UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Denis Bučković

# Glasovno vodenje računalnika s pomočjo sistema za razpoznavanje govora Sphinx-4

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2013

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00392/2013
Datum: 03.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DENIS BUČKOVIĆ**

Naslov: **GLASOVNO VODENJE RAČUNALNIKA S POMOČJO SISTEMA ZA RAZPOZNAVANJE GOVORA SPHINX-4**

**COMPUTER CONTROL WITH SPHINX-4 SPEECH RECOGNITION SOFTWARE**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Interakcija med človekom in strojem je v zadnjih nekaj letih močno napredovala. Sistemi za avtomatsko razpoznavanje govora omogočajo ljudem, da na povsem naraven način, to je s pomočjo govora, upravljajo razne naprave. Ena izmed uporab glasovnega upravljanja je tudi upravljanje računalnika. V okviru diplomskega dela predstavite sistem za razpoznavanje govora. Opišite sistem Sphinx-4 ter ga po lastnostih primerjajte z drugim sistemom za razpoznavanje govora Julius. V okviru tega izmerite tudi učinkovitost delovanja sistema Sphinx-4. V okviru drugega dela diplome s pomočjo sistema Sphinx-4 razvijte aplikacijo, ki razpozna človeški glas ter tako govorcu omogoča glasovno upravljanje računalnika.

Mentor: viš. pred. dr. Aljaž Zrnec

Dekan: prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

# IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Denis Bučković, z vpisno številko **63100022,** sem avtor diplomskega dela z naslovom:

*Glasovno vodenje računalnika s pomočjo sistema za razpoznavanje govora Sphinx-4*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aljaža Zrneca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

Ljubljani, dne Podpis avtorja:

*Najbolj se zahvaljujem staršem, saj so mi omogočili študij in me v času študija spodbujali in podpirali.*

*Zahvaljujem se svojemu mentorju viš. pred. dr. Aljažu Zrnecu za vso pomoč, prijaznost ter za napotke in nasvete pri izdelavi diplomske naloge.*

*Velika zahvala gre tudi moji sestri Doris za vso pomoč in podporo.*

# Kazalo vsebine

**Povzetek**

**Abstract**

**1 Uvod ..... 1**

**2 Govor ..... 3**
2.1 Predstavitev govora ..... 3
2.2 Tvorjenje govora ..... 4

**3 Sistem za razpoznavanje govora ..... 5**
3.1 Delitev sistemov glede na način razpoznave govora ..... 5
3.2 Učinkovitost sistemov ..... 6
3.3 Pregled orodji za razpoznavo govora po priljubljenosti ..... 7
3.4 Delovanje sistema za razpoznavo govora ..... 8
3.5 Zgradba splošnega sistema za razpoznavo govora ..... 9
3.6 Prednosti in slabosti sistemov za razpoznavo govora ..... 10
3.6.1 Prednosti ..... 10
3.6.2 Slabosti ..... 11
3.7 Pregled strojne opreme za potrebe razpoznavanja govora ..... 11
3.7.1 Zvočna kartica ..... 11
3.7.2 Mikrofon ..... 12
3.7.3 Procesor in pomnilnik ..... 12

**4 Sistema Sphinx-4 in Julius ..... 13**
4.1 Sistem Sphinx-4 ..... 13
4.1.1 Arhitektura sistema Sphinx-4 ..... 14
4.1.1.1 Ospredje ..... 15
4.1.1.2 Lingvist ..... 16
4.1.1.3 Dekodirnik ..... 17
4.2 Julius ..... 18
4.2.1 Arhitektura sistema Julius ..... 18
4.2.2 Značilnosti sistema Julius ..... 19

**5 Primerjava in testiranje ..... 21**
5.1 Primerjava Sphinx-4 in Julius ..... 21
5.1.1 Sistemske zahteve ..... 21
5.1.2 Treniranje / učenje ..... 21
5.1.3 Upravljalnik iskanja ..... 22
5.1.4 Glavne tehnike iskanja ..... 22
5.1.5 Primerjava ostalih značilnosti ..... 23
5.2 Testiranje in rezultati ..... 24

**6 Razvoj aplikacije za upravljanje računalnika z govornimi ukazi ..... 27**
6.1 Slovnica ..... 28
6.2 Konfiguracijska datoteka ..... 29
6.3 Uporabniški vmesnik ..... 31

**7 Sklepne ugotovitve ..... 35**

**Literatura ..... 36**

# Kazalo slik

Slika 1: Organi, ki sodelujejo pri tvorbi oz. obliki govora. ..... 4
Slika 2: Delovanje sistema za razpoznavo govora. ..... 8
Slika 3: Zgradba splošnega sistema. ..... 9
Slika 4: Zgradba sistema Sphinx-4. ..... 14
Slika 5: Sphinx-4 ospredje - zgradba. ..... 15
Slika 6: Iskalni graf. ..... 16
Slika 7: Zgradba sistema Julius. ..... 18
Slika 8: Uporabniški vmesnik programa Live. ..... 24
Slika 9: Dodani parametri, potrebni za testiranje. ..... 25
Slika 10: Prikaz rezultatov. ..... 26
Slika 11: Slovnica. ..... 28
Slika 12: Del slovarja besed. ..... 29
Slika 13: Nastavitev poti slovarja. ..... 30
Slika 14: Nastavitev poti slovnice. ..... 30
Slika 15: Uporabniški vmesnik aplikacije. ..... 31
Slika 16: Opozorilno okno. ..... 32
Slika 17: Pregled ukazov. ..... 32
Slika 18: Del kode za zagon programov. ..... 33
Slika 19: Del kode za izvršitev ukazov. ..... 34

# Kazalo tabel

Tabela 1: Primerjava značilnosti med Sphinx-4 in Julius..... 23

# Seznam uporabljenih kratic

| | |
|---|---|
| API | Aplikacijski programski vmesnik (angl. *Application Programming Interface*) |
| ARPA | Agencija za raziskavo projektov (angl. *Advanced Research Projects Agency*) |
| BNF | Oblika za zapis sintakse jezikov, ki se uporabljajo v računalništvu (angl. *Backus-Naur Form*) |
| CMU | Univerza Carnegie Mellon (angl. *Carnegie Mellon University*) |
| CPU | Centralna procesna enota (angl. *Central Processing Unit*) |
| DFA | Naprava za razpoznavanje jezikov (angl. *Deterministic Finite Automaton*) |
| HMM | Prikriti Markov model (angl. *Hidden Markov Model*) |
| HTK | Orodje za razpoznavanje govora (angl. *Hidden Markov Model Toolkit-Speech Recognition Toolkit*) |
| JSAPI | Java aplikacijski programski vmesnik za razpoznavanje govora (angl. *Java Speech Application Programming Interface*) |
| JSGF | Java aplikacijski programski vmesnik za razpoznavanje govora - format slovnice (angl. *Java Speech Application Programming Interface Grammar Format*) |
| LVCSR | Sistemi z velikimi slovarji, ki omogočajo razpoznavo tekočega govora (angl. *Large-Vocabulary Continuous Speech Recognition*) |
| PCM | Pulzno kodna modulacija (angl. *Pulse Code Modulation*) |
| SWT | Grafično orodje, ki se uporablja z Java platformo (angl. *Standard Widget Toolkit*) |
| WER | Stopnja napak besed (angl. *Word Error Rate*) |
| XHTML | Razširljiv jezik za označevanje nadbesedila (angl. *Extensible HyperText Markup Language*) |
| XML | Razširljiv označevalni jezik (angl. *Extensible Markup Language*) |

# Povzetek

Z razvojem sistemov za razpoznavanje človeškega govora se že vrsto let ukvarjajo tako ljudje iz akademskega kot strokovnega področja. Dva izmed sistemov za razpoznavanje govora sta tudi Sphinx-4 in Julius, ki ju v diplomskem delu med seboj primerjamo. S pomočjo sistema Sphinx-4 smo izdelali aplikacijo za upravljanje računalnika, pri katerem govorec v mikrofon pošilja zvočne signale, in s testnim orodjem izmerili njegovo učinkovitost na podlagi dobljene stopnje napak.

Na začetku diplomske naloge se seznanimo s človeškim govorom, ki predstavlja proces komunikacije. Sledi obširnejša predstavitev splošnega sistema za razpoznavanje govora, kjer se seznanimo s koncepti, ki nam v nadaljevanju omogočajo razumevanje sistemov Sphinx-4 in Julius. Namen poglavja je predstavitev delovanja sistema za razpoznavanje govora, prikaz njegovih prednosti in slabosti ter kaj ga omejuje oz. s katerimi problemi se spopada pri razpoznavi naravnega človeškega govora. Nato se seznanimo s sistemoma Sphinx-4 in Julius. Pri sistemu Sphinx-4 si ogledamo arhitekturo, ki nam pri praktičnem delu služi za lažje razumevanje delovanja sistema, nekoliko bolj grobo pa to predstavimo v sistemu Julius. Sledi poglavje, kjer primerjamo med seboj prej omenjena sistema. Tu prikažemo nekatere prednosti, slabosti, razlike in podobnosti med njima ter s testiranjem predstavimo rezultate učinkovitosti sistema za razpoznavo govora Sphinx-4. V zadnjem delu diplomske naloge predstavimo delovanje aplikacije in prikažemo izgled uporabniškega vmesnika.

**Ključne besede:**
človek, stroj, razpoznavanje govora, Sphinx-4, Julius, upravljanje računalnika

# Abstract

With the development of systems for recognizing human speech has been engaged people from academic and professional field for many years. Two of the systems for speech recognition as also Sphinx-4 and Julius, that in diploma work we compare each other. With the help of Sphinx-4, we created an application for the management of the computer in which the speaker sends sound signals to the microphone and with test tool we measured its efficiency in terms of the word error rate.

At the beginning of the thesis is acquainted with human speech, which is the process of communication. Followed by extensive presentation of the general speech recognition system, where we introduced with the concepts, that in continue enable us to understand the Sphinx-4 and Julius system. The purpose of this chapter is presentation operation of the speech recognition system, show it's strengths and weaknesses and what it restricts it or which problems it facing with recognizable natural human speech. Then acquainted with Sphinx-4 and Julius systems. At Sphinx-4 system we look the architecture, of which in the practical work serves for easier understanding functioning of the system, a little rougher, this we presented in the Julius system. Followed by a section, which compare each other the aforementioned system. Here we shows some strengths, weaknesses, differences and similarities between them and with testing we present the results of performance of the speech recognition Sphinx-4 system. The last part of the thesis we present the operation of applications and we show the user interface looks.

**Key words:**
human, machine, speech recognition, Sphinx-4, Julius, computer management

# 1 Uvod

Sistemi, ki se ukvarjajo z razpoznavanjem govora so se pojavili že v prejšnjem stoletju. Zgodnji sistemi so bili v primerjavi z današnjimi sistemi precej dragi in so za delovanje zahtevali precej zmogljive računalnike. Omenjeni sistemi so uporabljali diskretni govor. To pomeni, da je bil sistem implementiran tako, da je govorcu omogočal oz. je od govorca zahteval, da izgovori največ eno besedno hkrati, kjer so besede morale biti ločene s kratkimi premori. V zadnjih nekaj letih pa je področje razpoznavanja govora precej napredovalo. Sistemi razviti v zadnjih nekaj letih omogočajo razpoznavo tekočega govora in so tako bolj prilagojeni govorcu, saj mu omogočajo, da govori na bolj naraven način.

Človek pri govorni komunikaciji brez večjih težav razloči ženski glas od moškega. Ne glede na to ali je okolje v katerem se sporazumeva hrupno oz. šumno, razume govor neodvisno od govorca. Pri sistemih za avtomatsko razpoznavanje govora pa to povzroča precejšnje težave. Sistem, ki bi bil implementiran tako, da bi razpoznal govor brez omejitev (kot so npr. okolje, prostor, število možnih besed) oz. bi razpoznal govor, ne da bi ga pri tem motil šum, ki nastane v okolju, bo razvijalcem verjetno predstavljal problem še precej časa.

