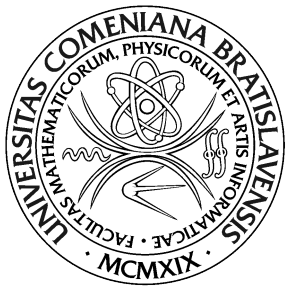
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



EXPERIMENTÁLNY ROZPOZNÁVAČ SPONTÁNNEJ REČI

Diplomová práca

2022 Bc. Andrej Kapusta

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

EXPERIMENTÁLNY ROZPOZNÁVAČ SPONTÁNNEJ REČI

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Marek Nagy, PhD.

Bratislava, 2022 Bc. Andrej Kapusta

Čestne prehlasujem, že túto diplomovú prácu som vypracoval samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa.

. . . . . .

Bratislava, 2022 Bc. Andrej Kapusta

Poďakovanie

Abstrakt

Kľúčové slová: rozpoznávanie hlasu, skryté markovovské modely, neurónové siete, Kaldi

Abstract

Keywords: speech recognition, hide markov models, neural networks, Kaldi

Obsah

1 Úvod

2 Základné koncepty rozpoznávania reči

Oblasť rozpoznávania hovorenej reči je v dnešnej dobe veľmi populárna a to aj napriek tomu, že vývoj v nej sa započal už okolo roku 1950. Jedným z prvých významných rozpoznávačov bol automatický rozpoznávač hlasu, ktorý vytvorili Davis, Biddulph, a Balashek v roku 1952. Rozpoznávač vedel rozpoznávať jednotlivé číslice, a naraz ho mohol používať len jeden používateľ [5]. Doba však pokročila, a v posledných rokoch môžme pozorovať veľký posun v oblasti rozpoznávania reči. Táto technológia začala meniť spôsob, akým žijeme či pracujeme a stala sa jedným z hlavných prostriedkov interakcie ľudí s mobilnými zariadeniami (napr. Siri, Google Now a Cortana). Príchod tohto nového trendu sa pripisuje značnému pokroku dosiahnutému v mnohých oblastiach [8]. Spomínaný posun v rozpoznávaní hlasu je spôsobený aj pokrokom v oblasti strojového učenia a neurónových sietí. V nasledujúcich častiach si priblížime základné pojmy spojené s princípom rozpoznávania reči a popíšeme si jednotlivé fázy v procese rozpoznávania.

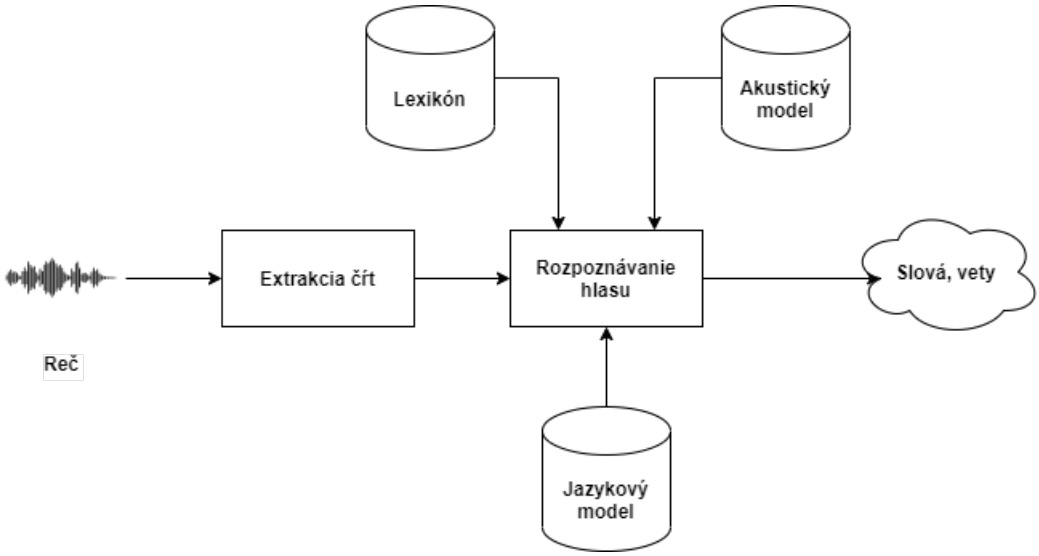
2.1 Základný princíp rozpoznávania reči

Na úvod je potrebné si zadefinovať samotný proces rozpoznávania reči. V jednoduchosti povedané, rozpoznávanie reči je proces transformácie akustického signálu na text. Tento proces vieme rozdeliť do niekoľkých podprocesov, ktoré si postupne predstavíme.

