Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Zvýšenie úspešnosti rozpoznávania izolovaných hlások využitím techniky modelovania šumu pozadia

bakalárska práca

2015

Ondrej Ritomský

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Zvýšenie úspešnosti rozpoznávania izolovaných hlások využitím techniky modelovania šumu pozadia

bakalárska práca

Študijný program: Informatika
Študijný odbor: 2508 Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: RNDr. Marek Nagy, PhD.

Bratislava, 2015
Ondrej Ritomský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Ondrej Ritomský
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Zvýšenie úspešnosti rozpoznávania izolovaných hlások využitím techniky modelovania šumu pozadia
Increase the recognition accuracy of isolated phones using approach of modeling background noise

Cieľ: Cieľom bude vylepšiť úspešnosť rozpoznávania izolovane vyslovených hlások slovenského jazyka. Rozpoznávač bude využitý v jednoduchej edukačnej aktivite-hre pre deti 1.ročníka. Zvýšenie úspešnosti bude založené na modelovaní a adaptovaní signálu-šumu pozadia. Zaujímavým prvkom bude skúmanie využitia informácie o tom, že rečníci sa nachádzajú pri hre (nahrávaní zvuku) v spoločnom priestore. Predpokladá sa lepšia eliminácia "crosstalk"-ov.

Aplikácia bude realizovaná pomocou Javascriptu, HTML5 a nodejs.

Vedúci:	RNDr. Marek Nagy, PhD.
Katedra:	FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry:	prof. Ing. Igor Farkaš, PhD.
Dátum zadania:	04.10.2014

Dátum schválenia: 28.10.2014 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Chcel by som poďakovať hlavne môjmu vedúcemu bakalárskej práce, RNDr. Marekovi Nagyovi PhD. za jeho nápady a trpezlivosť s mojou počiatočnou neznalosťou tematiky, ale aj všetkým žiakom, ktorý mi poskytli dostatok nahrávok na prácu.

Abstrakt

Bakalárska práca sa zaoberá zvyšovaním úspešnosti rozpoznávania hlások a vytvorením aplikácie, ktorá by mohla slúžiť žiakom základných škôl na precvičovanie vyslovovania hlások. Rečový signál je reprezentovaný metódou MFCC a vyskúšané sú tri rôzne metódy na jeho klasifikáciu. Použité experimentálne metódy na zvýšenie úspešnosti sú orezávanie nahrávky, klástering, vytvorenie modelu pozadia. Práca tiež popisuje možnosti obmien týchto metód, ktoré sa vyskúšali pri jej vypracovaní.

Práca taktiež popisuje aplikáciu, ktorá využíva popísaný rozpoznávač. Aplikácia je jednoduchá na pochopenie pre cieľovú skupinu a je spustiteľná v hociakom operačnom systéme, v ktorom je najnovšia verzia Google Chromu. Táto aplikácia je vytvorená s cieľom zrýchliť rozpoznávanie využitím viacerých vlákien. Taktiež je aplikácia schopná nahrávky zbierať a následne poskytnúť nástroje na pomoc pri ich spracovaní.

Kľúčové slová: zvuk, rozpoznávanie hlások, MFCC, model pozadia, klástering.

Abstract

Bachelor thesis is about increasing success rate of phones recognition and about creating an application, which could serve elementary school students to train phone pronunciation. Speech signal is represented by MFCC method and there are tried three different methods for his classification. Used experimental methods for increasing success rate are trimming records, clustering, creation of background model. Thesis also shows possible modifications applicable to those methods. Described modifications were tried in the time of creating this thesis.

Thesis also describes application, which is using described speech recogniser. The application is easy to understand for targeted group and is executable in any operating system with the newest version of Google Chrome browser. This application is created with a goal to improve speed of recognition using threads. Application is also capable of saving records and offers tools for help to process records.

Keywords: sound, speech-sound recognition, MFCC, background model, clustering.

Obsah

Úvod

Rozpoznávaniu reči sa venuje veľa prác a môžeme pozorovať programy schopné rozpoznávať reč s vysokou úspešnosťou. Na rozdiel od toho je rozpoznávanie izolovaných hlások iné. Pri rozpoznávaní slov tiež rozpoznávame hlásky alebo slabiky tvoriace slovo, ale spoliehame sa na kontext a teda pri použití veľkého slovníka nepotrebujeme rozpoznať všetky hlásky slova dokonalo. Na rozpoznávanie izolovaných hlások nemôžeme využiť kontext, pretože žiadny nemáme. Všetky informácie čo získame sú z rečového signálu a predchádzajúce rozpoznávanie nám nedáva žiadnu informáciu navyše. Z tohto dôvodu rozpoznávače dosahujú menšiu úspešnosť pri rozpoznávaní izolovaných hlások. Práve táto skutočnosť bude naša časť motivácie.

Rozpoznávanie izolovaných hlások sa môže zdať ako neatraktívny problém, keďže v aplikáciách využívajúcich rozpoznávanie reči chceme rozprávať plynulo a nie hláskovať. Ale pravdou je, že ak napríklad dieťa nevie ešte poriadne vyslovovať všetky hlásky, tak ani jeho celé slová nemusia byt zrozumiteľné pre rozpoznávač. A teda korektné vyslovovanie hlások bude naša druhá časť motivácie. Samotné rozpoznávanie reči bude prebiehať v prostredí so šumom, čo znižuje úspešnosť, ale rozpoznávanie v prostredí s minimálnym šumom je nepraktické a nemôžeme sa nato spoliehať.

Keď už vieme, akú máme motiváciu je jednoduché určiť cieľ nasledujúcej práce. Cieľom je vytvoriť rozpoznávač izolovaných hlások, ktorý sa následne budeme snažiť vylepšovať rôznymi experimentami. Tento rozpoznávač potom chceme použiť v aplikácii, ktorá môže pomôcť deťom v čítaní hlások.

Prvé tri kapitoly začnú všeobecne. V prvej sa pozrieme na komunikáciu

hovorenou rečou, ktorej sa účastní počítač. Popíšeme si históriu rozpoznávania reči a tiež na problémy, ktoré nás momentálne stále sprevádzajú. V druhej kapitole sa pozrieme ako delíme reč, ako vznikajú hlásky a ako ich v slovenskom jazyku delíme. V tretej kapitole sa pozrieme aký vplyv má šum na rozpoznávanie reči a aké metódy sa vo svete používajú na redukciu tohto vplyvu.

V štvrtej kapitole si popíšeme ako vytvoriť modely pre jednotlivé hlásky. Popíšeme si, že signál predspracujeme prevzorkovaním a Hammingovou funkciou okienka. Ďalej si popíšeme momentálne jednu z najpoužívanejších metód na reprezentovanie signálu, metódu MFCC. Kapitola skončí popisom, ako si vytvoríme modely použitím gausiánov.

V piatej kapitole si povieme o možnostiach rozpoznávania hlások a taktiež popíšeme experimenty, ktoré vyskúšame na zvyšovanie úspešnosti rozpoznávania. Pri týchto experimentoch uvedieme aj možné obmeny, ktoré majú rôzny vplyv na rozpoznávanie.

Predposledná šiesta kapitola bude o implementácii aplikácie. Spomenieme si z akých komponentov sa bude skladať, akú funkcionalitu ktorý komponent vykonáva a ako si vieme upraviť aplikáciu upravením parametrov.

V poslednej kapitole sa pozrieme na výsledky. Popíšeme si testy a spôsoby klasifikácie a vylepšenia, ktoré sme použili.

Na záver zhrnieme dosiahnuté výsledky a uvedieme možnosti pokračovania v tejto práci.

Kapitola 1

Komunikácia hovorenou rečou

V prvej kapitole si povieme, ako musíme komunikáciu rozdeliť na podúlohy, ak by sme chceli, aby sa komunikácie účastnil počítač. Tiež si povieme, aké problémy nastávajú pri rozpoznávaní reči počítačom a na konci kapitoly sa pozrieme na významné vynálezy a objavy.

1.1 Počítač ako účastník komunikácie

Hovorená reč patrí medzi základné prostriedky komunikácie dvoch a viac ľudí. Túto výmenu informácií vieme rozdeliť na postupnosť akcií. Človek si správu najprv pripraví v mozgu a následne ju vysloví. Následne poslucháč akustický rečový signál zachytí a pochopí. Takto sa postupnosť akcií opakuje, pričom aj z poslucháča sa môže stať rečník.

Situácia sa mení pokiaľ jeden z účastníkov komunikácie je počítač. V roli poslucháča musí počítač rečový signál spracovať, rozpoznať reč, pochopiť význam slov a viet. V roli rečníka musí byť schopný správu syntetizovať. Zatiaľ sa však nepodarilo vytvoriť počítač, ktorý by bol schopný viesť plnohodnotný dialóg s užívateľom bez obmedzení. Už dlhšie sa využívajú jednoduché systémy, ktoré vykonávajú len časť úloh existujúcich v komunikácií hovorenou rečou s počítačom. Tieto systémy dokážu rozpoznať malú množinu slov alebo slovných spojení, ktoré môžu slúžiť ako povely, príkazy pre program. Naprí-

klad automatický telefónny operátor, ktorý sa dá ovládať hlasom a tým šetrí čas. Ďalej existujú programy umožňujúce syntézu napísanej správy, alebo programy schopné rozpoznávať diktované vety. Problémy nastávajú, ak rečník správne neartikuluje, alebo rozpráva hovorovou rečou.

Príkladov a možností využitia komunikácie s počítačom hovorenou rečou je veľa. Vo väčšine prípadov nám zjednodušujú život a urýchľujú skoro každodenné činnosti. Avšak každý z programov vykonáva len niektoré podúlohy problematiky a to tiež nie dokonalo. Podrobnejšie o komunikácií s počítačom si môžete prečítať napríklad v knihe Jozefa Psutku [9].

1.2 Známe problémy pri rozpoznávaní reči

Komunikácia s počítačom je jedna z disciplín umelej inteligencie, ktorá ale zahŕňa oblasti fonetiky, akustiky, teórie informácie, spracovanie signálu, heuristického prehľadávania, matematickej lingvistiky a porozumenia. Každá oblasť, ako aj nezávislý vývoj týchto oblastí, prináša svoje problémy.

