri bigramskih modelih z manjšimi slovarji. V vseh primerih pa so te razlike majhne. Naredili smo tudi statistični test med pari rezultatov, dobljenimi z različnimi načini glajenja. Opravili smo test naključnega približanja z 10.000 prehodi. V vseh primerih je izvedeni test pokazal, da razlika ni statistično značilna. Najmanjša dobljena p-vrednost je bila 0,12, medtem ko bi za ugotovitev statistične značilnosti ta morala biti manjša od 0,05.

Na vseh postopkih razpoznavanja smo merili tudi trajanje razpoznavanja. Podatki so predstavljeni v tabeli 7.4. Rezultati so podani v faktorju realnega časa (RTF), ki predstavlja razmerje med trajanjem razpoznavanja in dolžino posnetkov, ki jih razpoznavamo.

Tabela 7.4: Faktorji realnega časa pri razpoznavanju v prvem prehodu.

Velikost slovarja	Faktor realnega časa			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	6,294	18,456	6,142	18,617
100k	7,819	23,421	7,520	23,367
200k	10,440	31,545	10,476	31,705
300k	12,661	37,089	12,876	37,704

Rezultati kažejo na to, da se pri petkratnem povečanju slovarja, časovna zahtevnost razpoznavanja poveča za faktor približno 2, pri prehodu z bigramskega na trigramski model pa se časovna zahtevnost poveča za faktor približno 3. Ne opazimo pa bistvenih razlik glede na način glajenja, ki smo ga uporabljali pri gradnji modela.

Iz rezultatov uspešnosti razpoznavanja in faktorjev realnega časa lahko sklepamo, da je v primeru testne množice BNSI ugodneje uporabljati modele z Good-Turingovim glajenjem. Prav tako smo videli, da je povečanje slovarja prineslo večje izboljšanje uspešnosti razpoznavanja ob manjšem povečanju faktorja realnega časa kot pa prehod z bigramskega na trigramski model. Videli smo, da je to povečanje približno ustrezalo zmanjšanju deleža besed izven slovarja. Ker je delež besed izven slovarja pri slovarju velikosti 300k samo še 1,02 %, lahko sklepamo, da nadaljnje povečevanje velikosti slovarja ne bo več prinašalo bistvenih izboljšav v uspešnosti razpoznavanja.

Na faktor realnega časa vplivajo drugi dejavniki. Predvsem je pomembna strojna oprema. Ker smo hoteli imeti rezultate, ki so med seboj primerljivi, so bila vsa razpoznavanja izvedena na isti strojni opremi. Še en faktor, ki je nekoliko povečal faktor realnega časa, je način izpisa rezultatov. Uporabili smo namreč izpis rezultatov v obliki besedne mreže, ki predstavlja iskalni prostor ob zaključku razpoznavanja posameznih segmentov. Ta način je bil potreben, da smo lahko tvorili več hipotez, ki smo jih uporabili v drugem prehodu razpoznavanja.

Pred samim izvajanjem razpoznavanja na testni množici je bilo potrebno izvesti še optimizacijo parametrov razpoznavanja. To smo storili s pomočjo razvojne množice. Za utež jezikovnega modela smo dobili vrednost 15, za utež vrinjene besede (angl. word insertion penalty) pa smo dobili vrednost -15.

Bistveni del naše naloge je, da zgradimo dvoprehodni algoritem razpoznavanja, ki bo uporabljal množico jezikovnih modelov. Ker bo pri tem potrebno tudi v nadaljevanju izvajati optimizacije raznih parametrov, ki so na voljo za algoritem razpoznavanja, bomo tudi v nadaljevanju potrebovali razvojno množico in sezname hipotez razpoznavalnika za

Tabela 7.5: Rezultati razpoznavanja v prvem prehodu z različnimi jezikovnimi modeli na razvojni množici BNSI.

Velikost slovarja	Delež napak [%]			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	33,83	30,87	33,79	31,32
100k	31,43	28,59	31,68	29,15
200k	30,03	27,10	30,21	27,67
300k	29,56	26,62	29,67	26,98

Tabela 7.6: Faktorji realnega časa pri razpoznavanju v prvem prehodu na razvojni množici BNSI.

Velikost slovarja	Faktor realnega časa			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	6,902	21,550	9,978	23,539
100k	8,336	27,404	8,412	30,054
200k	11,402	36,725	11,603	39,307
300k	13,712	43,374	14,111	45,991

govorne segmente v njej. Zato smo vse rezultate prvega prehoda razpoznavanja ponovili še na razvojni množici. Ti rezultati so prikazani v tabelah 7.5 in 7.6.

Če si ogledamo rezultate uspešnosti razpoznavanja in faktorje realnega časa na razvojni množici, ugotovimo, da ima razvojna množica za razpoznavanje podobne lastnosti, kot pa testna množica. Opazimo namreč podobne razlike v rezultatih glede na velikost slovarja, red modela in vrsto glajenja kot pa pri testni množici.

Primerjava rezultatov med testno in razvojno množico pa pokaže, da so rezultati na razvojni množici nekoliko boljši. Za to je lahko več razlogov. Pomemben razlog je bil, da so bili parametri za razpoznavanje optimizirani ravno na razvojni množici. Drugi razlog pa so razlike v množicah. Tukaj lahko le posumimo, da se v razvojni množici nahaja večje število govornih segmentov, ki so lažji za razpoznavanje, kot pa v testni množici.

7.3 Seznami N najboljših hipotez

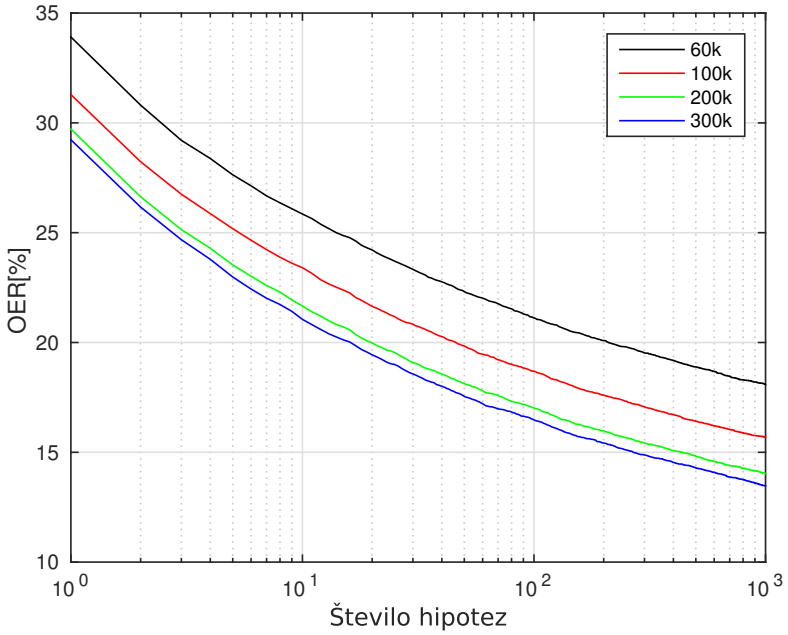
Izvedli smo 32 kombinacij razpoznavanj: dve različni množici govornih segmentov (razvojna in testna), dva reda modela, štiri velikosti slovarjev in dva načina glajenja. V nadaljevanju smo vse rezultate razpoznavanja pretvorili v sezname N najboljših hipotez, kjer smo za N izbrali število 1000.

Nato je sledila analiza števila napak v teh seznamih. V ta namen najprej definirajmo napako oraklja (OER, angl. oracle error rate). Za vsak govorni segment imamo po

prvem prehodu razpoznavanja na voljo seznam N hipotez. Vsako od teh hipotez lahko ocenimo glede na število napak, ki jih vsebuje. Sedaj pa predpostavimo idealni sistem razpoznavanja v drugem prehodu (orakelj), ki bo vedno znal izbrati tisto hipotezo, ki vsebuje najmanj napak. Običajno bo to ena izmed hipotez, ki se nahajajo pri vrhu seznama, saj je seznam urejen po oceni iz prvega prehoda. Ta sistem lahko opišemo z enačbo:

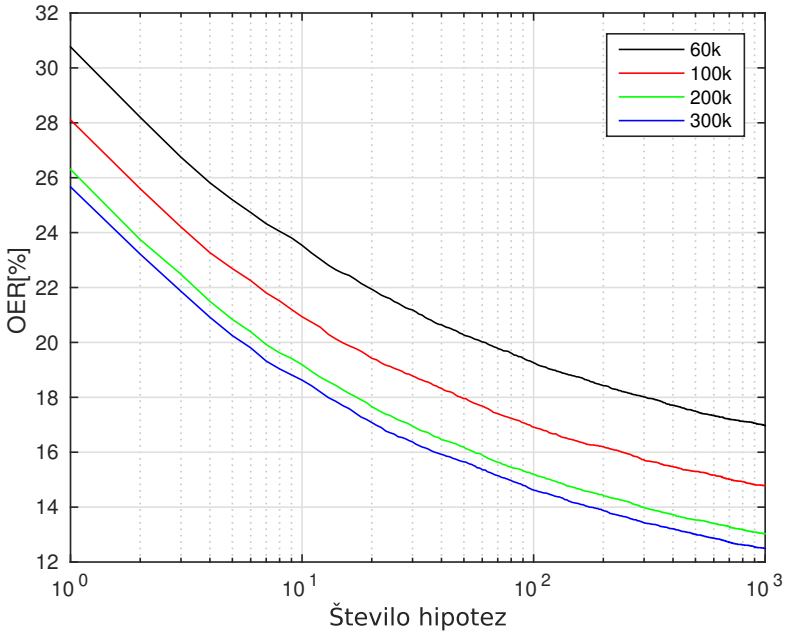
$$OER_i = \min_j \{D(H_{i,j}, R_i)\}, \tag{7.1}$$

kjer je OER_i napaka oraklja za i-ti govorni segment, $H_{i,j}$ je j-ta hipoteza za i-ti govorni segment, R_i je referenčna transkripcija za i-ti segment in D je funkcija, ki predstavlja Levenshteinovo razdaljo na nivoju besed. Celotno napako oraklja dobimo s seštevkom napak oraklja za vse govorne segmente. Napaka oraklja predstavlja spodnjo mejo za rezultate uspešnosti razpoznavanja v drugem prehodu. Dokazati je mogoče, da je napaka oraklja padajoča funkcija glede na število hipotez N, kar pomeni, da je z daljšimi seznami N najboljših hipotez teoretično mogoče doseči boljše rezultate razpoznavanja v drugem prehodu. V praksi pa s številom hipotez narašča tudi možnost, da bomo v drugem prehodu izbrali hipotezo, ki vsebuje več napak. Zato bomo v razpoznavanju v drugem prehodu poskušali optimizirati tudi število hipotez v drugem prehodu.



Slika 7.1: Napaka oraklja na testni množici BNSI (bigramski modeli).