Celý proces je zahájený vznikom, resp. detekciou reči používateľa. V praxi sa môžme stretávať s nástrojmi na rozpoznávanie reči, ktoré pracujú v reálnom čase alebo môžu pracovať s už existujúcou audio, alebo audio-vizuálnou nahrávkou,

ktorú spracujú a vytvoria jej textový prepis. V oboch prípadoch je následný proces rozpoznávania totožný.

Na nasledujúcom obrázku č.1 sa nachádza blokový diagram znázorňujúci postup rozpoznávania reči. Na obrázku si môžme všimnúť, že samotnému rozpoznávaniu hlasu ešte predchádza proces extrakcie čŕt zo vstupnej reči.



Obrázok 1: Základná bloková schéma procesu rozpoznávania reči

Samotné rozpoznávanie je zároveň závislé na akustickom modeli, jazykovom modeli a na lexikóne, pričom tieto komponenty musia byť vytvorené, resp. natrénované ešte pred rozpoznávaním hlasu. Výsledkom rozpoznávania sú nakoniec slová alebo postupnosť slov v podobe viet.

Pozrime sa teraz na rozpoznávanie hlasu z formálnejšieho pohľadu. Cieľom rozpoznávania je predikcia postupnosti slov W

$$W = \{w_1, w_2, ..., w_n\}$$

pre daný signál hovorenej reči X

$$X = \{x_1, x_2, ..., x_n\}$$

pričom cieľom je nájsť najoptimálnejšiu postupnosť slov $\hat{W}$ pri ktorej je zadná pravdepodobnosť najväčšia.

$$\hat{W} = \underset{W}{\operatorname{argmax}} P_{\Lambda,\Gamma}(W|X)$$

V uvedenom vzťahu sa vyskytuje parameter Λ, ktorý reprezentuje akustický model a parameter Γ, ktorý reprezentuje jazykový model. Pri použití Bayesovského pravidla dostane vzťah:

$$P_{\Lambda,\Gamma}(W|X) = \frac{p_\Lambda(X|W)P_\Gamma(W)}{p(X)}$$

, ktorý má po úprave nasledujúcu formu:

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, p_\Lambda(X|W)P_\Gamma(W)$$

,kde $p_\Lambda(X|W)$ je pravdepodobnosť že pre vstupný signál bude predikovaná postupnosť W pri akustickom modely Λ a $P_\Gamma(W)$ je pravdepodobnosť predikovanej postupnosti W pri jazykovom modeli Γ. Takto popísaný model nám hovorí, že sa snažíme maximalizovať pravdepodobnosť, že vstupný signál bude odpovedať vytvorenej predikcii.[8]

2.2 Modely

V predchádzajúcej časti sme dotkli pojmov akustický a jazykový model. Tieto modely hrajú v procese rozpoznávania dôležitú úlohu a preto je potrebné sa pri

nich zastaviť a bližšie si ich špecifikovať.

2.2.1 Akustický model

Akustický model nám poskytuje odhad pravdepodobnosti $P(X|W)$, teda s akou pravdepodobnosťou bol zaznamenaný signál X, ak bola povedaná postupnosť slov W. V tomto momente je potrebné si uvedomiť, spracovávaná reč sa môže značne líšiť, a rôzne slová rôzne znejú a to aj v rámci rovnakého jazyka. Z tohoto dôvodu je potrebné reprezentovať akustický model tak, aby sa vedel vysporiadať s vysokou mierou variability v akustických dátach. Tento problém vedia vyriešiť metódy akustického modelovania, ktorými sú skryté markovovské modely (HMM), hlboké neurónové siete (DNN) alebo ich kombinácia (HMM-DNN).

2.2.2 Jazykový model

Jazykový model je štatistický model, ktorý predstavuje rozdelenie pravdepodobnosti nad sekvenciou slov.