Problémov týkajúcich sa rozpoznávania ľudskej reči je hneď niekoľko. Každý človek má iný hlas. Muži mávajú hlas hlbší, ženy a deti naopak vyšší. Zafarbenie hlasu má každý jedinec iné. Niekto môže mať hlas chrapľavý, niekto zase piskľavý. Každý má tiež iné tempo rozprávania, iný prízvuk. Preto je problém vytvoriť systém schopný rozpoznať reč ľubovoľnej osoby. Rozdielnu hlasitosť dokážeme skoro ignorovať, ale šepkanie a krik tiež menia hlas. Tieto problémy sú spôsobené rozdielmi v hlasových ústrojenstvách a rozdielnou artikuláciou. Rozlišnostiam hlasu detí sa venuje napríklad práca Antona Urbaníka [11].

Opakované vyslovenie toho istého slova je rôzne a to je ďalší problém. Človeka je ovplyvnený mnohými faktormi, napríklad fyzickým stavom. Chorý človek bude slovo vyslovovať inak, pretože sa môže stať, že vzduch nebude schopný prejsť nosovou dutinou. Vekom sa hlas tiež môže meniť. Emocionálny stav tiež ovplyvňuje spôsob vyslovenia, nahnevaný alebo vystrašený človek bude mat iný hlas ako niekto pokojný. Samotná výslovnosť slova sa mení

aj vďaka koartikulácii. Záleží na tom, aké slovo bolo povedané pred ním a aké bude povedané po ňom. Tie môžu ovplyvniť vyslovenie začiatku a koncu slova. Dokonca aj samotné hlásky môžu mať rozdielnu dĺžku vyslovenia a byť vyslovené inak podľa toho aké hlásky sa nachádzajú okolo nich.

Vplyv prostredia je tiež problém pri rozpoznávaní reči. V pozadí sa môže napríklad stavať, alebo sa môže niekto iný rozprávať. Hlasné rozprávanie v pozadí je naozaj veľký problém, lebo rozpoznávač reči nemusí vedieť odlíšiť rozprávajúcich. Nesmieme zabudnúť ani na ozvenu. Všetky tieto vplyvy sa zmiešajú a sčítajú so signálom. Reálne nedosiahneme v pozadí absolútne ticho, takže na šum nesmieme zabudnúť.

Rozpoznávacie zariadenia zatiaľ nedokážu využiť funkcionalitu, ktorú používa človek aj keď len podvedome. Pochopenie významu správy nie je ovplyvnené farbou hlasu, alebo hlasitosťou. Túto informáciu človek dokáže filtrovať. Naopak význam predchádzajúcich slov nám pomôže v rozpoznaní nasledujúcich slov. Intonácia môže zmeniť význam vety a kontext nám pomáha v pochopení celej správy. Takúto funkcionalitu žiaľ rozpoznávače nemajú. Nepoznajú význam slov, len sa snažia rozpoznať slovo za slovom.

Niektorá funkcionalita mozgu by sa dala využiť na zlepšenie rozpoznávania. Napríklad by sme paralelne mohli rozpoznávať aj celé slovné spojenia okrem izolovaných slov. To nás vedie k ďalšiemu problému. Väčšina programov vyžaduje spracovanie reči v reálnom čase, preto zlepšovanie rozpoznávania pridávaním funkcionalít, zväčšuje čas odozvy programu. Každoročné zlepšovanie hardvéru čas spracovania znižuje, ale veľa ľudí, firiem a inštitúcií stále používa staršie počítače. Vývojári by sa mali snažiť programy čo najviac optimalizovať, využívať nové efektívnejšie algoritmy a paralelizáciu.

Všetky tieto problémy prispievajú k chybovosti prijatej správy. To spôsobuje problém pri porozumení správy, pretože problematika porozumenia prirodzeného jazyka bola vyvíjaná nezávisle. V nej predpokladali, že vstup je správa v písomnej forme bez chýb.

1.3 História rozpoznávania reči

Tvorba systémov, ktoré by umožňovali komunikáciu medzi človekom a strojom je problém, ktorý človeka zaujímal ešte dávno predtým ako vznikli prvé počítače. V 18. storočí nezávisle popísali von Kempelen a Kratzyenstein prvé experimenty s mechanickým rečovým syntetizátorom, ktorý dokázal vytvoriť zvuk samohlások. Od vtedy vzniklo veľa experimentov a návrhov, ktoré sa zaoberali syntézou, analýzou a rozpoznávaním signálu. Jedným z významných objavov bol vynález spektrografu Lawrencom Kerstom. Spektrograf dokázal vizuálne reprezentovať akustické spektrum a iné signály. Spektrograf sa stále používa napríklad pri spracovaní reči alebo analyzovaní zvukov zvierat. Veľkým objavom boli samozrejme počítače. Vytvorilo sa veľa metód založených na digitalizácií signálu. Rozvoj týchto metód bol závislý od rozvoja výpočtových systémov. Rozvoj objavu Fourierovej analýzy bol tiež závislý od ostatných objavov. Bola známa už v 18. storočí, ale diskrétna Fourierova transformácia dostala pozornosť až s objavom počítačov. Fourierova transformácia vyjadrí zvukový signál, ako kombináciu jednoduchých vĺn závislých od frekvencie. Jej širšie využitie je vidieť s vynájdením signálových procesorov.

Podobne boli vynájdené a vylepšované algoritmy na klasifikáciu slov. Významné boli metódy založené na dynamickom programovaní ako napríklad dynamic time warping. Bola to často používaná metóda, ktorá sa používala na klasifikáciu izolovaných slov a slovných spojení. Po nej vznikol prístup modelovania slov alebo častí slov skrytými Markovovými modelmi. Tento prístup sa stal atraktívnym napríklad pri metóde klasifikovaní slov z veľkých slovníkov. Viac o Markovových modeloch sa môžete dočítať v rigoróznej práci Mareka Nagyho [7].

Kapitola 2

Stavebné jednotky reči

V tejto kapitole si povieme, čím je naša reč tvorená. Taktiež sa pozrieme na podrobnejšiu tvorbu a delenie hlások.

2.1 Fón, fonéma, hláska, slabika, slovo

Fonému považujeme za najmenšiu zvukovú jednotku reči. Fonémy navzájom vieme odlíšiť podľa spôsobu a miesta vzniku, artikulujúceho orgánu alebo sluchového dojmu. Fonologické výskumy ukázali, že existuje okolo 12 univerzálnych odlišovacích príznakov, ktoré sa aktívne využívajú vo svetových jazykoch. Dá sa to objasniť fyziológiou hlasového ústrojenstva, kde sú artikulačné orgány schopné vytvoriť okolo dvanásť rôznych polôh.

Ďalšia stavebná jednotka je slabika, ktorá je tvorená postupnosťou foném. Každá slabika obsahuje inú postupnosť foném a štruktúra každej slabiky je presne daná. Slabiky skladáme do slov, ktoré už majú aj informačnú hodnotu. Slovanské jazyky obsahujú okolo 2500 až 3500 slabík a 45000 až 50000 slov.

Okrem foném poznáme fón, ktoré skúma fonetika. Fón v jazyku nemá rozlišovaciu schopnosť. Ak v slove vymeníme fonému za inú, budeme počuť rozdielne slovo. Na rozdiel od toho fóny nemusia ešte nerozlišujú slová, napríklad písmeno n v slovách noha a jednotka vyslovujeme rôzne, teda počujeme rôzne fóny, ale ich výmena nezmení význam slova. Ďalej budeme označovať

fóny a fonémy spoločne slovom hláska.

2.2 Tvorba a delenie hlások

Proces vytvorenia reči začína v pľúcach a končí odídením zvuku z úst. Pľúca sú zdrojom hlasovej energie. Vytlačený vzduch z pľúc prechádza rečovými orgánmi, medzi ktoré patria hlasivky, hrdelná, ústna a nosná dutina, mäkké a tvrdé podnebie, zuby a jazyk.

Hlasivky sú zdrojom všetkých znelých zvukov. Medzi znelé zvuky patria samohlásky a znelé spoluhlásky. Aby sme mohli vytvoriť jednotlivé zvuky, hlasivky sa zužujú a rozťahujú. Vytlačený zvuk z pľúc tlačí na hlasivky a tie sa stávajú pružnými a začínajú kmitať. Frekvencia kmitania sa pohybuje medzi 150 až 400 Hz. Je rôzna u detí, mužov a žien, kvôli rôznej veľkosti tlaku vyvíjaného na hlasivky a svalového napätia hlasiviek. Táto frekvencia kmitov hlasiviek charakterizuje základní tón hlasu, ktorý je prítomný pri vytváraní znelých zvukov.

Pri vyslovovaní samohlások je priechod vzduchu čo najvoľnejší. Okrem základného tónu sa objavujú aj zosilňujúce tóny, ktoré sa nazývajú formanty. Zmeny ich výšky a intenzity sú spôsobené perami, čeľusťou, jazykom a mäkkým podnebím. Pri všetkých slovenských samohláskach mäkké podnebie zabraňuje prúdeniu vzduchu do nosnej dutiny.

Spoluhlásky sa líšia od samohlások tým, že v akustickom spektre hlások je prítomný aj charakteristický šum. Pri vytváraní spoluhlások vydychovaný vzduch naráža o prekážky vytvorené artikulačnými orgánmi a vytvára vzduchovú turbulenciu. Prekážky delíme na úplné a čiastočné. Zrušenie úplnej prekážky spôsobuje šum, ktorý je vytvorený náhlim vypustením vzduchu prekážkou. Takýto šum môžeme prirovnať k malému výbuchu. Úplná prekážka je podstatou záverových spoluhlások, v slovenčine to sú *m, n, ň, d, ď, t, ť, b, p, k, g*. Naopak čiastočné prekážky nezatvárajú cestu úplne, ale len ju zužujú. Pri priechode vzduchu zúženou cestou, vzniká trecí šum. Čiastočná prekážka je podstatou úžinových spoluhlások, medzi ktoré patria ostatné

spoluhlásky teda *l, ľ, r, s, š, z, ž, f, v, ch, h, j*. Spoluhláskam *c, č, dz, dž* sa hovorí polozáverové a vyskytujú sa u nich oba typy prekážok.

Spoluhlásky môžeme tiež deliť na základe znelosti. Poznáme spoluhlásky znelé a neznelé. Znelé sú pri tvorení sprevádzané hlasivkovým tónom. Neznelé naopak nesprevádza hlasivkový tón. Dôvodom je, že pri vyslovovaní neznelých spoluhlások, sú hlasivky od seba vzdialené a prepúšťajú vzduch podobne ako pri voľnom dýchaní. Nosná dutina sa pri väčšine tvorby spoluhlások nezapája, pretože mäkké podnebie cestu do nej uzatvára. Ale existujú spoluhlásky, pri ktorých mäkké podnebie neuzavrie cestu do nosnej dutiny. Voláme ich nosové spoluhlásky a sú to *m, n , ň*.