Na slikah 7.1 in 7.2 je prikazana napaka oraklja na testni množici BNSI za bigramske in trigramske modele z Good-Turingovim glajenjem. Podobne grafe dobimo za modele z Knesser-Neyevim glajenjem in za rezultate na razvojni množici BNSI. Po pričakovanjih vidimo padajoče funkcije glede na število hipotez za vse štiri velikosti slovarja. Najhitreje padajo funkcije na začetku grafa, kar dodatno kaže na to, da bomo z zviševanjem majhnega



Slika 7.2: Napaka oraklja na testni množici BNSI (trigramski modeli).

števila hipotez v drugem prehodu dosegali večje izboljšave rezultata kot pa kasneje z zviševanjem že velikega števila hipotez. Iz vseh grafov vidimo tudi, da napaka oraklja pri 1000 hipotezah znaša približno polovico deleža napak v prvem prehodu. Natančnejši podatki za delež napak pri 1000 hipotezah so podani v tabelah 7.7 in 7.8. Podobne rezultate dobimo tudi, če analizo napak ponovimo za sezname, dobljene s Knesser-Neyevim glajenjem, in za vse sezname, dobljene na razvojni množici.

Tabela 7.7: Napaka oraklja na testni množici BNSI pri 1000 hipotezah.

Velikost slovarja	Napaka oraklja [%]			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	18,08	16,97	18,08	17,30
100k	15,68	14,78	15,72	15,09
200k	14,03	13,03	14,13	13,46
300k	13,48	12,50	13,62	12,98

Na tej točki smo se začeli odločati, katere rezultate bomo uporabljali v drugem prehodu razpoznavanja. Do sedaj prikazani rezultati tako za osnovni rezultat prvega prehoda, kot za lastnosti N najboljših seznamov, kažejo na doprinos večjega slovarja in večjega reda modela k uspešnosti razpoznavanja, ne kažejo pa bistvenih razlik glede na metodo glajenja.

Odločili smo se, da bomo v nadaljevanju uporabljali N najboljše sezname, ki so bili dobljeni s slovarjem 300k in trigramskim modelom z Good-Turingovim glajenjem v prvem prehodu, ker ti dajejo najboljše rezultate. Ker smo že v [12] pokazali, da lahko

Tabela 7.8: Napaka oraklja na razvojni množici BNSI pri 1000 hipotezah.

Velikost slovarja	Napaka oraklja [%]			
	Good-Turingovo glajenje		Knesser-Neyevo glajenje	
	Bigramski	Trigramski	Bigramski	Trigramski
60k	20,54	19,14	20,28	19,43
100k	18,21	16,70	18,42	17,26
200k	16,36	14,99	16,65	15,49
300k	15,81	14,33	16,04	14,73

v drugem prehodu skoraj nadoknadimo doprinos trigramskega modela v prvem prehodu, smo se odločili, da bomo v nadaljevanju uporabljali tudi sezname, dobljene z bigramskim modelom ter enakim slovarjem in postopkom glajenja. Vseh ostalih seznamov N najboljših hipotez od te točke naprej nismo uporabljali.

Pred uporabo morfoloških modelov smo vse hipoteze označevali z označevalnikom Obeliks. Natančnosti označevanja ne moremo preveriti, saj nimamo na voljo ročno označenih hipotez. Preverili pa smo njegovo hitrost. Podobno kot za razpoznavanja v prvem prehodu smo izračunali faktor realnega časa, ki predstavlja razmerje med časom, ki je potreben za označevanje besedila, in časom, v katerem je bilo besedilo izgovorjeno. Faktor je imel vrednosti med 26,07 in 26,67.

7.4 Preprosti morfološki modeli

Po postopku, opisanem v podpoglavju 4.2, smo zgradili 244 jezikovnih modelov za besedno vrsto in 244 modelov za oznake MSD. Za vse modele smo najprej izračunali perpleksnost na razvojni množici BNSI. V tabeli 7.9 so prikazani rezultati za perpleksnost za najboljši model za besedno vrsto in najboljši model za oznako MSD. V tabeli so navedene tudi poti sestopanja teh modelov. Vidimo, da sta poti dolgi le 3 oz. 4 faktorje, medtem ko smo testirali modele s potmi dolžine do 7. To pomeni, da daljše poti ne dajejo vedno boljšega rezultata.

Tabela 7.9: Najmanjši perpleksnosti za preproste morfološke modele.

Modeliran faktor	Model	Perpleksnost
Besedna vrsta	$P(P_0\|M_0, M_{-1}, P_{-1})$	2,214
Oznaka MSD	$P(M_0\|P_0, M_{-1}, P_{-1}, M_{-2})$	3,628

Opazimo tudi, da se v poteh sestopanja obeh modelov nahajajo različni faktorji. Če pogledamo vse modele za besedno vrsto, ki vsebujejo le besedno vrsto prejšnjih besed v poteh sestopanja, dobimo najmanjši rezultat perpleksnosti 11,764, kar je bistveno več od rezultata v tabeli 7.9. Podobno dobimo najmanjši rezultat za perpleksnost 18,365 pri modelih za oznake MSD, ki uporabljajo le oznake MSD v poteh sestopanja.

7.5 Razpoznavanje s preprostimi morfološkimi modeli

V prvih preizkusih drugega prehoda smo uporabljali izbrane preproste morfološke modele. Uredili smo jih glede na njihovo perpleksnost na razvojni množici. Izbrali smo 5 modelov z najnižjo perpleksnostjo za faktor besedne vrste in 5 modelov za faktor oznake MSD. Skupno smo izbrali 10 preprostih modelov z morfološkimi informacijami. K rezultatom dobljenim s seznami, kjer smo v prvem prehodu uporabljali trigramske jezikovne modele, smo dodali še originalne rezultate iz prvega prehoda: akustično verjetnost, verjetnost iz besednega modela in število besed v hipotezi. Pri rezultatih na seznamih N najboljših hipotez, dobljenih z bigramskimi modeli v prvem prehodu, pa smo verjetnost jezikovnega modela nadomestili z verjetnostjo iz trigramskega modela. Bigramskih jezikovnih modelov v drugem prehodu nismo več uporabljali.

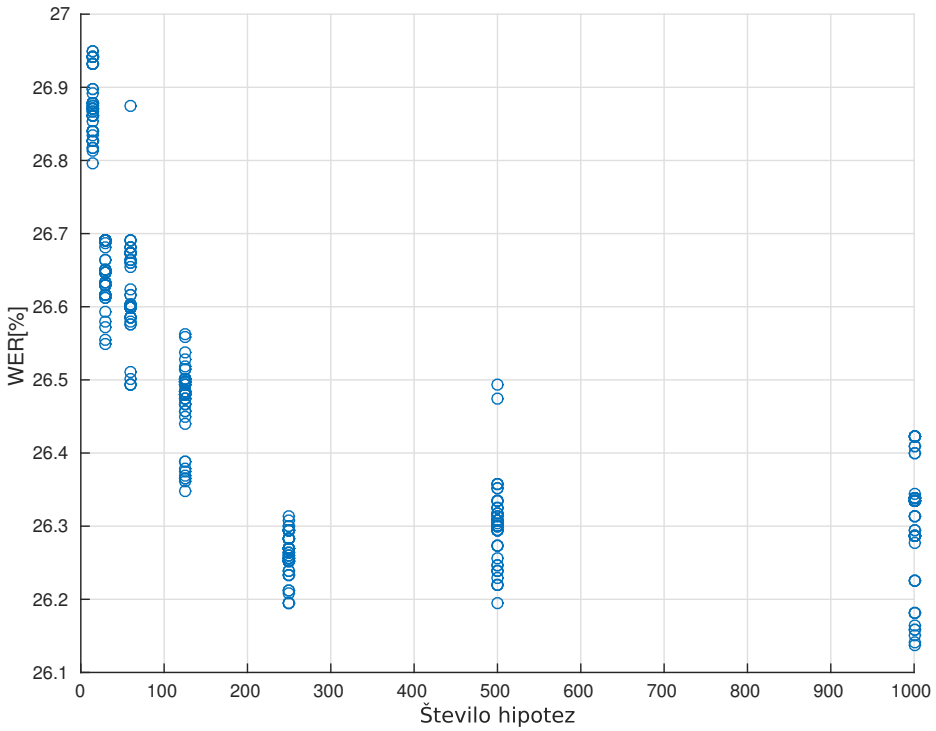
V drugem prehodu smo preizkušali kombinacije verjetnosti iz zgoraj omenjenih modelov. Pri tem smo se omejili na le en model za besede, en model za besedne vrste in en model za oznake MSD. To nam skupno daje 25 možnih izbir modelov.

Preverili smo doprinos modelov za besedne vrste, ki upoštevajo le besedne vrste prejšnjih besed, in modelov za oznake MSD, ki upoštevajo le oznake MSD prejšnjih besed (glej enačbo 4.1). Redi modelov segajo od bigramskega do 6-gramskega. Prvo razpoznavanje je potekalo na razvojni množici, ki smo jo ocenili z vsemi modeli, ki jih obravnavamo. Nato smo izvedli algoritem za optimizacijo uteži 5.2 z različnimi kombinacijami modelov, različnimi začetnimi vrednostmi parametrov in različnim številom upoštevanih hipotez. Za vsak zagon optimizacijskega algoritma smo dobili rezultat v obliki končnih vrednosti uteži in pripadajočega števila napak v razpoznavanju.

7.5.1 Začetne vrednosti za optimizacijski algoritem

Ker je zaradi velikega števila možnih faktorjev in uteži tudi optimizacija uteži dolgotrajen proces, smo poskušali zmanjšati število zagonov algoritma za modele, ki upoštevajo tako besedno vrsto kot oznako MSD. V ta namen smo preverili že dobljene rezultate glede na posamezne parametre.

Na sliki 7.3 smo prikazali delež napak na razvojni množici po optimizaciji glede na število obravnavanih hipotez. Slika prikazuje rezultate za primer uporabe modelov za oznake MSD različnih redov in različnih začetnih vrednosti parametrov. Za upoštevano število hipotez smo izbirali 30, 60, 125, 250, 500 in 1000. Rezultati so bili pridobljeni iz N najboljših seznamov, kjer smo v prvem prehodu uporabljali bigramski jezikovni model. Podobne rezultate prikazuje slika 7.4, kjer pa smo ponovno ocenjevali N najboljše sezname, ki so bili dobljeni s trigramskim jezikovnim modelom v prvem prehodu. Podobne rezultate lahko dobimo z modeli za besedne vrste.