Jazykový model nám hovorí, aká je závislosť medzi slovami alebo skupinou slov, ktoré znejú podobne. V praxi totiž nastáva problém, kedy dve odlišné postupnosti slov znejú podobne, a preto potrebujeme vedieť štatisticky vyjadriť ich závislosť. Rovnako ako pri akustickom modeli, tak aj pri jazykovom je potrebné ho najprv natrénovať na trénovacej sade dát. Vzhľadom na obrovské množstvo všetkých rôznych postupností slov nie je možné disponovať takou trénovacou sadou, ktorá by ich všetky pokryla, a preto by na konci jazykovému modelu chýbalo množstvo postupností. Riešením tohto problému je použitie n-gramov, ktoré definujú pravdepodobnosť postupnosti n za sebou idúcich slov. Ak je $n = 1$, tak potom hovoríme o unigrame, ktorý je vo svojej podstate slovník slov a počtu výskytov konkrétneho slova v danej postupnosti. Spomínané n-gramy sú pre každý jazyk

rôzne a sú vytvárané rôznymi jazykovednými ústavmi a inými inštitúciami. V prípade slovenského jazyka boli vytvorené rôzne n-gramy Jazykovedným ústavom Ľudovíta Štúra Slovenskej akadémie vied v Bratislave, pričom tieto n-gramy sú súčasťou Slovenského národného korpusu.

2.3 Skryté markovovské modely (HMM)

[TEORIA] [OBRAZKY]

2.4 Neurónové siete

[TEORIA] [OBRAZKY]

2.5 Nástroje na rozpoznávanie hlasu

Proces vytvorenia a natrénovania modelu rozpoznávaču hlasovej reči je v celku zložitý. Problematika rozpoznávania hlasu však už nie je nová, a preto máme k dispozícii rôzne nástroje, ktoré nám pri vytváraní modelu vedia pomôcť a uľahčiť prácu. Pokúsili sme sa analyzovať voľne dostupné toolkit-y a zhodnotiť, ktorý z nich je najvhodnejší pre riešenie nášho problému. V Nasledujúcich častiach uvádzame prehľad vybraných najrelevantnejších nástrojov:

- HTK toolkit
- CMU Sphinx
- Kaldi

Pri popise jednotlivých nástrojov uvádzame ich krátku špecifikáciu, funkcionalitu a výhody, prípadne nevýhody vzhľadom na naše potreby.

2.5.1 HTK toolkit

Nástroj HTK toolkit slúži na vytváranie a manipuláciu so skrytými markovovskými modelmi (HMM). HTK sa primárne používa na výskum rozpoznávania reči, aj keď sa používal na množstvo ďalších aplikácií vrátane výskumu syntézy reči, rozpoznávania znakov či sekvenovania DNA.

Tento nástroj patrí k jedným z najznámejších toolkitov, ktoré poskytujú utility na vytváranie modelov pre rozpoznávanie reči. V posledných rokoch už na tomto nástroji nie je aktívny vývoj a posledná významná zmena v podobe pridania podpory pre neurónové siete je z roku 2015. Komunita vývojárov, ktorý na tomto nástroji pracovali, sa dnes podieľajú na vývoji iných známych nástrojov, ako je napríklad CMU Sphinx alebo Kaldi.

Ako už bolo spomenuté, nástroj HTK sa používa hlavne na výskum rozpoznávania reči, a jeho jadrom sú HMM. Okrem toho nástroj poskytuje už aj podporu pre neurónové siete.

HTK toolkit je možné považovať za jednu z vhodných možností pre použitie v prípade výskumu v oblasti rozpoznávania reči. Poskytuje významnú funkcionalitu predovšetkým s podporou HMM a dodnes je využívaný na pozadí mnohých webových stránok. Jeho nevýhodou je, že na ňom neprebieha aktívny vývoj a s tým súvisí aj znížená aktivita komunity používateľov. Z tohto dôvodu sme sa rozhodli, že nástroj HTK v ďalšie fáze tejto práce nepoužijeme.

2.5.2 CMU Sphinx

CMU Sphinx predstavuje skupinu rôznych vývojárskych knižníc a nástrojov zameraných na rozpoznávanie reči, ktorých spojením môžme vytvárať aplikácie orientované na prácu s hovorenou rečou. Vyvíjaný začal byť okolo roku 2015, no v poslednom období neboli zaznamenané nové zmeny či funkcionality. Tento ná-

stroj sa skladá z nasledujúcich štyroch častí a každá z nich poskytuje odlišnú funkcionalitu.

