Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rozpoznávanie reči v zjednodušenom anglickom jazyku pre účely robotickej aplikácie

Bakalárska práca

2019
Dávid Šuba

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rozpoznávanie reči v zjednodušenom anglickom jazyku pre účely robotickej aplikácie

Bakalárska práca

Študijný program:	2511 Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	prof. Ing. Igor Farkaš, Dr.

Bratislava, 2019
Dávid Šuba

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Dávid Šuba
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Rozpoznávanie reči v zjednodušenom anglickom jazyku pre účely robotickej aplkácie
Speech recognition in simplified English language for a robotic application

Anotácia: V interakcii človeka s robotickým systémom vzniká prirodzená potreba komunikovať v prirodzenom jazyku, často v nejakej konkrétnej doméne. V súčasnosti existuje niekoľko systémov, najmä pre angličtinu, ktoré sa dajú pre taký účel použiť, prípadne dotrénovať pre potreby užívateľa s cieľom maximalizovať presnosť rozpoznania slov, nezávisle od hovoriaceho.

Cieľ: 1. Naštudujte si problematiku rozpoznávania reči, princíp skrytých markovovských modelov a umelých neurónových sietí, a oboznámte sa so systémom HTK Toolkit.
2. Dotrénujte a otestujte systém HTK na vami pripravenej dátovej množine (oznamovacie a príkazové anglické vety týkajúce sa opisu objektov na scéne, viacero hovoriacich).
3. Doprogramujte potrebné skripty potrebné pre nasadenie systému do prevádzky.

Literatúra: Jurafsky D., Martin J. (2008) Speech and Language Processing, Pearson International Edition.
HTK Toolkit, Cambridge, UK, http://htk.eng.cam.ac.uk/

Vedúci: prof. Ing. Igor Farkaš, Dr.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 24.10.2018

Dátum schválenia: 24.10.2018 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie: čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím uvedenej literatúry.

V Bratislave dňa: .....

.....
Dávid Šuba

Poďakovanie: Chcel by som sa poďakovať môjmu školiteľovi prof. Ing. Igorovi Farkašovi, Dr. za možnosť venovať sa tejto zaujímavej téme a ochotu pomôcť, keď som potreboval. Taktiež ďakujem všetkým, ktorý mi poskytli potrebné zvukové nahrávky. V neposlednom rade chcem poďakovať mojim spolubývajúcim, ktorí so mnou mali trpezlivosť, keď som aj v neskorých nočných hodinách testoval systém.

Abstrakt

Bakalárska práca sa venuje rozpoznávaniu reči pomocou skrytých Markovovských modelov. Je tu popísaná teória okolo danej problematiky a navrhnutý rozpoznávací systém pomocou nástroja HTK, ktorý je natrénovaný a otestovaný na vlastnej dátovej množine. Systém je zameraný na niekoľko typov anglických viet, ktoré slúžia ako povely robotickému zariadeniu. Zaoberá sa tiež rôznym experimentom na zvýšenie úspešnosti rozpoznávania.

Ďalej je tu popísaná aplikácia, ktorá detekuje reč zo zvukového signálu a používa natrénovaný systém na rozpoznanie povelov. Aplikácia zabezpečuje komunikáciu medzi používateľom a robotickým zariadením a umožňuje upravovať parametre detekcie reči.

Kľúčové slová: rozpoznávanie reči, HTK, skryté markovovské modely, detekcia reči

Abstract

Bachelor thesis deals with the speech recognition based on hidden Markov models. There is described theory around given issue and designed recognition system by means of HTK toolkit, which is trained and tested on own dataset. System is focused on several types of english sentences, which serves as commands for robotic device. It also deals with different experiments for increasing successfulness of recognition.

Furthermore there is described application, which detects speech from audio signal and uses trained system for commands recognition. Application covers communication between user and robotic device and allows adjusting parameters of speech detection.

Keywords: speech recognition, HTK, hidden Markov models, speech detection

Obsah

Úvod

V dnešnej modernej dobe hrajú technológie dôležitú úlohu v našich životoch. Stretávame sa a interagujeme s nimi na každom kroku. Mobilné telefóny sa stali našou neoddeliteľnou súčasťou, vlastníme inteligentné autá, domy či dokonca chladničky. Vzniká potreba intuitívnej komunikácie medzi človekom a zariadením. Od ťažkopádnych klávesníc sa presúvame k dotykovým obrazovkám alebo hlasovému ovládaniu. Práve ovládanie hlasom zažíva v dnešných dňoch veľký pokrok a je to trend, ktorý sa bude len stupňovať. Je to z toho dôvodu, že komunikácia rečou je najprirodzenejšia forma výmeny informácií medzi ľuďmi a nie sú potrebné žiadne špeciálne zručnosti alebo vedomosti, aby sa človek naučil používať takto ovládané zariadenie. Na úspešné nasadenie takéhoto systému, však musíme zvládnuť problém rozpoznania reči. Musíme byť schopný získať z audio signálu jeho textový prepis.

Táto úloha nie je v informatike nová a v priebehu desaťročí sa vystriedali rôzne prístupy a metódy pri jej riešení. V posledných pár rokoch však nastala revolúcia v oblasti rozpoznávania reči kvôli vysokej úspešnosti metód hlbokého učenia. Takéto systémy sú kombináciou veľkého množstva dát a výpočtovej sily. Väčšinou sú závislé na internetovom prístupe, keďže rozpoznávanie je realizované na vzdialených serveroch. Ak tieto zdroje nie sú k dispozícii, je treba hľadať inde. Štatistický prístup pomocou skrytých Markovovských modelov nie je síce nový, ale s rôznymi vylepšeniami dokáže stále konkurovať moderným technológiám, hlavne v prípadoch, keď nedisponujeme spomenutými zdrojmi.

Hlavným cieľom našej práce je navrhnúť reálne použiteľný systém na ovládanie robotickej ruky hlasom, založeným na skrytých Markovovských modeloch. V prvej kapitole sa budeme zaoberať históriou tejto problematiky, rozličnými technológiami použitými na rozpoznávanie reči a podobným prácam, ktoré boli zamerané na túto tému. Ďalej sa pozrieme na spôsob parametrizácie signálu pre potreby rozpoznávania reči a rôzne analýzy signálu zamerané na detekciu reči. Predstavíme si princíp skrytých Markovovských modelov a algoritmy, ktoré nám slúžia na prácu s nimi. Taktiež si povieme niečo o nástroji HTK, ktorý nám zjednodušuje prácu s Markovovskými modelmi a poskytuje nám možnosti na ich vytvorenie, trénovanie a otestovanie.

V posledných dvoch kapitolách si predstavíme návrh nášho systému, problémy s

ktorými sme sa museli zaoberať a samotnú implementáciu rozpoznávača a výslednej aplikácie.

1. Úvod do problematiky

V tejto kapitole sa budeme najskôr venovať histórii rozpoznávania reči. Pozrieme sa na rôzne prístupy k nej a na súčasný stav vývoja. Nakoniec si predstavíme rôzne podobné systémy a práce, ktoré sa venovali tejto téme. Nasledujúca kapitola bola zostavená podľa zdrojov [3], [4] a [1].

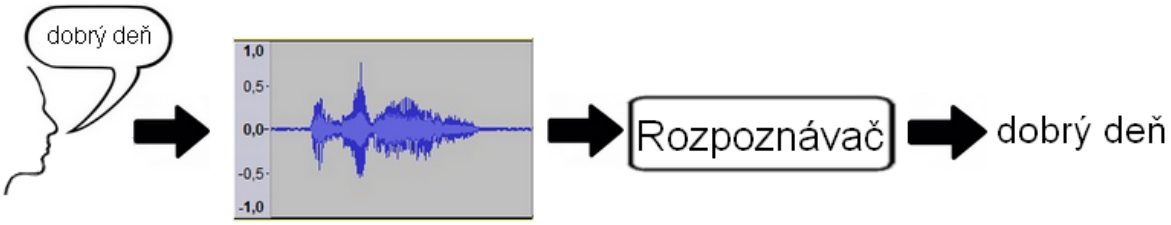
1.1 Úvod do rozpoznávania reči

Problém rozpoznávania reči (angl. automatic speech recognition, ASR) vznikol spoločne s vývojom modernej výpočtovej techniky. Hoci ešte stále nie je uspokojivo vyriešený, dnešné systémy sú dostatočne úspešné pre ich použitie v niektorých oblastiach.

Proces rozpoznania reči je náročný a oblastí, ktorými sa pri ňom musíme zaoberať je niekoľko a siahajú do viacerých oborov.

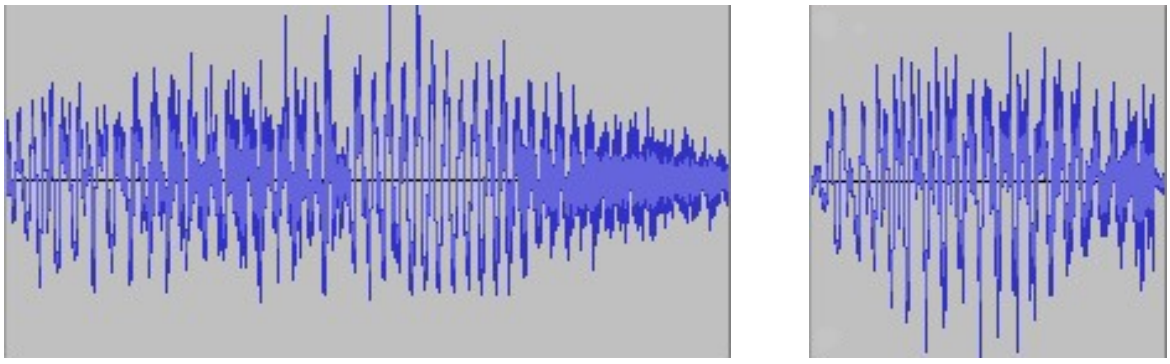
- Akustika - ako vzniká slovo v ľudskom hlasovom trakte, jeho šírenie rôznymi prostrediami a nakoniec spracovanie ušným ústrojenstvom.
- Spracovanie signálu - problematika digitalizovania signálu, odstraňovania šumu, získania vhodných informácií potrebných pre rozpoznanie reči.
- Lingvistika - stavba jazyka, rozdelenie slov na slabiky a hlásky, vzťah medzi zvukom (fonetika) a významom (sémantika) .
- Rozpoznávanie vzorov - počítačové algoritmy a metódy na klasifikáciu dát, hľadania podobností.

Počítač musí byť schopný zachytiť zvukovú vlnu, šíriacu sa vzduchom, spracovať ju, rozdeliť na slová, prípadne hlásky a poskytnúť používateľovi prepis vyslovenej reči (obr. 1).



Obr. 1: Rozpoznanie slovného spojenia “dobrý deň” ASR systémom

Zložitých problémov ktoré nám pri rozpoznávaní reči vyvstanú je niekoľko. Každý človek má iný hlas. Ženy väčšinou vyšší, muži nižší. Dokonca aj jednej osobe sa mení hlas v priebehu dňa. Iný má keď ráno vstane a iný napríklad počas choroby. Naša reč sa líši aj v tempe. Môžeme slovo vysloviť rýchlejšie, pomalšie. Znenie slova sa mení aj v závislosti od emócii, od použitého kontextu. Na obrázku 2 vidíme porovnanie zvukovej vlny slova "máš"v dvoch rôznych vetách, vyslovených tým istým rečníkom hneď za sebou. Na obidvoch obrázkoch je zobrazená len vlna slova “máš”. Tvar je síce podobný, ale dĺžka slova vo vete "Máš sa dobre?"je dvojnásobne dlhšia oproti vete “Ako sa máš?”. Ďalší faktor vstupujúci do do procesu rozpoznávania je vplyv pozadia. Absolútne tiché prostredie je nepoužiteľné, lebo sa nedá dosiahnuť v reálnom nasadení systémov. Vždy musíme počítať so šumom pochádzajúcim z nedokonalosti mikrofónu alebo z prostredia. Rozhovor v pozadí, otvorenie dverí alebo hučanie ventilátora zmenia charakter zvukovej vlny. Modely ASR systémov musia byť robustné a snažiť sa eliminovať tieto nežiaduce javy.



((a)) **Máš** sa dobre? ((b)) Ako sa **máš**?

Obr. 2: Porovnanie zvukových vĺn slova **máš** v 2 rôznych vetách.

1.1.1 Typy rozpoznávania

ASR systémy môžme rozdeliť podľa niekoľkých základných kritérií. Môžme rozpoznávať samotné hlásky, izolované slová alebo plynulú reč. Rozpoznávanie hlások je nároč-

nejšie, kvôli absencii kontextu, ktorý nám môže pomôcť určiť pravdepodobnosť priradenia hlásky k danej zvukovej sekvencii. Podobný princíp platí aj pri rozoznávaní izolovaných slov a plynulej reči, zloženej z viet. Pri definovanej gramatike jazyka vieme vypočítať pravdepodobnosť výskytu daného slova vo vete alebo vylúčiť vety, ktoré nespadajú do zvolených kritérií. Pri plynulej reči je však problém s rozličnými variantami toho istého slova podľa použitia, ako sme si ukázali na obrázku 2.

Hlavne v minulosti boli ASR systémy závislé na rečníkovi. To znamená, že boli schopné rozpoznať reč iba jedného, resp. niekoľkých ľudí. Bolo bežné, že po kúpe komerčného softvéru na transkripciu reči, musel používateľ prečítať a nahrať pripravený text a tak dotrénovať softvér na jeho konkrétny hlas. Dnes je snaha vyvíjať systémy nezávislé na rečníkovi. Je to možné hlavne vďaka dostupnému množstvu dát a zlepšeniu hardvéru.