Razpoznavanje govora je pravzaprav zelo zapleten proces. Izgovorjave oz. vokalizacije govorcev se razlikujejo glede na naglas, izgovorjavo, hrapavost, glasnost in hitrost. Pri izgovorjavi se govor popači zaradi hrupa in odmeva, ki ga ustvari okolica in le-ta povzroči dodatno težavnost in problem pri razpoznavi govora.

Namen diplomske naloge je raziskati področje razpoznavanja govora in ugotoviti katere prednosti prinaša splošni sistem za razpoznavo govora ter s katerimi problemi se spopada pri razpoznavi naravnega človeškega govora. Cilj diplomske naloge je raziskati učinkovitost sistema za razpoznavo govora Sphinx-4, ga primerjati z drugim sistemom za razpoznavo govora, to je Julius, poiskati podobnosti in razlike med njima in izdelati aplikacijo pri kateri govorec govorno upravlja računalnik. S pomočjo testnega orodja, ki je priložen sistemu Sphix-4 bomo testirali njegovo delovanje ter poskušali dobiti WER (angl. *Word Error Rate*), ki predstavlja stopnjo učinkovitosti sistema.

Ker predstavlja govor pomembno vlogo pri sistemih za razpoznavanje govora, se mu bomo najprej na kratko posvetili in razjasnili kako sploh človek proizvaja govor ter kaj govor v

resnici je, saj nam bo to služilo za lažjo in boljšo nadaljnjo predstavitev in razlago pri sistemih za razpoznavanje govora.

# 2 Govor

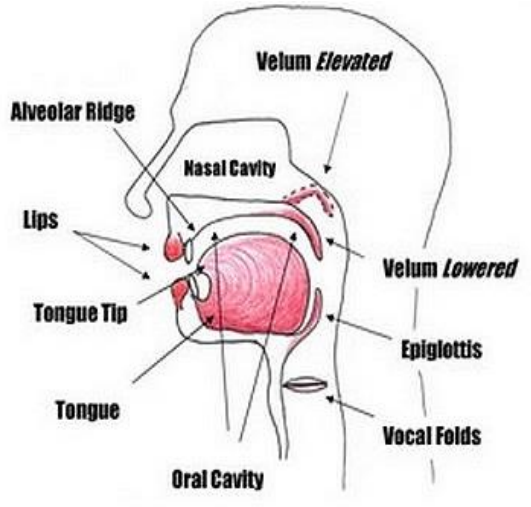
## 2.1 Predstavitev govora

Govor je eden izmed najbolj zanimivih področji obdelave signalov. Vokalizirana oblika človeške komunikacije oz. govorni signal predstavlja govor. Govor je človeku vrojen, zato tudi najbolj primeren način komuniciranja. Je optimalna zvočna komunikacija, oblikovana z ritmom besed in stavkov [1]. Večina se sploh ne zaveda kompleksnosti in težavnosti procesa govorne komunikacije. Več desetletne raziskave govornega signala so pokazale, da je govor dinamičen proces. Če želimo sposobnost govorne komunikacije prenesti tudi na stroj, moramo posamezne stopnje procesa komunikacije ustrezno razdeliti na področja govorne tehnologije. Te razdelimo v tri skupine:

- avtomatsko razpoznavanje govora,
- pretvorba besedila v govor in
- procesiranje shranjenega govornega signala (sinteza govora s pomočjo že posnetega govornega signala).

Avtomatsko razpoznavanje govora omogoča tekstovni zapis. Področje na katero se nanaša avtomatsko razpoznavanje govora, je področje, ki se ukvarja s komunikacijo govorca in stroja in s tem skuša doseči, da bi komunikacija potekala na povsem naraven način, torej govorimo o razpoznavanju človeškega govora.

## 2.2 Tvorjenje govora




Slika 1: Organi, ki sodelujejo pri tvorbi oz. obliki govora.

Slika 1 nam prikazuje organe, ki tvorijo strukturo govora. Govor predstavlja sistem glasov in glasovnih kombinacij, ki jih posameznik proizvaja s svojimi govornimi organi [1]. Pri tvorbi človeškega glasu sodelujejo organi, ki jih imenujemo govorila: trebušna prepona, rebra in mišice prsnega koša, oba režnja pljuč, sapnici s sapnikom, grlo, stene žrelne, ustne in nosne votline, jezik, zobje in ustnice [5].
Govor se tvori na sledeči način: Zračni tok iz pljuč povzroči tresenje glasilk, ki ustvarijo akustično nihanje. Le-to se v preostalih delih vokalnega trakta oblikuje in ojača ter izhaja skozi ustno in nosno odprtino kot govor.

Glede na strukturo govora se pri razpoznavanju le-tega uporabljajo trije modeli:

- Akustični model: vsebuje statistično predstavitev različnih zvokov, ki tvorijo vsako besedo v jezikovnem modelu. Vsak ločen zvok ustreza fonemu.

- Fonetični slovar: vsebuje preslikavo iz besed v foneme.

- Jezikovni model: se uporablja za omejevanje besednega iskanja. Opredeljuje katera beseda lahko spremlja prejšnjo že znano besedo in pomaga, da precej omeji proces primerjanja oz. ujemanja z odstranjevanjem besed, ki niso verjetne. Najpogosteje uporabljeni jezikovni modeli so N-gram jezikovni modeli.

# 3 Sistem za razpoznavanje govora

Sistem za razpoznavanje govora lahko opredelimo kot sistem, ki prepozna in pretvori izgovorjene besede oz. akustični signal, ki ga oseba z govorjenjem v mikrofon proizvede v tekstovni zapis (to so lahko besedne zveze, besede, fonemi, stavki, itd.). Sistem se lahko prav tako uporablja za upravljanje računalnika ali drugih naprav s pomočjo govornih ukazov, kar pomeni, da je uporaba npr. tipkovnice pri takšnih sistemih odveč, saj pretvorba govora v besedilo zamenja le-to. To je še posebej koristno za osebe s posebnimi potrebami, katerim uporaba tipkovnice predstavlja problem. Sistem za razpoznavo govora poznamo tudi pod imenom »avtomatski sistem za razpoznavo govora« ter tudi kot sistem »govor v besedilo«.

Osnovni sistemi za razpoznavanje govora imajo omejen besednjak (slovar) besed in besednih zvez, ki jih prepoznajo, če so izgovorjene zelo jasno (potrebna je pazljivost pri izgovorjavi besed, hitrosti in narečju, prav tako pa mora biti prostor v katerem človek proizvede akustični signal čim manj zvočno moten). Bolj razviti oz. kompleksni sistemi pa imajo možnost sprejeti naraven govor (tak sistem bi bil npr. razpoznavanje govora v šumnem okolju [6]).

Sistemi za razpoznavanje govora so lahko odvisni ali neodvisni od govorca. Sistemi, ki so odvisni od govorca, delujejo samo za eno osebo. Oseba mora najprej programsko opremo naučiti besed, tako da lahko potem računalnik analizira govor te osebe. To pogosto pomeni, da mora človek prebrati nekaj strani besedila v računalnik, preden lahko uporablja programsko opremo za razpoznavanje govora. Ti sistemi so bolj natančni in enostavni za razvoj, vendar pa niso prilagodljivi. Sistemi, ki so neodvisni od govorca pa so zasnovani tako, da razpoznajo katerikoli govor brez predhodnega treniranja programske opreme, kar pomeni da so lahko v interakciji z različnimi osebami, vendar pa se ti sistemi težje implementirajo in so precej dražji ter v splošnem manj natančni kot sistemi, ki so od govorca odvisni. Sistemi neodvisni od govorca zato omejujejo slovnico, ki jo uporabljajo. Z uporabo manjšega seznama besed je bolj verjetno, da sistem pravilno razpozna izgovorjene besede.

## 3.1 Delitev sistemov glede na način razpoznave govora

Glede na način razpoznave govora delimo sisteme za razpoznavanje govora v tri skupine. To so [2, 3]:

- sistemi za razpoznavanje izoliranih (posameznih) besed,
- sistemi za razpoznavanje tekočega govora in
- sistemi za razpoznavanje vezanega govora.

Pri razpoznavanju izoliranih besed sistem zahteva premor med vsako izgovorjeno besedo. Pri vsaki besedi mora biti začetek in konec natančno določljiv. Premori med besedami izgovorjenimi v stavku morajo biti dolgi vsak 200ms ali več. Gre za najpreprostejšo obliko razpoznavanja, saj je končno pozicijo besede lažje najti in izgovorjava besede tako rekoč ne vpliva na drugo izgovorjeno besedo.

Pri razpoznavanju tekočega govora so besede med seboj povezane in niso ločene s premori oz. prekinitvami. Sistemi, ki uporabljajo ta način razpoznave govora so kompleksni za uporabo, saj nimajo nobenih omejitev, katere pa moramo pri razpoznavanju izoliranih besed upoštevati (to so: premori med besedami, začetek in konec besede mora biti natančno določljiv, itd.). Težava s katero se soočajo ti sistemi je ta, da fonemi in besede vplivajo na sosednje foneme in besede in tako otežujejo razpoznavanje. Neprekinjen govor je težje obvladovati zaradi različnih vplivov. Težko je najti začetno in končno pozicijo besed. Sistem se mora med delovanjem tako prilagajati govorcu. Prilagajati se mora v primeru slabše izgovorjave govorca, stopnji oz. hitrosti govora (hitrejše govorjenje predstavlja precejšen problem) ter ločevanju besed.

Pri razpoznavanju vezanega govora je govor sestavljen iz izoliranih besed pri čemer je vhod tekoč govor. Ta način razpoznavanje govora je zelo podoben razpoznavanju izoliranih besed, ki pa dovoljuje, da ločene izjave tečejo skupaj (angl. *run-together*) oz. jih izgovorimo skupaj z minimalnimi premori med njimi.

## 3.2 Učinkovitost sistemov

Delovanje sistemov za razpoznavanje govora se običajno ocenjuje z vidika natančnosti in hitrosti. Natančnost je običajno ocenjena s stopnjo napak razpoznanih besed (WER), lahko pa jo ocenimo oz. izmerimo tudi s stopnjo napak posameznih besed (angl. *Simple Word Error Rate*) ali s stopnjo uspešnosti ukazov (angl. *Command Success Rate)*, medtem ko se hitrost avtomatskih sistemov za razpoznavanje govora meri s pomočjo realnega faktorja časa (angl. *Real Time Factor*).

Stopnjo napak besed izračunamo kot:

$$WER = \frac{Z + I + V}{N} \quad \text{ali} \quad WER = \frac{Z + I + V}{Z + I + P} \tag{3.1}$$

kjer posamezni simboli pomenijo:

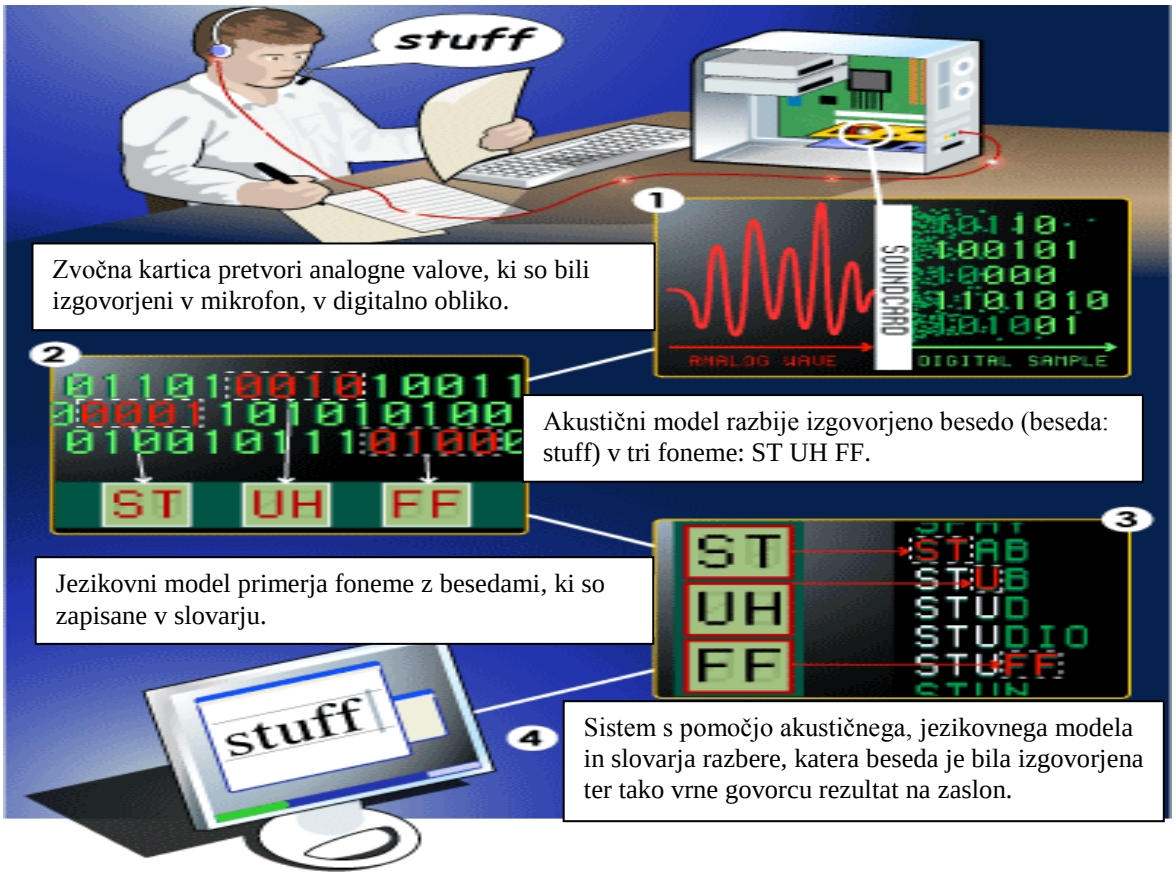
Z - število zamenjav oz. število zamenjanih besed,
I - število izbrisanih besed,
V - število vstavljenih besed,
P - število pravilnih besed,
N - število besed v referenci, ki ga izračunamo kot Z+I+P.