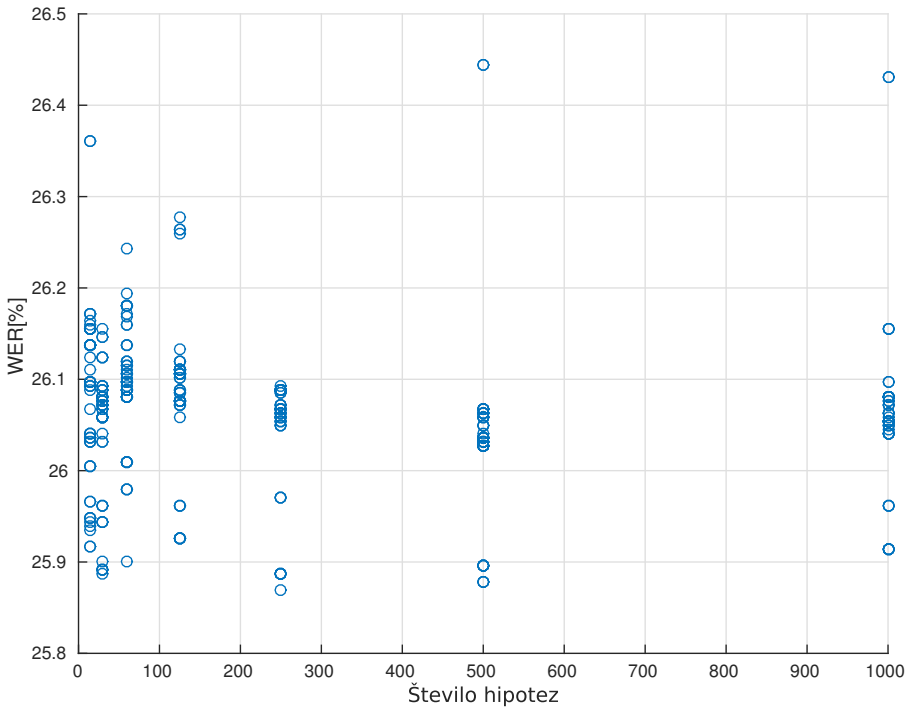


Slika 7.3: Vrednosti deleža napak glede na število hipotez (dobljene z bigramskimi modeli) pri uporabi modelov za oznake MSD, ki uporabljajo faktorje oznake MSD, po optimizaciji parametrov.

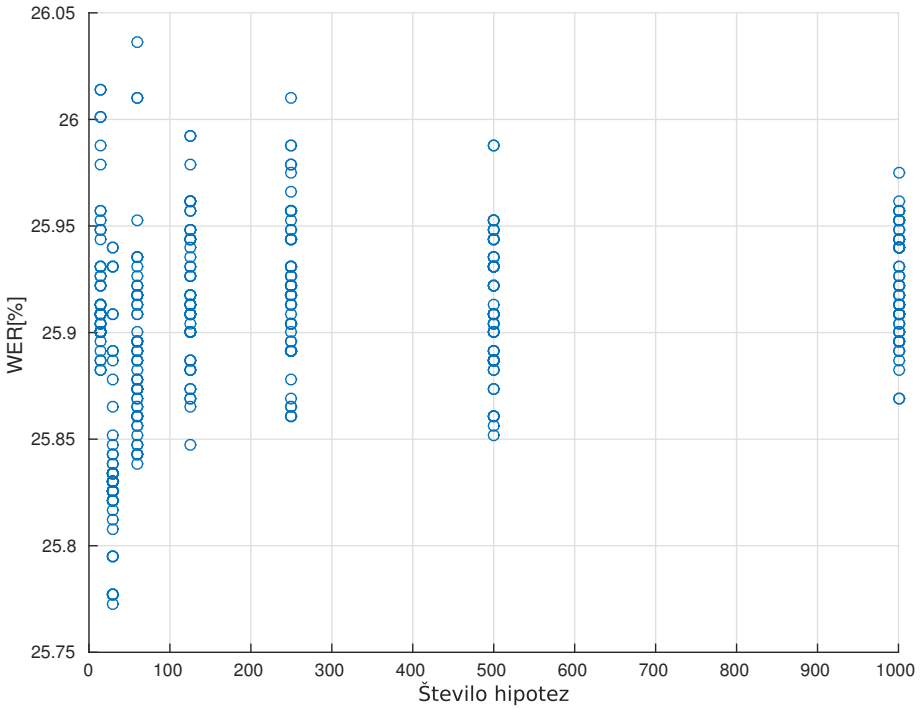
Glede na rezultate na sliki 7.3 lahko sklepamo, da izbira 125 ali manj ni primerna, ker daje slabše rezultate. Rezultati pri uporabi 250 in 500 hipotez so skoraj enaki. Kadar izbiramo med tema številoma, ni ugodno vzeti 500 hipotez, ker ne dobimo boljših rezultatov, se pa približno dvakrat podaljša čas izvajanja algoritma optimizacije, katerega časovna zahtevnost linearno narašča s številom hipotez. Nekoliko v dvomih smo lahko pri izbiranju med 500 in 1000, kjer vidimo manjše izboljšanje pri najboljših rezultatih, vendar spet za ceno dvakrat večje časovne zahtevnosti. Podatki na sliki 7.4 kažejo nekoliko drugačno sliko. Tukaj se zdi, da je najprimernejše število hipotez 250, podatki pri manj hipotezah pa niso veliko slabši. Glede na rezultate, ki jih dobimo z modeli za besedne vrste, se ponovno lahko odločamo med 250 in 500 hipotezami. Pri tem lahko še opazimo, da so rezultati na sliki 7.3 v absolutnem merilu najboljši, kar pomeni, da dajejo modeli za oznake MSD, uporabljeni na rezultatih s trigramskimi modeli, v prvem prehodu najboljša izboljšanja natančnosti razpoznavanja.

Nekoliko drugačne rezultate pa vidimo na sliki 7.5, ki prikazuje delež napak po optimizaciji z modeli oznak MSD, ki uporabljajo besedne vrste in oznake MSD. Vidimo, da najmanjše deleže napak dobimo pri številu hipotez 30. Glede na vse te rezultate smo se odločili, da se bomo v prihodnje pri optimizaciji uteži in številu hipotez omejili na 250 hipotez.

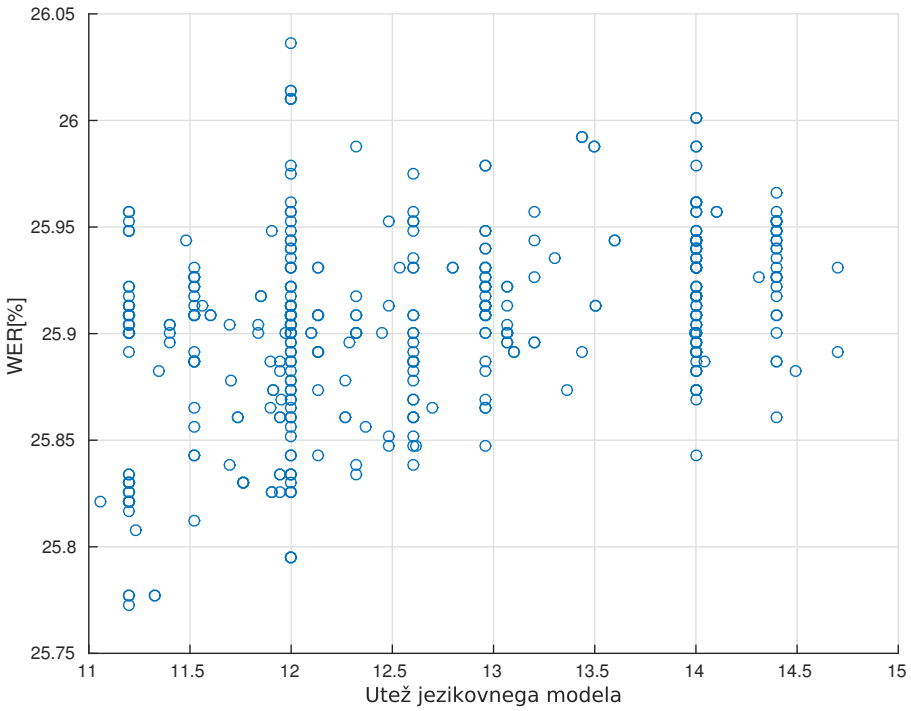
Slika 7.6 kaže dobljene uteži trigramskega jezikovnega modela besed, dobljene po



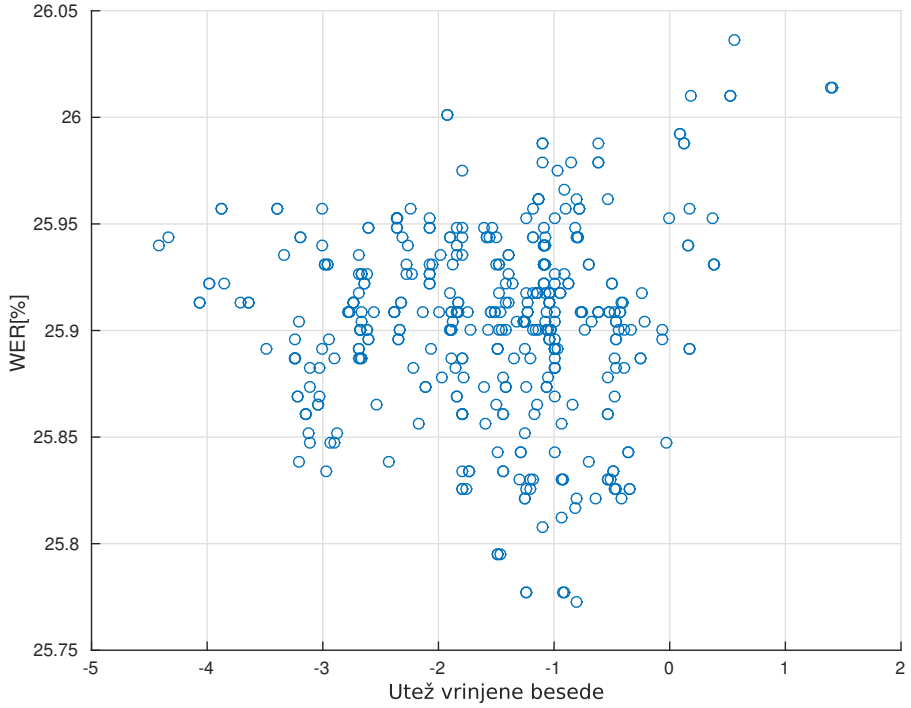
Slika 7.4: Vrednosti deleža napak glede na število hipotez (dobljene s trigramskimi modeli) pri uporabi modelov za oznake MSD, ki uporabljajo faktorje oznake MSD, po optimizaciji parametrov.



Slika 7.5: Vrednosti deleža napak glede na število hipotez pri uporabi modelov za oznake MSD, ki uporabljajo faktorje besedne vrste in oznake MSD, po optimizaciji parametrov.



Slika 7.6: Vrednosti deleža napak glede na utež jezikovnega modela pri uporabi modelov za oznake MSD, ki uporabljajo faktorje besedne vrste in oznake MSD, po optimizaciji parametrov.



Slika 7.7: Vrednosti deleža napak glede na utež vrinjene besede pri uporabi modelov za oznake MSD, ki uporabljajo faktorje besedne vrste in oznake MSD, po optimizaciji parametrov.

optimizaciji, kjer smo zraven uporabljali model za oznake MSD. Vidimo, da se optimalne uteži gibljejo med 11 in 14. Slika 7.7 pa prikazuje optimalne vrednosti za utež vrinjene besede. Tukaj vidimo, da so optimalne vrednosti med -3 in 0. V nadaljevanju bomo te okvirne vrednosti optimalnih parametrov in zgoraj ugotovljenega optimalnega števila hipotez uporabljali za določanje začetnih vrednosti v algoritmu za optimizacijo uteži 5.2. Ker bomo v vsakem nadaljnjem optimizacijskem postopku uporabljali drugačen morfološki model, ni bilo smiselno postavljati ožjih okvirjev za začetne vednosti njegove uteži.