Spoluhlásky sa tiež delia na párové a nepárové. Párové spoluhlásky tvoria dvojice, v ktorej je artikulácia oboch spoluhlások rovnaká. Líšia sa účasťou znelosti. V každej dvojici je jedna spoluhláska znelá a druhá neznelá. Nepárové spoluhlásky sú všetky znelé a nemajú neznelý opak.

Kapitola 3

Vplyv šumu na rozpoznávanie reči

V tejto kapitole sa pozrieme aký má šum vplyv na reprezentáciu reči a aké metódy sa používajú na redukciu nezhôd v rozpoznávaní reči spôsobených vplyvom šumu.

3.1 Delenie vplyvu šumu

Čistá reč je v praktických aplikáciách nereálna požiadavka. Takmer vždy je vstup znečistený šumom pozadia, ktorý spôsobuje zníženie úspešnosti rozpoznávacích aplikácií. Zníženie presnosti rozpoznávania môže spôsobiť až nepoužiteľnosť takýchto aplikácií. Preto sa rozpoznávanie reči s rušným pozadím (alebo aj robustné rozpoznávanie reči) stalo dôležitou časťou výskumu rozpoznávania reči.

Vplyv rušivého pozadia rozdeľujeme na dve kategórie. Prvou kategóriou je strata informácie, ktorá je spôsobená náhodným výskytom rušenia. Počas rozprávania nás môže niečo prehlušiť alebo niekto iný môže rozprávať zároveň s nami. Strate informácie nevieme zabrániť. Aj keď dokážeme indentifikovať rozprávajúceho, nedokážeme rozpoznať čo rozpráva ak s ním rozpráva zároveň niekto iný. Druhou kategóriou je skreslenie reprezentujúceho priestoru.

Toto skreslenie spôsobené vplyvom pozadia, spôsobí rozdiely medzi čistými trénovacími a testovacími podmienkami. Akustické modely sa trénujú s rečou s čistým pozadím a preto nemodelujú reč so šumom presne. Tejto kategórii sa venuje viacero metód, ktoré sa snažia zredukovať rozdiely medzi čistými a rušivými podmienkami. Ani úspešným zredukovaním rozdielov, nedokážeme zabrániť strate informácií z prvej kategórie.

3.2 Metódy redukcie vplyvu šumu na rozpoznávanie

Väčšina metód vytvorených na prispôsobenie aplikácií rozpoznávajúcich reč, sa snaží znížiť rozdiely medzi trénovacími a testovacími podmienkami. Tieto metódy môžeme rozdeliť do troch skupín.

Prvou skupinou sú metódy, využívajúce robustné reprezentovanie reči. Ak reč modelujeme spôsobom, ktorý je minimálne ovplyvnený šumom, môžeme rozdiely medzi čistými a rušivými podmienkami ignorovať. V tejto skupine si môžeme spomenúť metódu zdvíhacích okienok, ktorá pri reprezentácií reči založených na Linear Prediction Coefficients (LPC-cepstrum) redukuje prínos rádovo nižších koeficientov, ktoré sú ovplyvnené šumom viac. Ďalšia metóda je využitie Mel Frequency Cepstral Coeffictients (MFCC), ktoré dosahujú lepšie výsledky pod vplyvom rušivého pozadia. Viac o MFCC si povieme v ďalšej kapitole, kde si aj podrobne prejdeme ako fungujú.

Do druhej skupiny patria metódy, ktoré urobia z rušenej verzie odhad čistej reči a túto potom rozpoznávajú použitím modelov natrénovaných s čistými podmienkami. V tejto skupine existuje metóda spektrálneho odčítania. Za predpokladu, že reč a šum nekorelujú a že šum má viac nemenné vlastnosti ako reč, šum môže byť čiastočne odstránený. Efektivita závisí od dobrého odhadu vlastností šumu pozadia. Túto skupinu voláme metódy kompenzujúce šum.

Tretia skupina zahŕňa metódy, ktoré adaptujú čisté modely na rušné pozadie. Do tejto skupiny patrí napríklad metóda znečistenia trénovacej da-

tabázy. Trénovanie reči s rušivým pozadím je veľmi efektívne, má to však svoje nedostatky. Dopredu nedokážeme odhadnúť aký šum bude pri používaní takýchto modelov a nevieme povedať, aký šum sa vyskytol pri trénovaní. Trénovanie v reálnom čase nepomôže, pretože môže trvať moc dlho. Riešenie týchto problémov spočíva napríklad v natrénovaní reči pri rôznych šumoch.

Tieto skupiny si môžeme zadefinovať aj presnejšie. Označme x reprezentáciu čistej reči. Vplyv šumu spôsobí skreslenie a zmení x na y. Táto skreslená reč y je rozpoznávaná čistým modelom λ_x. Prvá skupina s robustným modelom očakáva y, takže negatívny vplyv na rozpoznávanie je minimálny. Druhá a tretia skupina predpokladá, že y je skreslená verzia čistej reči $y = F(x)$, kde F modeluje skreslenie spôsobené šumom. Kompenzačné metódy odhadnú inverznú skresľujúcu T^{-1} a vytvoria odhad čistej reči x':

$$x' = T^{-1}(y) \tag{3.1}$$

Rozpoznávanie je následne urobené odhadom čistej reči x' a čistých modelov λ_x. Adaptujúca tretia skupina naopak aplikuje odhadnutú funkciu T na modely:

$$\lambda'_y = T^{-1}(\lambda_x) \tag{3.2}$$

Rozpoznávanie použije skreslenú reč y a odhadnuté modely skreslenej reči λ'_y

Viac o metódach na rozpoznávanie reči v rušivom prostredí si môžete pozrieť napríklad v práci [4], ktorá sa zaoberá hlavne kompenzačnými metódami.

Kapitola 4

Reprezentácia hlások

V tejto kapitole sa pozrieme na spôsob reprezentovania hlások, metódou Mel Frequency Cepstral Coefficients (MFCC). Popíšeme každú funkciu, ktorú sme použili, od prevzorkovania signálu až po vytvorenie modelov hlások z trénovacích vzoriek. Začneme unifikáciou signálu, potom si povieme ako funguje MFCC a nakoniec sa pozrieme na vytvorenie modelov a ostatné informácie, ktoré sa oplatí si uložiť alebo predpočítať.

4.1 Predspracovanie signálu

Nech už naša aplikácia beží na hoci akom počítači, potrebujeme aby vstup do série funkcií MFCC bol zjednotený. Prvou požiadavkou je aby sme na spracovanie časového úseku dĺžky t použili rovnako veľa vzoriek. Problémom je, že každá zvuková karta, či už na procesore alebo externá, používa inú vzorkovaciu frekvenciu. Inak povedané na vstup dostávame rôzne veľa vzoriek za sekundu. Tento problém vyriešime prevzorkovaním a po ňom si povieme o funkcii okienka.

4.1.1 Prevzorkovanie

V dnešnej dobe sú bežné vzorkovacie frekvencie 48 kHz alebo 44.1 kHz. Frekvencia 48 kHz sa používa pri ukladaní zvuku na DVD. Tieto hodnoty sú pre nás však privysoké a to z viacerých dôvodov. Ak by sme chceli používať 48 000 vzoriek za sekundu, tak by používatelia s nižšími frekvenciami nedokázali poskytnúť dostatok vzoriek na spracovanie. Toto sa dá riešiť duplikovaním vzoriek, no na kvalite to nepridá, len nahrávka zaberie viac miesta v pamäti alebo na disku a zvýši zaťaženie siete. Takúto vysokú kvalitu však nepotrebujeme, pretože veľká vzorkovacia frekvencia znamená akú najväčšiu zvukovú frekvenciu dokážeme zreprodukovať, a vyššie zvukové frekvencie sú pre človeka nepočuteľné. Preto nám bude stačiť vzorkovacia frekvencia 16 kHz. Aj keď kvalita nahrávky sa zhorší, ostane stále dostačujúca. Okrem toho nám takáto frekvencia šetrí miesto. Pri trvaní nahrávky 2 sekúnd, vzorkovacej frekvencii 16 kHz, 4 bytov potrebných na reprezentovanie nahrávky a vynásobením:

$$zabraná\_pamäť = t * vzorkovacia\_frekvencia * počet\_bytov\_na\_vzorku \tag{4.1}$$

dostaneme, 128000 bytov oproti 384000 bytom, ktoré by sme potrebovali pri použití frekvencie 48 kHz. Samozrejme ak máme len jednu nahrávku, tak to nie je veľa, ale pre sieť a server, ktorý obsluhuje viacero používateľov je to rozdiel. Teraz, keď už vieme akú vzorkovaciu frekvenciu chceme, stačí aby program zistil ako rýchlo dostáva vzorky a vypočítame *pomer* týchto frekvencií:

$$pomer = vzorkovacia\_frekvencia/nová\_vzorkovacia\_frekvencia \tag{4.2}$$

Vzorky zo vstupu a pomer dáme prevzorkovacej funkcii a tá nám vráti vzorky s novou vzorkovacou frekvenciou. Táto funkcia funguje jednoducho, z indexu i sa posunie o *pomer* a ak je to celé číslo tak pridá vzorku $x_{i+pomer}$ do výstupného pola. Ak nový index nie je celé číslo, tak vypočíta vážený priemer zo vzoriek $x_{\lfloor i+pomer \rfloor}$ a $x_{\lceil i+pomer \rceil}$ a ten pridá do výstupného pola.

Predtým ako sa pozrieme na funkciu okienka a ostatné postupy si potrebujeme vstup rozdeliť do menších segmentov. Čím kratší segment, tým lepšie budeme môcť reprezentovať väčší časový úsek, ale zároveň budeme mať menej vzoriek pre jeden segment. Preto by menšia vzorkovacia frekvencia ako 16 kHz mohla byť dosť nekvalitná. Na segment dĺžky 25 ms tak dostaneme 400 vzoriek. Dĺžku 25 ms sme nevybrali náhodne, je často používaná a po práci so 100 ms segmentami sme aj zistili prečo. O tom si však povieme neskôr. Okrem 25 ms segmentov sa používajú aj trochu kratšie segmenty, napríklad 13 ms. Je dôležité spomenúť, že ak prvý segment začína v čase 0, tak druhý segment nezačne v čase 25ms, ale v čase 12.5ms. Pomôže to zlepšiť situáciu, keď nahrávku rozdelíme v nevhodnom momente.