## 3.3 Pregled orodji za razpoznavo govora po priljubljenosti

- Dragon NaturallySpeaking,
- Microsoft Speech,
- IBM ViaVoice,
- MacSpeech,
- Philips,
- SpeechWorks,
- Tellme Networks,
- Julius,
- CMUSphinx in
- CSLU Toolkit.

Seznam zajema tako odprtokodna kot tudi komercialna orodja. Kot nam prikazuje seznam sta se tudi sistema CMUSphinx in Julius uvrstila med prvih deset najbolj priljubljenih sistemov za razpoznavanje govora.

## 3.4 Delovanje sistema za razpoznavo govora




Slika 2: Delovanje sistema za razpoznavo govora.

Razpoznavanje govora v osnovi deluje (Slika 2) kot cevovod, ki zvočne signale (govor govorca) pretvori v PCM (angl. *Pulse Code Modulation*) digitalni zvok. Slednji se nato pretvori v besedo ali besedni niz.. Elementi cevovoda so:

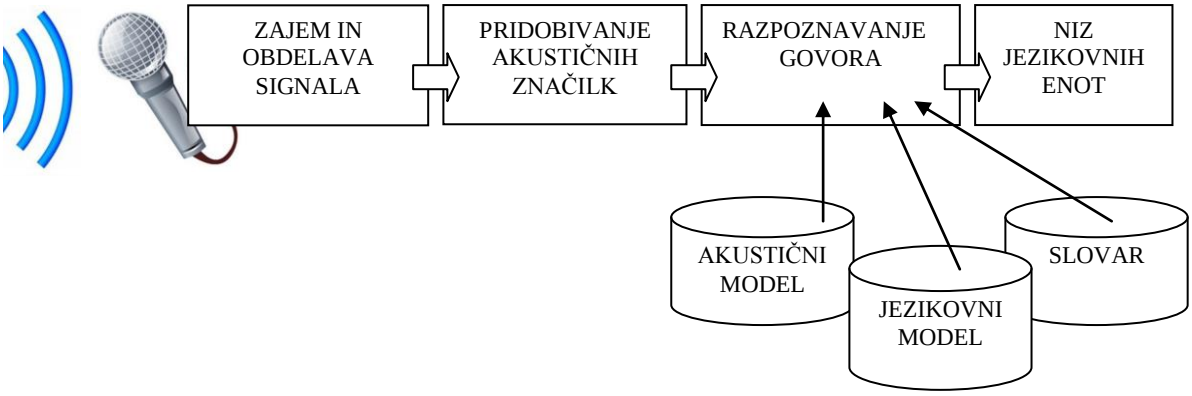
- modul za preoblikovanje PCM digitalnega zapisa zvoka v boljšo zvočno predstavitev,
- modul, ki z uporabo slovnice omogoča, da sistem ve katere glasove lahko pričakuje,
- modul za razbiranje fonemov, ki so bili izgovorjeni in
- modul za pretvorbo fonemov v besede.

Digitalni zvok je tok amplitud, ki je vzorčen približno 16.000-krat na sekundo. Za lažjo prepoznavo vzorcev se PCM digitalni zvok preoblikuje v frekvenčni prostor. Transformacija se opravi s pomočjo hitre Fourierjeve transformacije.

Digitalni zvok ne vsebuje le govorne podatke (višina, glasnost, tihi predeli) ampak tudi hrup iz okolja - šum. Šumi lahko motijo proces razpoznavanja govora, zato se mora sistem za razpoznavanje govora prilagodili okolju v katerem je zvok proizveden. Njegova prva naloga je, da pretvori digitalni zvok, ki prihaja iz zvočne kartice v obliko, ki je najprimernejša za nadaljnjo analizo. Ko je govor v ustrezni obliki, sistem začne iskati najboljša ujemanja. To počne z upoštevanjem besede in besednih zvez, ki jih pozna (slovnica), skupaj s svojim znanjem o okolju, v katerem deluje. Poznavanje okolja je na voljo v obliki akustičnega modela. Ko določi najverjetneje ujemanje besede, ki jo je govorec izgovoril, z besedo zapisano v slovarju, vrne razpoznano besedo kot besedni niz. Velikost slovarja vpliva na kompleksnost sistema in na obseg možnih besed, ki jih lahko sistem razpozna. Aplikacije, ki ne potrebujejo bogatega nabora besed, uporabljajo majhne slovarje, druge pa zahtevajo zelo velike slovarje. Slovarje glede na velikost delimo na:

- majhen slovar - več deset besed,
- srednje velik slovar - stotine besed,
- velik slovar - tisoče besed in
- zelo velik slovar - več deset tisoč besed.

## 3.5 Zgradba splošnega sistema za razpoznavo govora




Slika 3: Zgradba splošnega sistema.

Slika 3 prikazuje zgradbo splošnega sistema za razpoznavo govora. Potek razpoznave govora se začne, ko govorec pošlje akustični signal v mikrofon. Sledi parametrizacija [6], ki obsega zajem in obdelavo signala ter pridobivanje in izračun akustičnih značilk. Tu signal s pomočjo kratko časovne Fourierjeve transformacije pretvorimo v močnostni spekter. Iz vzorcev vhodnega govornega signala se v postopku frekvenčne ali časovne analize najprej oblikuje

začetna predstavitev signala. Sledi razpoznavanje govora. Naloga razpoznavalnika govora je poiskati najbolj verjetni niz besed za zajeti vhodni govor [7]. Iskanje izvedemo s pomočjo iskalnih algoritmov, ki vhodni govorni signal predstavljen z značilkami, modelirajo s pomočjo informacij iz akustičnega in jezikovnega modela. Pri iskanju najbolj verjetnega niza besed ni moč pregledati celotnega iskalnega prostora, ampak ga z različnimi hevurističnimi metodami omejujemo. Razlikujemo statično omejevanje (npr. drevesna predstavitev slovarja) in dinamično omejevanje iskalnega prostora (npr. snopovno omejevanje, pogled-naprej v jezikovni model ipd.).

## 3.6 Prednosti in slabosti sistemov za razpoznavo govora

Noben sistem za razpoznavanje govora ni 100-odstotno uspešen zaradi različnih dejavnikov, ki vplivajo nanj. Ti sistemi tako prinašajo prednosti kot slabosti.

### 3.6.1 Prednosti

- Pomoč osebam s posebnimi potrebami: ena izmed najbolj opaznih prednosti razpoznavanja govora je uporaba s strani oseb (invalidi), ki imajo težave z rokami (poškodbe), saj jim ti sistemi omogočajo delo z računalniki ali drugimi napravami, ne da bi pri tem uporabljali vhodne naprave (kot so: tipkovnica in miška).

- Višja storilnost: npr. pri pisanju dokumenta. Oseba, ki uporablja program za razpoznavanje govora, lahko dokument napiše precej hitreje, saj sistemi za razpoznavanje govora zapisujejo besede v dokument navidezno sočasno z izgovorjavo govora. Takšni sistemi lahko tudi izboljšujejo sposobnost posameznika, da opravlja več nalog hkrati.

- Govor je naraven način za interakcijo: uporaba tipkovnice je pri uporabi teh sistemov odveč.

- Ni potrebnega predhodnega usposabljanja: npr. pri uporabi tipkovnice se mora oseba najprej naučiti upravljati le-to, pri sistemih za razpoznavanje govora je predhodno usposabljanje nepotrebno, ker gre za način komunikacije, ki je človeku naraven.

### 3.6.2 Slabosti

- Nizko razmerje med signalom in šumom: sistem mora slišati izgovorjeno besedo razločno in jasno brez dodatnega hrupa, ki bi pri tem oviral signal, kar pomeni, da mora oseba delati v čim manj hrupnem prostoru. Razpoznavanje govora deluje najbolje, če je mikrofon v bližini. Bolj oddaljeni mikrofoni (npr. namizni ali stenski) povečujejo možnost za napake. Do nizkega razmerja signal/šum lahko pride tudi, če je zvočna kartica, ki zagotavlja vhod za mikrofon nižje kakovosti. Pogosto takšne zvočne kartice niso dovolj zaščitene pred električnimi signali, ki jih proizvajajo druge naprave. To lahko povzroča dodatno brnenje ali šum v signalu.

- Prekrivanje govora: sistemi imajo težave pri ločevanju simultanega govora več uporabnikov.

- Intenzivna uporaba procesorske moči: vodenje statističnih modelov npr. (angl. *Hidden Markov Model-HMM*), potrebnih pri razpoznavanju govora, zelo obremenjuje procesor. Težave poleg zapletenih besed ali besednih zvez, ki upočasnjujejo odzivni čas, predstavljajo tudi slovarji, ki hranijo besede, saj zavzamejo veliko prostora na disku.

- Enakoglasnice: enakoglasnica je beseda, ki se v primerjavi z drugo besedo enako izgovarja vendar ima drugačen pomen oz. se napiše drugače zato sistemi za razpoznavanje govora težko razlikujejo le-te.

## 3.7 Pregled strojne opreme za potrebe razpoznavanja govora

### 3.7.1 Zvočna kartica

Govor zahteva razmeroma nizko pasovno širino, zato so že 16-bitne zvočne kartice srednje visoke kakovosti primerne za uporabo. Zvočne kartice s čisto (angl. *cleanest*) A/D (iz analognega v digitalno) pretvorbo so priporočljive, vendar pa je čistost digitalnega vzorca pogosto odvisna predvsem od kakovosti mikrofona, še bolj pa od hrupa v okolju. Električni hrup oz. šum iz monitorjev, PCI rež, trdih diskov, itd., ne vpliva toliko na kakovost zvoka v primerjavi z računalniškimi ventilatorji, glasnega (težkega) dihanja v mikrofon ali hrupnega okolja.

### 3.7.2 Mikrofon

Mikrofon je akustično električni pretvornik, ki pretvarja zvok v električni signal. Pri uporabi avtomatskega razpoznavalnika govora je kakovost mikrofona ključnega pomena. Namizni mikrofoni niso ravno najboljša izbira, saj ne dajejo pričakovanih rezultatov. Ti običajno zajamejo več hrupa iz okolja oz. prostora, kjer proizvajamo akustični signal, kar sistemom za razpoznavanje govora povzroča precejšnje težave. Ročni mikrofoni prav tako niso najboljša izbira. Čeprav omejujejo količino hrupa v okolju, se ti najpogosteje uporabljajo pri aplikacijah, ki zahtevajo spreminjajoče se govornike. Najpogostejša ter tudi najboljša izbira so običajno slušalke kombinirane z mikrofonom (angl. *headset*). Te sprejmejo v primerjavi z drugimi mikrofoni zelo malo hrupa iz ozadja, saj je mikrofon ves čas blizu ust.