Ko smo definirali okvirne podatke za primerne uteži, izvedemo še optimizacijo parametrov za drugi prehod, kjer uporabimo oba morfološka modela – model za oznake MSD in model za besedne vrste. V nadaljevanju bomo pregledali najboljše rezultate po optimizaciji in pripadajoče optimalne vrednosti parametrov.

7.5.2 Rezultati na razvojni množici

Tabela 7.10 prikazuje najboljše rezultate razpoznavanja po optimizaciji uteži na razvojni množici glede na različne jezikovne modele v prvem in drugem prehodu. V prvem prehodu z bigramskimi modeli je bil delež napak 29,56 %. V drugem prehodu smo ta rezultat lahko izboljšali na 26,05 %, kar predstavlja 11,87 % relativno izboljšanje. Pomemben del tega izboljšanja je uporaba trigramskega besednega modela. V prvem prehodu s trigramskimi modeli pa je bil delež napak 26,62 %. Po dodajanju morfoloških modelov in optimizaciji parametrov smo ta delež zmanjšali na 25,77 %, kar predstavlja 3,19 % relativno izboljšanje, ki pa je tokrat posledica uporabe samo morfoloških modelov.

Tabela 7.10: Delež napak na razvojni množici ob uporabi enega morfološkega modela po optimizaciji parametrov.

Besedni model	Modeliran faktor	Uporabljeni faktorji	WER [%]
Bigramski	Oznake MSD (M)	M	26,14
Bigramski	Oznake MSD (M)	M+P	26,05
Bigramski	Besedna vrsta (P)	P	26,23
Bigramski	Besedna vrsta (P)	M+P	26,19
Trigramski	Oznake MSD (M)	M	25,87
Trigramski	Oznake MSD (M)	M+P	25,77
Trigramski	Besedna vrsta (P)	P	26,03
Trigramski	Besedna vrsta (P)	M+P	25,83

V obeh primerih je bilo največje izboljšanje doseženo pri uporabi modelov za oznake MSD, ki so kot faktorje uporabljali tako oznake MSD kot besedne vrste. Tudi iz ostalih rezultatov je razvidno, da so modeli za oznake MSD bolj koristni za zmanjšanje deleža napak. Najmanjši doprinos pa dobimo z modeli za besedne vrste, ki kot faktorje uporabljajo le besedne vrste.

Tabela 7.11 prikazuje najboljše rezultate razpoznavanja po optimizaciji parametrov, kjer smo uporabljali dva morfološka modela v drugem prehodu. Tudi tokrat so rezultati

prikazani glede na različne jezikovne modele v prvem prehodu. V drugem prehodu pa smo uporabljali dve kombinaciji modelov. V prvi kombinaciji smo uporabili modela za oznake MSD in besedne vrste, ki sta oba uporabljala le eno vrsto faktorjev (istoležni faktorji), v drugi kombinaciji pa sta oba modela uporabljala obe vrsti faktorjev (vsi faktorji).

Tabela 7.11: Delež napak na razvojni množici ob uporabi dveh morfoloških modelov po optimizaciji parametrov.

Besedni model	Uporabljeni faktorji	WER [%]
Bigramski	Istoležni faktorji	26,18
Bigramski	Vsi faktorji	26,00
Trigramski	Istoležni faktorji	25,86
Trigramski	Vsi faktorji	25,71

Najboljši rezultat smo dosegli pri uporabi trigramskih besednih modelov v prvem prehodu in drugi kombinaciji morfoloških modelov, kjer uporabljamo vse faktorje. Dosegli smo relativno izboljšanje 3,42 % glede na prvi prehod razpoznavanja.

Glede na rezultate v tabelah 7.10 in 7.11, smo se odločili za dva scenarija izbire modelov za končni preizkus delovanja preprostih morfoloških modelov. Najprej smo izbrali razpoznavanje v drugem prehodu, kjer smo uporabili trigramski besedni jezikovni model in model za oznake MSD, ki uporablja vse faktorje. V tabeli 7.12 so prikazane optimizirane vrednosti parametrov za ta scenarij, ki dajo najboljša rezultata v tabeli 7.10 (26,05 % in 25,77 %). Ob tem smo zapisali tudi strukturo obeh modelov, s katerima smo dosegli ta rezultata.

V drugem scenariju smo izbrali razpoznavanje v drugem prehodu, kjer smo uporabili besedni jezikovni model ter dva morfološka modela – en model za oznake MSD in en model za besedne vrste. V tabeli 7.13 so prikazane optimizirane vrednosti parametrov še za ta scenarij. Izbrali smo vrednosti parametrov, s katerimi smo dosegli rezultata 26,00 % in 25,71 % v tabeli 7.11.

Tabela 7.12: Optimalne vrednosti parametrov za prvi scenarij razpoznavanja s preprostimi morfološkimi modeli.

Besedni jezikovni model	Bigramski	Trigramski
Število hipotez	1000	30
Utež jezikovnega modela	13,82	11,2
Utež morf. modela	2,29	2,88
Utež vrinjene besede	−0,88	−0,80
Struktura modela	$P(M_0\|P_0, M_{-1}, P_{-1}, M_{-2})$	$P(M_0\|P_0, P_{-1}, M_{-1}, M_{-2}, P_{-2})$

7.5.3 Rezultati na testni množici

Potem ko smo vrednosti uteži optimirali na razvojni množici, smo te vrednosti in pripadajoče modele preizkusili na testni množici. V tabeli 7.14 so prikazani rezultati raz-

Tabela 7.13: Optimalne vrednosti parametrov za drugi scenarij razpoznavanja s preprostimi morfološkimi modeli

Besedni jezikovni model	Bigramski	Trigramski
Število hipotez	500	30
Utež jezikovnega modela	13,0667	13,72
Utež morf. modela za oznake MSD	2,1037	2,2848
Utež morf. modela za besedne vrste	−3,2	−3,6176
Utež vrinjene besede	−1,21453	−1,8756
Struktura modela za oznake MSD	$P(M_0\|P_0, M_1, P_1, M_2)$	$P(M_0\|P_0, M_1, P_1, M_2)$
Struktura modela za besedne vrste	$P(P_0\|M_0, P_1, P_2)$	$P(P\|M_0, P_1, M_1)$

poznavanja v drugem prehodu. Rezultati so podani za oba scenarija uporabe preprostih morfoloških modelov in pri uporabi bigramskega in trigramskega jezikovnega modela v prvem prehodu. Zraven je zapisano tudi relativno izboljšanje rezultatov glede na prvi prehod.

Tabela 7.14: Rezultati razpoznavanja na testni množici z besednimi modeli (W), preprostimi morfološkimi modeli za oznake MSD (M) in preprostimi morfološkimi modeli za besedne vrste (P).

Model v 1. prehodu	Modeli v 2. prehodu	WER [%]	Relativno izboljšanje [%]
2g	W+M	25,84	+11,56
2g	W+M+P	26,01	+10,99
3g	W+M	25,86	−0,84
3g	W+M+P	25,62	+0,07

Iz rezultatov lahko razberemo, da v drugem prehodu dobimo večja izboljšanja pri rezultatih, ki so bili dobljeni z bigramskimi modeli v prvem prehodu. To je delno posledica uporabe trigramskih jezikovnih modelov in delno posledica uporabe morfoloških modelov v drugem prehodu. Iz rezultatov, ki so bili dobljeni s trigramskimi jezikovnimi modeli v prvem prehodu, vidimo po drugem prehodu le manjše spremembe. V primeru uporabe samo morfološkega modela za oznake MSD je prišlo celo do poslabšanja rezultata. Iz teh rezultatov lahko sklepamo, da uporaba preprostih morfoloških modelov prinese le nebistvene spremembe rezultatov v drugem prehodu razpoznavanja.

7.6 Modeli s kontekstno odvisno strukturo

Med postopkom učenja strukture modelov s kontekstno odvisno strukturo smo tvorili in testirali veliko množico različnih modelov, od katerih pa smo na koncu obdržali le majhno množico. V tabeli 7.15[2] so prikazani podatki za število pregledanih modelov glede na red modela in modeliran faktor. Na koncu tabele imamo tudi podatke za število modelov za posamezni faktor in vse rede modela.

[2]V teh in nadaljnjih tabelah se oznake P, M oz. W nanašajo na modele s kontekstno odvisno strukturo, ki modelirajo besedne vrste, oznake MSD oz. besede.

Tabela 7.15: Število tvorjenih in izbranih modelov s kontekstno odvisno strukturo.

Red modela	Faktor	Št. testiranih modelov	Št. končnih modelov
1	M	1	1
1	W	4	4
2	P	15	12
2	M	55	16
2	W	176	28
3	P	705	30
3	M	1.758	36
3	W	4.411	41
4	P	17.385	44
4	M	21.981	35
4	W	37.373	46
5	P	124.049	48
5	M	108.068	37
5	W	140.087	43
Vsi	P	131.729	108
Vsi	M	118.092	95
Vsi	W	158.071	119
Vsi	Vsi	407.892	322

Iz podatkov lahko razberemo, da se število testiranih modelov hitro veča s povečevanjem razpoložljivih faktorjev. Število končnih modelov je bilo že vnaprej navzgor omejeno s 50, saj imamo toliko razredov kontekstov, pri tem uporabljamo en model na kontekst. Vidimo pa, da so števila končnih modelov manjša od petdeset. To pomeni, da smo tudi po zadnjem združevanju imeli nekaj različnih razredov, v katerih smo izbrali isti končni model. Tudi te razrede bi lahko združili in obdržali isti model za združeni razred. Nadaljnje združevanje bi le po obsegu nekoliko poenostavilo razpoznavanje v drugem prehodu, ne bi pa imelo vpliva na rezultate razpoznavanja.

Tabela 7.16 prikazuje podatke o perpleksnosti nabora modelov za posamezne faktorje. Perpleksnost je računana na razvojni množici BNSI. Navedeni sta minimalna in maksimalna perpleksnost izmed perpleksnosti vseh modelov za podan faktor. Navedena je tudi povprečna perpleksnost vseh modelov. Iz podatkov vidimo, da imajo modeli za besedno vrsto precej podobne perpleksnosti. Večja odstopanja vidimo pri modelih za oznake MSD in za besede. Posebej izstopajoč je eden izmed modelov za besede, ki ima perpleksnost več kot 1000, medtem ko imajo vsi ostali perpleksnost manj kot 250.

Tabela 7.16: Perpleksnosti modelov s kontekstno odvisno strukturo.