4.1.2 Hammingovo okienko

Hammingovo okienko je funkcia, ktorá všetky vzorky x_i z intervalu ováhuje váhou v_i a vzorky mimo intervalu váhou $v_j = 0$. My sme ale už rozdelili vstup na segmenty a naša funkcia dostane jeden segment, ktorý prezentuje celý interval. Ako sa teda počíta váha vzorky? Konkrétne pre Hammingovo okienko je to vzorcom:

$$v\left(i\right) = \alpha - \beta \cos\left(\frac{2\pi i}{N-1}\right) \tag{4.3}$$

kde N je dĺžka segmentu a s hodnotami $\alpha = 0.54, \beta = 0.46$.

Otázkou stále ostáva, prečo vôbec používame funkciu okienka. Dôvodom je fakt, že sme potrebovali vstup rozdeliť na segmenty a Diskrétna Fourierova transformácia, ktorá sa použije následne, predpokladá, že vstup je jedna perióda, periodického signálu. Lenže počet períód v segmente nemusí byť celé číslo a to môže spôsobiť nespojitosť medzi koncom jedného a začiatkom druhého segmentu. Aplikovaním Hammingovho okienka, sa budú kraje segmentov približovať k nule a teda prechod medzi segmentami bude plynulejší. To v konečnom dôsledku zlepší výsledky rozpoznávania.

Presnejšie vysvetlenie funkcii okienka si môžete pozrieť napríklad v tejto

práci [1]. Okrem toho sa tam dá dočítať aj o Fourierovej Transformácii, o ktorej si povieme niečo aj my teraz.

4.2 Mel Frequency Cepstral Coefficients

Reprezentácia signálu pomocou metódy MFCC je jedna z najpoužívanejších. Je to z dôvodu, že oproti LPC sa snažia napodobiť ľudské ucho, ktoré vníma jednotlivé frekvencie signálu nelineárne, konkrétne nižšie frekvencie vníma citlivejšie. V predchádzajúcej kapitole sme si povedali, že MFCC je metóda, ktorá robustnejšie reprezentuje reč a preto ju môžeme považovať za prvú metódu, ktorú využijeme na lepšie rozpoznávanie reči so šumom v pozadí. Jednoduchý návod môžeme nájsť na stránke [2].

4.2.1 Diskrétna Fourierova Transformácia

Fourierova Transormácia je ďalšia funkcia, ktorú aplikujeme na každý segment. Môžeme ju pokladať za začiatok celej metódy MFCC. Fourierova Transformácia zmení časovú doménu funkcie na frekvenčnú doménu. Inak povedané, rozloží funkciu na ováhovanú sumu sínusov a kosínusov. Ale my reprezentujeme signál konečnou množinou vzoriek a preto sa v spracovaní zvuku používa Diskrétna Fourierova Transformácia. Tú počítame nasledujúcim predpisom:

$$X_k = \sum_{j=1}^{N-1} \left(x_j \left(\cos(\frac{2\pi}{N}) + i \sin(\frac{2\pi}{N}) \right) \right) \tag{4.4}$$

Týmto rozdelíme spektrum na N frekvencií, kde každá je reprezentované jedným komplexným číslom z postupnosti $X_0, ..., X_{N-1}$. Reálna zložka nám udáva amplitúdu frekvencie a imaginárna veľkosť fázového posunu. Teraz môžeme skúmať frekvencie, ktoré sú pre nás dôležité a nepodstatné naopak ignorovať.

Priame implementovanie tejto funkcie je ale nedostačujúce. Keďže, každú hodnotu X_k počítame prejdením celého segmentu, dostávame časovú zloži-

tosť $O(N^2)$. A to je pri spracovaní v reálnom čase pomalé. Našťastie existuje Rýchla Hartleyho Trasformácia [10], nazývaná aj Rýchla Fourierova Transormácia(RFT), ktorá využíva metódu rozdeľuj a panuj. Tá vždy rozdelí vzorky na párne a nepárne a tie sa spracujú samostatne. Teda hĺbka rekurzie je $\log_N$ a časová zložitosť $O(N \log_N)$. RFT predpokladá, že vstup je dĺžky 2^n pre $n \in N$. To sme doteraz síce nepredpokladali, ale stačí doplniť segment nulami.

Z komplexných čísiel nakoniec vypočítame amplitúdové spektrum so vzťahom $a_i = \sqrt{r_i^2 + i_i^2}$, kde r_i je reálna zložka komplexného čísla X_i a i_i je imaginárna zložka.

4.2.2 Mel škála a koeficienty kepstra

Mel škála je spôsob ako sa MFCC snaží napodobiť ľudské ucho. Najprv si prevedieme frekvencie na jednotku mel vzorcom:

$$m = 1127 \log_e \left(1 + \frac{f}{700}\right) \tag{4.5}$$

a tým si vytvoríme banku filtrov. Tieto filtre si môžeme predpočítať, pretože pre ich vytvorenie nám stačí vedieť najnižšiu a najvyššiu frekvenciu, počet výsledných filtrov a počet hodnôt amplitúdového spektra. Začiatočnú hranicu sme určili na 300 a naša vzorkovacia frekvencia 16000 určuje najvyššiu frekvenciu na 8000, pretože to je najvyššia frekvencia, ktorú vieme reprezentovať pri tejto vzorkovacej frekvencii. Počet filtrov sme určili na 40 a počet hodnôt sa dozvieme pri počítaní MFCC prvého segmentu. Takto sme si pripravili hranice každého filtra. Pre nižšie frekvencie sú filtre užšie, pre vyššie frekvencie sú širšie a teda budú aplikované na väčší rozsah frekvencii.

Teraz, keď dostaneme hodnoty amplitúdového spektra pre daný segment, môžeme aplikovať vypočítané filtre. Jednotlivé hodnoty z amplitúdového spektra patriace do daného filtra, ováhujeme hodnotami filtra a zosumujeme. Toto zopakujeme pre každý filter. Nakoniec vrátime 40 hodnôt pre dané filtre, ktoré funkcia vypočítala.

Nakoniec si z týchto hodnôt vypočítame výsledný vektor koeficientov pomocou kosínusovej transformácie:

$$c_i = \sqrt{\frac{2}{n}} \sum_{j=1}^{n} \left(a_j \cos\left(\frac{\pi j}{n} (j - 0.5) \right) \right) \tag{4.6}$$

kde a_j sú vypočítané amplitúdy z filtrov a n je ich počet, teda 40. Dimenzia výsledného vektoru je 12.

4.2.3 Energia

Okrem reprezentácie segmentu vektorom MFCC, si spočítame ešte energiu segmentu. Vypočítame ju pre daný segment ešte pred aplikovaním Hammingovho okienka nasledujúcim vzorcom:

$$e = \sum_{i=1}^{n} x_i^2 \tag{4.7}$$

Energia segmentu nie je súčasť MFCC, ale ako informácia o segmente sa nám bude hodiť.

4.3 Vytvorenie modelov

Keď už vieme ako reprezentovať reč, môžeme vytvoriť model pre dané hlásky. Modelom pre konkrétnu hlásku h bude gausián λ vytvorený z množiny H, tvorenou n vektormi c. Predpokladáme, že pre každú hlásku h množina H obsahuje len také vektory c, ktoré naozaj reprezentujú danú hlásku. Ako prvé si spočítame priemer $\mu = \mu_1, \mu_2, ..., \mu_{12}$:

$$\mu_i = \frac{1}{n} \sum_{j=1}^{n} c_{j,i} \tag{4.8}$$

Ďalej si potrebujeme spočítať rozptyl a to vo viacrozmernom prípade je kovariančná matica Σ:

$$\Sigma = \frac{1}{n} \sum_{j=1}^{n} (c_j - \mu)^T (c_j - \mu) \tag{4.9}$$

Nás budú zaujímať len hodnoty na diagonále, teda hodnoty Σ_{ij} kde $i = j$. Inšpirovali sme sa kódom v HTK Book [12], kde tiež ignorujú ostatné hodnoty. Tieto hodnoty budeme ďalej označovať ako vektor $\sigma^2 = \sigma_1^2, \sigma_2^2, ..., \sigma_{12}^2$ a nazývať rozptyl. Týmto spôsobom si vytvoríme model pre tieto hlásky:

$$a, b, d, \check{d}, e, f, g, h, ch, i, j, k, l, \check{l}, m, n, \check{n}, o, p, r, s, \check{s}, t, \check{t}, u, v, z, \check{z} \tag{4.10}$$

Môžeme si všimnúť, že niektoré hlásky chýbajú, sú to:

$$c, \check{c}, dz, d\check{z}, q, x \tag{4.11}$$

Je to z dôvodu, že po zvukovej stránke sú tvorené kombináciou ostatných hlások a to nasledujúco:

$$c = t + s$$
$$\check{c} = t + \check{s}$$
$$dz = d + z$$
$$dz = d + \check{z}$$
$$q = k + v$$
$$x = k + s$$

A preto si pre ne nevytvoríme model, miesto toho použijeme ostatné modeli na ich testovanie. Keby sme ich vytvorili, tak napríklad modely hlások c a s by boly veľmi podobné a bol by problém ich rozlíšiť.

Ostatné hlásky slovenskej abecedy nebudeme testovať vôbec. Ale vytvoríme si jeden model pre ticho resp. pozadie v ktorom je šum minimálny.

Kapitola 5

Rozpoznávanie hlások

V tejto kapitole si povieme ako budeme rozpoznávať hlásky použitím vytvorených modelov. Vysvetlíme si aj metódy na zvyšovanie úspešnosti. Tabuľky a obrázky v tejto kapitole sú ilustračné a slúžia na lepšie vysvetlenie metód.

5.1 Základné rozpoznávanie

Aby sme mohli testovať náš rozpoznávač potrebujeme vedieť, akú hlásku používateľ naozaj povedal a aká hláska je v nahrávke podľa našej klasifikácii. Ak sa rovnajú tak klasifikoval správne, ak nie nesprávne. Klasifikáciu môžeme popísať ako jednoduchú funkciu, ktorá priradí nahrávke x hodnotu $h = K(x)$. Najprv si ukážeme priamočiary spôsob klasifikácie bez rôznych spôsobov vylepšení.