### 3.7.3 Procesor in pomnilnik

Zaradi velike količine digitalnega filtriranja in obdelave signalov, ki poteka v sistemih za razpoznavanje govora, so sistemi lahko močno odvisni od hitrosti obdelave. Logično je, da pri hitrejšem CPU (angl. *Central Process Unit*) in pri večji količini pomnilnika dosegamo boljše rezultate. Sistem za razpoznavanje govora je možno narediti že z enojedrnim procesorjem pri 100 MHz in 16 M pomnilnika, vendar ko uporabljamo velike slovarje, kompleksne sisteme razpoznavanja ali veliko število vzorcev, potrebujemo hitrejšo obdelavo, zato moramo imeti na voljo najmanj 1 GHz procesor in 1 GB pomnilnika.

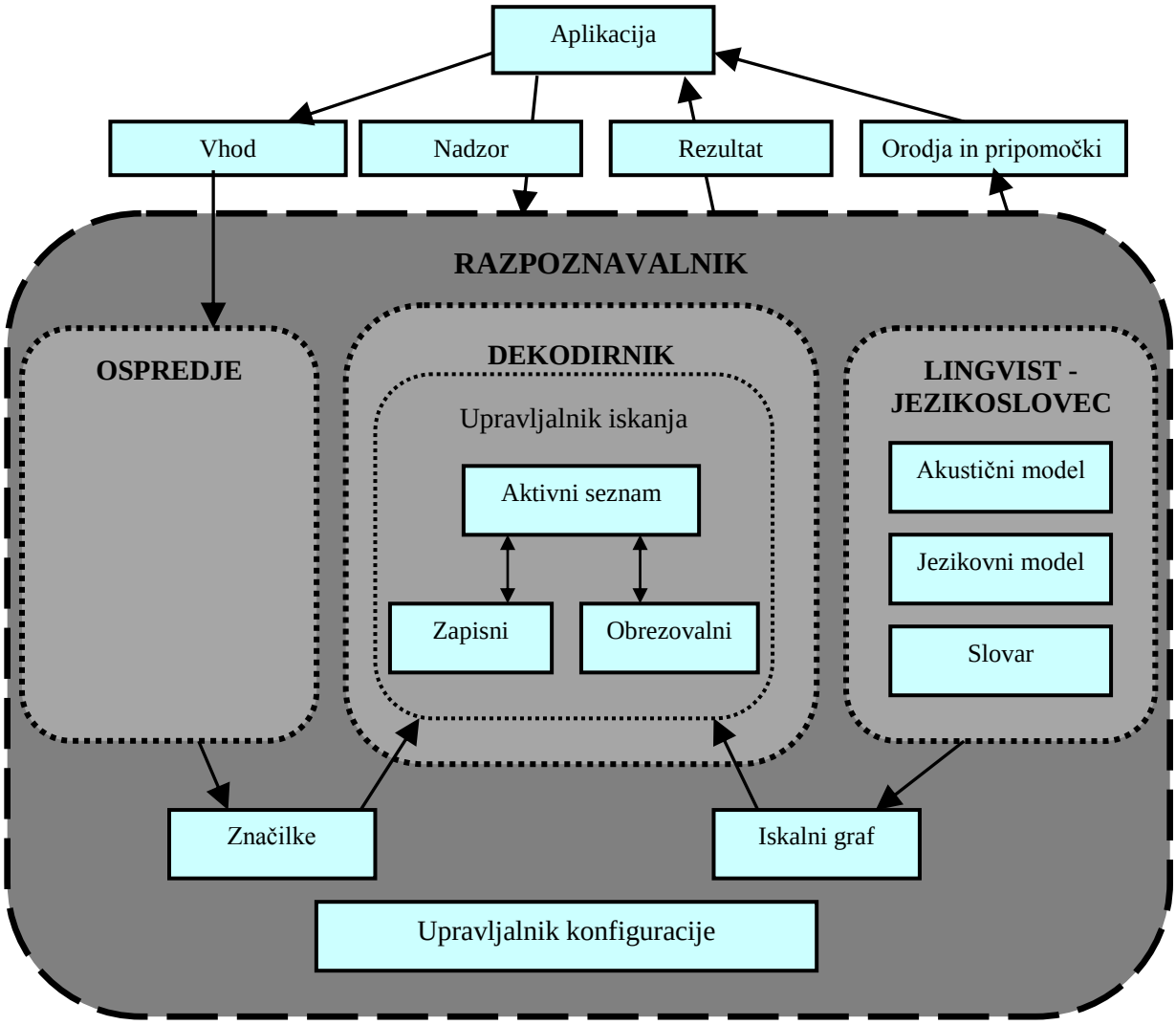
# 4 Sistema Sphinx-4 in Julius

## 4.1 Sistem Sphinx-4

Sistem Sphinx-4 je sistem za razpoznavanje govora, ki temelji na uporabi pristopa prikritega Markovega modela (angl. *Hidden Markov Model*). Odprtokodni sistem Sphinx-4 je bil razvit na univerzi Carnegie Mellon in je zadnji razvit sistem izmed vseh sistemov, ki spadajo med CMUSphinx sisteme. Zasnova Sphinx-4 temelji na vzorcih, ki so nastali z razvojem preteklih sistemov [8]. Sphinx-4 sistem se od prejšnjih CMUSphinx sistemov razlikuje v smislu modularnosti, prilagodljivosti in algoritmičnih vidikov. V primerjavi s prejšnjimi CMUSphinx sistemi uporablja tudi novejše strategije iskanja.

Pri sistemu Sphinx-4 je pomembno, da vemo, kakšna je njegova arhitektura in delovanje. Ker se bomo v praktičnem delu tudi sami dotaknili tega sistema, si bomo bolj podrobno ogledali njegovo zgradbo. Sistem Sphinx-4 sestavljajo tri osnovni moduli (angl. *primary moduls*), ki se jim bomo v nadaljevanju, poleg ostalih komponent, bolj podrobno posvetili.

### 4.1.1 Arhitektura sistema Sphinx-4

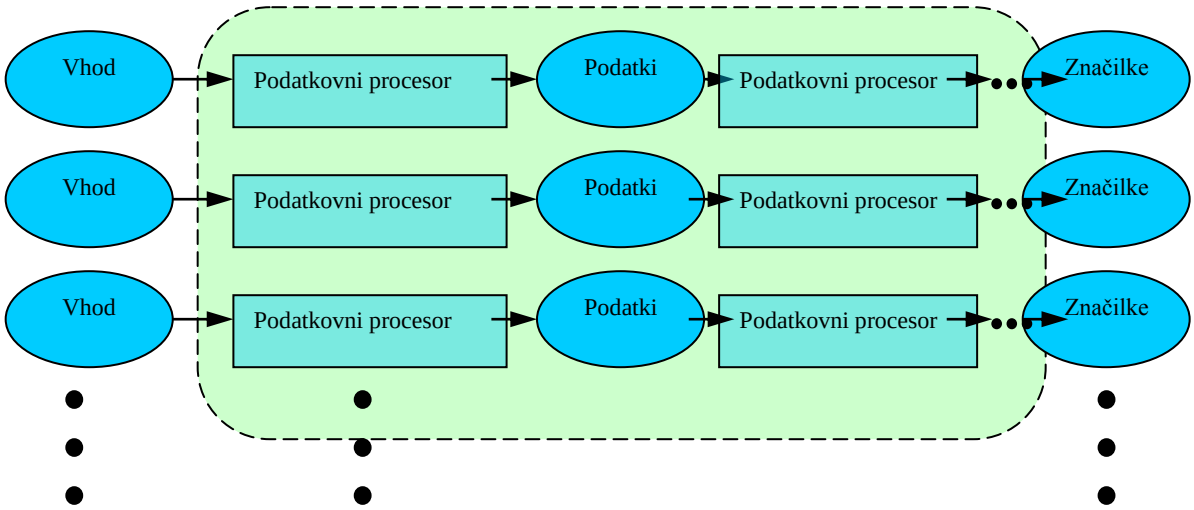



Slika 4: Zgradba sistema Sphinx-4.

Zgornja shema (Slika 4) prikazuje zgradbo sistema Sphinx-4. Posamezni element, prikazan na sliki 4, lahko enostavno zamenjamo, kar nam omogoča različne implementacije modulov, pri čemer nam drugih delov sistema ni potrebno spreminjati [8]. Arhitektura, na kateri je zasnovan sistem Sphinx-4, temelji na treh modulih: ospredju (angl. *fornt-end*), dekodirniku (angl. *decoder*) in lingvistu - jezikoslovcu (angl. *linguist*). Omenjeni moduli nato zgradijo svoje podkomponente oz. podmodule.

#### 4.1.1.1 Ospredje

Ospredje (Slika 5) opravlja digitalno obdelavo signalov vhodnih podatkov (govorni signal, ki ga zajame mikrofon se običajno vzorči s 16 kHz). Sprejme vhodni zvočni signal in ga parameterizira v zaporedje izhodnih značilk (angl. *features*).



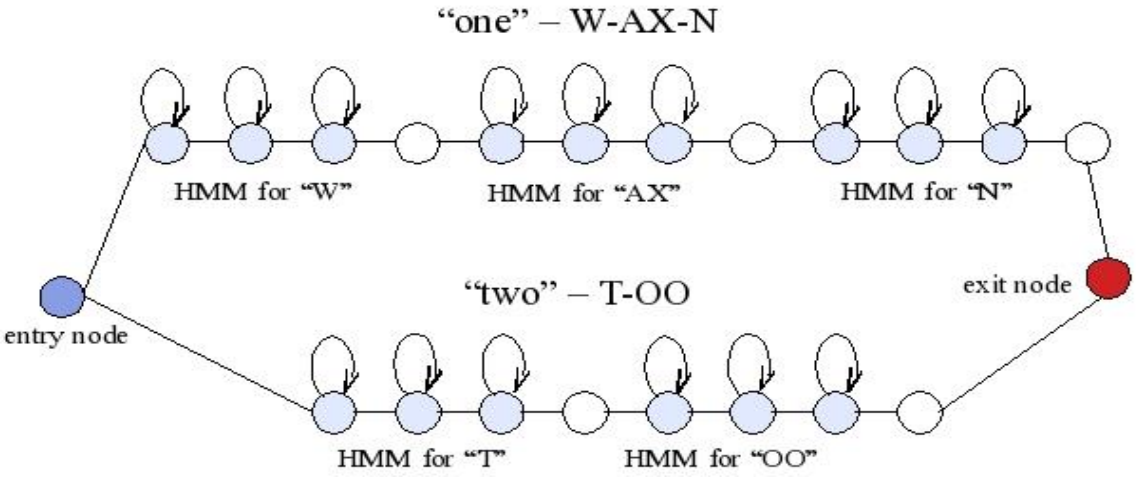

Slika 5: Sphinx-4 ospredje - zgradba.

Ospredje sistema Sphinx-4 povezuje in vsebuje eno ali več vzporednih verig podatkovnih procesov (angl. data processor), ki komunicirajo med seboj. Naloga podatkovnih procesov je obdelovanje vhodnih signalov. Podpora več vzporednim verigam omogoča hkratno oz. sočasno računanje različnih tipov parametrov istih ali različnih vhodnih signalov. Vsak podatkovni proces v modulu zagotavlja oz. vsebuje vhod in izhod, ki se lahko povežeta na drugi podatkovni proces. To omogoča gradnjo poljubno dolgih zaporednih verig. Generične podatkovne objekte predstavljajo vhodi in izhodi vsakega podatkovnega procesa, ki enkapsulirajo obdelane podatke kot tudi označevalce, ki npr. označujejo konec govora. Zadnji podatkovni proces v verigi je odgovoren za izdelavo podatkovnih objektov, sestavljenih iz parameteriziranih signalov, ki jih imenujemo tudi značilke, katere uporablja dekodirnik. Komunikacija med objekti podatkovnih procesov temelji na postopku, ki ga imenujemo povleci zasnovo (ang. pull design). Ta postopek omogoča podatkovnim procesorjem medpomnjenje.