Zaujíma nás aj veľkosť slovnej zásoby. Koľko slov je rozpoznávač schopný identifikovať. Samozrejme je jednoduchšie rozlišovať medzi niekoľkými povelmi, ako poznať jadro slovnej zásoby jazyka. Väčšina moderných systémov s veľkou slovnou zásobou je preto založená na rozpoznaní foném, resp. viacerých spojených foném - trifónov, bifónov, difónov. Z nich sa potom vyskladajú slová. Rozpoznávač by mal byť teda schopný rozpoznať aj neznáme slovo, na ktoré nebol špeciálne trénovaný.

Ďalšie kritérium ASR systému je či bude transkripcia prebiehať v reálnom čase. Ak chceme ovládať hlasom nejaké zariadenie, reakcia systému musí byť okamžitá aj za cenu miernych nepresností. Naopak pri generovaní prepisu videa je výhodnejšie nechať systém dlhšie pracovať a získať tak kvalitnejší text.

1.2 História

Použitie rozpoznávania reči je v poslednom čase veľmi skloňované, hlavne kôli produktom veľkých spoločností napr. Alexa, Siri alebo Google Assistant, ktoré sú spoľahlivo ovládané hlasom. O túto problematiku sa však zaujímali informatici už desaťročia dozadu a jej vývoj prešiel rôznymi objavmi a zmenami, ktoré vyústili do dnešného stavu. O podrobnejších informáciách k tejto téme sa môžme dočítať v práci [4].

1.2.1 Začiatky

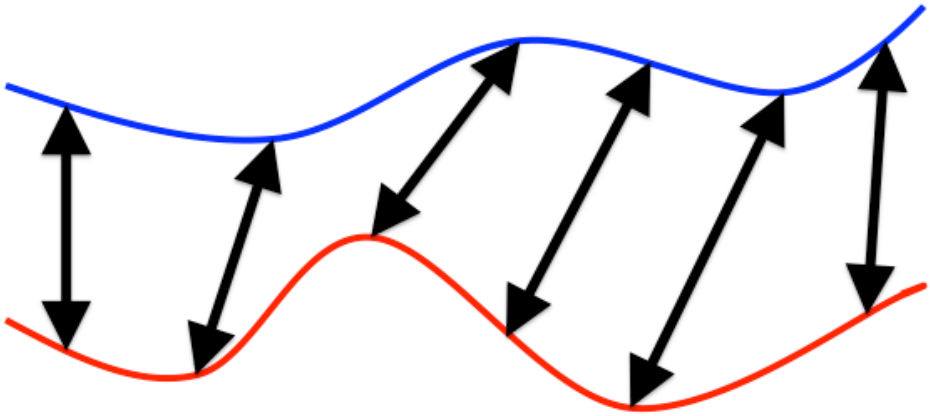
Za prvý pokus o ASR systém môžme považovať rozpoznávač “Audrey” z roku 1952 zostrojený v Bell Laboratories. Vedel rozpoznať číslice do desať vyslovené jedným rečníkom. Založený bol na rozpoznaní formantov, špičiek vo zvukovom spektre, počas samohlások každej číslice. Ďalšie výskumy sa zameriavali na rozpoznanie niekoľkých samohlások a spoluhlások, čo sa ukázal ako dobrý postup, keďže z hlások vieme vyskladať neskôr slová. Slovná zásoba však bola stále obmedzená na desať, dvadsať izolovaných

slov.

1.2.2 Dynamic time warping

Posun nastal vymyslením algoritmu Dynamic time warping (DTW), ktorý slúži na hľadanie podobnosti medzi dvoma lineárnymi vstupmi líšiacimi sa v rýchlosti, za pomoci dynamického programovania.

Rovnaké slovo vyslovené dvoma rôznymi rečníkmi sa väčšinou líši v tempe. Niekto rozpráva rýchlejšie a niekto pomalšie. DTW "ohýba" časovú os, aby zrovnal vstupné signály.



Obr. 3: Ukážka zarovnávania dvoch signálov pomocou ohýbania časovej osi [26]

Neznáme slovo sa samostatne porovnáva so všetkými slovami v danom slovníku. Vyberie sa to, s ktorým má najväčšiu zhodu. Prístup je teda klasifikačný, z čoho plynie problém, že hoci sa slovo nemusí nachádzať vôbec v rozpoznávanom slovníku, ani sa na žiadne podobať, algoritmus ho vždy priradí k najpodobnejšiemu. Tento postup sa osvedčil najme pri rozpoznávaní menšieho počtu izolovaných slov. Veľkosť slovnej zásoby, ktorú ASR systém dokázal rozpoznať ale stúpla na niekoľko desiatok slov.

1.2.3 Skryté Markovovské modely

Ďalší veľký skok v ASR systémoch priniesli skryté markovovské modely (angl. hidden Markov model, HMM). Tomuto riešeniu sa budeme podrobne venovať v tejto práci. Začali sa používať v 80-tych rokoch a tento nový štatistický prístup bol považovaný za najlepší až do nedávnej minulosti. S rozličnými vylepšeniami sa HMM ešte stále používajú v niektorých komerčných softvéroch.

1.2.4 Neurónové siete

Používanie umelých neurónových sietí (angl. artificial neural networks, ANN) v oblasti umelej inteligencie je dnes veľmi moderné. Tento model však bol známy už v 50-tych rokoch. Úspešné pokusy o jeho aplikáciu v oblasti rozpoznávania reči sa dosiahli v 80-tych a 90-tych rokoch minulého storočia. ANN neboli použité samostatne, ale vylepšovali konkrétne problémy v štatistických HMM systémoch, napríklad odhad rozdelenia pravdepodobnosti alebo parametrizácia vstupného signálu. Predstavený bol HMM/MLP model (MLP - multi layer perceptron) [5], ktorý načrtol budúcnosť použitia ANN v ASR, ale vtedajší hardvér a algoritmy na učenie nepostačovali, aby dokázali konkurovať systémom založených na skrytých Markovovských modeloch.

Odvtedy vývoj pokročil a pomerne dobrú úspešnosť dosahovali aj systémy založené hlavne na neurónových sieťach. Samotný klasifikátor je realizovaný ANN, ale ešte stále je použité predspracovanie signálu alebo parametrizácia signálu algoritmami, aké sa používajú napríklad pri HMM systémoch. Budeme si o nich hovoriť v kapitole 2. Nejde teda o rozpoznávanie "od začiatku do konca" (angl. end-to-end) neurónovými sieťami.

Rozpoznávaniu čísel slovenského jazyka za pomoci neurónových sietí sa venuje práca Vojtecha Slovika [6]. Použil trojvrstvovú klasifikačnú ANN s jednou vrstvou skrytých neurónov. Použil aj techniku "okna do minulosti a budúcnosti", ktorý eliminuje problém s časovým kontextom reči, teda že výslovnosť hlásky závisí aj od hlásky pred ňou a po nej. Podarilo sa mu dosiahnuť dobrú úspešnosť okolo 90% pre 16-slovný slovník a 8 rôznych rečníkov.

Skutočnú revolúciu do problému ASR však priniesla metóda hlbokého učenia (angl. deep learning) v posledných pár rokoch. K natrénovaniu DNN siete (z angl. deep neural network) je potrebné veľké množstvo výpočtovej sily a veľké množstvo dát. Dnes to už nie je problém, vďaka vývoju špecializovaných grafických kariet a technologickým gigantom ako napríklad Google, ktorý vie pohodlne zozbierať množstvo hovorenej reči od svojich používateľov.

DNN "end-to-end" rozpoznávače sú schopné väčšej abstrakcie nad vstupným signálom, teda lepšie využijú kontext časti signálu ako predchádzajúce rozpoznávače, ktoré ho analyzujú po malých častiach. Navyše, ako potvrdil tento experiment [7], pri trénovaní nie je potrebný fonetický prepis tréningových dát, čo predstavuje veľké ušetrenie práce pri príprave dátocých množín. DNN systémy svojou úspešnosťou rozpoznávania slov a plynulej reči už prevyšujú klasické HMM modely a práve týmto smerom sa uberá aktuálny vývoj ASR.

1.3 Podobné práce a systémy

Automatické rozpoznávanie reči je zaujímavý problém pre komerčnú sféru, kôli možnosti ovládať zariadenia hlasom. Už sme spomínali systémy Google Assistant, Siri a Alexa, ktoré dosahujú vysokú úspešnosť pre plynulú reč a veľkú slovnú zásobu. Pravdepodobne fungujú na technikách DNN. Je však potrebný internetový prístup. To ich robí v niektorých situáciách nepoužiteľnými. Systémy bez potreby internetu majú vyššiu nepresnosť alebo obmedzenú slovnú zásobu.

Oblasti rozpoznávania reči sa venovali aj iné bakalárske práce, napríklad [8]. Cieľ bol rozpoznať slovenské hlásky s využitím rôznych techník, ako napríklad modelovanie šumu pozadia. Dosiahnutá úspešnosť sa pohybovala len okolo 50%, ale podarilo sa dosiahnuť mierne zlepšenie po použití optimalizačnej metódy klasteringu. Túto techniku by sme mohli využiť aj v našej práci.

Rozpoznávaním reči založenom na zmesiach gausiánov sa zaoberala práca [9]. Oproti štandardným gausiánom používa hlbokú architektúru. To znamená, že použijeme viac vrstiev gausiánskych zmesí nad sebou. Ide o kombináciu techník hlbokého učenia a štandardného HMM-GMM prístupu (angl. Gaussian mixture models, GMM). Na vyhodnotenie úspešnosti tohto prístupu, bol systém natrénovaný na rozpoznávanie slovenských číslic. Dosiahol mierne zlepšenie o necelé percento oproti modelu bez hlbokej architektúry. Nevýhodou tohoto modelu sú zvýšené časové nároky na trénovanie, ktoré môže trvať aj desaťkrát dlhšie. Na základe poznatkov o hlbokých architektúrach môžme predpokladať, že navrhnutý model bude úspešnejší pri dostatočne veľkej trénovacej množine, avšak pri nedostatku trénovacích dát bude mať väčšiu chybovosť ako štandardný model.

2. Spracovanie signálu

Spracovanie audio signálu je prvým článkom v každom ASR systéme či už je založený na HMM, ANN, DTW alebo inom princípe. Zameriame sa na metódy týkajúce sa detekcie reči a parametrizácie signálu.

Vstupom pre rozpoznávanie reči je síce meniaci sa rečový audio signál, ale keď ho rozdelíme na dostatočne malé úseky (10-30ms) môžme ho považovať za stacionárny, vzhľadom k tomu, že náš hlasový aparát nemení svoju konfiguráciu v takomto krátkom časovom úseku. Môžeme teda považovať rečový signál za postupnosť stavov, ktorých je konečný počet. Následne sa snažíme získať informácie opisujúce tieto časti signálu. Parametre získané z týchto úsekov nazývame príznaky alebo pozorovania. Spôsobov akým sa signál parametrizuje, aby sme z neho získali čo najviac informácii o reči, je viacero, ale najpoužívanejšie - LPC a MFCC sa zakladajú na spektrálnej analýze signálu. Tiež si predstavíme koncept Hammingovho okienka.

Ďalej sa pozrieme na analýzu signálu za účelom detekcie reči. Detekcia reči je oblasť, ktorá úzko súvisí s rozpoznávaním reči, hlavne pri systémoch, ktoré pracujú v reálnom čase. Správna extrakcia úsekov s rečou má veľký vplyv na výsledky rozpoznávača. Predstavíme si niekoľko analýz signálu, ktoré môžme použiť pri riešení tohto problému. Kapitola bola spracovaná na základe zdrojov [25], [10], [17] a [18].

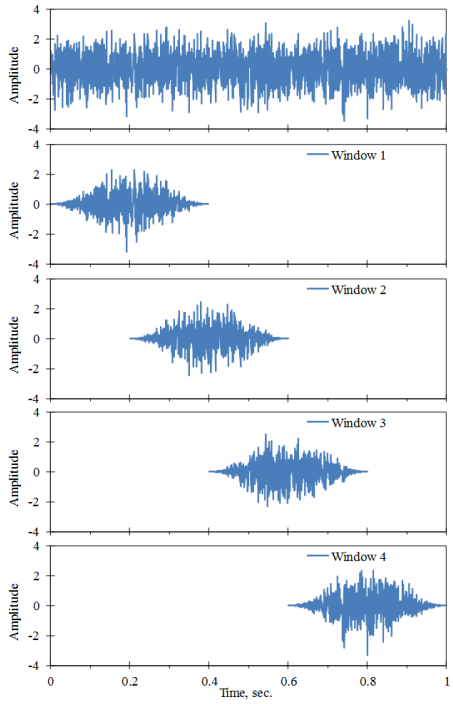
2.1 Hammingovo okienko

Ako sme si už povedali, rečový signál analyzujeme po krátkych úsekoch, rádovo v desiatkach milisekúnd. Ak by sme len jednoducho rozdelili zvukovú vlnu na pravidelné časti požadovanej dĺžky, stratili by sme kontext a prišli by sme o časť informácií. Preto sa používajú prekrývajúce sa "okienkové" funkcie (angl. window functions).
V oblasti spracovania signálu pre ASR systém je najpoužívanejšie Hammingovo okienko. Je to funkcia definovaná predpisom 2.1.

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{M-1}) \quad 0 \leq n \leq M-1 \tag{2.1}$$

kde M je veľkosť okienka.