- Pocketsphinx — jednoduchá knižnica na rozpoznávanie v jazyku C
- Sphinxbase — pomocná knižnica, ktorú využíva Pocketsphinx
- Sphinx4 — nastaviteľný rozpoznávač implementovaný v jazyku Java
- Sphinxtrain — trénovacie nástroje pre akustický model

Podporuje množstvo jazykov ako je americká a anglická angličtina, francúzčina, mandarínčina a iné, no zároveň je možné vytvoriť nové modely aj pre iné jazyky. CMU Sphinx podporuje skryté markovovské modely ale aktuálne nemá podporu hlbokých neurónových sietí [1].

2.5.3 Kaldi

Kaldi je open-source nástroj, ktorý je určený na prácu s dátami, ktoré obsahujú hovorenú reč. Svoje uplatnenie našiel predovšetkým v oblasti rozpoznávania hlasovej reči a rozpoznávania hovoriacich osôb. Nástroj Kaldi je vyvíjaný už od roku 2013 a neustále na ňom participuje široká komunita vývojárov z oblasti rozpoznávania reči. Jadro tohto nástroju je implementované v jazykoch C/C++ a Fortran, avšak pre jeho lepšiu použiteľnosť je vytvorené rozhranie v podobe Bash-u a Python skriptov [3][9].

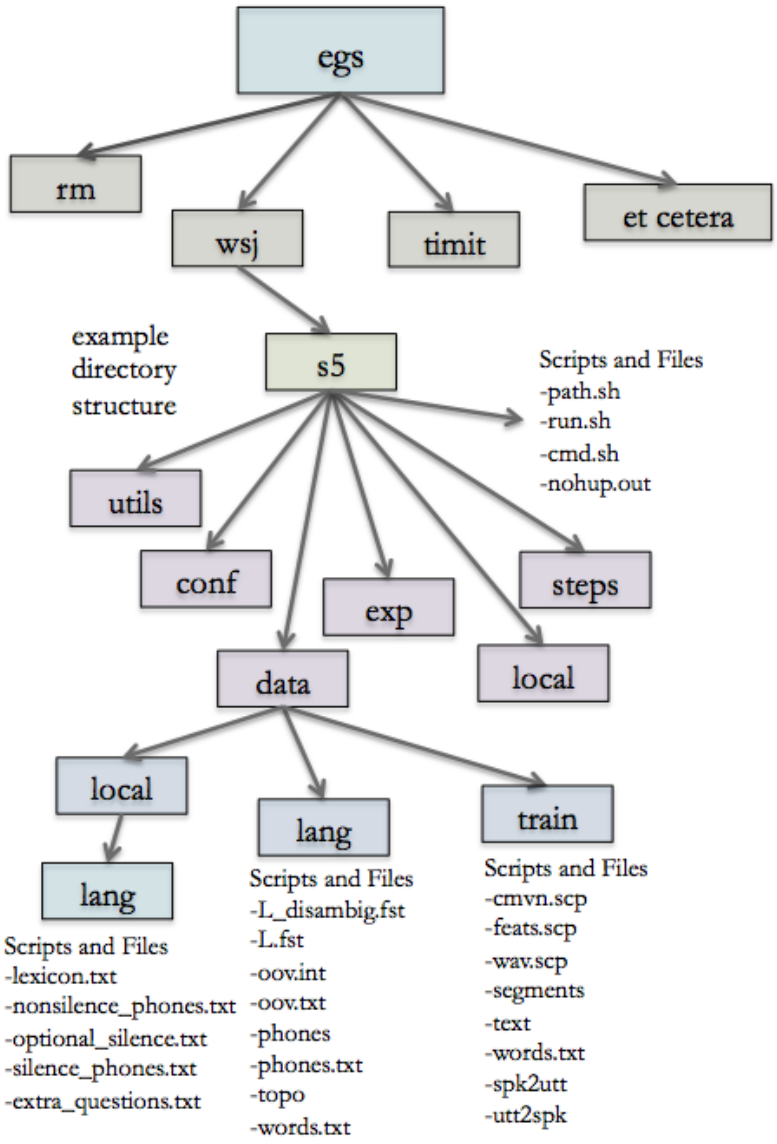
Medzi prednosti Kaldi patrí predovšetkým jeho využiteľnosť a široká škála rôznych pomocných programov v podobe prítomnosti skrytých markovovských modelov a neurónových sietí, ktoré v prípade zvyšných nástrojoch nie sú vôbec, prípadne nie sú dotiahnuté na takú úroveň, ako je to v prípade nástroju Kaldi.