Operacije, ki jih podatkovni procesi nad vhodnimi signali izvajajo, so: diskretna kosinusna in fourierova transformacija, funkcije v povezavi z okni (Hammingova in Hannova okna), mel-frekvenčno filtriranje, linearno napovedano kodiranje in druge.

#### 4.1.1.2 Lingvist

Lingvist generira iskalni graf, ki ga med iskanjem uporablja dekodirnik. Lingvist prevede vse, ne glede na tip, jezikovne modele skupaj z informacijo izgovorjave dobljene iz slovarja ter informacijo o strukturi enega ali več akustičnih modelov v iskalni graf. Graf je usmerjen in posamezno vozlišče (iskalno stanje) je lahko emisijsko ali neemisijsko. V emisijsko stanje lahko pridemo na podlagi akustičnih značilk, medtem ko v neemisijska stanja na ta način ni mogoče priti. Neemisiska stanja uporabljamo za predstavitev višjih jezikovnih konstruktov, kot npr. besede ali fonemi. Verjetnost prehoda iz enega vozlišča (stanja) v drugo predstavljajo povezave med njimi.




Slika 6: Iskalni graf.

Iskalni graf (Slika 6), ki ga lingvist po določenih kriterijih ustvari (npr. slovnice), uporablja znanje iz akustičnega modela, slovarja in jezikovnega modela, ki so sestavni del lingvista. Opis akustičnega modela, jezikovnega modela in slovarja smo povzeli po članku [7].

Naloga akustičnega modela je razpoznavanje fonemov v akustičnem signalu oz. čim bolj učinkovito predstaviti akustično-fonetične parametre glasu. Tu se predvsem uporablja prikriti Markov model (HMM). Za učenje akustičnih modelov HMM potrebujemo govorne baze, ki vsebujejo posnetke govora in natančen (fonemski) prepis tega, kar je bilo izgovorjeno. Ker se izgovorjava istega fonema razlikuje glede na njegova sosednja fonema, se danes za akustično modeliranje najpogosteje uporabljajo trifonski akustični modeli, ki upoštevajo tudi levi (predhodni) in desni (sledeči) fonem oz. kontekst fonema. Podobna stanja akustičnih modelov med seboj združujemo, da zmanjšamo kompleksnost sistema. Pri razpoznavanju tekočega govora so meje med besedami težko določljive, saj jih ne označujejo premori. Slovar določa

izgovorjave posameznih besed, ki jih najdemo v jezikovnem modelu. Izgovorjave »zlomijo« (angl. *break*) besede v zaporedja podbesednih enot, ki se hranijo (nahajajo) v akustičnem modelu. Tudi slovar, ki vsebuje nabor besed, ki jih razpoznavalnik mora razlikovati, močno raste, zato samo akustični model ni dovolj uspešen. Potreben je dodaten vir znanja o jeziku, ki mu pravimo jezikovni model. Strukturo jezikovnega modela lahko predstavimo z dvema kategorijama slovnic. V prvo kategorijo slovnic uvrščamo N-gram model (določa verjetnost naslednjih besed na podlagi predhodnih besed). V drugo kategorijo slovnic pa spada slovar, ki ga predstavimo z grafi (vozlišča predstavljajo posamezno besedo, povezave pa prehode med besedami - prehodi so opredeljeni glede prehodnih verjetnosti). Medtem ko akustični model računa verjetnosti na ravni fonemov oz. trifonov, jezikovni model računa verjetnosti na ravni besed. Za učenje jezikovnega modela ne potrebujemo več baze izgovorjav, ampak velike elektronske zbirke besedil, ki obsegajo več milijonov besed. Implementacija sistema Sphinx-4 vključuje različne formate jezikovnega modela:

- SimpleWordListGrammar,
- JSGFGrammar,
- LMGrammar,
- FSTGrammar,
- SimpleNGramModel in
- LargeTrigramModel.

#### 4.1.1.3 Dekodirnik

Osnovna naloga dekodirnika je uporaba značilk iz ospredja (modula) v povezavi z iskalnim grafom, ki ga tvori lingvist in s tem ustvariti oz. generirati rezultat. V pomoč so mu različni moduli, eden izmed njih je upravljalnik iskanja (angl. *search manager*), ki poenostavlja postopek dekodiranja. Upravljalnik iskanja s pomočjo vhodnih značilk ocenjuje poti skozi iskalni graf in jih z uporabo obrezovalca omejuje. Dekodirnik vsebuje tudi podmodule, to so:

- Aktivni seznam (angl. *active list*): vsebuje seznam parametrov, ki predstavljajo vsa stanja v iskalnem grafu, ki so aktivna v trenutnem okvirju značilk.

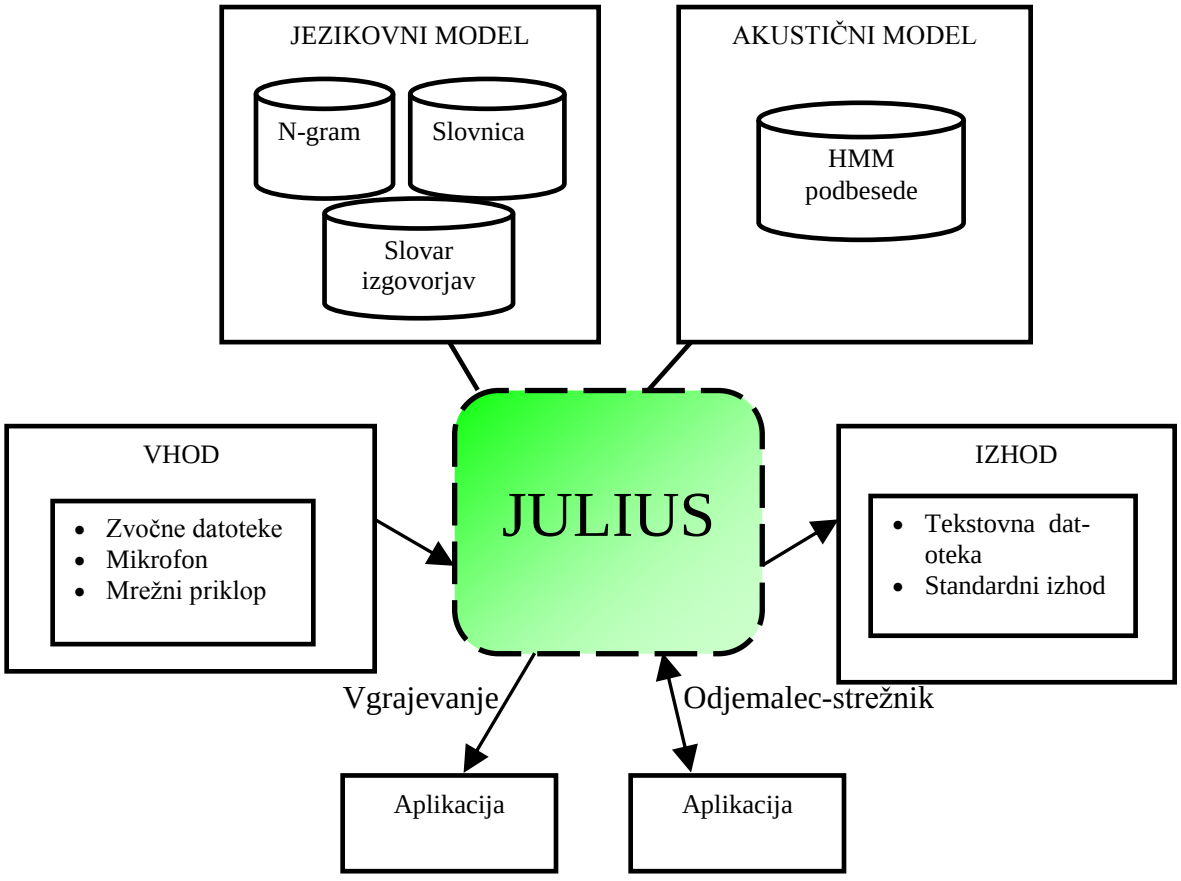
- Zapisni (angl. *scorer*): beleži oz. zapisuje trenutni okvir značilk.

- Obrezovalni (angl. *pruner*): omejuje aktivni seznam v skladu z določenimi strategijami.

## 4.2 Julius

Julius je japonski raziskovalni projekt, ki ga je razvil Interactive Speech Technology Consortium [10]. Je visoko zmogljiv, odprtokodni razpoznavalnik govora (dekodirnik) [4]. Njegove glavne lastnosti so večnamenskost, prilagodljivost in razširljivost. Julius je sposoben zlahka zgraditi sistem za razpoznavanje govora, ki združuje slovar, jezikovni in akustični model, ene ali več deset tisoč tekoče izgovorjenih besed. V realnem času opravlja zvezno razpoznavanje govora velikega števila besed zelo učinkovito, z relativno majhno porabo sistemskih resursov. Temelji na jezikovnem modelu N-gram in je kontekstno odvisen od prikritega Markovega modela. Glavne tehnike iskanja kot so drevesni slovarji, Gaussovo obrezovanje, Gaussov izbor, itd., so vključene v celoti. Poleg učinkovitosti iskanja je zgrajen tako, da je neodvisen od modela struktur. Razvit je za platformo Linux in Unix sisteme, deluje pa tudi v okolju Windows.

### 4.2.1 Arhitektura sistema Julius




Slika 7: Zgradba sistema Julius.

Iz zgornje slike (Slika 7) lahko vidimo, da sistem Julius, prav tako kot sistem Sphinx-4, za svoje delovanje potrebuje slovar, akustični in jezikovni model. Kot vhod sprejme tako zvočne (avdio) datoteke kot tudi akustični signal preko mikrofona. Prav tako podpira samodejno delitev vhoda z dolgimi premori, kjer se odkrivanje premorov izvaja na podlagi stopenj oz. nivojev in z ničelnimi navzkrižnimi pragi (angl. *zero cross thresholds*). Akustični model uporablja HMM za vsako podbsedeno enoto. Julius podpira jezikovni model za N-gram, slovnico in tekoče izgovorjene besede.

### 4.2.2 Značilnosti sistema Julius

Glavne značilnosti sistema so:

- Lahko izvede razpoznavanje s slovarjem, ki vsebuje več kot 65.000 besed.
- Delež razpoznavnosti za dvajset tisoč izgovorjenih besed je več kot 95%.
- Sistem uporablja 2-gram in 3-gram jezikovni model.
- Lahko uporablja monofone, trifone ali mešanico vezanih trifonov. Ob zagonu sistem samodejno zazna vrsto uporabljenega modela.
- Nizka poraba pomnilnika.
- Zaporedno dekodiranje, ki omejuje vnos (besede) s kratkimi premori.
- Hitro razpoznavanje izoliranih besed.
- Kot vhod lahko uporablja poleg govornih datotek tudi datoteke HTK (angl. *Hidden Markov Model Toolkit*) formata.
- V celoti podpira HTK HMM opredelitev datotek.

# 5 Primerjava in testiranje

## 5.1 Primerjava Sphinx-4 in Julius

Odprtokodna sistema Julius in Sphinx-4 sta zelo znana orodja za razpoznavo govora. Za delovanje so elementi kot so slovar, jezikovni in akustični model, pomembni dejavniki pri obeh sistemih. Arhitektura obeh sistemov je na prvi pogled podobna, vendar iz slike 4 in 7 vidimo kar nekaj razlik med njima. Razlike niso vidne samo v arhitekturi, temveč se kažejo tudi pri drugih opravilih in dejavnostih. V nadaljevanju si bomo ogledali lastnosti, prednosti in slabosti obeh sistemov ter ju med seboj primerjali [4, 8].

### 5.1.1 Sistemske zahteve

Julius je v celoti napisan v programskem jeziku C, sistem Sphinx-4 pa v programskem jeziku Java, kar nam pove oz. lahko sklepamo, da sistem Sphinx-4 nekoliko bolj zahteven oz. potraten. To se pri sistemu Sphinx-4 predvsem kaže pri računanju in pomnilniških zahtevah, saj mora za vsako besedo zgraditi svoj lastni HMM model.
Poraba pomnilnika pri sistemu Julius je precej nižja od sistema Sphinx-4. Zahteva manj kot 32 MB. Za razpoznavanje dvajset tisoč izgovorjenih besed s 3-gram jezikovnim modelom pa potrebujemo le 64 MB pomnilnika.