Faktor	Minimalna	Maksimalna	Povprečna
P	9,182	11,480	10,086
M	3,636	9,464	4,737
W	46,486	1182,51	91,719

Podatki v tabeli 7.16 pa ne nakazujejo končne perpleksnosti, ki jo dobimo, kadar

Tabela 7.17: Realne perpleksnosti modelov s kontekstno odvisno strukturo na razvojni množici BNSI.

Red modela	Modeliran faktor		
	P	M	W
1	/	6,581	38,867
2	8,392	3,490	33,536
3	7,275	3,147	31,294
4	6,561	2,970	29,447
5	6,231	2,876	27,947

upoštevamo nabor razredov kontekstov, različne modele, ki ustrezajo razredom, in število pojavnic v množici, na kateri smo računali perpleksnost, ki pripadajo posameznim razredom. Ustrezni rezultati so prikazani v tabeli 7.17. Ker smo rešitve učenja – nabor razredov in pripadajočih modelov – gradili za vsak red modela in vsak kontekst, smo tudi rezultate podali na tak način. Iz podatkov o perpleksnosti je razvidno padanje z redom modela. Tudi pri prehodu na 5-gramski model je še vedno prisotno opazno zmanjšanje perpleksnosti (približno 3 do 5 %). Opazimo tudi, da že pri bigramskih modelih perpleksnost modelov s kontekstno strukturo pade pod minimalno perpleksnost posameznih modelov, kot je prikazano v tabeli 7.16.

Primerjava s preprostimi morfološkimi modeli je nekoliko težavnejša. Pri modelih za besedno vrsto vidimo manjše perpleksnosti pri preprostih modeli, kar je posledica uporabe trenutne besedne vrste, ki je v modelih s kontekstno odvisno strukturo ne uporabljamo. Ponovno pa vidimo, da modeli za oznake MSD, ki so se sicer pokazali za bolj uporabne pri razpoznavanju govora, že pri bigramskem modelu dajejo manjšo perpleksnost.

7.7 Razpoznavanje z modeli s kontekstno odvisno strukturo

Prvi korak v testiranju uspešnosti razpoznavanja govora z modeli s kontekstno odvisno strukturo je ponovno ocenjevanje vseh hipotez razvojne množice z vsemi posameznimi izdelanimi modeli. Ker se je ponovno ocenjevanje hipotez z veliko množico modelov pokazalo kot časovno zahtevno, smo se glede na ugotovitve pri uporabi preprostih morfoloških modelov omejili na 250 hipotez. Ocenjevanje hipotez s posameznimi modeli sicer ni zahtevno, vendar smo vse skupaj tvorili 322 modelov. Različni modeli imajo različno časovno zahtevnost. Povprečno vrednost faktorja realnega časa pa ocenjujemo na približno 0,013 za en model oz. 0,65 za 50 modelov, kolikor jih pride v poštev pri posameznem modeliranem faktorju.

Tukaj velja pripomniti, da smo z vsemi modeli ocenjevali vse besede v hipotezah. S tem smo dobili najbolj splošno uporabne rezultate. Iz dobljenih rezultatov smo nato izluščili le tiste verjetnosti, ki pridejo pri posamezni besedi v poštev. V primeru implementacije sistem, ki bi bil v praksi bolj uporaben, bi bilo potrebno tvoriti orodje, ki bi ob

ocenjevanju neke besede uporabljalo le tiste modele, ki so primerni. Takšen sistem bi lahko bistveno hitreje deloval, vendar bi imel večjo zahtevnost glede pomnilniškega prostora in bi vračal tudi manj informacij, primernih za nadaljnjo analizo delovanja modelov.

Za vsako naučeno strukturo modelov smo nato iz dobljenih rezultatov poiskali ustrezne vrednosti in tako tvorili končno verjetnost modela s kontekstno odvisno strukturo. Tako smo tvorili 14 rezultatov za unigramske do 5-gramske modele za besedno vrsto, oznako MSD in besedo. Izvzeli smo le unigramski model za besedno vrsto, ker ta nima nobenega faktorja, glede na katerega bi lahko izdelali model.

7.7.1 Rezultati na razvojni množici

Po ocenjevanju hipotez v razvojni množici smo ponovno uporabljali algoritem za optimizacijo uteži jezikovnih modelov. Optimizacijski algoritem smo izvajali na več različnih naborih modelov. Začetne vrednosti algoritma smo izbrali glede na izkušnje z uporabo preprostih morfoloških modelov in na podlagi nekaj poskusnih iteracij algoritma z modeli s kontekstno odvisno strukturo. Ponovno smo za vsako kombinacijo modelov izbrali več različnih začetnih vrednosti algoritma.

Tabela 7.18: Rezultati razpoznavanja na razvojni množici z enim modelom s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene z bigramskim modelom v prvem prehodu.

Red modela	Modeliran faktor		
	P	M	W
1	/	26,23	26,26
2	26,27	26,19	26,24
3	26,22	25,94	26,20
4	24,47	25,80	26,20
5	24,47	25,71	26,16

Tabela 7.19: Rezultati razpoznavanja na razvojni množici z enim modelom s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene s trigramskim modelom v prvem prehodu.

Red modela	Modeliran faktor		
	P	M	W
1	/	25,99	25,97
2	26,01	59,89	25,94
3	25,91	25,61	25,92
4	25,11	25,41	25,92
5	23,58	25,25	25,91

Najprej smo kombinirali verjetnosti iz prvega prehoda z le enim modelom s kontekstno odvisno strukturo. Rezultati deleža napačno razpoznanih besed so prikazani v

Tabela 7.20: Rezultati razpoznavanja na razvojni množici z dvema modeloma s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene z bigramskim modelom v prvem prehodu.

Red modela	Modeliran faktor		
	P+M	P+W	M+W
2	26,14	26,25	26,16
3	25,95	26,20	25,90
4	25,19	25,40	25,72
5	24,08	24,39	25,63

Tabela 7.21: Rezultati razpoznavanja na razvojni množici z dvema modeloma s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene s trigramskim modelom v prvem prehodu.

Red modela	Modeliran faktor		
	P+M	P+W	M+W
2	25,86	25,95	25,78
3	25,54	25,83	25,50
4	24,70	24,98	25,34
5	22,98	23,44	25,19

tabelah 7.18 in 7.19. Prikazani so najboljši rezultati po optimizaciji parametrov. Rezultati v tabeli 7.18, kjer smo v prvem prehodu uporabili bigramske modele, kažejo relativno izboljšanje rezultatov glede na prvi prehod med 11,1 in 17,2 %. Rezultati v tabeli 7.19, kjer smo v prvem prehodu uporabili trigramske modele, kažejo izboljšanja med 2,29 in 14,42 % relativno.

Nadaljevali smo s kombinacijo dveh modelov s kontekstno odvisno strukturo. Ustrezni rezultati so podani v tabelah 7.20 in 7.21. Dosegli smo izboljšanja med 11,2 in 18,53 % relativno oz. med 2,51 in 13,7 % relativno. Zadnji rezultati se nanašajo na uporabo vse treh modelov z morfološko strukturo. Rezultati so prikazani v tabelah 7.22 in 7.23. Dosegli smo izboljšanja med 11,3 in 19,0 % relativno ter med 2,91 in 14,06 % relativno. V vseh preizkusih, kjer smo kombinirali več modelov z morfološko odvisno strukturo, smo se omejevali na kombinacije modelov istega reda.

V vseh prikazanih rezultatih opažamo zmanjševanje deleža napak z večanjem reda modela. Pri tem vidimo večji doprinos k uspešnosti s strani modelov za besedne vrste in oznake MSD. Opažamo tudi, da imamo boljše rezultate, kadar uporabljamo več modelov s kontekstno odvisno strukturo. Najboljši rezultat dobimo pri uporabi vseh treh 5-gramskih modelov s kontekstno odvisno strukturo. Rezultat 22,87 % je za 14,1 % relativno boljši od pripadajočega rezultata, ki smo ga dosegli v prvem prehodu razpoznavanja.

Po samem postopku optimizacije smo tudi preverili območja, v katerih ležijo optimalne vrednosti uteži. Ugotovili smo, da se ne razlikujejo bistveno od območij, ki smo jih ugotovili pri razpoznavanju s preprostimi morfološkimi modeli.

Tabela 7.22: Rezultati razpoznavanja na razvojni množici s tremi modeli s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene z bigramskim modelom v prvem prehodu.

Red modela	Delež napak
2	26,22
3	25,84
4	25,11
5	23,93

Tabela 7.23: Rezultati razpoznavanja na razvojni množici s tremi modeli s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene s trigramskim modelom v prvem prehodu.

Red modela	Delež napak
2	25,84
3	25,44
4	24,54
5	22,87

7.7.2 Rezultati na testni množici

Na podlagi najboljših rezultatov, dobljenih na razvojni množici smo izbirali optimalne vrednosti uteži. Rezultati kažejo, da je optimalna vrednost števila hipotez odvisna od reda modela v prvem prehodu. Pri uporabi bigramskih modelov se kot optimalna vrednost izkaže 250, pri uporabi trigramskih pa 30. Optimalne vrednosti uteži jezikovnega modela se vedno gibljejo okoli vrednosti 14. Optimalne vrednosti modelov s kontekstno odvisno strukturo pa so precej manjše in se razlikujejo glede na modeliran faktor. Za oznako MSD so optimalne vrednosti v območju med 2 in 4, za besedo pa 0,2 in 1,5. Optimalne vrednosti za model besednih vrst pa so celo negativne in se gibljejo med -4 in -8.

Nato smo s temi vrednostmi testirali modele s kontekstno odvisno strukturo na testni množici. Rezultati so prikazani v tabelah 7.24 in 7.25. V obeh tabelah smo prikazali le po en rezultat za vsak nabor modelov različnih redov. Zapisali smo najboljši rezultat in red modelov, pri katerem smo ga dobili.

Tabela 7.24: Rezultati razpoznavanja na testni množici z modeli s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene z bigramskim modelom v prvem prehodu.

Modeli	Red modelov	Delež napak [%]	Relativno izboljšanje [%]
P	5	26.51	9.26
M	4	25.84	11.56
W	5	26.07	10.78
P+M	5	25.92	11.27
P+W	5	26.38	9.71
M+W	4	25.92	11.30
Vsi	5	26.01	10.99

Tabela 7.25: Rezultati razpoznavanja na testni množici z modeli s kontekstno odvisno strukturo. Uporabili smo hipoteze, dobljene s trigramskim modelom v prvem prehodu.

Modeli	Red modelov	Delež napak [%]	Relativno izboljšanje [%]
P	3	25.90	−1.01
M	5	25.39	0.99
W	5	25.46	0.70
P+M	5	25.61	0.14
P+W	5	25.95	−1.20
M+W	2	25.20	1.73
Vsi	5	25.45	0.74

Pri rezultatih na seznamih N najboljših besed, ki smo jih dobili z uporabo bigramskih modelov v prvem prehodu, smo povsod opazili bistveno izboljšanje rezultata, ki je delno posledica uporabe trigramskih jezikovnih modelov in delno posledica uporabe morfoloških modelov v drugem prehodu. Vidimo pa, da so izboljšanja manjša tam, kjer uporabljamo model za besedne vrste, iz rezultatov pa je tudi razvidno, da ima največji doprinos model za oznake MSD. Te ugotovitve ustrezajo ugotovitvam pri uporabi preprostih morfoloških modelov.