Okienko "kĺže" po signále o hodnotu menšiu ako je jeho dĺžka, čiže sa bude prekrývať a ováhuje nám jednotlivé hodnoty amplitúdy vo výseku. Ukážku práce takéhoto okienka môžme vidieť na obrázku 4.



Obr. 4: Ukážka kĺzajúceho sa Hammingovho okienka [23]

2.2 Lineárne prediktívne kódovanie

Lineárne prediktívne kódovanie (angl. linear predictive coding, LPC) je metóda založená na predpoklade, že rečová vzorka môže byť odhadnutá ako lineárna kombinácia predchádzajúcich vzoriek. LPC modeluje rečový signál ako zdroj, ktorý je generovaný hlasivkami a filter tvorený krkom a ústami, definovaný podľa vzťahu 2.2. Snaží sa oddeliť tieto rezonancie od seba a lokalizovať už spomínané formanty - špičky vo zvukovom spektre.

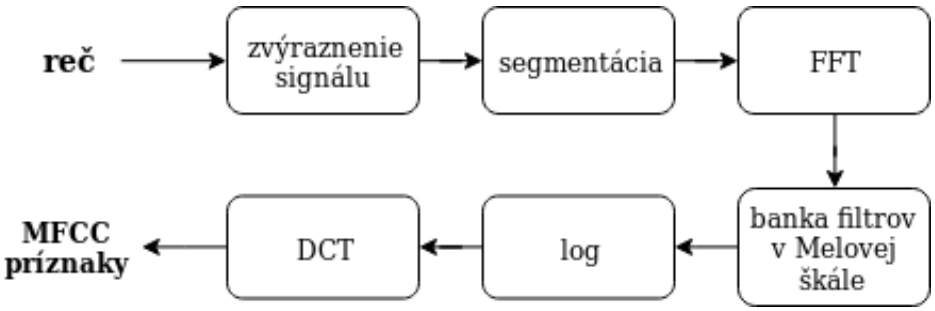
$$H(z) = \frac{1}{A(z)} \tag{2.2}$$

kde $A(z) = 1 + a_1 z^{-1} + ... + a_p z^{-p}$ je polynóm rádu p, pričom $p = 2k + 1$, k je počet formantov. Model sa nazývame, že je rádu p. Typické hodnoty p sú 10 pre vzorkovaciu frekvenciu 8kHz, pre vyššie frekvencie napr. 16. [16]

Ak máme model reči a chyby predikovaného signálu oproti skutočnému signálu, vieme reprodukovať signál, z ktorého boli tieto parametre vypočítané. Preto sa LPC používa aj na kompresiu rečového signálu, hlavne pri systémoch s nízkou prenosovou rýchlosťou. Na parametrizáciu signálu pre potreby rozpoznávania reči sa používajú parametre vymodelovaného filtra a_i.

2.3 Melove kepstrálne koeficienty

Melove kepstrálne koeficienty (angl. Mel-frequency cepstral coefficients, MFCC) je v súčastnosti asi najpoužívanejšia technika získania príznakov z rečového signálu. Jednotlivé fázy tohto procesu môžme vidieť na obrázku 5.



Obr. 5: Blokový diagram získavania MFCC príznakov

Prvá fáza, *zvýraznenie signálu*, je filter, ktorý zvýrazní vyššie frekvencie, ktoré boli potlačené počas vytvárania zvuku v ľudskom hlasovom trakte. Výpočet nového signálu je nasledovný:

$$S(n) = X(n) - a \cdot X(n-1) \tag{2.3}$$

kde $S(n)$ je zvýraznená vzorka signálu, $X(n)$ je pôvodná vzorka, $X(n-1)$ je predchádzajúca vzorka a a je koeficient zvýraznenia, väčšinou v intervale $<0.95, 1>$.

Druhá fáza je rozdelenie vstupného signálu na rámce. Tu sa používa Hammingovo okienko spomínané v časti 2.1.

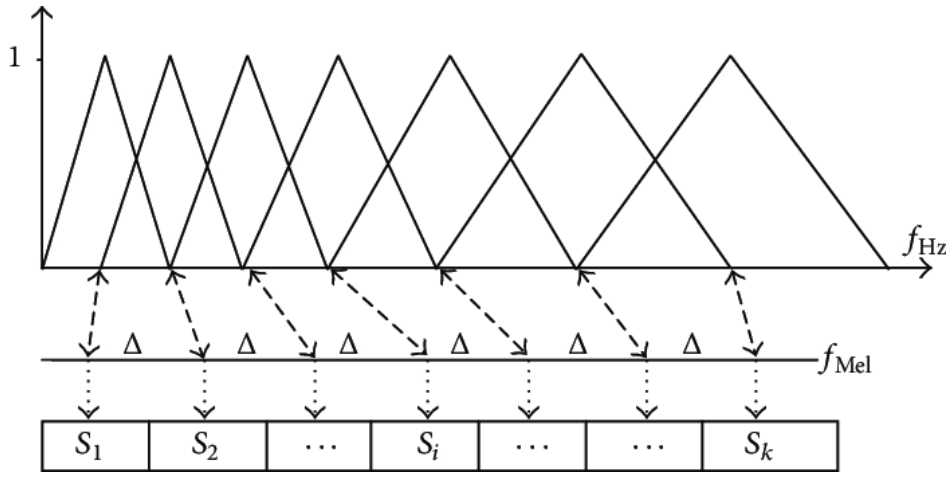
Tretia fáza je rýchla fourierova transformácia (angl. fast fourier transform, FFT). FFT je rýchla implementácia diskrétnej fourierovej transformácie (angl. discrete fourier transform, DFT), ktorá mení časovú doménu signálu do frekvenčnej domény. DFT je veľmi dôležitý nástroj v spracovaní signálu. Často je výhodnejšie pracovať s frekvenčným spektrom, ako so zvukovou vlnou vo forme sínusoidy. FFT vypočíta magnitúdu frekvencie vstupného rámca.

Štvrtou fázou je analýza pomocou banky filtrov v Melovej škále. Pozorovaním sa zistilo, že ľudské ucho vníma frekvencie nelineárne, preto sa frekvencie získané FFT prevedú do Melovej škály, ktorá sa viac snaží priblížiť k ľudskému vnímaniu frekvencií. Prevod frekvencie f sa realizuje nasledujúcim výpočtom:

$$M = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.4}$$

Ďalej vytvoríme banku prekrývajúcich sa trojuholníkových filtrov, ktoré sú rozložené pozdĺž Melovej škály ako je znázornené na obr. 6. To znamená, že filtre pri

vyšších frekvenciách budú širšie, ako tie pri nižších. Výstup každého filtra je príslušne ováhovaná suma spektrálnych komponentov.



Obr. 6: Analýza frekvenčného spektra pomocou banky filtrov [24]

Na výstup banky filtrov sa aplikuje logaritmus a následne diskrétna kosínusová transformácia (angl. discrete cosine transform, DCT). Tento proces slúži na prevod z Melovej spektrálnej domény naspať na časovú. DCT je definovaná vzorcom 2.5.

$$X_k = \alpha \sum_{i=0}^{N-1} x_i \cdot \cos\{\frac{(2i+1)\pi k}{2N}\} \tag{2.5}$$

kde α je konštanta závislá na N.

Týmto spôsobom dostaneme MFCC vektory príznakov pre rečový signál. Často sa ako dodatočná informácia pridá ešte suma energie rámca, delta a akceleračné koeficienty.

Delta a akceleračné koeficienty sa vypočítajú z MFCC koeficientov a dávajú nám informáciu ako sa menili hodnoty MFCC koeficientov v čase. Výpočet delta koeficientov sa podľa [19] realizuje nasledovne:

$$d_t = \frac{\sum_{i=1}^{n} w_i(c_{t+1} - c_{t-1})}{2\sum_{i=1}^{n} w_i^2} \tag{2.6}$$

kde n je veľkosť okienka, z ktorého sa počítajú koeficienty, w_i je regresný koeficient a c_t je pôvodný MFCC koeficient.

Akceleračné koeficienty sa počítajú rovnako, len ich počítame z delta koeficientov namiesto MFCC koeficientov.

2.4 Analýza signálu za účelom detekcie reči

Problém detekcie reči spočíva v označení intervalov signálu, ktoré obsahujú ľudský hlas. Prvotné spracovanie audia je podobné ako pri parametrizácii. Vstupný signál sa rozsegmentuje na niekoľkomilisekundové rámce a pre každý sa vypočíta parameter podľa zvolenej analýzy. Následne je možné určiť prah, ktorý oddeľuje reč od iných zvukov.

ZCR (angl. zero crossing rate) je analýza založená na počítaní prechodov zvukovej vlny cez nulu, teda koľkokrát sa zmení znamienko signálu. Segmenty, v ktorých sa hovorí by mali mať ZCR nižší, keďže pozostávajú z viac-menej konštantných frekvencií, hlavne počas trvania samohlások. Výpočet pre rámce $x(n) = 0, 1...N-1$ môžme definovať nasledovne:

$$ZCR = \frac{1}{2N}\sum_{n=0}^{N-1}(sgn(x(n)) - sgn(x(n-1))) \tag{2.7}$$

kde znamienková funkcia $sgn(x(n))$ je definovaná takto:

$$sgn(x(n)) = \begin{cases} 1, & x(n) \geq 0 \\ -1, & x(n) < 0 \end{cases} \tag{2.8}$$

SFM (angl. spectral flatness measure) vyjadruje "plochosť"magnitúdového spektra. Matematicky to môžme vyjadriť ako podiel geometrického a aritmetického priemeru magnitúdového spektra. SFM by sa malo blížiť k 1 pre zašumený signál.

$$SFM = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} \tag{2.9}$$

Ďalší druh analýzy používa rýchlu Fourierovu transformáciu, ktorú sme už spomínali. Pomocou nej nájdeme v segmente dominantnú frekvenciu. Tá by mala byť pre signál obsahujúci reč vyššia ako pre signál bez reči.

Krátkodobá energia (angl. short-term energy, STE) je definovaná nasledovne:

$$STE = \sum_{n=0}^{N-1} x(n)^2 \tag{2.10}$$

. STE analýza vychádza z logického predpokladu, že reč má vyššiu amplitúdu ako šum. Filtruje signál iba na základe amplitúdy, takže zachytí ľubovolný hlasný zvuk, nielen reč.

3. Skryté markovovské modely

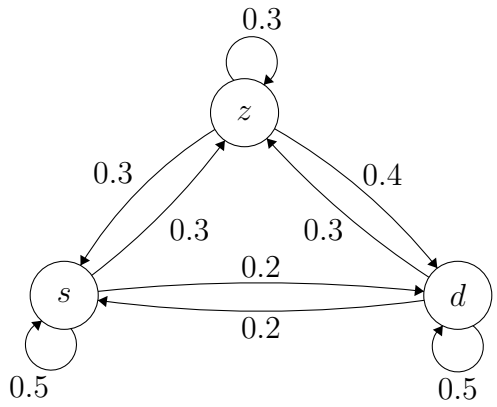
Skryté markovovské modely(HMM - z angl. hidden markov models) boli donedávna najpoužívanejším nástrojom na rozpoznávanie reči. Svoje postavanie na ASR scéne si držali od začiatku 90. rokov až do príchodu metód hlbokého učenia. V tejto kapitole si predstavíme teóriu okolo HMM a algoritmy, ktoré nám slúžia na prácu s nimi. Čerpali sme zo zdrojov [1], [3], [10] a [19].

HMM sú rozšírením markovovských reťazcov. Markovovské reťazce sa snažia modelovať náhodné procesy ako postupnosti stavov a pravdepodobnosti prechodov medzi týmito stavmi. Založené sú na predpoklade, že budúcnosť závisí len na aktuálnom stave, nie na minulosti.

Klasický príklad použitia markovovských reťazcov je predpoveď počasia. Predpokladajme, že počasie je vždy definované jedným z nasledujúcich stavov: slnečno (s), zamračené (z), dážď (d). Ďalej predpokladajme, že pravdepodobnosti prechodov medzi jednotlivými stavmi sú nasledovné:

$$\mathrm{A} = \begin{array}{c} \\ s \\ z \\ d \end{array} \begin{array}{c} \begin{array}{ccc} s & z & d \end{array} \\ \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.3 & 0.3 & 0.4 \\ 0.2 & 0.3 & 0.5 \end{pmatrix} \end{array}$$

Takto definovaný model si môžme predstaviť ako konečný automat zobrazený na obrázku 7.



Obr. 7: Markovovský reťazec M na predpoveď počasia

Skúsme vypočítať pravdepodobnosť, že najbližších 5 dní bude počasie nasledovné: s, s, z, d, z, ak dnes prší. Chceme vypočítať pravdepodobnosť sekvencie stavov $X = \{d, s, s, z, d, z\}$ z modelu M. Budeme vychádzať z vyššie spomenutého predpokladu, že nasledujúci stav, závisí iba od aktuálneho stavu.

$$\begin{aligned} P(X|M) &= P(d, s, s, z, d, z|M) \\ &= P(d) \cdot P(s|d) \cdot P(s|s) \cdot P(z|s) \cdot P(d|z) \cdot P(z|d) \\ &= \pi_d \cdot a_{ds} \cdot a_{ss} \cdot a_{sz} \cdot a_{zd} \cdot a_{dz} \\ &= 1 \cdot 0.2 \cdot 0.5 \cdot 0.3 \cdot 0.4 \cdot 0.3 \\ &= 0.0036 \end{aligned} \tag{3.1}$$

kde π_d je pravdepodobnosť, že dážď bude začiatočný stav. V našom prípade je to 1, lebo vieme, že dnes prší.