5.1.1 Klasifikácia segmentu

Nazačiatku celá nahrávka prejde procesom reprezentácie, ktorý sme si popísali v predchádzajúcej kapitole. Teda nahrávku rozdelíme na segmenty dĺžky 25 ms a každému segmentu vypočítame MFCC vektor c. Následne si zoberieme všetky uložené modely pre hlásky *a, b, d, ď, e, f, g, h, ch, i, j, k, l, ľ, m, n, ň, o, p, r, s, š, t, ť, u, v, z, ž* a model *ticha*. Teraz každému seg-

mentu vypočítame pre každý model λ hodnotu p, ktorá nám hovorí s akou pravdepodobnosťou patrí daný segment do daného modelu. Na počítanie pravdepodobnosti použijeme viacrozmerné normálne rozdelenie:

$$f\left(c_1, ..., c_{12}\right) = \frac{1}{\sqrt{\left(2\pi\right)^{12}\left|\Sigma\right|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}\left(x-\mu\right)\right) \tag{5.1}$$

A segment klasifikujeme ako hlásku, ktorá mala najväčšiu pravdepodobnosť.

Segment	**Klasifikácia**
...	
i-3	U
i-2	O
i-1	O
i	O
i+1	O
i+2	Ticho
i+3	Ticho
...	

Tabuľka 5.1: Klasifikácia hlásky O

Segment	**Klasifikácia**
...	
i-3	U
i-2	U
i-1	O
i	O
i+1	O
i+2	U
i+3	U
...	

Tabuľka 5.2: Klasifikácia hlásky O

5.1.2 Klasifikácia celej nahrávky

Každému vektoru c sme priradili hlásku h pri ktorej modely mal segment najväčšiu pravdepodobnosť p. Teraz túto postupnosť priradených hlások a ich pravdepodobností musíme interpretovať. Nato môžeme použiť viac spôsobov a my si popíšeme tri.

Prvý spôsob, ktorý určite každého napadne je maximálna početnosť. Spočítame si pre každý model, koľko segmentov sme klasifikovali ako daný model. Nahrávku klasifikujeme ako hlásku, pri ktorej bolo najviac segmentov klasifikovaných ako daný model hlásky. Samozrejme segmenty označené ako

ticho ignorujeme, pretože očakávame, že ich bude najviac, keďže nahrávka má dĺžku 2s a hláska okolo 100ms. Tento spôsob má svoje nevýhody aj výhody. Napríklad môže byť problém, že vyhodnotí postupnosť v tabuľke 5.2 ako u aj keď to malo byť o. Ďalším problémom sú hlásky, pre ktoré sme model nevytvorili, ako q, kde potrebujeme mať nájdené aj k a v. Takéto vyhodnotenie by pri nich dávalo menší zmysel, aj keď môže stále fungovať.

Druhým spôsobom je nájdenie najdlhšej postupnosti. Taktiež segmenty označené *tichom* ignorujeme. Výhoda tohto spôsobu je vidieť v rozpoznávaní už spomínaných zložených hlások ako q, kde ak zistíme, že najväčšia postupnosť je v, môžeme sa pozrieť pred jej začiatok a hladať k. Ak ho nájdeme, tak nahrávku klasifikujeme ako q, ak nie tak ako v. Tento postup môže byť samozrejme otočený, pri nájdení postupnosti k sa pozrieme za jej koniec a budeme hľadať v.

Posledný spôsob, okrem priradenej hlásky h, bude využívať aj pravdepodobnosť p, preto ho budeme nazývať najväčšia pravdepodobnosť. Pre každý model si vypočítame sumu a produkt pravdepodobností segmentov označených daným modelom. Výslednú hodnotu k dostaneme:

$$k = \frac{\sum_{i=1}^{n} p_i}{\prod_{i=1}^{n} p_i} \tag{5.2}$$

Nahrávku vyhodnotíme modelom pre ktorý je výsledná hodnota k najväčšia.

5.2 Experimenty na zvyšovanie úspešnosti rozpoznávania

Popísané rozpoznávanie by mohlo fungovať, keby sme predpokladali, že okrem rozprávajúceho nebude počuť v nahrávke nič iné. To však nie je reálne a vo väčšine prípadov môže niekto v pozadí rozprávať alebo môže byť v pozadí šum. Už vieme, že ak rozpráva niekto zároveň s používateľom rozpoznávača,

tak môže viesť k strate informácií a s tým nedokážeme nič urobiť. Preto sa pokúsime zlepšiť aspoň ostatné prípady.

5.2.1 Modelovanie pozadia

Prvou metódou na zvýšenie úspešnosti, ktorý vyskúšame, je modelovanie pozadia. Už sme si jedej model pozadia, ktorý sme nazvali *ticho*, vytvorili. Ten však nedokáže pokryť všetky pozadia, pretože sme sa snažili aby nahrávky z ktorých bol vytvorený, obsahovali čo najnižšie množstvo šumu. Tento model sa však môže stať nepoužiteľným, keďže stačí zmeniť miestnosť v ktorej sa nachádzame a môže sa stať, že žiadny segment nebude klasifikovaný ako *ticho*, aj keď vo veľkej časti nahrávky nikto nerozprával. A to môže viesť k zhoršeniu rozpoznávania.

Na vyriešenie tohto problému sa pokúsime namodelovať pozadie pri štarte aplikácie. Získame jednu nahrávku, pri ktorej budeme tak trochu dúfať, že používateľ nič nevravel a že zachytíme šum pozadia. Z tejto nahrávky potom vytvoríme samostatný model, ktorý budeme nazývať *pozadie*. Nevytvoríme ho rovnako ako predchádzajúce modely, ale bude to podobné. Aby sme zabránili pokazeniu modelu v prípadoch, kde napríklad polovicu nahrávky niekto rozpráva, použijeme vážený priemer a rozptyl. Priemer teda vypočítame nasledujúcou rovnicou:

$$\mu = \frac{\sum_{j=1}^{n} p_j c_j}{\sum_{j=1}^{n} p_j} \tag{5.3}$$

kde p_j je pravdepodobnosť, že vektor c_j patrí do modelu *ticha*. A rozptyl Σ vypočítame vzťahom:

$$\Sigma = \frac{\sum_{j=1}^{n} p_j \left(c_j - \mu\right)^T \left(c_j - \mu\right)}{\sum_{j=1}^{n} p_j} \tag{5.4}$$

kde opäť budeme potrebovať len diagonálu matice Σ . Ováhovaním pravdepodobnosťou p_j zabezpečíme, že segmenty v ktorých sa rozpráva, budú mať

malú váhu pri vytvorení modelu *pozadia* . V tabuľke 5.3, môžeme vidieť, že

Segment	Klasifikácia
...	
i-3	D
i-2	S
i-1	S
i	S
i+1	A
i+2	A
i+3	Ticho
...	

Tabuľka 5.3: Klasifikácia bez namodelovaného pozadia

Segment	Klasifikácia
...	
i-3	Pozadie
i-2	Pozadie
i-1	Pozadie
i	S
i+1	A
i+2	A
i+3	Pozadie
...	

Tabuľka 5.4: Klasifikácia s modelom pozadia

rozpoznávač s nedostatočne mohutným *tichom*, rozpozná väčšinu segmentov ako nejakú hlásku. Naopak v tabuľke 5.4 vidíme ako túto nepresnosť môže pomôcť odstrániť nový model.

5.2.2 Vytvorenie modelu ə

Druhá metóda súvisí so zvukovou stránkou spoluhlások. Pri vyslovení niektorých spoluhlások ako napríklad *t, d, p, k* atď., môžeme počuť vyslovenie ə, čo je po zvukovej stránke podobné samohláske *e*. No nie je to úplne to isté. Preto si vytvoríme spolu s ostatnými modelmi ešte jeden model. Tento model sa nebude vytvárať za behu aplikácie, ale vytvorí sa z trénovacích vzoriek. Budeme ho označovať model *ee*. Takisto ako model *ticha* a *pozadia* ho budeme pri klasifikácii ignorovať.

Touto metódou chceme dosiahnuť, aby sme spoluhlásky neklasifikovali ako *e*. V tabuľke 5.5 vidíme nahrávku, ktorá by bola klasifikovaná ako *e*, keďže samotná spoluhláska je veľmi krátka. V tabuľke 5.6 je pridaný nový

model *ee*, ktorý nám pomohol zistiť, že v skutočnosti bolo vyslovené *k*. Ak sú modely *e, ee* vytvorené kvalitne, tak by nemalo nedochádzať k pokazeniu rozpoznávania hlások *e*.

Segment	Klasifikácia
...	
i-2	K
i-1	K
i	E
i+1	E
i+2	E
...	

Tabuľka 5.5: Klasifikácia bez modelu EE

Segment	Klasifikácia
...	
i-2	K
i-1	K
i	EE
i+1	EE
i+2	EE
...	

Tabuľka 5.6: Klasifikácia s modelom EE

5.2.3 Orezávanie nahrávky za pomoci energie segmentu

Ďalšia metóda orezávanie sa pokúša odhadnúť v ktorej časti nahrávky sa rozpráva. Následne túto časť oddelí a klasifikácia sa aplikuje len na ňu. Táto metóda navyše zmenší čas celého procesu rozpoznávania. Nahrávka trvá dve sekundy a z toho samotná hláska len 50 až 150 ms., preto je zbytočné rozpoznávať aj segmenty mimo tohto výseku.

Ako už vieme, okrem vektoru MFCC si počítame aj energiu segmentu. A práve túto energiu teraz využijeme. Je zjavné, že reč človeka za mikrofónom bude mať väčšiu energiu ako samotné pozadie. Navyše ak si prezrieme vypočítanú energiu zistíme, že aj reč človeka v pozadí, ktorý rozpráva nahlas, je oveľa nižšia. Preto táto metóda dokáže dosť presne určiť, v ktorej časti používateľ rozprával.

Nazačiatku si nájdeme segment s_j s najväčšou energiou $maxE$. Následne normalizujeme všetky hodnoty energie $e_i = e_i/maxE$. Tento segment bude

označovať koniec výseku. Začiatok výseku určíme tak, že pôjdeme od segmentu s maximálnou hodnotou energie späť a budeme pridávať segmenty až pokiaľ nájdeme segment pre ktorý platí $e_i < 0.01$ a teda segment e_{i+1} bude určovať začiatok výseku. Normalizovaním energie sme zabezpečili, že všetky hodnoty e_i budú v rozmedzí 0 až 1.