V tabeli 7.25, kjer smo uporabljali sezname N najboljših besed, ki smo jih dobili z uporabo trigramskih modelov v prvem prehodu, so izboljšanja rezultatov manjša. Ponovno vidimo, da imajo modeli za oznake MSD največji doprinos, medtem ko se zdi, da modeli za besedne vrste celo zmanjšujejo uspešnost razpoznavanja.

Podrobnejša analiza rezultatov na testni množici je pokazala, da se optimalne vrednosti uteži modelov precej razlikujejo glede na vrednosti, izbrane na podlagi rezultatov na razvojni množici. To kaže na različnost obeh množic, ki vpliva na postopek optimizacije parametrov. Največje razlike se pojavijo pri utežeh za modele besednih vrst, kjer dobimo pozitivne vrednosti, ki so dejansko tudi bolj smiselne od negativnih. Razlike pri parametrih za druge modele so tudi prisotne, vendar so manjše. Tukaj predvidevamo, da so negativne optimalne vrednosti na razvojni množici posledica postopka učenja strukture modelov na tej isti množici. V namen testiranja te predpostavke bi bilo potrebno imeti še dodatno neodvisno razvojno množico, ki bi jo uporabljali za optimizacijski postopek. Tukaj bi bilo pomembno, da bi nova razvojna množica bila čim bolj podobna testni množici, s čimer bi lahko dosegali bolj zanesljive rezultate.

Rezultate v tabeli 7.25 lahko uporabimo za potrditev osnovne teze doktorske naloge, da lahko z uporabo morfoloških modelov s kontekstno odvisno strukturo izboljšamo rezultate razpoznavanja. Primerjava z rezultati v tabeli 7.14 kaže tudi na to, da modeli s kontekstno odvisno strukturo dajejo boljše rezultate od preprostih morfoloških modelov. Postopek optimizacije je pokazal, da imajo modeli s kontekstno odvisno strukturo potencial še dodatno povečati uspešnost v primeru ustrezne razvojne množice. Na podlagi vseh rezultatov na testni množici ocenjujemo, da bi lahko dosegli izboljšanja rezultatov do 3 %.

7.8 Analiza rezultatov

Rezultate, opisane v prejšnjem podpoglavju, smo podrobneje analizirali glede na razlike v prvem in drugem prehodu. Izbrali smo rezultat prvega prehoda, kjer smo uporabljali trigramske jezikovne modele in slovar velikosti 300k. Kot rezultat drugega prehoda smo izbrali najboljši rezultat, ki smo ga dobili iz hipotez, ki ustrezajo izbranemu rezultatu prvega prehoda. Najprej smo preverili statistično značilnost izboljšanja rezultata. Uporabili smo test naključnega približevanja (angl. approximate randomization) z 10.000 testnimi primeri, za stopnjo statistične značilnosti pa smo izbrali 0,05. Ugotovljena je bila p-vrednost 0,0055, ki nakazuje statistično značilnost rezultata.

V tabeli 7.26 so navedena osnovna števila različnih napak in sprememba med prvim in drugim prehodom. Iz podatkov je razvidno, da so jezikovni modeli v drugem prehodu močno zmanjšali število izbrisanih besed, medtem ko se je število vrinjenih in zamenjanih besed nekoliko povečalo. Rezultati so dobljeni z uporabo Levenshteinove razdalje med izbrano hipotezo in referenčno transkripcijo.

Tabela 7.26: Primerjava rezultatov iz prvega in drugega prehoda z uporabo Levenshteinove razdalje.

Vrsta napake	1. prehod	2. prehod	Razlika
Izbrisane besede	1417	1189	-228
Vrinjene besede	372	471	99
Zamenjane besede	4042	4071	29
Vse napake	5831	5731	-100

Analiza frekvenc posameznih napak je pokazala, da se vse vrste napak najpogosteje pojavljajo pri kratkih besedah, ki so tudi sicer najpogostejše v testni množici. Pri tem kratke besede izstopajo predvsem pri izbrisanih in vrinjenih besedah, manj pa pri zamenjanih. Ne pojavljajo pa se bistvene razlike glede na prehod razpoznave in izbiro poravnave.

Analizirali smo tudi število napak na nivoju lem. Ugotovili smo, da je v vseh primerih delež napak manjši za 10 do 11 %. Na podlagi tega rezultata lahko predvidevamo, da je približno desetina vseh napak posledica napačno prepoznane besedne oblike, vendar pravilno razpoznane leme.

V tabeli 7.27 so enaki rezultati izračunani z uporabo razširjene Levenshteinove razdalje, opisane v enačbi 6.2. Sami rezultati uspešnosti se bistveno ne razlikujejo. Vendar pa razširjena Levenshteinova razdalja daje drugačno poravnavo med hipotezo in referenčno transkripcijo, ki je bolj primerna za nadaljnjo analizo napak, ki jih bomo obravnavali v nadaljevanju

V tabeli 7.28 so prikazani podrobnejši rezultati za napake različnih besednih vrst. Navedeni so podatki za število napak v prvem in v drugem prehodu ter delež teh napak

Tabela 7.27: Primerjava rezultatov iz prvega in drugega prehoda z uporabo razširjene Levenshteinove razdalje.

Vrsta napake	1. prehod	2. prehod	Razlika
Izbrisane besede	1446	1216	−230
Vrinjene besede	401	498	97
Zamenjane besede	4013	4043	30
Vse napake	5860	5757	−103

na skupno število izbrane besedne vrste v testni množici. Navedeni sta tudi absolutna in relativna sprememba števila napak med prehodoma. Iz podatkov vidimo, da se največja zmanjšanja napak pojavijo pri veznikih in predlogih, kar kaže na to, da so modeli uspešni pri ocenjevanju pomožnih besednih vrst. Večja zmanjšanja imajo tudi glagoli, pridevniki, prislovi in zaimki. Samostalniki in števniki pa sta edini besedni vrsti, pri katerih se število napak nekoliko poveča.

Tabela 7.28: Delež napak po besednih vrstah.

Besedna vrsta	1. prehod		2. prehod		Razlika	
Samostalnik	1559	23,20 %	1563	23,26 %	4	+0,25 %
Glagol	1101	28,22 %	1079	27,66 %	-22	−2,00 %
Pridevnik	547	20,67 %	538	20,33 %	-9	−1,65 %
Prislov	414	24,84 %	403	24,18 %	-11	−2,66 %
Zaimek	525	34,84 %	511	33,27 %	-14	−2,67 %
Števnik	204	27,31 %	207	27,71 %	3	+1,47 %
Predlog	688	27,53 %	666	26,65 %	-22	−3,19 %
Veznik	553	28,16 %	526	26,78 %	-27	−4,88 %
Členek	257	24,76 %	254	24,47 %	-3	−1,17 %
Medmet	8	100,00 %	4	50,00 %	-4	−50,00 %
Neznano	4	66,67 %	6	100,00 %	2	+50,00 %

Tabela 7.29: Število napačno razpoznanih besed pri pravilno razpoznanih besednih vrstah in lemah.

Besedna vrsta	1. prehod		2. prehod	
Samostalnik	296	34.58 %	291	34.64 %
Pridevnik	135	57.94 %	126	53.58 %
Glagol	183	48.93 %	177	47.58 %
Zaimek	47	52.22 %	46	51.11 %
Števnik	16	20.78 %	16	21.33 %
Skupaj	667	41.53 %	656	40.72 %

V tabeli 7.29 so prikazani še rezultati za število napak na pregibnih besednih vrstah, kjer je bila prepoznana pravilna besedna vrsta in pravilna lema, vendar napačna končnica. Te napaka kažejo na napačno razpoznane končnice. Podan je tudi delež teh napak glede na vse pravilno razpoznane besedne vrste. Iz podatkov lahko vidimo, da je približno 40 % napak, ki se pojavijo pri teh besednih vrstah, posledica napačno razpoznane končnice. V

drugem prehodu se je skupno število napak na pregibnih besednih vrstah zmanjšalo, prav tako se je nekoliko zmanjšal tudi delež napačno razpoznanih besed. Ta rezultat dodatno potrjuje, da morfološki jezikovni modeli s kontekstno odvisno strukturo pripomorejo k zmanjšanju napačno razpoznanih oblik besede.

Poglavje 8

Zaključek

V doktorski nalogi smo obravnavali jezikovno modeliranje za razpoznavanje govora pregibnih jezikov. V teh jezikih, mednje spada tudi slovenščina, običajno opazimo manjšo uspešnost razpoznavanja, ki je delno posledica napačno prepoznanih besednih oblik pregibnih besednih vrst. Ker besedna oblika nosi informacijo o slovničnih kategorijah besede, smo rezultate razpoznavanja poskušali izboljšati z uporabo morfoloških modelov. V doktorski nalogi smo tako zasnovali sistem razpoznavanja tekočega govora, ki temelji na uporabi več jezikovnih modelov.

Uporabili smo že uveljavljene besedne n-gramske modele. V nalogi smo predstavili rezultate razpoznavanja z različnimi redi modelov, različnimi velikostmi slovarja in različnimi načini glajenja. Prišli smo do pričakovanih ugotovitev, da povečanje reda modela in povečanje slovarja prineseta izboljšanje rezultatov razpoznavanja. Ugotovili smo tudi, da ima način glajenja le majhen vpliv na uspešnost razpoznavanja in da je uspešnost s Knesser-Neyevim glajenjem, ki velja sicer za bolj ugodno od Good-Turingovega, na slovenski govorni bazi Broadcast News slabša. Celoten sistem razpoznavanja smo zasnovali kot dvoprehodni algoritem. Potem ko smo v prvem prehodu uporabili besedne n-gramske modele in tvorili seznam N najboljših hipotez, smo v drugem prehodu izvajali razpoznavanje z morfološkimi modeli.

Morfološke modele smo zasnovali kot faktorizirane. Definirali smo morfološke modele s kontekstno odvisno strukturo, v katerih sta nabor začetnih faktorjev in pot sestopanja odvisna od besedne vrste besede, ki se ocenjuje, in besednih vrst predhodnih besed (njenega konteksta). Definicija takšnih modelov predvideva poljuben nabor začetnih faktorjev v poti sestopanja iz nabora vseh faktorjev in poljubno pot sestopanja skozi izbrane faktorje za vsako možno zaporedje besednih vrst – kombinacij zaporedij POS. Zaradi tega smo se pri učenju strukture modelov soočali z velikim iskalnim prostorom za strukturo modela in pomanjkanjem učnih podatkov za veliko množico možnih kombinacij zaporedij POS.