V tomto jednoduchom príklade vieme jednoznačne určiť aké je počasie, tak, že sa pozrieme z okna. Niektoré procesy ale nevieme pozorovať priamo (sú nám skryté) a informácie o nich získavame skrze iné procesy, ktoré produkujú nejaké pozorovania. Napríklad si predstavme, že sme v miestnosti bez okien a o tom aké je počasie nás informuje osoba, ktorá niekedy klame. Skutočná postupnosť stavov počasia nám je skrytá a poznáme len pozorovania od danej osoby, ktoré sú s určitou pravdepodobnosťou pravdivé. Teda jednotlivé stavy nebudú reprezentovať skutočný stav počasia, ale budú generovať (emitovať) pravdepodobnosti s akými bolo v príslušnom stave odpozorované dané pozorovanie. Takto sme rozšírili markovovské reťazce na skryté markovovské modely.
Podľa [1] môžme zadefinovať HMM ako nasledovnú päticu:

$$M = (Q, A, O, B, \pi) \tag{3.2}$$

kde:

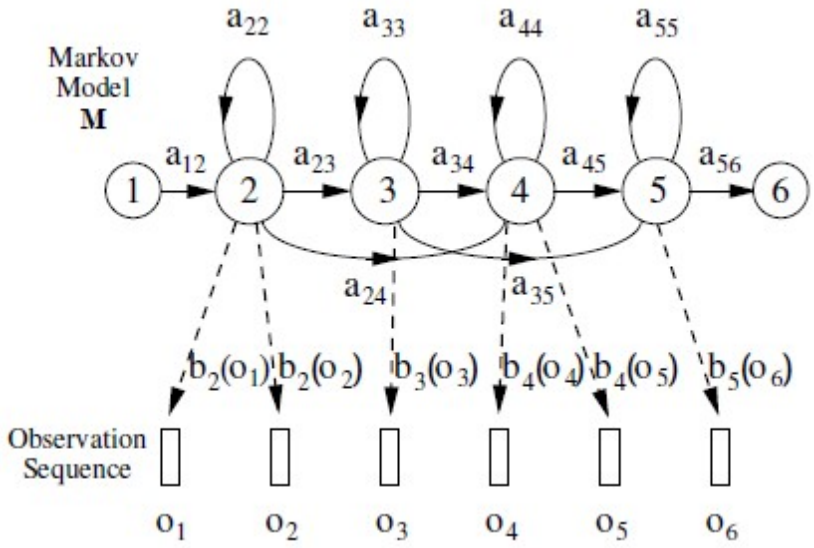
- Q je množina všetkých stavov, $Q = q_1, q_2...q_n$
- A je matica pravdepodobností prechodov, $A = a_{11}...a_{ij}...a_{nn}$, a_{ij} je pravdepodobnosť prechodu zo stavu i do stavu j, z toho vyplýva $\sum_{j=1}^{n} a_{ij} = 1, \forall i$
- O je sekvencia t pozorovaní zo slovníka $V = v_1, v_2...v_v$, $O = o_1, o_2...o_t$
- B je pravdepodobnosť generovania pozorovaní, $B = b_i(o_j)$ je pravdepodobnosť vygenerovania pozorovania o_j v stave i
- π je počiatočné rozdelenie pravdepodobnosti, $\pi = \pi_1, \pi_2...\pi_n$, π_i je pravdepodobnosť, že počiatočný stav bude i, musí platiť, že $\sum_{i=1}^{n} \pi_i = 1$

Z praktických dôvodov sa v nástroji HTK pridáva ešte jeden špeciálny stav na začiatok a na koniec modelu. Tieto stavy negenerujú pozorovania a prechody medzi

nimi sa dejú v nulovom čase. Výhoda je, že umožňujú spájať samostatné HMM do väčších celkov. Z toho vyplýva, že π_i stráca zmysel, lebo sa vždy začne v špeciálnom stave a tieto pravdepodobnosti sa presunú do prechodov medzi pôvodnými stavmi a špeciálnym začiatočným stavom.

Na obrázku 8 môžme vidieť takto upravený model ako prechádza cez sekvenciu stavov $X = 1, 2, 2, 3, 4, 4, 5, 6$ a počíta pravdepodobnosť vygenerovania postupnosti pozorovaní $O = o_1, o_2...o_6$.

$$P(O, X|M) = a_{12}b_2(o_1)a_{22}b_2(o_2)a_{23}b_3(o_3)a_{34}b_4(o_4)a_{44}b_4(o_5)a_{45}b_5(o_6)a_{56} \tag{3.3}$$



Obr. 8: generovanie pozorovaní HMM modelom [10]

Postupnosť stavov nám pri HMM ale nie je známa. Poznáme len pozorovanie O a pravdepodobnosť vygenerovania tohto pozorovania modelom M vypočítame podľa [10] ako sumu pravdepodobností prechodov cez všetky možné sekvencie stavov $X = x(1), x(2), x(3)...x(T)$:

$$P(O|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t)a_{x(t)x(t+1)} \tag{3.4}$$

kde $x(0)$ je špeciálny vstupný stav a $x(T+1)$ je špeciálny koncový stav.

Jednoduchšia možnosť, ktorá sa v praxi používa, je aproximovanie výsledku a to tak, že nájdeme len najpravdepodobnejšiu sekvenciu stavov. Teda takú, ktorej pravdepodobnosť, že bude vygenerovaná O, je najvyššia:

$$\hat{P}(O|M) = \max_X \{a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t)a_{x(t)x(t+1)}\} \tag{3.5}$$

3.1 HMM a rozpoznávanie reči

Rozpoznávanie reči je ukážkový prípad použia HMM. Postupnosť stavov (vyslovené slová, vety) nepoznáme, poznáme len pozorovanie O - audio signál. Vstupom do HMM samozrejme nemôže byť spojitý audio signál, preto sa parametrizuje, ako sme opísali v kapitole 2.
Rozpoznanie slova, môžme definovať ako:

$$arg\max_i\{P(w_i|O)\} \tag{3.6}$$

kde w_i je slovo patriace do rozpoznávaného slovníka. Po použití Bayesovej vety dostaneme:

$$P(w_i|O) = \frac{P(O|w_i)P(w_i)}{P(O)} \tag{3.7}$$

Keďže hľadáme maximum, $P(O)$ môžme z menovateľa odstrániť, lebo bude pre každé w_i rovnaké. Teda hľadáme:

$$arg\max_i\{P(O|w_i)P(w_i)\} \tag{3.8}$$

$P(O|w_i)$ môžme počítať pomocou HMM, ak priradíme ku každému slovu HMM model a budeme ich považovať za ekvivalentné
$P(w_i)$ je pravdepodobnosť výskytu slova v rozpoznávanom jazyku, určuje ju jazykový model.

Podľa vzťahu 3.8, hľadáme maximálnu pravdepodobnosť. Je teda zrejmé, že takto definovaný ASR systém funguje ako klasifikátor. Jeho triedy sú slová zo slovnej zásoby a každú rozpoznávanú sekvenciu pozorovaní priradí do jednej z nich, konkrétne do triedy, ktorej model dosiahol maximálnu pravdepodobnosť.

Keďže sme do modelov pridali špeciálny začiatočný a koncový stav môžme modely ľubovolne spájať. V praxi sa nepoužíva jeden HMM model pre každé slovo, ak rozpoznávame veľkú slovnú zásobu museli by sme natrénovať veľa modelov a aj samotné rozpoznávanie by bolo výpočtovo náročné. Riešenie je natrénovať modely pre fonémy. Foném je najmenšia zvuková jednotka, realizácia hlásky. Je ich obmedzený počet a hľadané slovo získame pospájaním takýchto modelov. Samotná hláska trvá však krátko a chýba jej kontext, preto sa viac osvedčilo modelovať bifóny alebo trifóny, čo sú postupnosti dvoch, resp. troch foném.
Spájanie modelov je však možné aj na úrovni slov. Týmto spôsobom môžme definovať gramatiku jazyka, keď pospájaním modelov slov vytvoríme automat, ktorý umožňuje iba určité sekvencie slov.

3.2 Výstupné rozloženie pravdepodobnosti

Pravdepodobnosť generovania pozorovania o_j v stave i, $b_i(o_j)$ je daná distribúciou pravdepodobnosti. Môže byť použité diskrétne rozdelenie, definované nasledovne:

$$b_j(o_t) = P_j[o_t] \tag{3.9}$$

kde $P_j[o_t]$ je pravdepodobnosť vygenerovania pozorovania o_t v stave j.

Bežne sa ale používa spojité rozloženie pravdepodobnosti, hlavne viacrozmerné Gaussovo, definované vzťahom 3.10 [10].

$$b_j(o_t) = \mathcal{N}(o_t, \mu, C_j) = \frac{1}{\sqrt{(2\pi)^n |C_j|}} \exp\Big\{ -\frac{1}{2}(o_t - \mu)^\top C_j^{-1}(o_t - \mu)\Big\} \tag{3.10}$$

kde μ je vektor stredných hodnôt, n je dimenzia o a C je kovariančná matica.
Dimenzia gausiánu závisí od veľkosti vektora príznakov o. Ak použijme metódu parametrizácie signálu MFCC s deltami a akceleračnými koeficientami v nástroji HTK, tak dostaneme rozmer vektora 39 - 13 MFCC koeficientov, 13 delta koeficientov, 13 akceleračných koeficientov.

Ďalšie vylepšenie je použitie zmesi gausiánov (angl. Gaussian mixture model, GMM) vo výstupnej pravdepodobnosti namiesto jedného. Pri jednom gausiáne predpokladáme, že modelované pozorovanie je symetrické a unimodálne. To samozrejme pri modelovaní reči nie je pravda skoro nikdy a preto sa používa zmes gausiánov, ktorá je schopná modelovať aj asymetrické a viac modálne dáta. Výstupná pravdepodobnosť vyjadrená M- zložkovou zmesou je definovaná nasledovne:

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(o_t, \mu_{jm}, C_{jm}) \tag{3.11}$$

kde c_{jm} je váha m-tého komponentu pre stav j a musí platiť $\sum_{m=1}^{M} c_{jm} = 1, \quad c_{jm} > 0$.

3.3 Inicializácia HMM

Na začiatku treba hodnoty A, B pre HMM nejako odhadnúť. Čím presnejšie sa A a B odhadnú, tým menší čas je potrebný na natrénovanie modelov. Jedna z možností je použiť náhodné parametre a následne ich postupom spomínaným v 3.4 vylepšovať.

Nástroj HTK však používa rozumnejší spôsob odhadu výstupnej pravdepodobnosti B. Trénovacie dáta pre konkrétny model rovnomerne rozdelí medzi jednotlivé stavy a použije ich priemer ako nové hodnoty pre μ_j a C_j. Tento spôsob inicializácie je však

vhodný len pre HMM s ľavo-pravou alebo doprednou topológiou. Výpočet inicializácie parametrov pre vektor príznakov O, dĺžky T:

$$\mu_j = \frac{1}{T}\sum_{t=1}^{T} o_t \tag{3.12}$$

$$C_j = \frac{1}{T}\sum_{t=1}^{T}(o_t - \mu_j)(o_t - \mu_j)^\top \tag{3.13}$$

Potom pomocou Viterbiho algoritmu, ktorý je podrobne opísaný v sekcii 3.5, nájde najpravdepodobnejšiu cestu v modeli, znova rozdelí trénovacie pozorovania a prepočíta hodnoty strednej hodnoty a kovariančnej matice. Tento postup sa opakuje, kým sa nový odhad líší od starého.

3.4 Trénovanie HMM

Najpoužívanejší spôsob vylepšovania parametrov HMM je Baum-Welchov odhad. Princíp je podobný ako pri inicializácii parametrov opísanom v 3.3. Snažíme sa maximalizovať pravdepodobnosť vygenerovania postupnosti pozorovaní O modelom M. Odhad opakujeme, kým platí:

$$P(O|M') \geq P(O|M) \tag{3.14}$$

kde M' označuje nový model s upravenými parametrami.
Vzťah odhadu parametrov je nasledovný:

$$\mu_j = \frac{\sum_{t=1}^{T} L_j(t)o(t)}{\sum_{t=1}^{T} L_j(t)} \tag{3.15}$$

$$\Sigma_j = \frac{\sum_{t=1}^{T} L_j(t)(o_t - \mu_j)(o_t - \mu_j)^\top}{\sum_{t=1}^{T} L_j(t)} \tag{3.16}$$

kde $L_j(t)$ označuje pravdepodobnosť, že sa model nachádza v čase t v stave j pri generovaní pozorovaní O, teda $P(x(t) = j|O, M)$.
Rozdiel oproti vzťahom 3.12 a 3.13 je ten, že jednotlivé pozorovania nie sú priradené ku konkrétnemu stavu, ale prispievajú k odhadu parametrov vo všetkých stavoch v pomere pravdepodobnosti, že sa model bude nachádzať v stave, ktorý upravujeme, keď bolo o_t pozorované.
Pravdepodobnosť $L_j(t)$ sa dá efektívne vypočítať forward-backward algoritmom pre model M s veľkosťou množiny stavov $|Q| = N$ a veľkosťou vektora pozorovaní $|O| = T$.