Náš prístup samozrejme nie je dokonalý, môže sa stáť, že energia segmentov pred segmentom s_j bude vysoká a úsek ostane stále veľký. Tiež môže byť problém, že sme koniec výseku určili skôr ako sa hláska skončila. Našťastie to ovplyvní skôr samohlásky, ktoré sú na tom lepšie ako spoluhlásky, ktoré by posunutie konca kazilo. V prípade, že by bol výsek veľmi krátky, posunieme koniec, aby mal výsek aspoň 100ms.

Segment	**Klasifikácia**	**Energia**
...		
i	H	<0.01
i+1	CH	0.1
i+2	CH	0.5
i+3	CH	0.9
j	H	1
...		

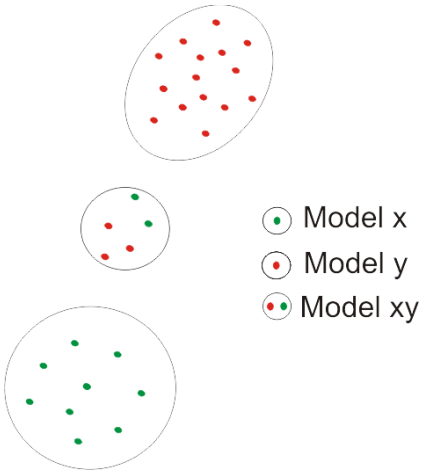
Tabuľka 5.7: Klasifikácia celého segmentu pred orezávaním

Segment	**Klasifikácia**	**Energia**
i+1	CH	0.1
i+2	CH	0.5
i+3	CH	0.9
j	H	1

Tabuľka 5.8: Klasifikácia malého výseku

5.2.4 Klástering

Klástering bude posledná metóda, ktorú si popíšeme. Táto optimalizačná metóda sa aplikuje na naše natrénované modely. Spočíva vo fakte, že niektoré modely nie sú úplne čisté a niektoré modely hlások sú veľmi blízko pri sebe.



Obr. 5.1: Ukážka clusterov

Pri tejto metóde vieme vymyslieť veľa obmien. Pri všetkých postup začína tak, že zoberieme modely a všetky vektory z ktorých boli vytvorené. Majme modely dvoch hlások, ktoré označíme x a y. Každému vektoru v vypočítame pravdepodobnosť, ktorá hovorí, že patrí do daného modelu. Ak je pravdepodobnosť pre model x väčšia ako pre model y tak označíme vektor v_x, inak v_y. Keďže tieto dva modely mohli byť blízko pri sebe, môže sa stať, že niektoré vektory, ktoré boli použité na vytvorenie modelu x majú väčšiu pravdepodobnosť pre y a z v_x sa stalo v_y, alebo naopak. Takýto vektor budeme označovať $v_{x->y}$. A teraz môžeme znovu vytvoriť modely x a y viacerými spôsobmi. Môžeme vytvoriť model x len z vektorov v_x alebo z vektorov v_x a $v_{y->x}$. Taktiež môžeme vytvoriť nové modely, napríklad model xy z vektorov $v_{y->x}$, $v_{x->y}$ ako vidíme na obrázku 5.1, ktorý by sme pri klasifikácii ignorovali. Alebo vytvoriť model, x_2 z vektorov $v_{x->y}$, ktorý by slúžil na klasifikáciu konkrétnej hlásky spolu s novovytvoreným modelom x. V tabuľkách 5.9 a

Segment	Klasifikácia
i-4	CH
i-3	CH
i-2	CH
i-1	K
i	CH
i+1	CH
i+2	CH
i+3	K
i+4	B
i+5	EE
i+6	EE
i+7	EE

Tabuľka 5.9: Pred clusterizáciou

Segment	Klasifikácia
i-4	CH
i-3	CH
i-2	CH
i-1	CH
i	CH
i+1	CH
i+2	CH
i+3	CH
i+4	EE
i+5	EE
i+6	EE
i+7	EE

Tabuľka 5.10: Po clusterizácii

5.10 môžeme vidieť ako by vylepšenie modelov mohlo pomôcť.

Kapitola 6

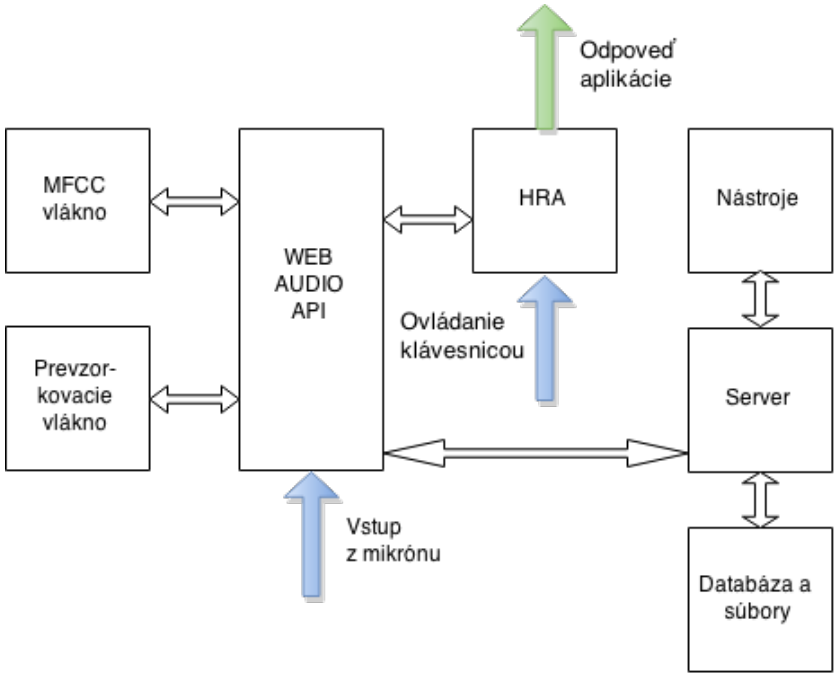
Implementácia

V predchádzajúcich kapitolách sme si vysvetlili ako samotné rozpoznávanie prebieha. V tejto kapitole sa pozrieme na návrh našej aplikácie, čo všetko dokáže. Začneme popísaním logických častí a ich komunikáciou a skončíme popísaním parametrov.

6.1 Návrh a použité nástroje

Náš program sa neskladá len zo samotnej aplikácie bežiacej na strane používatela, kde sa bude využívať rozpoznávač. Okrem aplikácie nám beží server na stiahnutie aktuálnych modelov a ukladanie nahrávok. Taktiež samotná hra nie je len hotový produkt, ale aj spôsob ako nazbierať dostatok nahrávok na vytvorenie modelov. Tiež sme vytvorili nástroje na prácu s nahrávkami a testovanie rozpoznávača. Na obrázku 6.1 môžeme vidieť všetky komponenty z ktorých sa náš program skladá. Šípky medzi nimi znázorňujú komunikáciu a plné šípky znázorňujú vstup od používateľa a výstup.

Hra je implementovaná s kombináciou html, css, javascript. Vďaka tomu všetko sa všetko počíta u používateľa v prehliadači. Na prácu so vstupným signálom používame knižnicu WebAudioApi [5]. Táto knižnica je jedna z najlepších čo sa týka práce so zvukom v javascripte. Na prevzorkovanie a počítanie vektorov MFCC sme použili vlákna, v javascripte sa nazývajú Worker.



Obr. 6.1: Všetky komponenty

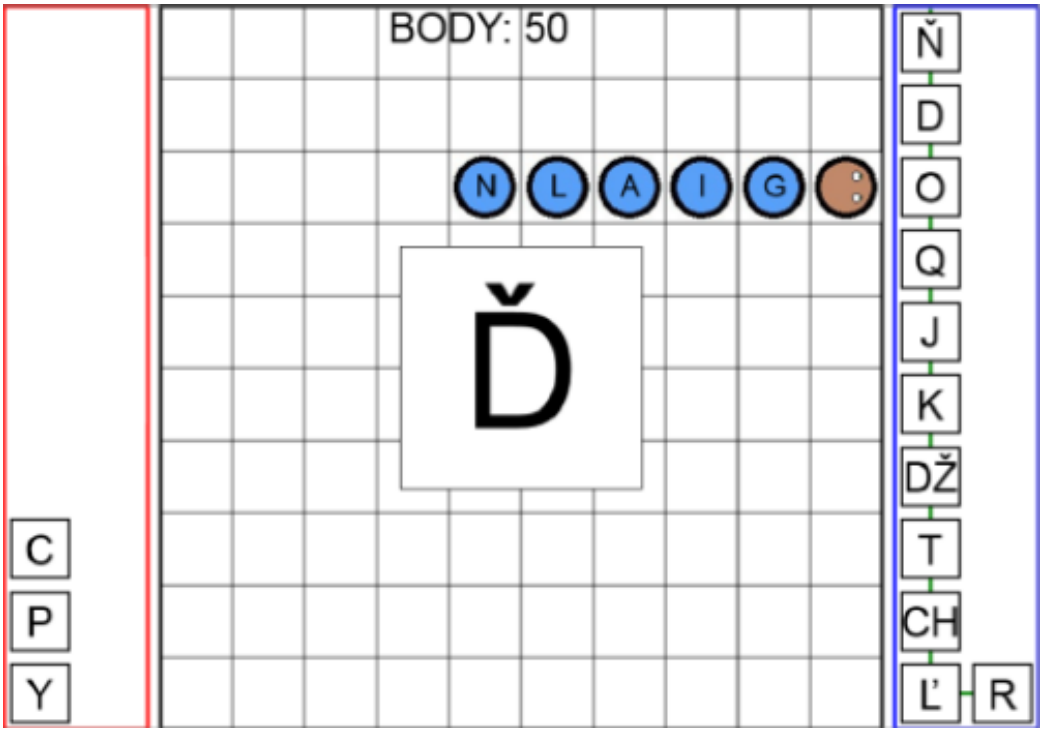
Tým sme docielili paralelizáciu a zabránili aby spracovávanie brzdilo samotnú hru.

Server je naprogramovaný v nodejs a komunikuje s hrou prostredníctvom socket.io [3]. Server je napojený na databázu MySQL. Samotné nástroje síce majú jednoduchý user interface pre administrátora na strane klienta, ale spracovávanie nahrávok a testy bežia na servery. Okrem nami vytvorených nástrojov, sme na prácu s nahrávkami používali program Audacity.