### 5.1.2 Treniranje / učenje

Pri obravnavi sistemov, ki imajo obsežen besednjak, je treniranje oz. učenje modelov zelo pomembno. Kot pri vsakem sistemu za razpoznavanje govora, ocenjevanje HMM temelji na količini podatkov, ki so na voljo za trening. Sistem Sphinx-4 uporablja za učenje akustičnih modelov orodje SphinxTrain [16], ki je že vključeno v sistem. SphinxTrain je odprtokodno orodje za učenje akustičnih modelov. Orodje je namenjeno družini sistemov CMUSphinx pri razpoznavanju tekočega govora. Pred treniranjem moramo najprej ustvariti datoteke s podatki, kot so: datoteko z značilkami, kontrolno datoteko, transkripcijsko datoteko, slovar besed, slovar mašil ter seznam fonemov.
Za treniranje v sistemu Julius opravi oz. je zanj zadolženo orodje HTK [15]. HTK je orodje za ustvarjanje oz. izgradnjo prikritih Markovih modelov. Uporablja se predvsem pri

razpoznavanju govora, čeprav je prisoten tudi v drugih aplikacijah, ki uporabljajo HMM modele, kot npr. pri sintezi govora.

### 5.1.3 Upravljalnik iskanja

Upravljalnik iskanja s pomočjo vhodnih značilk ocenjuje poti skozi iskalni graf in jih z uporabo obrezovalca omejuje.

Upravljalniki iskanja implementirani v sistemu Sphinx-4 so:

- SimpleBreadthFirstSearchManager,
- WordPruningBreadthFirstSearchManager,
- BushderbySearchManager in
- ParallelSearchManager.

Za upravljanje iskanja, sistem Julius uporablja hevristični iskalno mrežno drevo (angl. *tree-trellis heuristic search)* v kombinaciji z levo-desnim (iz leve proti desni) sinhronskim snopovnim iskanjem (angl. *synchronous beam search*), ter iskanje z desne proti levi v kombinaciji s skladovnim dekodirnim iskanjem (angl. *stack decoding search*).

### 5.1.4 Glavne tehnike iskanja

Glavne tehnike iskanja pri sistemu Sphinx-4 so Viterbi, Bushderby in vzporedno dekodiranje. Julius ima za razliko od Sphinx-4 bolj bogat nabor tehnik iskanja. Glavne iskalne tehnike, ki so vključene v sistem Julius so drevesni slovarji, Gaussovo obrezovanje, navzkrižni kontekst besed, snopovna iskalna ovojnica ter Gaussov izbor.

### 5.1.5 Primerjava ostalih značilnosti

| Sistem / Značilnosti | Sphinx-4 | Julius |
|---|---|---|
| **Treniranje in format akustičnega modela** | Sphinx | HTK |
| **Več-vzorčno razpoznavanje** | Ne podpira | Podpira |
| **Nastavitve/konfiguracija** | Monolitno | Modularno |
| **Odprtokodni sistem** | Da | Da |
| **Format jezikovnega modela** | ARPA (angl. *Advanced Research Projects Agency*) - (N-gram, LM slovnica, FST slovnica) in JSGF | ARPA-N-gram, DFA (angl. *Deterministic Finite Automaton*) slovnica, posamezne besede in uporabniško definirane funkcije |
| **Pristop razpoznavanja** | HMM | HMM |

Tabela 1: Primerjava značilnosti med Sphinx-4 in Julius.

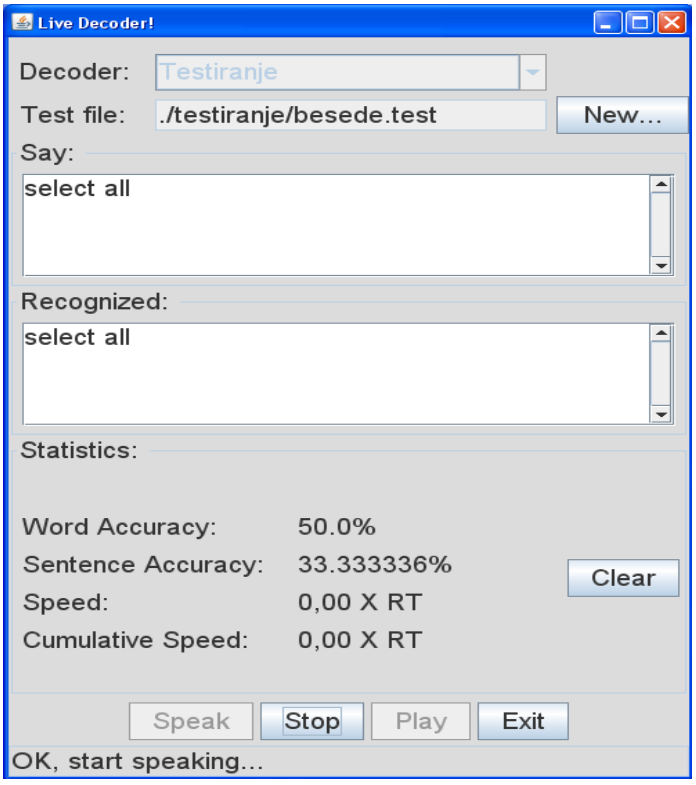
V tabeli 1 smo primerjali še ostale značilnosti med sistemom Sphinx-4 in Julius. Julius se za razliko od sistema Sphinx-4 osredotoča na obravnavo narekovanih opravil in ne na vodenje in nadzor. V primerjavi s sistemom Sphinx-4 ima zelo dobro zasnovan API (angl. *Application Programming Interface*) dekodirnik, prav tako ima tudi več možnosti za avdio (zvočni) vhod in je enostavnejši kot sistem Sphinx-4, ko primerjamo velikost sistema in model. Sistem Sphinx-4 za razliko od sistema Julius ne vsebuje le razpoznavanje govora, temveč vključuje tudi učenje sistema, medtem ko Julius opravlja samo dekodiranje (razpoznavanje). Prednost sistema Sphinx-4 pred orodjem Julius je ta, da nudi veliko več dokumentacije, katera razvijalcem pri uporabi le-tega služi za lažjo implementacijo lastnih modulov. Večina modulov predstavlja vmesnik [11]. Iskalni graf, upravljalnik iskanja, akustični model, slovar, jezikovni model, lingvist, aktivni seznam, obrezovalec in zapisovalec so Java vmesniki. Našteti vmesniki sestavljajo sistem Sphinx-4, ki pa jih razvijalci lahko spreminjajo oz. popravljajo ali novo zgradijo ter jih tako prilagodijo svoji aplikaciji.

## 5.2 Testiranje in rezultati

Za testiranje učinkovitosti in prikaz rezultatov sistema Sphinx-4 smo uporabili orodje oz. program Live, ki ga nudi sistem Sphinx-4. Program Live smo poganjali preko lupine operacijskega sistema (angl. *command prompt*). Pred samim zagonom programa, moramo imeti pravilno nameščen JSAPI (angl. *Java Speech Application Programming Interface*) [12] in Apache ant [9]. S pomočjo ant [11] ukaza lahko preko lupine operacijskega sistema zgradimo Sphinx-4 iz izvorne distribucije.

Program Live lahko zaženemo na tri načine:

- z ukazom »ant live«: s tem ukazom govorec sam določi začetek in konec govora,
- z ukazom »ant live-ep«: s tem ukazom govorec določi začetek govora, program pa konec zazna sam in
- z ukazom »ant live-free«: s tem ukazom program sam določi začetek in konec izgovorjene besede govorca. Program se v primerjavi s prvima dvema načinoma samodejno premakne na naslednjo izjavo, ki jo mora govorec izreči.




Slika 8: Uporabniški vmesnik programa Live.

Program Live (Slika 8) uporablja testno datoteko v kateri so zapisane besede ali zaporedje besed, ki jih govorcu prikaže na zaslon. Besede so zapisane po pravilih, ki so navedene v slovnici, ki jo uporablja posamezna testna datoteka. Program nam ponuja različne testne datoteke, ki jih lahko v »Decoder« izbirnem oknu izberemo, v primeru, da hočemo prikazati rezultate za posamezne besede, tekoče besede, črkovanje besed, itd. Lahko pa si pripravimo tudi svojo testno datoteko. V primeru, da hočemo prikazati rezultate učinkovitosti sistema na svoji pripravljeni testni datoteki, moramo popraviti oz. dodati nekaj parametrov (Slika 9) v datoteko »decoders.list« v kateri so navedeni vsi testi.

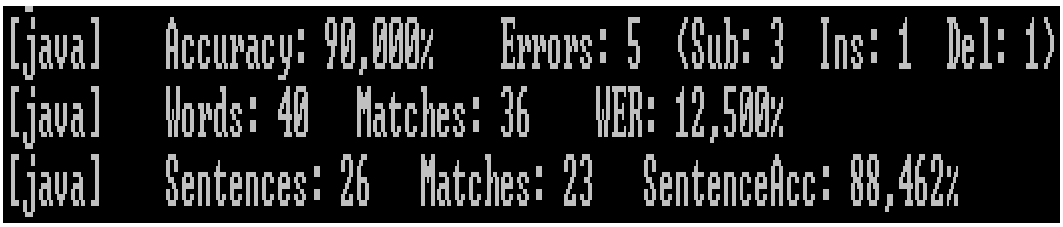
```
# Testiranje
testiranje.name= Testiranje
testiranje.configFile=./testiranje/nastavitve.config.xml
testiranje.testFile=./testiranje/besede.test
```

Slika 9: Dodani parametri, potrebni za testiranje.

Parametre, ki jih moramo določiti so:

- name: ime, ki bo vidno v »Decoder« izbirnem polju,
- configFile: je konfiguracijska datoteka, ki je enaka datoteki uporabljeni v aplikaciji. Spremembe se pojavijo le pri nastavitvi poti do slovnice in
- testFile: testna datoteka z ukazi/besedami.

Besede, ki jih govorec mora izreči program prikaže v polju z imenom »Say«, izgovorjeni ukazi govorca, pa so prikazani v polju z imenom »Recognized«. Učinkovitost sistema program računa tako, da besedo/e, ki jih/jo je govorec izrekel program primerja z besedo/ami zapisano/e v polju »Say«. V primeru, da se izgovorjena beseda govorca ujema z besedo, ki je zapisana v polju »Say« program zabeleži to besedo kot pravilno, v nasprotnem primeru pa to besedo razvrsti v eno izmed treh kategorij napak, kot so: zamenjana, izbrisana ali vstavljena.
Testna datoteka, ki smo jo pripravili, da bi dobili rezultate učinkovitosti sistema, je obsegala 40 besed. Besede v testni datoteki so sestavljale posamezne in povezane besede, ki so bile zapisane po pravilih, navedene v slovnici »grammar commands«.




Slika 10: Prikaz rezultatov.

Slika 10 prikazuje rezultat za 40 izgovorjenih besed. Vidimo, da se je sistem izkazal kot zelo učinkovit, saj je kar 90%, to je 35 besed, razpoznal pravilno. Pod »Errors« je program zabeležil napake. Program je zabeležil skupaj pet napak: tri zamenjave, eno vstavljanje in eno brisanje. Program Live nam izračuna tudi WER, ki je znašal 12,5% in ga izračunamo po formuli (3.1). Manjši kot je WER, tem bolj učinkovit je sistem.

# 6 Razvoj aplikacije za upravljanje računalnika z govornimi ukazi

S pomočjo sistema za razpoznavo govora Sphinx-4 smo izdelali aplikacijo, ki razpozna govor. CMUSphinx nam ponuja veliko dokumentacije in testnih primerov, ki so nam pomagali pri izgradnji le-te. Nekateri Sphinx-4 testi in predstavitve uporabljajo JSAPI, ki ponuja standardni vmesnik in funkcionalnost tako za razpoznavanje govora kot tudi za sintezo govora. JSAPI deluje kot srednji sloj med osnovnim razpoznavalnikom govora in dejansko aplikacijo. Je del Java Media, ki vsebuje Java Sound, Java, Java Speech in druge programske vmesnike. Najpomembnejše opravilo za razvijalce govornih aplikacij z uporabo Java Speech API je zasnova slovnice. V Java Speech API je razpoznavalnik govora ustvarjen v paketu javax.speech. Ta ustvarja dogodke (angl. *events*), kot so spremembe v stanju, dogodki o napakah, zvočni vhodni in izhodni dogodki, ki jih Java Speech API s pomočjo JavaBeans upravlja.