Doprednýsmer (forward) označíme $\alpha_j(t)$ a je to pravdepodobnosť definovaná nasledovne:

$$\alpha_j(t) = P(o_1, o_2...o_t, x(t) = j|M) \tag{3.17}$$

teda pravdepodobnosť pozorovania prvých t príznakov a skončenia v stave j. $\alpha_j(t)$ sa dá vypočítať rekurzívnym vzorcom [10]:

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_t(t-1)a_{ij}\right] b_j(o_t) \tag{3.18}$$

Počiatočné podmienky pre vzťah 3.18 sú:

$$\alpha_1(1) = 1 \tag{3.19}$$

lebo vždy začíname v špeciálnom začiatočnom stave a:

$$\alpha_j(1) = a_{1j}b_j(o_1) \tag{3.20}$$

pre $1 < j < N$. Záverečná podmienka je definovaná nasledovne:

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN} \tag{3.21}$$

Spätný smer (backward) označíme $\beta_j(t)$ a je to pravdepodobnosť:

$$\beta_j(t) = P(o_{t+1}, o_{t+2}...o_T|x(t) = j, M) \tag{3.22}$$

Vzorec na výpočet je rekurzívny ako v doprednej časti [10]:

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij}b_j(o_{t+1})\beta_j(t+1) \tag{3.23}$$

Počiatočná podmienka je:

$$\beta_i(T) = a_{iN} \tag{3.24}$$

pre $1 < i < N$. Záverečná podmienka je:

$$\beta_1(1) = \sum_{j=2}^{N-1} a_{1j}b_j(o_1)\beta_j(1) \tag{3.25}$$

Keď máme vypočítané oba smery α_j a β_j, môžme ich použiť na výpočet $L_j(t)$ nasledovne [10]:

$$\begin{aligned} L_j(t) &= P(x(t) = j|O, M) \\ &= \frac{P(O, x(t) = j|M)}{P(O|M)} \\ &= \frac{\alpha_j(t)\beta_j(t)}{\sum_{i=2}^{N-1} \alpha_i(t)\beta_i(t)} \end{aligned} \tag{3.26}$$

Ukázali sme si ako sa aktualizuje parameter B. Aby bolo trénovanie modelu kompletné, treba nám ešte aktualizovať parameter A. Použijeme už uvedené pravdepodobnosti $\alpha_j(t)$, $\beta_i(t)$, $L_j(t)$ a zadefinujeme novú pravdepodobnosť $L'_{ij}(t)$, ktorá značí pravdepodobnosť, že model bude v stave i v čase t a v stave j v čase $t+1$:

$$L'_{ij}(t) = P(x(t) = i, x(t+1) = j|O, M) \tag{3.27}$$

Vypočítame ju nasledovne:

$$L'_{ij}(t) = \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(o_{t+1})}{\sum_{i=2}^{N-1}\sum_{j=1}^{N-1}\alpha_i(t)a_{ij}\beta_j(t+1)b_j(o_{t+1})} \tag{3.28}$$

Potom nám stačí aktualizovať A podľa vzťahu [21]:

$$a_{ij} = \frac{\sum_{t=1}^{T-1} L_{ij}(t)}{\sum_{t=1}^{T-1} L'_{ij}(t)} \tag{3.29}$$

Uvedené vzťahy platia pre trénovanie HMM modelu pre jedno pozorovanie O. Keď ich máme viacero, môžeme upravovať parameter postupne pre každé pozorovanie. Nežiaduci následok tohto postupu je adaptovanie modelu na posledné trénované dáta. Lepšia možnosť je trénovať model naraz pre všetky pozorovania [20]. Predchádzajúce vzťahy je ale treba rozšíriť pre viacero pozorovaní $O_1...O_n$. Viac informácií o tomto prípade sa môžeme dozvedieť v práci [20].

3.5 Viterbiho algoritmus

Ako sme si už spomenuli v sekcii 3.1, HMM v rozpoznávaní reči hrajú úlohu určovania pravdepodobnosti $P(O|w_i)$, ak ku každému slovu priradíme jeden HMM model a budeme ich považovať za ekvivalenté. Teda $P(O|w_i) = P(O|M)$, kde M je HMM model pre slovo w_i. Taktiež sme si spomínali, že $P(O|M)$ sa počíta vzťahom 3.4, čo je ale výpočtovo náročné, preto sa výsledok aproximuje hľadaním najpravdepodobnejšej cesty v modeli pre dané pozorovanie O. Viterbiho algoritmus nám slúži presne na túto úlohu. Je definovaný rekurzívne a efektívne sa dá implementovať pomocou dynamického programovania.

Nech $v_j(t)$ je pravdepodobnosť najpravdepodobnejšej cesty, ktorá sa nachádza v čase t v stave j. Vypočítame ju nasledovne pre model M a N stavmi:

$$v_j(t) = \max_i^N\{v_i(t-1)a_{ij}b_j(o_t)\} \tag{3.30}$$

$$v_1(1) = 1 \tag{3.31}$$

$$v_j(1) = a_{1j} b_j(o_1) \tag{3.32}$$

pre $1 < j < N$. Výsledná pravdepodobnosť bude:

$$\hat{P}(O|M) = v_N(T) = \max_i^N \{v_i(T) a_{iN}\} \tag{3.33}$$

Keďže produkt násobenia pravdepodobností bude veľmi malé číslo, ktoré môže podtiecť, používa sa logaritmický súčet namiesto súčinu, ktorý zachová pomer a navyše je výpočtovo jednoduchší. Namiesto výpočtu 3.30 dostaneme:

$$v'_j(t) = \max_i^N \{v'_i(t-1) + \log(a_{ij}) + \log(b_j(o_t))\} \tag{3.34}$$

Ak si okrem maximálnej pravdepodobnosti budeme v rekurzii pamätať aj stav, ktorý ju dosiahol, čiže:

$$arg \max_i^N \{v'_i(t-1) + \log(a_{ij}) + \log(b_j(o_t))\} \tag{3.35}$$

budeme vedieť spätne zrekonštruovať najpravdepodobnejšiu postupnosť stavov.

4. HTK

Z predchádzajúcej kapitoly vidíme, že skryté Markovovské modely sú pomerne zložitý koncept a správna a efektívna implementácia by nebola triviálna. Preto sa vyvinuli nástroje, ktoré nám prácu s nimi uľahčujú. V tejto kapitole si povieme niečo o nástroji HTK [2] (z angl. hidden Markov model toolkit)a uvedieme si aj iné systémy umožňujúce prácu s HMM.

4.1 HTK toolkit

HTK je nástroj vyvinutý na Cambridge University. Primárne je určený na výskum oblasti rozpoznávania reči, ale použitie môže nájsť aj v iných oblastiach používajúcich HMM, napríklad rozpoznávanie znakov alebo DNA sekvenovanie. Poskytuje viacero nástrojov na spracovanie signálu, inicializáciu HMM modelov, ich tréning a analýzu výsledkov. Takisto je k HTK vydaná podrobná dokumentácia aj s príkladmi použitia [2]. Uvedieme si základné a najpoužívanejšie nástroje v rôznych fázach vývoja ASR systému pomocou HMM.

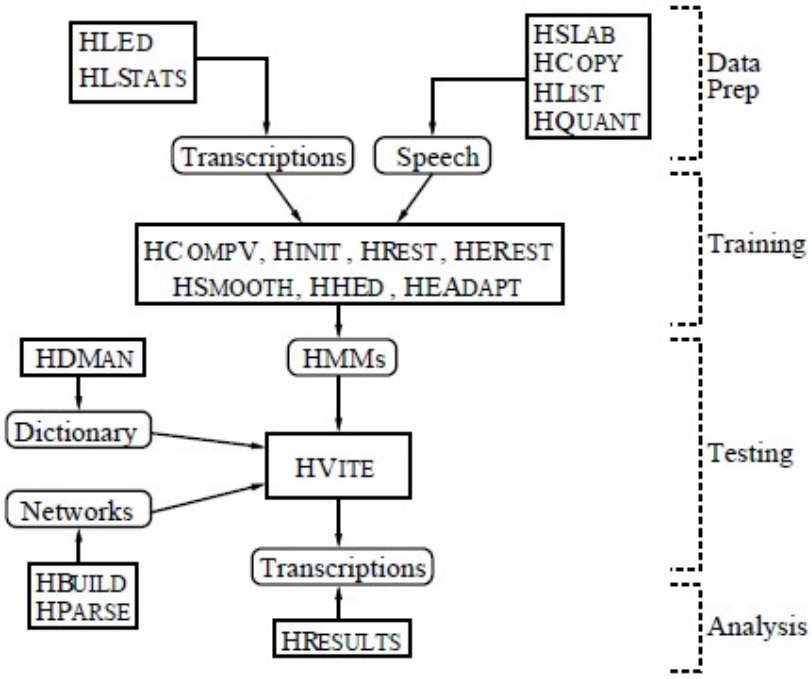
Príprava dát: na začiatku treba pripraviť trénovaciu a testovaciu množinu nahrávok spolu s ich transkripciou - HSLAB. Na manipuláciu s nahrávkami ako je ich spájanie, delenie a ich parametrizáciu slúži HCopy. Na konvertovanie a upravovanie súborov s transkripciou reči slúži HLed.

Trénovanie: Po vytvorení vhodnej HMM architektúry a vytvorení prototypov modelov (ručne editovateľné textové súbory) ich môžme inicializovať. Na inicializovanie a prvotný odhad parametrov pomocou Viterbiho algoritmu má systém HTK nástroj HInit. Potom môžme použiť HRest alebo HERest, ktoré používajú Baum-Welch algoritmus na úpravu odhadov parametrov modelu. HHEd slúži na úpravu jednotlivých HMM, napríklad pridanie gausiánov do jednotlivých stavov. Následne treba celý model pretrénovať znova, už uvedenými nástrojmi.

Rozpoznávanie: utilita HVite realizuje samotné rozpoznávanie. Vstupom je súbor definujúci gramatiku jazyka (aké sekvencie slov sú dovolené), sieť HMM, slovník výslovnosti slov a neznámy audio súbor. Za pomoci Viterbiho algoritmu sa zistiťuje prepis slova, ktorý uloží do súboru. Alternatívou k HVite je HDecode, ktorý sa hodí pre väčšie

slovníky.

Analýza: Posledný krok je zistenie úspešnosti rozpoznávania pomocou HResults. Tento nástroj porovná výstup z HVite a správny prepis audia za pomoci dynamického programovania. Spočíta chyby nahradenia, odstránenia a vloženia. Na obrázku 9 môžme vidieť spomínané fázy a nástroje poskytované HTK toolkitom.



Obr. 9: Rôzne nástroje poskytované HTK toolkitom podľa [10]

4.2 Podobné nástroje

Hoci je HTK už pomerne starý nástroj, neustále zlepšenia (posledná nová verzia je z roku 2015), podrobná dokumentácia a veľa voľne dostupných materiálov robia z neho konkurencie schopný softvér.

O niečo novší je systém Kaldi [12]. Oproti HTK má výhodu voľnejšej licencie a dá sa viac upraviť podľa potrieb. Na druhej strane mu chýba podrobná a zrozumiteľná dokumentácia akou disponuje HTK. Podrobnejšej analýze týchto dvoch nástrojov sa venovala táto práca [9]. Na vykonanom teste dosiahli porovnateľné výsledky, Kaldi o niečo lepšie.

Posledným nástrojom, ktorý spomenieme je CMUSPhinx [13]. Jeho výhodou je podpora jazykov Java a Python, čo ho robí použiteľným v mnohých prostriediach. Taktiež ponúka odľahčenú verziu Pocketsphinx, ktorá beží aj na operačnom systéme Android, teda je použiteľná v mobilných aplikáciách. Je však viac kompaktný a neponúka také

podrobné možnosti stavby modelov a ich trénovania ako HTK alebo Kaldi.

4.3 Inštalácia HTK

Návod na inštaláciu HTK na operačných systémoch Linux aj Windows, je dostupný online [2]. Inštalácia by mala byť pomerne priamočiara, autor práce však mal s ňou problémy na linuxovej distribúcii Debian, preto sa rozhodol uviesť pár bodov, ktoré odstránia prípadné problémy.

Nástroje HTK sú 32-bitové aplikácie, čo spôsobuje problémy s grafickými knižnicami na 64-bitových systémoch. Jediný spôsob, ktorý sa osvedčil bolo vynechať z kompilácie nástroj HSlab, ktorý jediný používa tieto knižnice. Skript **configure** bolo treba spustiť nasledovne:

```
$ configure --without-x --disable-hslab
```

HSlab slúži na nahrávanie reči a tvorbu súborov s transkripciou reči na trénovanie. Alternatíva k nemu je napríklad program Wavesurfer [14] alebo Audacity [15] .

Druhou chybou, ktorá sa vyskytla po spustení príkazu **make install** bolo nesprávne odsadenie v skripte MakeFile. V adresári **htk/HLMTools** ho treba upraviť. Na riadku 77 vymažeme odsadenie tvorené medzerami a nahradíme ich tabelátorom.

5. Návrh

Hlavným cieľom našej práce bolo navrhnúť a implementovať funkčný systém na rozpoznávanie reči, týkajúci sa opisu objektov na scéne v angličtine. Rozpoznávač by mal slúžiť ako hlasové ovládanie systému robotickej ruky a kamery, ktorý rozpoznáva objekty na scéne, vie určiť ich polohu a následne s nimi hýbať. Keďže náš výsledný systém na rozpoznávanie povelov ruke bude reálne nasadený do prevádzky, musíme dosiahnuť úspešnosť blížiacu sa k 100%. Budeme sa snažiť zvýšiť úspešnosť rozpoznávania aj za cenu závislosti na rečníkoch a obmedzení používania na konkrétny hardvér.

Úloh, ktoré musíme vyriešiť je niekoľko:

- Vytvorenie dátovej množiny hlasových nahrávok podľa definovanej slovnej zásoby, potrebných na trénovanie a testovanie.
- Návrh skrytých Markovovských modelov pre účely rozpoznávania reči v nástroji HTK a ich natrénovanie.
- Detekcia reči a ticha zo vstupného signálu. Extrakcia intervalov, kde používateľ rozpráva.
- Vytvorenenie intuitívneho používateľského rozhrania pre interakciu s celým systémom.