Med samim postopkom učenja modelov smo uporabljali njihovo perpleksnost na izbrani razvojni množici kot merilo za njihovo uspešnost. Da smo to lahko storili smo

izpeljali enačbe za opis izračuna perpleksnosti za izbrane besede na nekem besedilu, ki imajo ustrezen kontekst. Modele smo učili na ročno označenem korpusu slovenskega jezika ssj500k, perpleksnost pa smo preverjali na razvojni množici slovenske govorne baze Broadcast News.

Da bi omejili množico struktur modelov, ki smo jih preizkušali, smo zasnovali algoritem za postopno gradnjo strukture modelov. Ta je deloval tako, da je strukturo modela začel graditi s faktorjem, ki je sam največ prinesel k uspešnosti modela. Najboljšim modelom z enim samim faktorjem smo nato iterativno dodajali naslednje faktorje. Za ta namen smo definirali način omejevanja iskalnega prostora, ki je bil zasnovan po zgledu snopovnega omejevanja v sinhronih algoritmih za razpoznavanje govora. Izločali smo strukture modelov, ki so dale rezultate, za več kot izbrano razliko slabše od najboljšega modela. Izločali smo tudi strukture modelov, ki so po dodajanju novega faktorja prinesle le majhna izboljšanja ali pa celo poslabšanja perpleksnosti. S takšnim postopkom smo lahko bistveno zmanjšali potrebno število učenj modelov na učni množici in njihovega testiranja na razvojni množici.

Da ne bi učili modelov za vse teoretično možne kombinacije zaporedij POS, smo najprej omejili nabor možnih kombinacij na tiste, ki se pojavijo v razvojni množici. Teh je bilo še vedno toliko, da ni bilo možno za vsako izmed njih zanesljivo preveriti uspešnosti modela z neko strukturo. V ta namen je bilo potrebno zmanjšati število različnih kombinacij, za katere preverjamo uspešnost modelov. Kombinacije zaporedij POS smo združevali v skupine, ki smo jih poimenovali razrede. Zasnovali smo algoritem združevanja kombinacij, ki temelji na preverjanju podobnosti razredov. Podobnost dveh razredov smo definirali na podlagi preseka množic struktur najuspešnejših morfoloških modelov v dveh razredih.

Da bi preverjali uporabnost izdelanih morfoloških modelov, smo jih preizkusili na testni množici slovenske govorne baze Broadcast News. Izdelali smo orodje za ocenjevanje hipotez, dobljenih v prvem prehodu, z morfološkimi modeli s kontekstno odvisno strukturo. Da smo lahko ustrezno uporabili rezultate modelov, smo morali izbrati ustrezne uteži modelov v končnem ocenjevanju hipotez. V ta namen smo razpoznavanje vzporedno izvajali še na razvojni množici baze. Na tej množici smo optimizirali uteži modelov in število upoštevanih hipotez. Ker je z uporabo več jezikovnih modelov narasla tudi množica kombinacij uteži, smo še razvili algoritem za njihovo hitro optimizacijo.

Ker se je med uporabo modelov postavilo vprašanje, ali bodo morebitna izboljšanja posledica uporabe kontekstno odvisne strukture morfoloških modelov ali pa posledica samo uporabe morfoloških informacij, smo celoten postopek razpoznavanja izvedli tudi s preprostimi morfološkimi modeli, ki nimajo kontekstno odvisne strukture. Po optimizaciji parametrov in izvedbe drugega prehoda razpoznavanja smo primerjali rezultate iz prvega prehoda, rezultate s preprostimi morfološkimi modeli in rezultate z modeli s kontekstno odvisno strukturo. Ugotovili smo, da nam je razpoznavanje s preprostimi morfološkimi modeli prineslo le nebistvena izboljšanja rezultatov razpoznavanja, medtem ko je uporaba

modelov s kontekstno odvisno strukturo prenesla večje razlike v uspešnosti razpoznavanja.

Povzamemo lahko, da smo potrdili osnovno tezo naše naloge, da lahko z morfološkimi modeli s kontekstno odvisno strukturo izboljšamo razpoznavanje tekočega govora z velikim slovarjem. Pri tem smo razvili algoritma za učenje strukture modelov in združevanje različnih kontekstov besede, ki so bili potrebni za praktično implementacijo takšnega sistema. Razvili smo tudi algoritem za optimizacijo uteži jezikovnega modela v ocenjevanju hipotez razpoznavalnika.

Dosegli smo zmanjšanje števila napak za 1,73 %. Na podlagi primerjave s preprostimi morfološkimi modeli lahko sklepamo, da je izboljšanje posledica vpeljave modelov s kontekstno odvisno strukturo. Primerjava z razvojno množico in interpretacija rezultatov optimizacije uteži pa kažeta na potencial večjega izboljšanja rezultatov v primeru boljšega ujemanja med razvojno in testno množico.

Literatura

[1] E. Alba, G. Luque, L. Araujo, *Natural language tagging with genetic algorithms*, Information Processing Letters, vol. 100, no. 5, pp. 173–182, 2006

[2] F. Alleva, X. Huang, M-Y. Hwang, *An improved search algorithm using incremental knowledge for continuous speech recognition*, Acoustics, Speech, and Signal Processing, 1993 IEEE International Conference on, pp. 307–310, 1993

[3] Š. Arhar, V. Gorjanc, *Korpus FidaPLUS: nova generacija slovenskega referenčnega korpusa*, Jezik in slovstvo, let. 52, št. 2, str. 95–110, 2007

[4] X.L. Aubert, *An Overview of Decoding Techniques for Large Vocabulary Continuous Speech Recognition*, Computer Speech & Language, vol. 16, no. 1, pp. 89–114, 2002

[5] A. E. Axelrod, *Factored Language Models for Statistical Machine Translation*, University of Edinburgh, 2006

[6] A. Biem, E. McDermott, S. Katagiri, *A Discriminatice Filter Bank Model for Speech Recognition*, Proceedings of the IEEE, ICASSP-96, pp. 545–548, 1995

[7] J. A. Bilmes, K. Kirchhoff, *Factored Language Models and Generalized Parallel Backoff*, Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, vol. 2, pp. 4–6, 2003

[8] Bisani, M. and Ney H. (2004). Bootstrap Estimates for Confidence Intervals in ASR Performance Evaluation. ICASSP 2004 Proceedings. Montreal, Quebec, Canada: ICASSP, I-409–412, 2004

[9] T. Brants, *TnT – A Statistical Part-of-Speech Tagger*, In Proceedings of the Sixth Conference on Applied Natural Language Processing. Seattle, Washington, pp. 224–231, 2000

[10] S. F. Chen, J. Goodman, *An Empirical Study of Smoothing Techniques for Language Modeling*, Harvard University, 1998

[11] O. Cheng, J. Dines, M. M. Doss, *A Generalized Dynamic Composition Algorithm of Weighted Finite State Transducers for Large Vocabulary Speech Recognition*, Acoustics, Speech and Signal Processing, ICASSP 2007 IEEE International Conference on, vol. 4, pp. IV-345–IV-348, 2007

[12] G. Donaj, Z. Kačič, *Širjenje slovarja in dvoprehodni algoritem v razpoznavalniku tekočega govora UMB Broadcast News*, Jezikovne tehnologije 2012, str. 48–51, 2012

[13] G. Donaj, Z. Kačič, *The use of several language models and its impact on word insertion penalty in LVCSR*, Speech and Computer, pp. 354–361, 2013

[14] T. Erjavec, V. Gorjanc, M. Stabej, *Korpus FIDA*, Jezikovne tehnologije za slovenski jezik, str. 124–127, 1998

[15] T. Erjavec, S. Krek, *Oblikoskladenjske specifikacije in označeni korpusi JOS*, Jezikovne tehnologije 2008, str. 59–53, 2008

[16] T. Erjavec, D. Fišer, S. Krek, S. Ledinek, *The JOS linguistically tagged corpus of Slovene*, 7th International Conference on Language Resources and Evaluations (LREC-10), pp. 1806–1809, 2010

[17] M. Gales, S. Young, *The Application of Hidden Markov Models in Speech Recognition*, Foundations and Trends in Signal Processing, vol. 1, no. 3, pp. 195–304, 2007

[18] L. Gillick, S. J. Cox, *Some Statistical Issues in the Comparison of Speech Recognition Algorithms*, ICASSP 1989 Proceedings. Glasgow, Scotland: ICASSP, 532–535, 1989

[19] J. Giménez, L. Màrquez, *SVMTool: A gen-eaeral POS tagger generator based on support vector machines*, In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04), pp. 43–46, 2004

[20] V. Goel, S. Kumar, W. Byrne, *Segmental minimum Bayes-risk decoding for automatic speech recognition*, Speech and Audio Processing, IEEE Transactions on, vol. 12, no. 3, pp. 234–249, 2004

[21] I.J. Good, *The population frequencies of species and the estimation of population parameters*, Biometrica, vol. 40, no. 3–4, pp. 237–264, 1953

[22] G. Gosztolya, A. Kocsor, *Speeding up dynamic search methods in speech recognition*, Innovations in Applied Artificial Intelligence, IEA/AIE' 2005 Proceedings of the 18th international conference on, pp. 98–100, 2005

[23] M. Grčar, S. Krek, K. Dobrovoljc *Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik*, Jezikovne tehnologije 2012, str. 89–94, 2012

[24] H. Hermansky, *Perceptual linear predictive (PLP) analysis of speech*, Journal of the Acoustical Society of America, vol. 87, no. 4, pp. 1738–1752, 1990

[25] T. Hirsimäki M. Kurimo, *Decoder Issues in Unlimited Finnish Speech Recognition*, Proceedings of the 6th nordic Signal Processing Symposium 2004, pp. 320–323

[26] T. Hirsimäki, M. Creutz, V. Siivola, M. Kurimo, S. Virpioja, J. Pylkkönen, *Unlimited vocabulary speech recognition with morph language models applied to Finnish*, Computer Speech & Language, vol. 20, no. 4, pp. 515–541, 2006

[27] T. Hirsimäki, J. Pylkkonen, M. Kurimo, *Importance of High-Order N-Gram Models in Morph-Based Speech Recognition*, IEEE Transactions on Audio Speech and Language Processing, vol. 17, no. 4, pp. 724–732, 2009

[28] T. Hori, S. Watanabe, A. Nakamura, *Search error risk minimization in Viterbi beam search for speech recognition*, Acoustics Speech and Signal Processing, ICASSP 2010 IEEE International Conference on, pp. 4934–4937, 2010

[29] X. Huang, A. Acero, H.-W. Hon, *Spoken Language Processing*, Prentice Hall, 2001

[30] S. Huet, G. Gravier, P. Sebillot, *Morpho-syntactic post-processing of N-best lists for improved French automatic speech recognition*, Computer Speech & Language vol. 24, no. 4, pp. 663–684, 2010

[31] P. Ircing, J.V. Psutka, J. Psutka, *Using Morphological Information for Robust Language Modeling in Czech ASR System*, IEEE Transactions on Audio, Speech and Language Processing, vol. 17, no. 4, pp. 840–847, 2009

[32] F. Jelinek, *Continuous Speech Recognition by Statisical Methods*, IEEE Proceedings, vol. 64, no. 4, pp. 532–556, 1976