Aplikacija je namenjena upravljanju računalnika, kjer govorec s pošiljanjem govornih signalov v mikrofon ustvari niz ukazov, ki jih sistem sprejme, razpozna in govorcu nato vrne rezultat na zaslon. Govorec lahko z besedami, ki jih izreče v mikrofon, brez uporabe tipkovnice, odpira razne programe, brska po Internetu, se premika po straneh, ureja besedilo, odpira nova okna, itd.

Za izgradnjo aplikacije potrebujemo:

- java razred za razpoznavo govora,
- slovnico (gram datoteko) in
- konfiguracijsko datoteko.

Aplikacijo smo v celoti razvili v programskem jeziku Java s pomočjo razvojnega orodja Eclipse [13]. Aplikacijo sestavljata dva razreda. Prvi razred skrbi za izgled uporabniškega vmesnika, drugi pa za razpoznavanje govora. Pri sami izgradnji vmesnika smo uporabili orodje WindowBuilder [14]. WindowBuilder, ki je sestavljen iz orodja SWT (angl. *Standard Widget Toolkit*) in Swing, je zelo enostaven pri oblikovanju Java grafičnega uporabniškega vmesnika. WindowBuilder je zgrajen kot vtičnik za orodje Eclipse. Vtičnik gradi abstraktno sintaktično drevo in grafično urejevalno okno (angl. *graphical editing framework*).

Razred, ki skrbi za razpoznavo govora sprejema in obdeluje govorne ukaze. Objekt ConfigurationManager zbira informacije o sistemu iz konfiguracijske datoteke ter tako pridobiva vire iz sistema. Metoda recognize() vrne Result objekt, ki nam služi, da (s pomočjo metode getBestFinalResultNoFiller(), ki je del objekta Result) dobimo rezultat oz. nam vrne najboljše ujemanje.

## 6.1 Slovnica

Slovnica razpoznavalniku pove, katere besede lahko pričakuje od govorca. S pravili, ki jih uporabimo v slovnici lahko povemo npr. kako si bodo besede sledile, koliko ponovitev neke besede lahko razponavalnik pričakuje, kdaj se lahko neka beseda pojavi (ali je beseda odvisna od predhodne besede ali ne), itd. Slovnica tudi precej zmanjša obseg besed, ki jih lahko govorec izgovori. Lahko rečemo, da je to ena od pozitivnih strani slovnice. Pozitivna pa zato, ker razpoznavalnik ve katere besede lahko pričakuje in je tako razpoznavanje hitrejše ter bolj učinkovito.

Format, ki smo ga uporabili za predstavitev slovnice je JSGF (angl. *Java Speech API Grammar Format*) [17]. JSGF uporablja BNF (angl. *Backus-Naur Form*) notacijo, ki se uporablja za tekstovno predstavitev slovnic v razpoznavalnikih govora za tehnologije XHTML (angl. *Extensible HyperText Markup Language*)+Voice. Za upravljanje računalnika smo zgradili slovnico (Slika 11) v katero smo zapisali besede in jim določili pravila.

```
#JSGF V1.0;

grammar commands;

public <navigation> =(Back | Forward | Home | Refresh);
public <currentPage>= (Page button | Page top | Scroll up | Scroll down |
Print | Save | Zoom in | Zoom out | Next | Enter | Text (one | two | three) );
public <editing>=(Copy | Cut| Delete | Paste | Select all);
public <window>=(Open <programs> | Close | New | Next window);
public <site> = (Site) (Facebook | Youtube | Google);
<programs>= (Word | Excel |Browser | Player | Paint | Control| Notepad);
```

Slika 11: Slovnica.

Slovnici na vrhu določimo ime - »commands«, da v konfiguracijski datoteki vemo na katero ime se bomo sklicevali. Vidimo, da slovnica vsebuje nabor pravil. V lomljenih oklepajih zapišemo ime pravila. V primeru, da postavimo v ospredje zapis public lahko besede, ki so napisane znotraj tega pravila izgovorimo neposredno, v primeru da nimamo v ospredju public (to so podpravila), teh besed ne bo sistem razpoznal, ne da bi pred tem izgovorili drugo/e besedo/e. To lahko vidimo iz slike 11, kjer npr. programe kot so Word, Excel, Notepad, itd. sistem ne bo razpoznal brez predhodno izgovorjene besede Open. Besede, ki jih lahko govorec izgovori so zapisane v okroglih oklepajih, ki so ločene s pokončnim oklepajem. Pri izgradnji slovnice lahko uporabimo tudi nabor drugih pravil, vendar v našem primeru so zadostovala leta.

## 6.2 Konfiguracijska datoteka

Komponente (jezikovni model, slovar, akustični model, itd.), ki sestavljajo sistem Sphinx-4 so v konfiguracijski datoteki predstavljene v formatu XML (angl. *Extensible Markup Language*).

Konfiguracijska datoteka določa naslednje:

- imena in tipe vseh komponent sistema,
- povezljivost komponent sistema v smislu katere komponente bodo medsebojno komunicirale in
- podrobno konfiguracijo za vsako od teh komponent.

V konfiguracijski datoteki smo morali popraviti pot do slovnice (Slika 13) in pot do slovarja (Slika 14), saj smo za potrebe aplikacije potrebovali dodatne besede v slovarju.

```
ZYWICKI                 Z IH W IH K IY
ZZZZ                    Z IY Z
ZZZZ(2)                 Z Z
{BRACE                  B R EY S
{LEFT-BRACE             L EH F T B R EY S
}CLOSE-BRACE            K L OW Z B R EY S
}RIGHT-BRACE            R AY T B R EY S
FACEBOOK                F EY S B UH K
GOOGLE                  G UW G AH L
NOTEPAD                 N OW T P AE D
YOUTUBE                 Y AW T Y UW B
```

Slika 12: Del slovarja besed.

Slovar, ki ga uporablja sistem je precej obsežen, vendar v primeru, da besedo, ki jo želimo izgovoriti ni v privzetem slovarju sistema Sphinx-4, te besede ne bo mogel sistem razpoznati, zato smo besede, ki smo jih potrebovali vstavili v slovar ter jim določili fonemski zapis (izgovorjavo), kot prikazuje slika 12.

```
<!-- ******************************************************* -->
<!-- The Dictionary configuration                            -->
<!-- ******************************************************* -->

<component name="dictionary"
    type="edu.cmu.sphinx.linguist.dictionary.FastDictionary">
    <property name="dictionaryPath"
 value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/cmudict.0.6d"/>
    <property name="fillerPath"
 value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/noisedict"/>
    <property name="addSilEndingPronunciation" value="false"/>
    <property name="allowMissingWords" value="false"/>
    <property name="unitManager" value="unitManager"/>
</component>
```

Slika 13: Nastavitev poti slovarja.

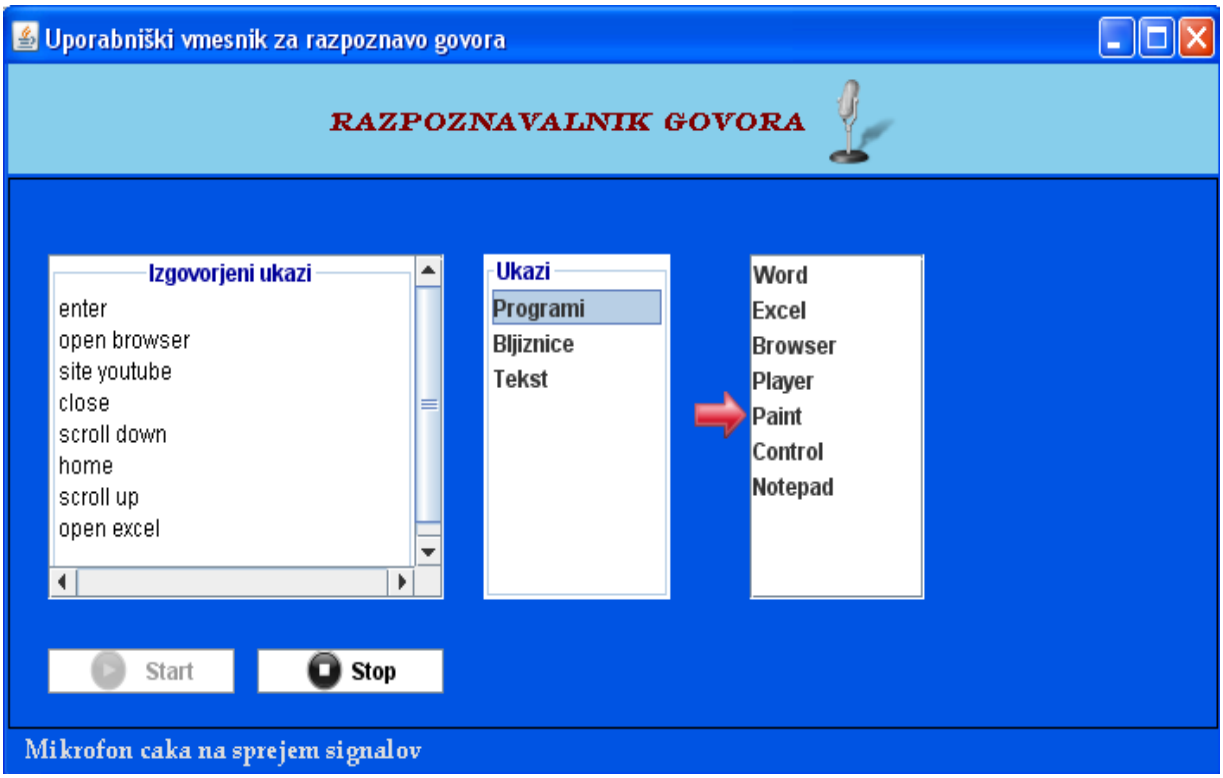
```
<!-- ******************************************************* -->
<!-- The Grammar  configuration                              -->
<!-- ******************************************************* -->

<component name="jsgfGrammar" type="edu.cmu.sphinx.jsgf.JSGFGrammar">
    <property name="dictionary" value="dictionary"/>
    <property name="grammarLocation" value="resource:/aplikacija/"/>
    <property name="grammarName" value="commands"/>
<property name="logMath" value="logMath"/>
</component>
```

Slika 14: Nastavitev poti slovnice.

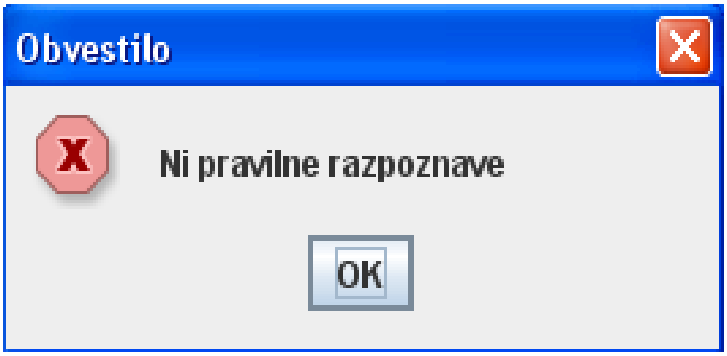
## 6.3 Uporabniški vmesnik

Slika 15 prikazuje uporabniški vmesnik aplikacije. Vmesnik nudi govorcu pregled izgovorjenih ukazov, opozarja na napake v primeru, da razpoznavanje ni bilo pravilno, izpisuje obvestila in omogoča pregled ukazov (to so ukazi, ki jih bo sistem razpoznal in jih lahko govorec izreče).