5.1 Slovná zásoba

Ako sme už spomínali, rozpoznávať budeme anglické slová. Celá slovná zásoba sa skladá z 19 slov, z ktorých je povolené robiť určité typy vetných konštrukcií. Rozpoznávať budeme tieto slová:
farby: *red, blue, green, yellow*
slovesá: *put, grasp, move, give-me, is*
objekty: *cube, block, cylinder*
príslovky: *left, right, middle, where*
predložky: *in, on*
členy[1]: *the*

[1] v slovenskom jazyku takýto slovný druh neexistuje, v anglickom je člen the slovný druh determiner

Dovolené typy viet môžme pomocou logických operátorov **&** a |, ktoré znamenajú **a** resp. **alebo** a premenných definovať nasledovne:

```
$color = red | blue | green | yellow;
$verb = put | grasp | move | give-me;
$object = cube | block | cylinder;
$adv = left | right;

(($verb $color $object) |

($color $object is ((in the middle) | (on the $adv))) |

(where is $color $object))
```

5.2 Dátová množina

Zostavenie správnej množiny trénovacích dát je veľmi dôležité pre správne fungovanie systému. HMM modely sa prispôsobia dátam rečníkov, na ktorých boli trénované a teda budú ich hlasy rozpoznávať s vyššou úspešnosťou. Dátová množina by však mala byť dostatočne veľká a rôznorodá, teda aby obsahovala čo najviac rôznych variantov vyslovení slova. Často práve od veľkosti a rôznorodosti závisí či je výsledný systém závislý na rečníkovi alebo nie.

Pri dátach, ktoré sú potrebné na trénovanie aj testovanie platí pravidlo: čím viac tým lepšie. Samozrejme dáta treba ale následne spracovať, priradiť k ním príslušné transkripcie potrebné na trénovanie, čo je časovo náročné. Preto sa často používajú veľké databázy zvukových nahrávok (angl. speech corpuses) s príslušným textovým prekladom. My sme sa však nerozhodli ísť týmto smerom, keďže takéto databázy obsahujú nahrávky rôznych kvalít, na rôznych mikrofónoch, obsahujú rôzny pomer šumu a reči. Toto je výhoda pri trénovaní všeobecných rozpoznávačov, určených pre veľa používateľov, veľa prostredí. Tým, že náš systém bude používaný v konkrétnych podmienkach, vieme tieto podmineky namodelovať aj pri nahrávaní dátovej množiny a tým maximalizovať úspešnosť rozpoznávania. Samozrejme pri zmene niektorého z našich predpokladov, napríklad pri použití vstavaného mikrofónu z laptopu namiesto headsetovej súpravy, bude náš systém nepoužiteľný.

5.3 Návrh HMM

HMM musíme navrhnúť tak, aby čo najlepšie modelovali slová, resp. časti slov, ktoré k nim prislúchajú. Pri návrhu skrytých Markovovských modelov záleží na na nasledu-

júcich parametroch:

- topológia - počet stavov v modeli, aké prechody sú dovolené medzi stavmi
- rozloženie výstupnej pravdepodobnosti - môžme zvoliť diskrétne rozdelenie alebo spojité, definované jedným gausiánom, alebo ich zmesou
- spájanie modelov - HMM môžme spájať do väčších celkov (slová, vety), je potrebné definovať pravidlá ich spájania

Následne je potrebné vybrať správne metódy inicializácie a trénovania modelov, ktoré maximalizujú pravdepodobnosť modelu pre prislúchajúce slovo.

5.4 Detekcia hlasu

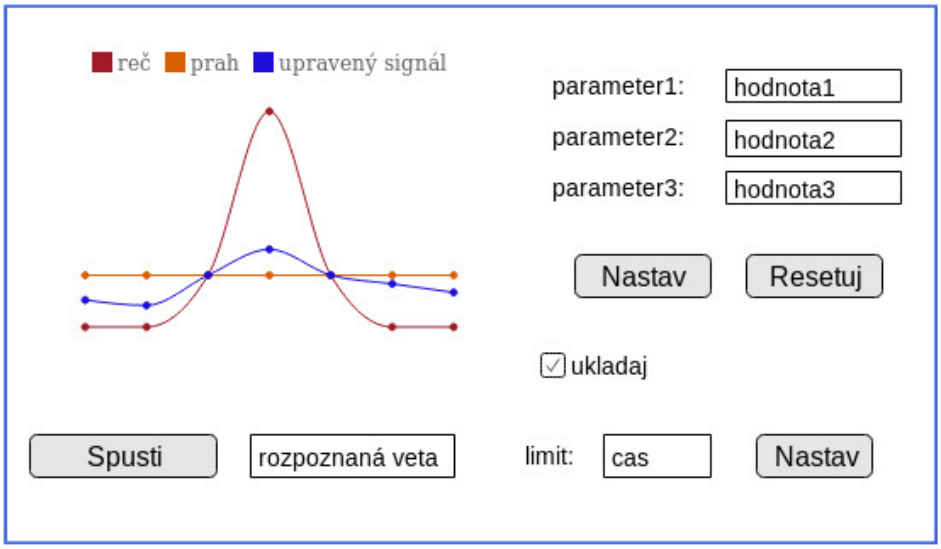
Na použitie systému koncovým používateľom je potrebné zaoberať sa aj detekciu reči. Musíme vedieť, kedy používateľ rozpráva a kedy je signál tvorený len šumom, prípadne vzdialenými hlasmi alebo inými zvukmi. Tento problém má skratku VAD (z angl. voice activity detection). VAD systémy môžu pracovať v reálnom čase alebo analyzovať už ukončenú nahrávku, čo je jednoduchšie. Nás bude samozrejme zaujímať prvá možnosť. VAD systém pracujúci v reálnom čase nemôže byť výpočtovo náročný, ale musí byť dostatočne presný.

5.5 Špecifikácia požiadaviek na výsledný systém

Nasledujúce body definujú aké požiadavky bude náš softvér spĺňať:

- Spustenie/zastavenie rozpoznávanie viet v reálnom čase. Systém sám vyextrahuje časť signálu s rečou dostatočnej dĺžky a rozpozná ju.
- Používateľ bude môcť zrušiť odoslanie rozpoznaného povelu robotickej ruke do časového limitu. Časový limit bude možné upravovať.
- Možnosť ukladať rozpoznané povely aj so zvukovým súborom.
- Možnosť upraviť parametre detekcie reči.

5.6 Používateľské rozhranie



Obr. 10: Návrh používateľského prostredia

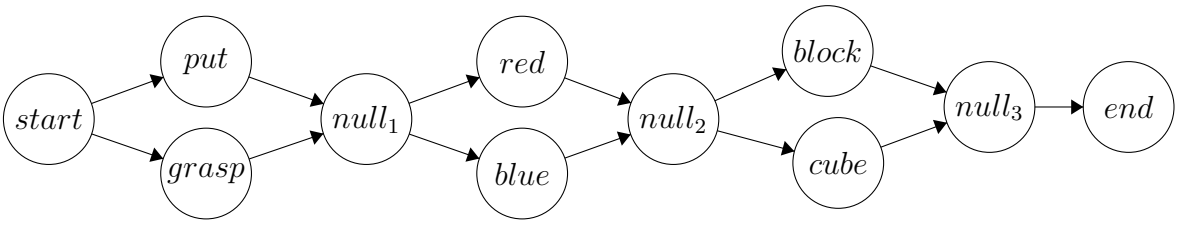
Na obrázku 10 je zobrazený návrh používateľského prostredia nášho systému. Obsahuje zobrazenie signálu upraveného pre potreby detekcie reči, prah, ktorý oddeľuje reč od signálu a krivku, ktorá oddeľuje, ktorá časť signálu bola označená ako povel. Bude slúžiť na spätnú väzbu používateľovi, aby vedel akou intenzitou má rozprávať, prípadne ako upraviť parametre detekcie reči, ak ich systém nedetekuje správne. Na pravej strane sa nachádzajú vstupné polia na zobrazenie a úpravu parametrov, spolu s tlačidlami na ich nastavenie a resetovanie. Pod nimi je zaškrtávacie pole na potvrdenie alebo zrušenie ukladania nahrávok. Spodný panel obsahuje tlačidlo na spustenie systému, úsek na zobrazenie rozpoznaného povelu a nástroje na úpravu časového limitu.

6. Implementácia a experimenty

V tejto kapitole sa budeme venovať implementácii systému, použitým postupom, technológiám a experimentom.

6.1 Gramatika jazyka

V sekcii 5.1 sme si pomocou logických operátorov a premenných definovali povolenú gramatiku viet. Z takto definovanej gramatiky vieme pomocou nástroja HParse vygenerovať sieť v SLF formáte (angl. standard laticce format), ktorá povoľuje prechody iba správnym typom viet. Keďže sieť vygenerovaná podľa nami definovaných pravidiel by mala až 43 stavov a 64 hrán ukážeme si vizualizáciu siete pre prvý typ vety: *($verb $color $object)* a všetky premenné obmedzíme iba na prvé dve hodnoty.



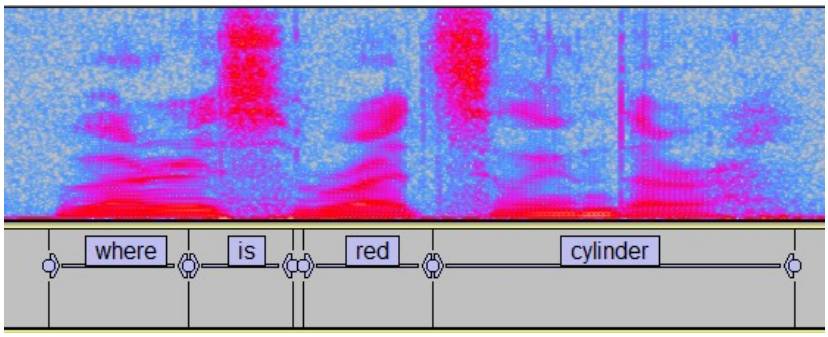
Obr. 11: Sieť pre gramatiku vety

Môžme si všimnúť, že boli vygenerované aj stavy, ktoré nereprezentujú žiadne HMM: $null_x$. Slúžia na optimalizáciu siete, konkrétne na zníženie počtu hrán. V tomto zjednodušenom príklade nám neušetrili žiadne hrany, dokonca jednu pridali, ale keby sme neobmedzili hodnoty premenných len na prvé dve, tak by bola optimalizácia viditeľná.

Na takto realizovanej sieti, kde každý stav je HMM model pre konkrétne slovo vieme realizovať Viterbiho algoritmus spomínaný v sekcii 3.5. Ak používame rozpoznávanie pomocou trifónov, tak nám pribudne ešte jedna úroveň siete: slová → trifóny → HMM modely.

6.2 Dátová množina

Na trénovanie sme zostavili množinu 54 viet, ktoré patria do našej gramatiky a snažili sme sa aby slová boli rovnomerne zastúpené a aby sa vyskytovali v rôznych kontextoch. Dokopy sme nahrali 13 minút a 26 sekúnd trénovacích nahrávok od 6 rôznych rečníkov. Získali sme tak 324 viet (1368 slov). Testovací scenár tvorilo 22 viet, ktoré sa nahrávali samostatne od trénovacích. Tie sme získali aj od 4 nových rečníkov, na ktorých nebol systém trénovaný. Nakoniec sme nahrali ešte testovací scenár od troch ženských rečníčok, ten však budeme posudzovať samostatne. Tieto testovacie dátové množiny si označíme nasledovne:
DM_z - testovacie nahrávky 6 známych rečníkov, 132 viet
DM_n - testovacie nahrávky 4 neznámych rečníkov, 88 viet
DM_{zn} - $DM_z \bigcup DM_n$, 220 viet (880 slov)
DM_d - testovacie nahrávky od 3 ženských rečníčok, 66 viet
K nahrávkam určeným na trénovanie bolo treba pridať časový preklad, tzn. presne označiť intervaly slov s ich transkripciou. Je to zdĺhavá práca a často je ťažké určiť presné hranice slov. Na obrázku 12 môžme vidieť spektogram[1] nahrávky a k nej priradený prepis v programe Audacity [15]. Následne treba ešte previesť prepis do HTK formátu, čo sme vyriešili jednoduchým python skriptom.

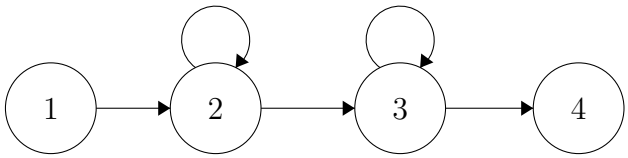


Obr. 12: Označovanie slov v trénovacích nahrávkach

6.3 HMM modely

Napriek tomu, že modely celých slov nie sú veľmi časté v praxi, my sme sa ich rozhodli použiť. Môžme si to dovoliť vzhľadom k malej a uzavretej slovnej zásobe. Očakávame vyššiu úspešnosť rozpoznávača, keďže bude o jednu úroveň spájania modelov menej.
Použili sme doprednú topológiu, teda prechod v HMM je možný len do najbližšieho nasledujúceho stavu alebo do aktuálneho stavu. Samozrejme okrem špeciálneho začiatočného a koncového stavu, ako môžme vidieť na obrázku 13.