Aplikácia bude súčasť multimediálnej čítanky [6], ako jedna z mnohých, ktoré sa tam dajú spustiť.

6.1.1 Komponenty

Komponenty programu môžeme rozdeliť na tri časti. Prvou časťou je samotná hra, ktorá rozpoznáva reč, spracováva vstup z klávesnice, riadi stav hry a dáva odpoveď užívateľovi či správne vyslovil hlásku. Pri spustení aplikácie, sa aplikácia napojí na server a vyžiada si modely. Po ich prijatí zapne mikrofón. Ak by prehliadač náhodou zablokoval mikrofón, aplikácia sa nespustí, pretože vstup z mikrofónu je hlavnou časťou aplikácie.



Obr. 6.2: Ukážka hry

Žiaci základnej školy hru snake nemusia poznať. Vybrali sme ju, pretože aj keď ju nepozná, je ľahká na pochopenie. Žiak ovláda hada šípkami a snaží sa nenaraziť do svojho chvosta. Keď zje písmeno, hra sa zastaví a písmeno sa zväčší. V tom momente musí písmeno prečítať. Ak ho prečíta správne pozadie písmena ozelenie na dobu jedenej sekundy, had sa zväčší o dané písmeno a

hráčovi sa pripočíta 10 bodov. Ak ho prečíta nesprávne pozadie očervenie, nedostane body a písmeno sa ukáže v ľavom stĺpci, ktorý je vidieť na obrázku 6.2.

Po technickej stránke čítanie písmena znamená zmena stavu. Trieda hry pošle informáciu do komponentu WebAudioApi. Ten vždy keď dostane nové dáta z mikrofónu, pošle správu vzorkovaciemu vláknu. Toto vlákno vstupné dáta prevzorkuje a rozdelí na segmenty požadovanej veľkosti. Každý vytvorený segment pošle ako správu späť. V prípade zbierania nahrávok, sa prezorkovaná nahrávka pošle na server, kde sa uloží na neskoršie spracovanie. V prípade hrania sa každý segment pošle do MFCC vlákna a vráti sa z neho vektor reprezentujúci tento segment a energie segmentu. Ak chceme, aj tieto vektory si môžeme rovno ukladať na server. V prípade, že to bol posledný vektor, pošleme všetky vektory s prislúchajúcimi energiami na klasifikáciu. Potom nám už stačí aby sme poslali správu späť hernej triede, ktorá zobrazí odpoveď.

Druhou časťou je samotný server. Ak sa používatelia pripoja len s cieľom hrať, tak server nič nepočíta. Odošle stránku a pri vyžiadaní pošle modely. Ak aplikáciu používame na zbieranie nahrávok, tak server tiež čaká na nahrávky, ktoré uloží na disk. Pokiaľ sa nepripojí administrátor, aby pracoval s nahrávkami, tak server je minimálne zaťažený.

Poslednou časťou je komponent obsahujúci nástroje na prácu s nahrávkami. Tieto nástroje by sme mohli rozdeliť tiež na viac skupín. Najprv si, ale povieme aké súbory sa nachádzajú na disku. Sú to *.raw* súbory, ktoré obsahujú celú nahrávku. Ďalej *.lab* súbory, obsahujúce informácie o danej nahrávke. Taktiež máme uložené *.mfcc* súbory, ktoré obsahujú vypočítané vektory MFCC, pre konkrétnu nahrávku. Prvá skupina nástrojov slúži na prechádzanie stromovej štruktúry disku a prezeranie si *.lab* a *.mfcc* súborov. Ďalšia skupina je na počúvanie *.raw* súborov. Vyžiadame si súbor od servera, ktorý nám ho pošle aj spolu jeho *.lab* súborom a my si ho vypočujeme. Máme možnosť súbor vymazať, vypočuť si ho znova, upraviť a zmeniť niektoré informácie v jeho *.lab* súbore. Táto skupina nástrojov slúži ako filter, na rýchle

odstraňovanie zlých nahrávok. Posledná skupina nástrojov slúži na prácu so všetkými nahrávkami. Máme funkciu na rozdelenie nahrávok na trénovacie a testovacie vzorky. Ďalej funkciu na vytvorenie modelov z trénovacích vzoriek. A nakoniec funkciu na testovanie vzoriek.

6.2 Parametre aplikácie

Cieľom implementácie je aplikácia, pre ktorú sa zapne server a už sa nemusíme o nič starať. To ešte neznamená, že sme vedeli všetky detaily už pri programovaní samotnej hry. Preto sme sa snažili aplikáciu programovať parametricky, aby sme mohli veci zmeniť len na jednom mieste a aplikácia sa bude chovať ako potrebujeme. Teraz si popíšeme niekoľko parametrov, ich nami nastavenú hodnotu a povieme si čo ovplyvňujú.

UPDATE_ RATE = 500 je čas v milisekundách, ako často sa hra aktualizuje a had sa pohne. Tento čas je zámerne dlhý, aby deti nemali problém s rýchlosťou hada. Obyčajne sa tento čas znižuje s časom, ale to nie je náš prípad, keďže ide primárne o zlepšovanie čítanie.

READING_UPDATES = 5 je počet aktualizácií, počas ktorých vidíme pred sebou písmeno na prečítanie. Pri našej hodnote **UPDATE_RATE** to je 2.5s.

ANSWER_UPDATES = 1 je počet aktualizácií na zobrazenie odpovede.

RECORD_TIME = READING_UPDATES * UPDATE_RATE - 300 je skutočný čas na prečítanie, pretože potrebujeme čas na klasifikáciu a nechceme hru prerušovať. V našom prípade dostaneme 2.2s, čo je opäť dosť veľa na hlásku trvajúcu 100ms, ale z nahrávok sme zistili, že hodnoty **READING_UPDATES** a **RECORD_TIME** sú tak akurát, aby deti

stíhali prečítať.

LETTERS = [′**a**′,′ **b**′,′ **c**′,′ **č**′,′ **d**′,′ **ď**′,′ **dz**′,′ **dž**′,′ **e**′,′ **f**′,′ **g**′,′ **h**′,′ **ch**′,′ **i**′,′ **j**′,′ **k**′,′ **l**′, ′**ľ**′,′ **m**′,′ **n**′,′ **ň**′,′ **o**′,′ **p**′,′ **q**′,′ **r**′,′ **s**′,′ **š**′,′ **t**′,′ **ť**′,′ **u**′,′ **v**′,′ **x**′,′ **y**′,′ **z**′,′ **ž**′] sú všetky písmená, ktoré sa v hre môžu objaviť.

GAME_LENGTH = 20 je počet písmeniek, ktoré sa prichystajú. Ak niekto prečíta posledné písmeno, hra sa ukončí. Samozrejme môže pokračovať, dostane ďalších 20 písmeniek a nahraté body mu ostanú.

UNIQUE = TRUE zabezpečí, že medzi pripravenými písmenami nebudú duplikáty.

WINDOW_LENGHT = 25 je dĺžka jedného segmentu v milisekundách. Pracovali sme aj s hodnotou 100, ale to je veľmi veľa.

SAVE_RAW = FALSE nám hovorí, či sa každá nahrávka pošle na server a uloží. Samozrejme sme ju používali, len počas trénovacej fázy, preto ju máme teraz nastavenú na **FALSE**.

SAVE_MFCC = FALSE && SAVE_RAW nám hovorí, či sa okrem na hrávky pošlú na server aj vypočítané vektory MFCC. Ak sa neposielajú samotné nahrávky, tak sa vektory nebudú ukladať aj keď nastavíme **TRUE**.

Okrem hlavných parametrov, existuje ešte pár ako nová vzorkovacia frekvencia, počet filtrov Mel škály a hranice energie škály. Tieto hodnoty sme však nastavali dopredu a nechceme ich meniť. Ich konkrétne hodnoty sme spomenuli v kapitole o reprezentácii hlások.

Tiež ešte máme pár hodnôt na ovplyvňovanie plátna, na ktoré sa hra vykresľuje. Napríklad jeho šírka, výška a veľkosť jedného štvorčeka.

Kapitola 7

Výsledky

V tejto poslednej kapitole sa pozrieme na výsledky testov. Povieme si, čo šlo dobre a čo naopak by sa dalo zlepšiť.

7.1 Testy

Pred samotnými výsledkami si povieme niečo viac o testoch. Nahrávky sme získali od prvákov základnej školy, konkrétne od piatich tried 1.A, 1.B, 1.C, 2.A a 3.C. Vďaka tomu môžeme rovno nahrávky rozdeliť na trénovacie a testovacie. Pri testoch budeme mať dve rôzne rozdelenia. Prvé rozdelenie používa nahrávky tried 1.B, 1.C a 3.C ako trénovacie a nahrávky 1.A, 2.A ako testovacie. Toto rozdelenie môžeme označiť ako najťažšie, keďže žiadny žiak sa nevyskytuje v oboch skupinách. Budeme ho nazývať testovanie s testovacími nahrávkami. Druhé rozdelenie bude používať 1.B, 1.C a 3.C ako trénovaciu a testovaciu zároveň. Toto rozdelenie by malo priniesť najlepšie výsledky, keďže nie len sú tí istí žiaci v trénovacej a testovacej skupine, ale zároveň tie isté nahrávky ktoré boli použité na vytvorenie modelov, budú použité pri testoch. Budeme ho nazývať testovanie na trénovacích nahrávkach.

V tabuľke 7.1 môžeme vidieť počty nahrávok použitých na trénovanie a testovanie. K testovaniu je ešte pridaný počet vektorov, ktoré boli použité na vytvorenie daného modelu. Pretože aj keď nahrávok môže byť dostatok,

ešte to nemusí znamenať, že aj vektorov je dosť. V niektorých pripadoch znamená jedna nahrávka len jeden alebo dva vektory, dokonca sú prípady, kde z nahrávky nemáme ani jeden vektor.

Pripomeňme si, že veľkosť jedného segmentu je 25 ms. Zo začiatku sme skúšali aj veľkosť 100 ms. Na samohlásky by to mohlo stačiť, ale na spoluhlásky určite nie. S takým veľkým segmentom sa stratí dosť veľa informácií a to je dôvod prečo sa také veľké segmenty nikde nepoužívajú. Samotný spôsob klasifikácie bude spresnený pri každom teste.