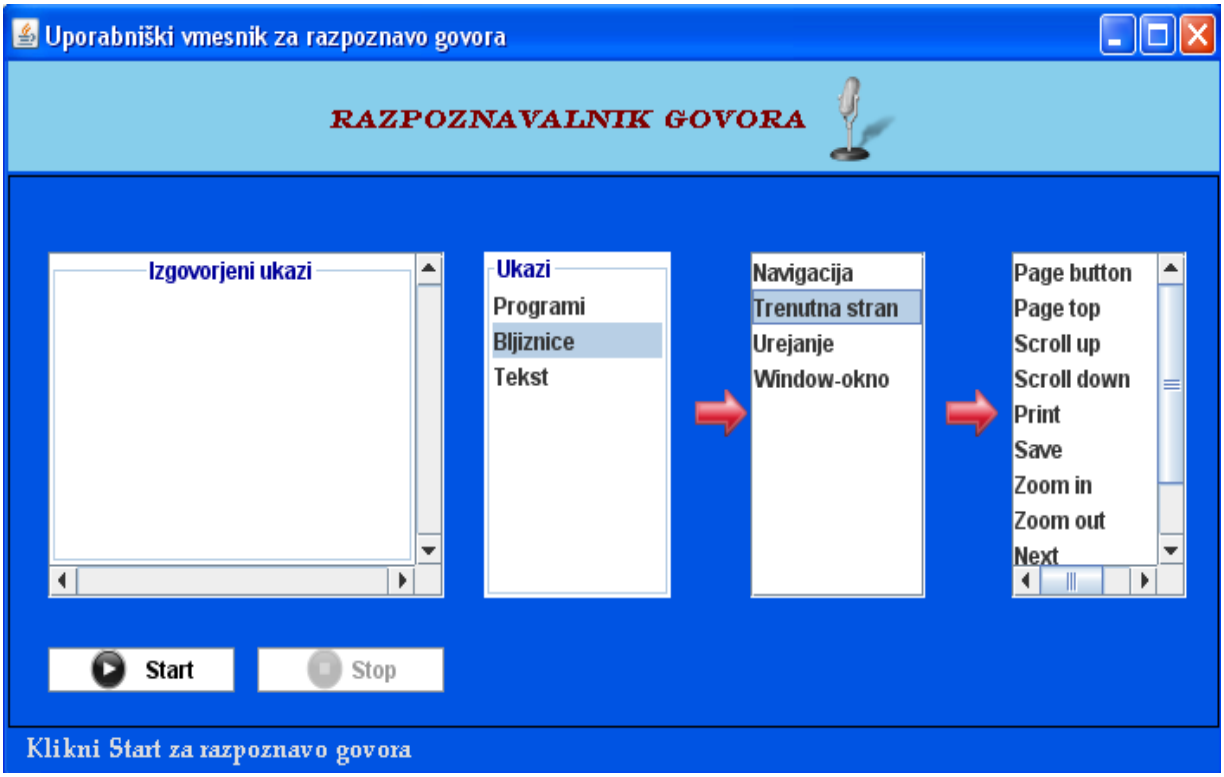

Slika 15: Uporabniški vmesnik aplikacije.

Kot vidimo iz slike 15 ima aplikacija dva gumba, statusno vrstico, okno z izgovorjenimi ukazi ter okno z ukazi, kjer ima vsak ukaz svoje podokno. V primeru, da govorec klikne na gumb »Start« se začne razpoznavanje govora in aplikacija čaka na sprejem ukazov. S tem tudi statusna vrstica v aplikaciji, govorca obvesti, da mikrofon čaka na sprejem signalov. S klikom na gumb »Stop« končamo razpoznavo, vendar s tem ne zapremo aplikacije in govorec lahko kadarkoli ponovno klikne na gumb »Start« za ponovno razpoznavanje govora. Razpoznani ukazi, ki se zapisujejo v okno, ki nosi naslov »Izgovorjeni ukazi« so pravzaprav besede, ki jih je govorec izrekel. V primeru, da je bila razpoznana beseda pravilna se v okno pojavi zapis te besede (razvidno iz slike 15), v nasprotnem primeru se pojavi opozorilno okno (Slika 16), ki govorca obvesti, da razpoznavanje ni bilo pravilno.




Slika 16: Opozorilno okno.

Okna, ki so desno od okna v katero se zapisujejo izgovorjeni ukazi govorca, služijo govorcu za pregled ukazov (Slika 17), ki so zajeti v slovnici (govorec je tako seznanjen z besedami, ki jih lahko izgovori). Okno z imenom »Ukazi« vsebuje tri opcije, to so: »Programi«, »Bljiznice« in »Tekst«. V primeru, da kliknemo na »Programi« se izpišejo programi, ki jih bo sistem zaznal v primeru, da jih govorec izreče.




Slika 17: Pregled ukazov.

Za odpiranje programov (Slika 18) smo uporabili objekt Process, kjer smo preko lupine operacijskega sistema, s pomočjo ukaza »start«, posamezen program spravili v zagon.

```
if(token.equals("word")){
    pr=Runtime.getRuntime().exec("cmd /c start winword");
}else if(token.equals("browser")){
    pr=Runtime.getRuntime().exec("cmd /c start firefox");
}else if(token.equals("player")){
    pr=Runtime.getRuntime().exec("cmd /c start wmplayer");
}else if(token.equals("paint")){
```

Slika 18: Del kode za zagon programov.

»Bljiznice« so razvrščene v štiri kategorije. Pod prvo imamo »Navigacija«, ki govorcu nudi nabor ukazov za usmerjanje po straneh, pod »Trenutna stran« imamo nabor ukazov, ki govorcu služijo predvsem za upravljane s trenutno stranjo oz. programom, bodisi, da imamo v zagonu program Word, Excel ali internetno stran. »Urejanje« nam prikazuje vrsto ukazov, ki so primerni predvsem za urejanje teksta v urejevalnikih besedila, kot je na primer kopiranje, lepljenje, brisanje, itd. Kot zadnja opcija, ki jo imamo pod »Bljiznice« je »Window« - okno, ki prikazuje nabor ukazov za delo z okni (odpiranje, zapiranje, itd.). V oknu z ukazi imamo možnost izbire tudi »Tekst«, kjer govorec npr. če izgovori »text one« se, če ima odprt urejevalnik besedila oz. okno, kjer lahko vpisuje tekst, izpiše tekst, ki se nahaja pod besedo »text one«.

Za izvršitev ukazov (Slika 19), ki jih imamo pod opcijama »Bljiznice« in »Tekst« smo uporabili objekt Robot, ki mu glede na sprejet ukaz oz. besedo, ki jo je govorec izrekel, povemo, kaj je njegova naloga oz. katere tipke mora pritisniti, da se bo ukaz izvršil in se tako prikazal rezultat govorcu na zaslon.

```
case "page":
    if(tokenizer.hasMoreTokens()){
       token=tokenizer.nextToken();
       if(token.equals("button")){
            robot.keyPress(KeyEvent.VK_END);
            robot.delay(20);
            robot.keyRelease(KeyEvent.VK_END);
       }else if(token.equals("top")){
            robot.keyPress(KeyEvent.VK_HOME);
            robot.delay(20);
            robot.keyRelease(KeyEvent.VK_HOME);
```

Slika 19: Del kode za izvršitev ukazov.

# 7 Sklepne ugotovitve

V diplomskem delu smo predstavili dva sistema za razpoznavo govora, to sta Sphinx-4 in Julius. V povezavi s sistemom Sphinx-4 smo izdelali aplikacijo, ki govorcu omogoča, da z govorjenem v mikrofon upravlja računalnik. Uporabniški vmesnik, ki je predstavljen v diplomski nalogi omogoča govorcu poleg pregleda izgovorjenih ukazov tudi pregled ukazov, ki jih lahko izreče oz. jih bo sistem zaznal.

Ko primerjamo predstavljena sistema ne moremo reči, da je nekateri boljši ali slabši. Natančnost in učinkovitost sta si zelo podobna, zlasti ko govorimo o sistemih LVCSR (angl. *Large-Vocabulary Continuous Speech Recognition*) kot sta tudi sistema Sphinx-4 in Julius. Sistem Sphinx-4 se je izkazal za zelo učinkovit ter preprost za uporabo, ki razvijalcu nudi dovolj dober pregled nad sistemom in mu tako omogoča izdelavo modulov oz. spreminjanje, dodajanje parametrov, ki jih potrebuje pri izdelavi aplikacije.

Nadaljnje izboljšave, ki bi jih aplikacija nudila govorcu je učenje sistema, kjer bi govorec lahko ustvaril profil za glas. To bi izboljšalo predvsem učinkovitost, saj nas bi s pomočjo lastnega profila, ki bi ga imel posamezen govorec, sistem bolje razumel in tako bi tudi lažje prepoznal izgovorjeno besedo. Težava, ki se bi lahko tu pojavila je ta, da bi lahko sistem deloval nekoliko počasneje, saj bi za posameznega govorca morali imeti shranjene izgovorjene glasove in bi tako postal precej obsežnejši. Ena izmed izboljšav bi bila tudi ta, da bi sistem govorcu omogočal razpoznavanje ne samo angleških besed, temveč bi bila prilagojena govorčevemu jeziku. Tu bi potrebovali več različnih jezikovnih modelov.

Razpoznavanje govora je še vedno zelo aktivno raziskovalno področje, ki bo v naslednjih letih zaradi razvoja tako strojne kot programske opreme verjetno še zelo napredovalo in s tem bodo sistemi tudi bolj prilagojeni pri razpoznavanju naravnega človeškega govora.

# Literatura

[1] D. Jelenc, »Osnovna vedenja o komunikaciji«, Pedagoška fakulteta, Ljubljana, 1998, pogl. 3.

[2] Z. Kačič, »Komunikacija človek-stroj«, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, 1995, pogl. 5.

[3] Z. Kačič, »Govorne tehnologije v telekomunikacijah«. Dostopno na: http://lojze.lugos.si/jota/KACIC_JOTA_24_02_2005.pdf, Ljubljana, 2005.

[4] A. Lee, T. Kawahara and K. Shikano. »Julius: An Open Source Real-Time Large Vocabulary Recognition Engine«. Dostopno na: http://julius.sourceforge.jp/paper/ri-eurospeech2001.pdf, 2009.

[5] D. Marc, K. Torkar Papež., »Kultura govorjene in zapisane besede ali retorika za današnjo rabo«, DZS, Ljubljana, 2006, pogl. 3.

[6] R. Rozman, »Nesimetrične okenske funkcije v sistemih za razpoznavo govora«, doktorska disertacija, Fakulteta za računalništvo in informatiko. Dostopno na: http://eprints.fri.uni-lj.si/772/1/doktorat_vse.pdf, Ljubljana, 2005.

[7] M. Sepesy Maučec, A. Žgank, »Speech Recognition in the Broadcast News Domain«, Fakulteta za elektrotehniko, računalništvo in informatiko. Dostopno na: http://www.pazu.si/dokumenti/25/2/2011/Anali_PAZU-1-Sepesy_Maucec_548.pdf, Maribor, 2011.

[8] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh and E. Gouvea, »Sphinx-4: A Flexible Open Source Framework for Speech Recognition«. Dostopno na: http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf, 2009.

# Spletni viri

[9] Apache ant (2013). Dostopno 22.3.2013 na: https://ant.apache.org/.

[10] Astem (2013), Interactive Speech Technology Consortium. Dostopno 19.5.2013 na: http://www.astem.or.jp/istc/index.

[11] CMUSphinx (2013), Sphinx-4 A speech recognizer written entirely in the Java™ programming language. Dostopno 5.3.2013 na: http://cmusphinx.sourceforge.net/sphinx4/.

[12] CMUSphinx (2013), JSAPI. Dostopno 22.3.2013 na: http://cmusphinx.sourceforge.net/sphinx4/doc/jsapi_setup.html.

[13] Eclipse (2013). Dostopno 5.4.2013 na: http://www.eclipse.org/.

[14] Eclipse (2013), WindowBuilder. Dostopno 5.4.2013 na: http://www.eclipse.org/windowbuilder/.

[15] HTK (2013). Dostopno 16.5.2013 na: http://htk.eng.cam.ac.uk/.

[16] Speech (2013), SphinxTrain. Dostopno 17.5.2013 na: http://www.speech.cs.cmu.edu/sphinxman/scriptman1.html.

[17] W3 (2013), Java Speech Grammar Format. Dostopno 18.3.2013 na: http://www.w3.org/TR/jsgf/.