[1] zobrazuje zmenu frekvencií v čase



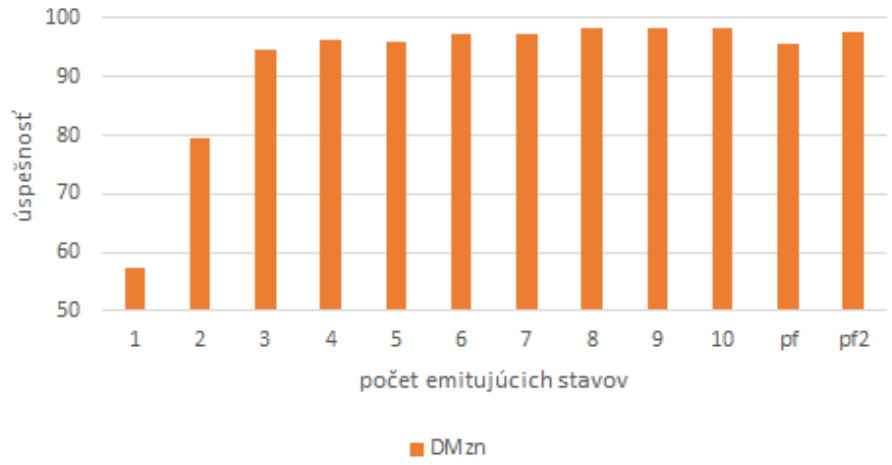
Obr. 13: Dopredná topológia HMM s 2 emitujúcimi stavmi

V HTK definujeme modely pomocou konfiguračných súborov, kde určíme počet stavov, definujeme stredné hodnoty a rozptyl gausiánov v jednotlivých stavoch. Takisto definujeme maticu s pravdepodobnosťami prechodov medzi stavmi a ňou určíme aj topológiu modelu. Príklad definovania gausiánu v stave a prechodovej matice pre dopredný model s 5 stavmi v konfiguračnom súbore:

```
<State> 2
   <Mean> 39
      0.0  0.0  0.0  ...
   <Variance> 39
      1.0  1.0  1.0  ...
...
 <TransP> 5
0.0  1.0  0.0  0.0  0.0
0.0  0.6  0.4  0.0  0.0
0.0  0.0  0.6  0.4  0.0
0.0  0.0  0.0  0.7  0.3
0.0  0.0  0.0  0.0  0.0
```

Pri inicializácii a trénovaní sme použili nástroje HVite a HRest. Princíp inicializácie pomocou HVite je vysvetlený v sekcii 3.3. Zo všetkých nahrávok určených na trénovanie sme vybrali úseky, ktoré boli označené prekladom slova, ktorého model sme práve trénovali a tie sa použili na inicializáciu modelov. Následne sme aplikovali Baum-Welchov algoritmus pomocou HRest-u, ktorý odhadol finálne parametre modelov. Podrobný popis toho procesu je popísaný v sekcii 3.4. Vyskúšali sme aj viacnásobné použitie Baum-Welch algoritmu, ktoré bolo použité v práci [20], ale dosiahli sme rovnaké výsledky ako pri jednom použití. Ako metódu parametrizácie signálu sme zvolili MFCC koeficienty, spolu s energiou segmentu, delta a akceleračnými koeficientami.

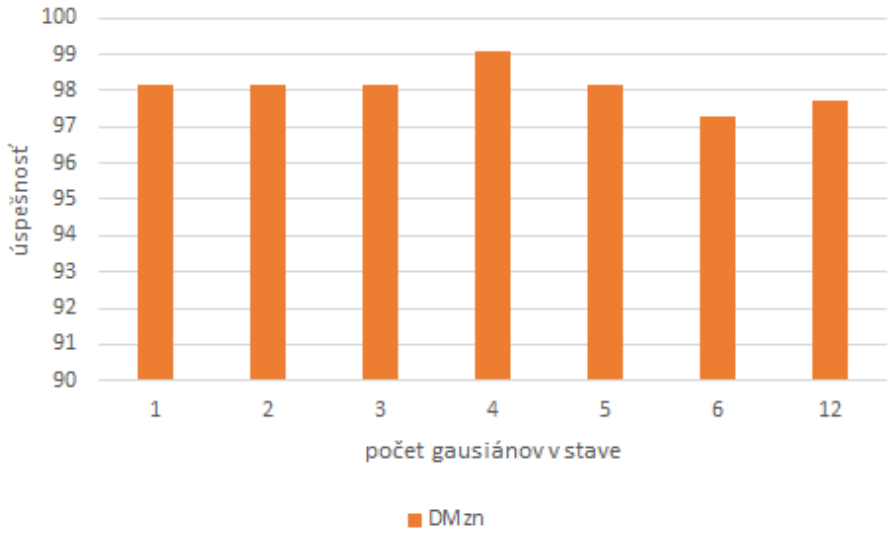
Dôležité je zvoliť správny počet stavov HMM, na grafe 14 je zobrazená úspešnosť nášho rozpoznávania pri rôznych počtoch emitujúcich stavov. pf a $pf2$ označujú modely, kde sa počet stavov rovná počtu foném v slove, resp. kde je počet stavov dvojnásobný oproti počtu foném. Takáto topológia je logická, keďže dlhšie slovo by malo byť lepšie modelované viacerými stavmi. Udávané percentá sú úspešnosti rozpoznania viet, to znamená, že ak jedno slovo vo vete je rozpoznané nesprávne, celá veta je označená ako nesprávna aj keď zvyšné slová boli správne.



Obr. 14: Úspešnosť rozpoznávania v závislosti od počtu emitujúcich stavov

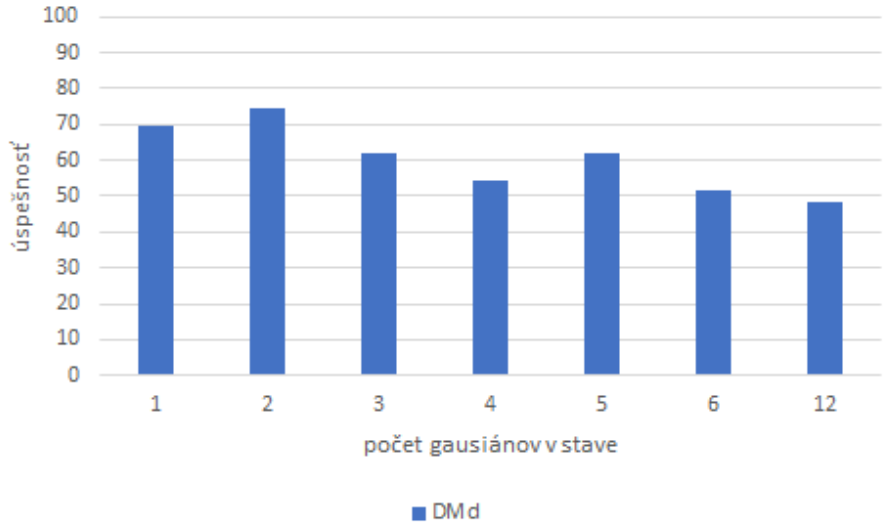
Najvyššia dosiahnutá úspešnosť bola 98.18% pri počte stavov 8, 9, 10. Úspešnosť rozpoznania slov bola v tomto prípade až 99.55%. Ďalej budeme používať najúspešnejší model s najmenším počtom stavov, teda model s fixným počtom stavov 8 pre každé slovo. Z toho, že modely s počtom stavov založeným na počte foném slova neboli úspešnejšie môžme usúdiť, že HMM nevadí vyšší počet stavov (napríklad 8 stavov pre krátke slovo "in"), dokonca dosahujú vyššiu úspešnosť. Samozrejme je to výpočtovo náročnejšie, to by sa však prejavilo až pri veľkej slovnej zásobe.

Ako sme si spomínali v sekcii 3.2, tak na výstupné rozloženie pravdepodobnosti sa používa Gaussovo rozdelenie s rozšírením na zmesi gausiánov. Na nasledujúcom grafe vidíme zmenu úspešnosti rozpoznávania viet vzhľadom k počtu gausiánov v stavoch.



Obr. 15: Úspešnosť rozpoznávania v závislosti od počtu gausiánov v stave

Vidíme, že pri zvýšení na štyri gausiány v stave, nám úspešnosť stúpla na 99.09%, čo je najlepší výsledok, ktorý sa nám podarilo dosiahnuť na testovacej množine DM_{zn}. Komplikáciou je, že dosahujeme vysokú úspešnosť a zmena počtu gausiánov sa nemá veľmi kde prejaviť. Vyšší počet gausiánov však znamená, že HMM budú lepšie modelovať dáta, na ktorých boli natrénované, ale zároveň sa môžu vzďaľovať neznámym dátam - rečníkom, na ktorých nebol systém trénovaný. Tento efekt môžme vidieť na nasledujúcom grafe. Zobrazuje rovnakú štatistiku ako grafe 15, ale pre dátovú množinu ženských rečníčok. Systém bol trénovaný iba na mužských hlasoch.



Obr. 16: Úspešnosť rozpoznávania v závislosti od počtu gausiánov v stave pre ženské rečníčky

Vidíme, že pri zvýšení na dva gausiány v stave nám ešte úspešnosť stúpla, ďalej však bola už len nižšia. Ako najúspešnejší volíme teda model s 2 gausiánmi v stave. Síce dosiahol nižšie percentá pre mužské hlasy ako model so štyrmi, ale úspešnosť rozpoznávania ženských hlasov bola až o 20% vyššia. To sa môže hodiť v situácii ak náš systém bude chcieť používať napríklad mužský rečník s vyšším hlasom.

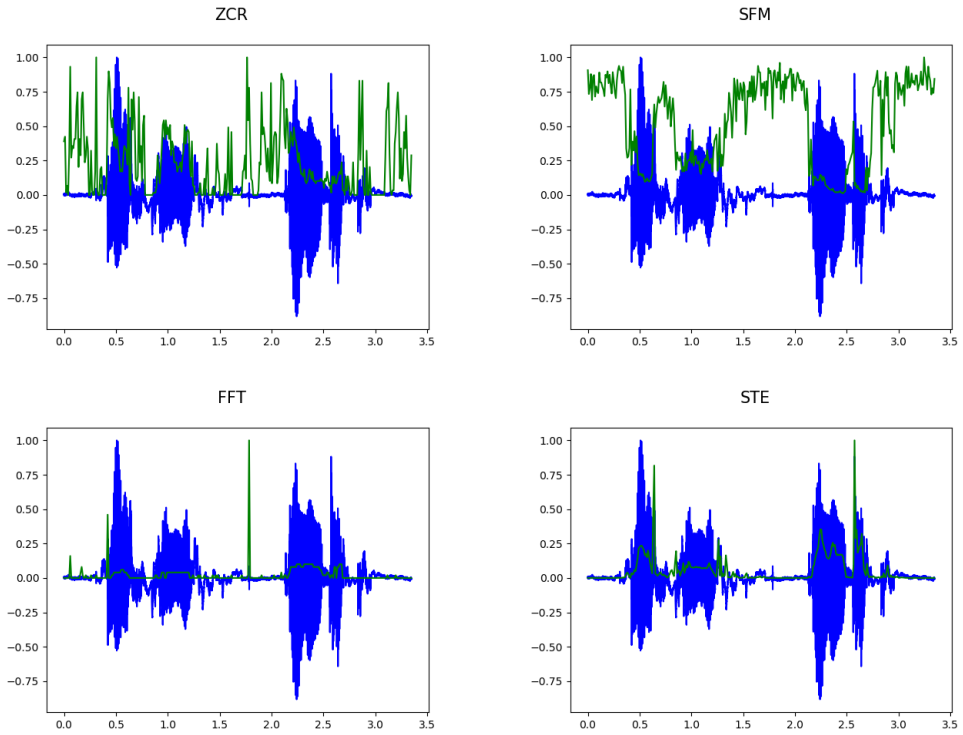
Proces inicializácie, trénovania a upravovania parametrov modelu pomocou nástojov HTK, sme automatizovali bash skriptami, keďže systém sme vyvíjali na operačnom systéme Linux.

6.4 Detekcia reči

Nástroj HTK ponúka vlastnú implementáciu VAD systému priamo prepojenú s nástrojmi na rozpoznávanie. Naše experimenty však ukázali veľké nedostatky. Takmer vždy bol odseknutý začiatok reči o niekoľko desatín sekundy, čo robí systém prakticky nepoužiteľný. Taktiež mu chýbajú funkcionality napríklad na uloženie častí nahrávky s

rečou, čo je dôležité pre testovanie. Keďže správne fungujúca detekcia reči je pre reálne použitie rozpoznávača reči absolútne kľúčová, rozhodli sme sa naprogramovať vlastný systém.

VAD systém pracujúci v reálnom čase nemôže byť výpočtovo náročný, ale musí byť dostatočne presný. Vyskúšali sme metódy analýzy signálu spomenuté v sekcii 2.4. Ich porovnanie na rovnakej zvukovej nahrávke je zobrazené na obrázku 17.



Obr. 17: Rôzne druhy analýzy signálu

Pri našich experimentoch sa osvedčili hlavne analýzy SFM a STE. Nakoniec sme sa rozhodli použiť práve krátkodobú energiu, ktorá je nielen najmenej náročná, ale najlepšie zachytávala okraje slov, ktoré splývali s okolitým šumom.