[33] Jiang, H., *Discriminative training of HMMs for automatic speech recognition: A Survey*, Computer Speech and Language, vol. 24, pp. 589–608, 2010

[34] S. Katz, *Estimation of probabilities from sparse data for the language model component of a speech recognizer*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 35, no.3, pp. 400–401, 1987

[35] K. Kirchhoff, et al, *Novel Speech Recognition Models for Arabic*, Johns-Hopkings University Summer Research Workshop 2002, Final Report

[36] K. Kirchhoff, D. Vergyri, J. Bilmes, K. Duh, A. Stolcke, *Morphology-based language modeling for conversational Arabic speech recognition*, Computer Speech & Language, vol. 20, no. 4, pp. 589–608, 2006

[37] K. Kirchoff, J. Bilmes, K. Duh, *Factored Language Models Tutorial*, http://ssli.ee.washington.edu/people/duh/papers/flm-manual.pdf, dostopano 1. 2. 2015

[38] N. Kitaoka, L. Ying, N. Takahashi, S. Nakagawa, *One-pass LVCSR algorithm using linear lexicon search and 1-best approximation tree-structured lexicon search* Signal Processing and Its Applications, ISSPA 2007 9th International Symposium on, pp. 1–4, 2007

[39] M. Kos, Z. Kačič, D. Vlaj, *Acoustic classification and segmentation using modified spectral roll-off and variance-based features Original Research Article*, Digital Signal Processing, vol. 23, no. 2, pp. 659–674, 2013

[40] C-H- Lee, L. R. Rabiner, *A Frame-Synchronous Network Search Algorithm for Connected Word Recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 11, pp. 1649–1658, 1989

[41] C. D. Manning, H. Schütze, *Fundamentals of natural language processing*, MIT Press, 1999

[42] L. Màrquez, H. Rodríguez, *Part-of-speech tagging using decision trees*, 1998

[43] B. Milner, *A Comparison of Front-end Configurations for Robust Speech Recognition*, Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on, pp.I-797–I-800, 2002

[44] M. Mohri, F. Pereira, M. Riley, *Weighted finite-state transducers in speech recognition*, Computer Speech & Language, vol. 16, no. 1, pp. 69–88, 2002

[45] A.E-D. Mousa, M.A.B. Shaik, R. Schlüter, H. Ney, *Sub-lexical language models for German LVCSR*, Spoken Language Technology Workshop (SLT), 2010 IEEE, pp. 171–176, 2010

[46] A.E-D. Mousa, Schlüter, R., Ney, H., *A hybrid Mophologically Decomposed Factored Language Models for Arabic LVCSR*, Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Volume 9, pp. 701–704, 2010

[47] A.E-D. Mousa, Shaik, M.A.B., Schlüter, R., Ney, H., *Morpheme Based Factored language Models for German LVCSR*, Interspeech 2011, pp. I-1053–1056

[48] J. Nouza, D. Nejedlova, J. Zdansky J. Kolorenc, *Very Large Vocabulary Speech Recognition System for Automatic Transcription of Czech Broadcast Programs*, Proceedings of Interspeech 2004, pp. 409–412, 2004

[49] J. Nouza, J. Zdansky, P. Cerva, J. Silovsky, *Challenges in Speech Processing of Slavic Languages (Case Studies in Speech Recognition of Czech and Slovak)*, COST 2102 Int. Training School 2009, pp. 225–241, 2010

[50] P. M. Nugues, *An Introduction to Language Processing with Perl and Prolog*, Springer, 2006

[51] S. Ortmanns, H. Ney, *The time-conditioned approach in dynamic programming search for LVCSR*, Speech and Audio Processing, IEEE Transactions on, vol.8, no.6, pp. 676–687, 2000

[52] L.R. Rabiner, *A Tutoririal on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989

[53] L. Rabiner, B-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993

[54] J. Ramírez, J. M. Górriz, J. C. Segura *Voice Activity Detection. Fundamentals and Speech Recognition System Robustness.* In M. Grimm and K. Kroschel. Robust Speech Recognition and Understanding, 2007.

[55] A. Ratnaparkhi, *A Maximum Entropy Model for Part-Of-Speech Tagging*, 1996

[56] S. Riezler, J.T. Maxwell III, *On Some Pitfails in Automatic Evaluation and Significance testing fot MT* ACL05 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization, 2005

[57] T. Rotovnik, *Avtomatsko razpoznavanje govora za pregibni jezik z velikim slovarjem besed z uporabo podbesednih modelov osnova-končnica*, doktorska disertacija, Maribor, 2004

[58] T. Rotovnik, M. Sepesy Maučec, Z. Kačič, *Large vocabulary continuous speech recognition of an inflected language using stems and endings*, Speech Communication, vol. 49, no. 6, pp. 437–452, 2007

[59] Sak, H., Saraçlar, M., Güngör, T., *Morphology-based and Sub-word Language Modeling for Turkish Speech Recognition*, 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 5402–5405

[60] R. Schlüter, *Investigations on Discriminative Training Criteria*, Ph.D. Dissertation, u RWTH Aachen University Technology, Aacher, Germany, 2000

[61] H. Schmid, *Part-of-speech tagging with neural networks*, In Proceedings of the 15th International Conference on Computational Linguistics (COLING 1994), pp. 172–176, 1994

[62] M. Sepesy Maučec, T. Rotovnik, M. Zemljak, *Modelling Highly Inflected Slovenian Language*, International Journal of Speech Technology, vol. 6, no. 3, pp. 245–257, 2003

[63] M. Sepesy Maučec, G. Donaj, Z. Kačič, *Improving statistical machine translation with additional language models*, 6th Language & Technology Conference, pp. 137–141, 2013

[64] A. Seward, *A fast HMM match algorithm for very large vocabulary speech recognition*, Speech Communication, vol. 42, no. 2, pp. 191–206, 2004

[65] M.A.B. Shaik, A.E-D. Mousa, R. Schlüter, H. Ney, *Using Morpheme and Syllable based Sub-Words for Polish LVCSR*, Proceedings of ICASSP 2011, pp. 4680–4683, 2011

[66] Tachbelie, M. Y., Menzel, W. *Capturing Worl-level Dependencies in Morpheme-based Language Modeling*, Procedings of the Second Workshop on African Language Technology, 2010, pp. 43–48

[67] V. Valtchev, *Discriminative Methods in HMM-based Speech Recognition*, Ph.D. Dissertation, Cambridge Unversity, UK, 1995.

[68] E.W.D. Whittaker, P.C. Woodland, *Language modelling for Russian and English using words and classes*, Computer Speech & Language, vol. 17, no. 1, pp. 87–104, 2003

[69] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. C. Woodland, *The HTK Book, version 3.4*, 2006

[70] S. Zablotskiy, K. Zablotskaya, W. Minker, *Some Approaches for Russian Speech Recognition*, 2010 Sixth International Conference on Intelligent Environments (IE), pp. 96–99, 2010

[71] J. Zhao, J. Hamaker, N. Deshmukh, A. Ganapathiraju, J. Picone, *Fast search algorithms for continuous speech recognition*, Southeastcon '99, Proceedings of the IEEE, pp. 36–39, 1999

[72] I. Zitouni, *Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition*, Computer Speech & Language, vol. 21, no. 1, Pages 88–104, 2007

[73] A. Žgank, D. Verdonik, Z. Kačič, *Slovenska baza BNSI Broadcast News za razpoznavanje tekočega govora*, Elektrotehniški vestnik, let. 75, št. 3, str. 85–90, 2008

[74] A. Žgank, M. Sepesy Maučec, *Razpoznavalnik tekočega govora UMB Broadcast News 2010: nadgradnja akustičnih in jezikovnih modelov*, Jezikovne tehnologije 2010, str. 28–31, 2010

Življenjepis kandidata

Rojen	14. 1. 1986	Ptuj
Šolanje	1993–2001	Osnovna šola Gorišnica
	2001–2005	II. gimnazija Maribor
	2005–2010	Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, univerzitetni študijski program Elektrotehnika, smer elektronika
	2006–2011	Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, univerzitetni študijski program Nepedagoška matematika
	2010–2015	Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, doktorski študijski program Elektrotehnika
Zaposlitev	2010–2014	Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko. Delovno mesto: mladi raziskovalec.
	2014–	Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko. Delovno mesto: visokošolski sodelavec – asistent.

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

IZJAVA DOKTORSKEGA KANDIDATA

Podpisani-a Gregor Donaj,

vpisna številka E9501382.

izjavljam,

da je doktorska disertacija z naslovom
Avtomatsko razpoznavanje govora za pregibni jezik z uporabo morfoloških jezikovnih modelov s kontekstno odvisno strukturo

- rezultat lastnega raziskovalnega dela,
- da predložena disertacija v celoti ali v delih ni bila predložena za pridobitev kakršnekoli izobrazbe po študijskem programu druge fakultete ali univerze,
- da so rezultati korektno navedeni in
- da nisem kršil-a avtorskih pravic in intelektualne lastnine drugih.

Podpis doktorskega-e kandidata-ke:

Obrazec RŠZ

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

IZJAVA O OBJAVI ELEKTRONSKE VERZIJE DOKTORSKE DISERTACIJE IN OSEBNIH PODATKOV, VEZANIH NA ZAKLJUČEK ŠTUDIJA

Ime in priimek doktoranda-ke: Gregor Donaj
Vpisna številka: E9501382
Študijski program: Elektrotehnika
Naslov doktorskega dela:
Avtomatsko razpoznavanje govora za pregibni jezik z uporabo morfoloških jezikovnih modelov s kontekstno odvisno strukturo

Mentor-ica: prof. dr. Zdravko Kačič
Somentor-ica:

Podpisani soglašam z objavo doktorske disertacije v Digitalni knjižnici Univerze v Mariboru.

Tiskana verzija doktorske disertacije je istovetna elektronski verziji, ki sem jo oddal-a v Digitalno knjižnico Univerze v Mariboru.

Podpisani-a hkrati izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum diplomiranja, naslov diplomskega dela) na spletnih straneh in v publikacijah Univerze v Mariboru.

Datum in kraj: 23.2.2015, Maribor

Podpis doktoranda-ke:

Obrazec RŠZ

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko
Smetanova ulica 17
2000 Maribor, Slovenija

FERI

IZJAVA KANDIDATOVEGA MENTORJA O USTREZNOSTI DOKTORSKE DISERTACIJE

Podpisani-a prof. dr. Zdravko Kačič, mentor doktorskemu kandidatu,

izjavljam, da je doktorska disertacija z naslovom Avtomatsko razpoznavanje govora za pregibni jezik z uporabo morfoloških jezikovnih modelov s kontekstno odvisno strukturo,

ki jo je izdelal doktorski kandidat Gregor Donaj,

v skladu z odobreno temo, Pravilnikom o pripravi in zagovoru doktorske disertacije ter mojimi navodili in predstavlja izviren prispevek k razvoju znanstvene discipline.

Datum in kraj:

23.2.2015, Maribor

Podpis mentorja-ice:

Obrazec RŠZ