Na porovnanie výsledkov sa môžeme pozrieť napríklad na dizertačnú prácu Mareka Nagyho [8]. Síce rozpoznáva slová, čo by sa mohlo zdať ťažšie, ale pri slovách sa dá využiť kontext. Preto na porovnanie, sa treba pozrieť na výsledky rozpoznávania samotných hlások.

7.1.1 Samohlásky

Prvým test bude len rozpoznávať samohlások, takže zoberieme len modely určené pre samohlásky a ticho. Na klasifikáciu nahrávky sme vyskúšali všetky tri spôsoby a všetky mali úplne rovnaké výsledky. V tabuľke 7.2 môžeme vidieť výsledky tohto testu.

7.1.2 Všetky hlásky

Ďalej budeme testovať rozpoznávanie všetkých hlások. Nebudeme uvádzať úspešnosť všetkých hlások, ale radšej uvedieme úspešnosť pri použití rôznych klasifikácií. Taktiež uvedieme úspešnosť pri testovaní na testovacích aj trénovacích nahrávkach. V tabuľke 7.3 vidíme, že pri rozpoznávaní všetkých hlások, majú všetky tri spôsoby klasifikácie podobnú úspešnosť. Stĺpec **pred klásteringom** používa orezávanie, model pozadia, model *ee* a stĺpec **po klásteringu** obsahuje výsledky po aplikovaní klásteringu na niektoré dvojice hlások. Použitá verzia klásteringu vyčistí konkrétne modely x, y a vytvorí aj modely x_2 a y_2.

V tabuľke 7.4 sme rozdelili hlásky do troch skupín. V prvej skupine sú

hlásky *a, e, i, o, u, ľ, ň, s, ch*, ktoré majú najvyššiu úspešnosť. V druhej sú *h, j, k, l, m, n, p, r, ť, v, š, z, ž* a v poslednej *b, f, g c, č, d, ď, dz, dž, q, t, x*. V tejto tabuľke uvádzame úspešnosť skupiny pri použití všetkých modelov ako v teste v tabuľke 7.3 a úspešnosť skupiny pri použití len tých modelov hlások, ktoré patria do danej skupiny. Tento krát ukážeme len výsledky pri použití pravdepodobnosti ako našej metódy klasifikácie.

Rozdelenie na skupiny nám ukázalo, že prvá skupina je použiteľná a druhá skupina po vyčistení a dotrénovaní modelov by bola tiež dobrá. Úspešnosť pri použití modelov danej skupiny sa nemôže brať ako výsledok rozpoznávača, keďže rozpoznávač nevie aká hláska sa má povedať a musí použiť všetky modely. Ale tento test nám môže pomôcť v analyzovaní. Napríklad úspešnosť hlásky *g*, ktorá je v poslednej skupine, značne poskočí, keď používame menej modelov. To znamená, že by ju išlo rozpoznavať celkom dobre, keby bol jej modely a modely hlások jej podobné lepšie.

Problémová tretia skupina obsahuje hlásky, ktorých výslovnosť je veľmi krátka ako napríklad hlásky *t, d, f, b* a hlásky, ktoré sú zložené, teda nemajú svoj model. Ich rozpoznanie je teda podmienené kvalitou dvoch hlások, kde vždy jedna má malú úspešnosť. Napríklad *dz, dž* je problém rozpoznať, keď nevieme rozpoznať ani samotné *d*.

7.2 Zhrnutie výsledkov

V tabuľkách sme videli, že úspešnosť rozpoznávania jednotlivých hlások je rôzna. Ideálne by teda bolo vytvoriť podmnožinu hlások, ktorú by sme mohli rovno použiť v hre a zvyšné modely hlások sa snažiť vylepšiť na dostatočnú úroveň. Pri pozeraní na počty vektorov v tabuľke 7.1 a výsledky v tabuľke 7.4, nepriamo vidíme koreláciu medzi nízkym počtom vektorov hlásky a jej úspešnosťou. To však ešte nemusí znamenať, že by sa úspešnosť všetkých hlások dala dostať na úspešnosť samohlások.

Ako sme spomenuli, rozdelenie na skupiny v tabuľke 7.4 a testovanie len s modelmi danej skupiny, nie je na reálne testovanie samotného rozpoznávača

možné, ale mohlo by sa využiť v situácii, kde vieme čo ideme testovať. A to je prípad našej aplikácie. Napríklad ak ideme rozpoznávať hlásku *ch*, tak by sme použili modely *ch, h, k, g, ee, pozadia*. Takto by sme mohli ľahšie odhaliť, či dieťa vyslovuje hlásku korektne a tiež pri vylepšovaní modelov lepšie vidieť či chybovosť klasifikácie klesá.

Nakoniec teda vidíme, že rozpoznávanie izolovaných hlások, nie je jednoduché, keďže si nevieme pomôcť kontextom a slovníkom. Našťastie aplikácia má iné testovacie podmienky, ako sme použili na testovanie úspešnosti rozpoznávača, a teda by sme ju mohli upraviť aby bola použiteľná v praxi.

Písmeno	**Na testovanie**	**Na trénovanie**	**Vektorov na trén.**
a	8	11	140
b	12	21	18
c	16	-	-
č	7	-	-
d	10	15	47
ď	5	10	23
dz	5	-	-
dž	9	-	-
e	6	12	138
f	11	6	58
g	10	19	38
h	7	15	79
ch	7	10	98
i	7	7	72
j	9	14	103
k	10	13	37
l	10	16	138
ľ	8	9	80
m	14	15	108
n	10	12	118
ň	6	9	49
o	5	10	115
p	7	17	11
q	1	-	-
r	7	4	24
s	8	15	159
š	12	10	118
t	13	14	14
ť	12	11	26
u	4	10	113
v	13	10	30
x	3	-	-
z	14	9	85
ž	7	25	184

Tabuľka 7.1: Počty nahrávok a vektorov

Písmeno	**Úspechov**	**Pokusov**	**Úspešnosť v %**
a	8	8	100
e	6	6	100
i	6	7	85
o	4	5	100
u	4	4	100
všetko	28	30	93

Tabuľka 7.2: Úspešnosť rozpoznávanie samohlások

Spôsob klasif.	**Úsp. na test. v %**		**Úsp. na trén. v %**	
	Pred klást.	**Po klást.**	**Pred klást.**	**Po klást.**
Pravdepodobnosť	33.44	34.81	47.36	47.6
Postupnosť	34.47	36.18	45.78	45.2
Početnosť	33.44	33.78	44.21	45.79

Tabuľka 7.3: Úspešnosť rozpoznávania všetkých hlások

Skupina	**Úspešnosť pri všetkých modeloch v %**	**Úspešnosť pri modeloch skupiny v %**
1.	71	86
2.	39	53
3.	4	18

Tabuľka 7.4: Úspešnosť samotných skupín

Záver

Cieľom mojej bakalárskej práce bolo implementovať rozpoznávač hlások a použiť metódy, ktoré by viedli k zvýšeniu úspešnosti rozpoznávania. Taktiež bol cieľ vytvoriť aplikáciu - hru, ktorá by tento rozpoznávač používala.

Na výsledkoch je vidieť, že úspešnosť rozpoznávača nie je najväčšia, ale to sa dalo očakávať, keďže rozpoznávanie izolovaných hlások nie je jednoduché. Vyskúšal som pomerne dosť obmien spomenutých metód na rozpoznávanie a pri niektorých bolo dosiahnuté aspoň malé zlepšenie. Za najúspešnejšiu považujem orezanie začiatku a konca nahrávok, pretože vo väčšine prípadov je nahrávka orezaná tak, že hlas ostaných detí, ktoré hovoria v pozadí v iných časoch, je odrezaný.

Aplikácia je fungujúca a hrateľná a po pri nej je vytvorených veľa nástrojov na prácu s nahrávkami. Rozpoznávač je reálne použiteľný v aplikácii pre samohlásky a niektoré spoluhlásky. Na niektoré spoluhlásky by rozpoznávač niekedy dával nesprávne odpovede a to by sa samozrejme žiakom nepáčilo.

V čom bol podľa mňa najväčší problém? Trénovacie nahrávky neboli čisté, čo by však nevadilo pretože by to v niektorých prípadoch len zväčšilo robustnosť modelov, ale v prípadoch niektorých spoluhlások, kde bolo vzoriek veľmi málo to vytvorilo zlé modely. Najviac by pomohlo dotrénovanie modelov, ktoré ale zaberie dosť času keďže z jednej nahrávky získame tak 1 až 2 vektory pre niektoré modely spoluhlások. Dotrénovanie a vyčistenie modelov by určite zdvihlo celkovú úspešnosť, ale myslím si, že rozpoznávanie niektorých spoluhlások by ostalo stále dosť neúspešné. Stále sa tiež dá pokračovať v obmenách spomenutých metód na zvyšovanie úspešnosti. Napríklad po-

sunúť začiatok a koniec orezávania nahrávky, alebo meniť počty iterácií v klásteringu.

Literatúra

[1] Understanding ffts and windowing. www.ni.com/white-paper/4844/en/pdf.

[2] Guide-mel-frequency-cepstral-coefficients-mfccs, navštívené 2015-05-01. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[3] Socket io, navštívené 2015-05-01. http://socket.io/.

[4] Angel De La Torre, Jose C Segura, Carmen Benitez, Javier Ramirez, Luz Garcia, and Antonio J Rubio. *Speech recognition under noise conditions: Compensation methods*. INTECH Open Access Publisher, 2007.

[5] W3C Audio Working Group. Web audio api, navštívené 2015-05-01. http://www.w3.org/TR/webaudio/.

[6] Marek Nagy. Multimediálna čítanka. http://mmcitanka.sk/.

[7] Marek Nagy. Skryté markovove modely a rozpoznávanie čísiic, 2004. http://dai.fmph.uniba.sk/~mnagy/documents/Nagy2004_rigo.pdf/.

[8] Marek Nagy. Počítačové rozpoznávanie reči a výuka čítania, 2010. https://stella.uniba.sk/zkp-storage/dpg/dostupne/FM/2010/2010-FM-KSRgqg/2010-FM-KSRgqg.pdf/.

[9] Josef Psutka. *Komunikace s počítačem mluvenou řečí*. Academia, 1995.

[10] Ronald F. Ullmann. An algorithm for the fast hartley transform, 1984. `http://sepwww.stanford.edu/theses/sep38/38_29.pdf`.

[11] A. Urbaník. Určovanie veku detí podľa rečového signálu, 2010.

[12] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying A Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book (for htk version 3.4). 2006.