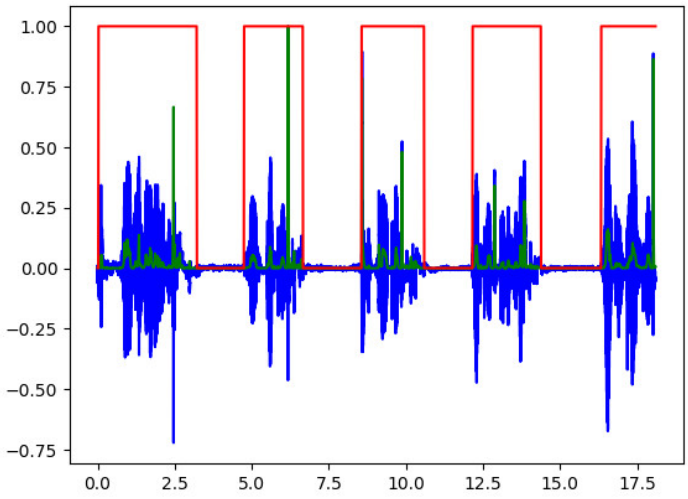
6.4.1 VAD algoritmus

Pri návrhu algoritmu na detekciu reči sme sa inšpirovali prácou [25] a ako hlavné kritérium sme použili krátkodobú energiu. Pseudokód hlavnej časti procesu je nasledovný:

VAD

rámce ← *vypočítaj rámce dĺžky 10ms zo vstupného signálu*
prah ← *úvodný STE prah*
šum ← *z prvých 30 rámcov vypočítaj priemernú hodnotu STE*
prah ← *prah* * log_{10}(*šum*)
intervaly ← *pole trojíc (stav, trvanie, signál), označuje súvislé intervaly reči/ticha*
for all *rámce* **do**
  ste ← *vypočítaj hodnotu STE*
  if *(ste - šum) > prah* **then**
    označ rámec ako reč
  else
    označ rámec ako ticho
    šum ← *0.8 * šum + 0.2 * ste*
  end if
  intervaly[n] ← *aktualizuj (stav, trvanie, signál)*
    ← *ignoruj ticho kratšie ako 10 rámcov za sebou*
    ← *ignoruj reč kratšiu ako 5 rámcov za sebou*
  if (*intervaly[n].stav = ticho* **and** *intervaly[n].trvanie > maximálna dĺžka ticha* **and**
  intervaly[n-1].trvanie = reč **and** *intervaly[n-1].trvanie > minimálna dĺžka reči*)
  then
    rozšír intervaly[n-1] o rámce s rečou na okrajoch
    ulož a rozpoznaj intervaly[n-1].signál
  end if
end for=0

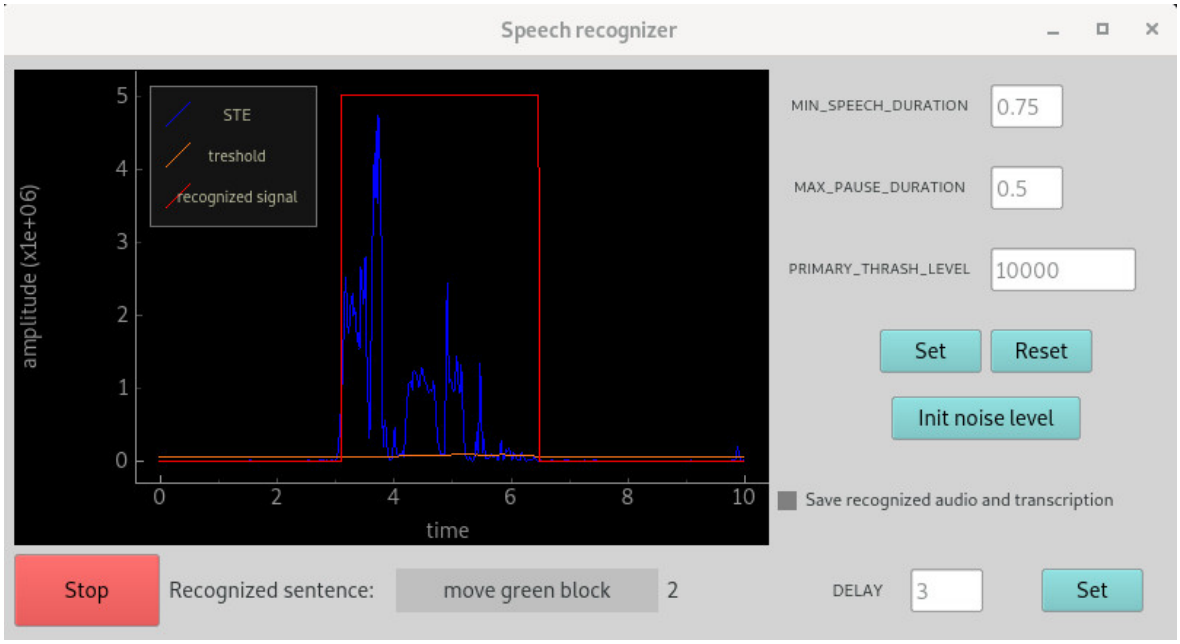
Ukážku práce opísaného algoritmu môžme vidieť na obrázku 18.



Obr. 18: Oddelenie častí signálu pre rozpoznávanie

Algoritmus sme implementovali v Pythone 3. Na prácu s audio signálom sme použili moduly pyaudio, wave a numpy.

6.5 Používateľské prostredie



Obr. 19: Používateľské rozhranie systému

Obrázok 19 zobrazuje používateľské rozhranie našej aplikácie. Na vývoj grafického prostredia sme sa rozhodli použiť modul PyQt4. Rieši prepojenie Pythonu 3 s multiplatformovou knižnicou Qt. Zobrazená je ukážka stavu, keď systém označil časť signálu ako reč a rozpoznal ju. Vedľa rozpoznanej vety beží časovač, do konca ktorého je možné zrušiť klávesou "medzera" odoslanie povelu do cieľového zariadenia. Blok napravo obsahuje nastavenia pre úpravu detekcie reči.

MIN_SPEECH_DURATION je minimálna dĺžka vety, ktorú sa systém pokúsi rozpoznať.

MAX_PAUSE_DURATION označuje maximálnu dĺžku ticha, ktorú môže používateľ spraviť medzi slovami v jednom poveli. Pri dlhšej pauze to systém vyhodnotí ako dve samostatné časti.

PRIMARY_THRASH_LEVEL označuje experimentálnu hodnotu prahu. V algoritme VAD je táto hodnota uvedená ako *úvodný STE prah*. Je možné tieto hodnoty upraviť a následne nastaviť, prípadne resetovať na pôvodné. Posledný prvok je tlačidlo na znovu vypočítanie šumu, ktorý bol pôvodne inicializovaný z prvých 30 rámcov (300ms).

Záver

V našej práci sme sa venovali rozpoznávaniu reči pomocou skrytých Markovovských modelov. Popísali sme teóriu potrebnú na ich inicializáciu, trénovanie a samotné rozpoznávanie. Pomocou nástroja HTK sme navrhli systém na rozpoznávanie niekoľkých typov anglických viet - povelov, zložených z 19 slov. Nahrali a spracovali sme vlastnú dátovú množinu zvukových nahrávok, na ktorých sme systém natrénovali a otestovali. Ten sme následne implementovali do výslednej aplikácie, ktorá slúži ako hlasové ovládanie robotického zariadenia. Kvôli zvýšeniu úspešnosti výsledného softvéru, sme sa rozhodli systém trénovať len na mužských rečníkoch.

Použili sme modely celých slov dopredenej topológie. Vyskúšali sme rôzne počty stavov a rôzne rozloženia výstupnej pravdepodobnosti. Najvyššia dosiahnutá úspešnosť rozpoznania bola 99.09% pri testovacej množine veľkosti 220 povelov. Pre samotné slová, ktorých bolo 880, bola úspešnosť 99.77%. Testovacia množina pozostávala z nahrávok 6 rečníkov, na ktorých bol systém trénovaný a 4 rečníkov, ktorý boli neznámi. Pre známych rečníkov sme dosiahli úspešnosť 100%, pre neznámych 97.72%. Môžme teda zhodnotiť, že náš systém je nezávislý na mužskom rečníkovi. Skúsili sme rozpoznať aj 3 ženské rečníčky a pri veľkosti testovacej množiny 66 povelov sme dosiahli najvyššiu úspešnosť 74.24%.

Súčasťou našej výslednej aplikácie je aj systém na detekciu reči založený na krátkodobej energii, ktorý vyextrahuje dostatočne dlhé úseky reči zo vstupného zvukového signálu z mikrofónu a tie sa následne rozpoznávajú. Používateľ interaguje s aplikáciou pomocou jednoduchého grafického rozhrania, ktoré zobrazuje všetky dôležité informácie a umožňuje upravovať niektoré parametre.

Všeobecnosti nášho rozpoznávača by určite pomohla väčšia trénovacia množina, obsahujúca viac rečníkov. Do budúcnosti by mohlo byť zaujímavé systém rozšíriť aj o ženské rečníčky a zachovať pri tom dosiahnutú úspešnosť. Prípadne porovnať úspešnosti našich modelov celých slov a modelov trifónov. Pravdepodobne by takto navrhnutý rozpoznávač dosiahol nižšiu úspešnosť, umožňoval by však ľahšie pridávať nové slová do slovnej zásoby.

Pri reálnom použití nedosiahne naša aplikácie takú veľkú úspešnosť ako pri testovaní, keďže musíme počítať so špeciálnymi prípadmi, napríklad zakašľanie, nesprávne

vyslovenie slova alebo hlasný rušivý zvuk v pozadí. Napriek tomu ju hodnotíme ako úspešnú a použiteľnú na svoj účel. Pri relatívne malých zdrojoch sa nám podarilo vytvoriť systém, ktorý je vhodnejší na ovládanie nášho robotického zariadenia, ako produkty od veľkých technologických firiem.

Hoci sa môže zdať, že v oblasti rozpoznávania reči prístupu pomocou skrytých Markovovských modelov už odzvonilo, ukázali sme, že stále majú svoje miesto. V úzko špecializovaných prípadoch, ako je práve ovládanie robotického zariadenia a pri nedostatku dát, im nemôžu metódy hlbokého učenia stále konkurovať.

Literatúra

[1] Jurafsky D., Martin J., Speech and Language Processing, Third Edition draft, 2018

[2] HTK, Hidden Markov Model Toolkit, http://htk.eng.cam.ac.uk/ [navštívené 8.1.2019]

[3] Juang B. H., Rabiner L. R., Fundamentals of speech recognition, Prentice-Hall International, 1993

[4] Juang B. H., Rabiner L. R., Automatic speech recognition–a brief history of the technology development, Georgia Institute of Technology, Atlanta, Rutgers University and the University of California, Santa Barbara, 2004

[5] Bourlard H., Morgan N., Connectionist Speech Recognition: A Hybrid Approach, The Kluwer International Series in Engineering and Computer Science, Boston, MA: Kluwer, 1994

[6] Rozpoznávanie foném čísel slovenského jazyka neurónovou sieťou, Slovik V., 2007, Diplomová práca, FMFI UK

[7] Graves A., Jaitly N., Towards End-to-End Speech Recognition with Recurrent Neural Networkss, International Conference on Machine Learning (ICML), 2014

[8] Ritomský O., Zvýšenie úspešnosti rozpoznávania izolovaných hlások využitím techniky modelovania šumu pozadia, 2015, Bakalárska práca, FMFI UK

[9] Šuppa M., Speech recognition based on deep Gaussian mixture models, 2016, Bakalárska práca, FMFI UK

[10] Young S., Evermann G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D., Valtchev V., Woodland P., The HTK Book (for HTK Version 3.4), 2006, Cambridge University Press

[11] Young S. J., Young S., The HTK hidden Markov model toolkit:Design and philosophy, 1993, University of Cambridge, Department of Engineering

[12] Kaldi toolkit, http://kaldi-asr.org/ [navštívené 19.1.2019]

[13] CMUSphinx speech recognition system, https://cmusphinx.github.io/ [navštívené 19.1.2019]

[14] Wavesurfer tool for sound, http://www.speech.kth.se/wavesurfer [navštívené 19.1.2019]

[15] Audacity audio software, https://www.audacityteam.org/ [navštívené 10.5.2019]

[16] Černocký H., Zpracování řečových signálů — studijní opora, 2006, Speech@FIT, Ústav počítačové grafiky a multimédií, Fakulta informačních technologií, Vysoké učení technické v Brně

[17] Deshmukh R.R., Saksamudre S.K., Isolated Word Recognition System for Hindi Language, JCSE-International Journal of Computer Sciences and Engineering, vol.4, Issue 7, July 2015

[18] Das B. P., Parekh R., Recognition of Isolated Words using Features based on LPC, MFCC, ZCR and STE, with Neural Network Classifiers, International Journal of Modern Engineering Research , Vol.2, Issue.3, May-June 2012

[19] Gales M., Young S., The Application of Hidden Markov Models in Speech Recognition, Cambridge University Engineering Department, 2008

[20] Nagy M., Počítačové rozpoznávanie reči a výuka čítania, 2010, Dizertačná práca, FMFI UK

[21] Psutka J., Komunikace s počítačem mluvenou řečí, ACADEMIA, Praha, 1995

[22] Dynamic time warping, https://blog.mide.com/dynamic-time-warping-to-estimate-location, [navštívené 26.5.2019]

[23] Hamming window, https://vru.vibrationresearch.com/lesson/averaging/, [navštívené 26.5.2019]

[24] Salhi L., Cherif A., Robustness of Auditory Teager Energy Cepstrum Coefficients for Classification of Pathological and Normal Voices in Noisy Environments, TheScientificWorldJournal. 2013. 435729. 10.1155/2013/435729, 2013

[25] Moattar M. H., Homayounpour M. M., Simple but efficient real-time voice activity detection algorithm, EUSIPCO, 2009

[26] DTW image, https://www.mathworks.com/matlabcentral/fileexchange/43156-dynamic-time-warping-dtw [navštívené 27.5.2019]