Z analýzy predchádzajúcich nástrojov pre nás vyplýva, že nástroj Kaldi bude najlepšou voľbou pri tvorbe rozpoznávaču spontánnej hlasovej reči.

Nakoľko sme sa rozhodli pre použitie tohto nástroju, pri jeho analýze sme zašli ešte hlbšie. Cieľom bolo pozrieť sa na jeho fungovanie a súborovú štruktúru, ktorej znalosť je kľúčová pre ďalšiu prácu či experimenty. Na nasledujúcom obrázku č.2 si môžme všimnúť súborovú štruktúru, rozvíjajúcu sa od adresáru *egs*. Adresár *egs* patrí medzi adresáre najvyššej úrovne, medzi ktoré ďalej patria aj *src, tools, misc* a *windows*. Adresár *egs* sa vyznačuje tým, že sa v ňom nachádzajú rôzne príklady trénovania rozpoznávačov, ktoré využívajú napríklad HMM alebo aj neurónové siete. Dostupné príklady využívajú korpus Wall Street Journal Corpus (*wsj*), TIMIT (*timit*), Resource Management (*rm*) ale aj mnohé iné. Po zvolení jedného z korpusov je k dispozícií viacero verzií označených ako *s1, s2* a pod. Verzia s najvyšším číslo je však považovaná za najaktuálnejšiu.

Na obrázku č.2 môžeme ďalej vidieť, po zvolení verzie ďalšiu úroveň adresárov a podadrsárov. O niečo podrobnejšie ich spomenieme v nasledujúcej kapitole. Kľúčovým pozorovaním je umiestnenie súborov v adresári dáta, kde sa uchovávajú trénovacie a testovacie dáta, ale aj súbory potrebné pre vytvorenie akustického a jazykového modelu.[4].

V rámci popísanej súborovej štruktúre sa nachádza početné množstvo príkladov, pričom tieto príkladu sú znovu použiteľné, upraviteľné a preto je možné ich použiť pre riešenie aj iného problému a nad inými vstupnými dátami.
Pri vytváraní vlastného projektu je možné vytvoriť čiastočne inú pod-adresárovú štruktúru, no ak budeme chcieť znovu-použiť už existujúce skripty z iných projektov, tak im túto štruktúru budeme musieť prispôsobiť.



Obrázok 2: Súborová štruktúra nástroja Kaldi [4]

3 Proces vytvorenia a natrénovania rozpoznávaču v Kaldi

V tejto kapitole sa budeme detailnejšie venovať nástroju Kaldi, ktorý sme si zvolili pre riešenie problému tejto práce. Nech už by sme použili akýkoľvek nástroj, vždy je potrebné sa s ním najprv oboznámiť a naučiť sa ho efektívne a hlavne správne používať. V nasledujúcich podkapitolách si ukážeme, ako vyzerá práca v Kaldi a prejdeme si jednotlivými časťami procesu vytvárania rozpoznávaču reči. Na začiatok si rozoberieme proces získania a pred-spracovania dát, proces vytvorenia jazykového a akustického modelu, a následne prejdeme na samotné trénovanie rozpoznávaču a vyhodnotenie výsledkov. Jednotlivé fázy procesu vytvárania rozpoznávaču reči si popíšeme aj v nadväznosti s praktickou ukážkou, ako je daný krok potrebné realizovať v nástroji Kaldi. Postup rozpoznávania budeme demonštrovať na nami vykonanom experimente, ktorý bol robený podľa oficiálneho návodu pre začiatočníkov [2]. Do jednotlivých krokov sa však pokúsime priniesť viac svetla, a jednotlivé kroky previažeme s teóriou z predchádzajúcej kapitoly.

3.1 Príprava zvukových dát

Pred zahájením práce s nástrojom Kaldi je potrebné získať dostatok zvukových dát, ktoré sa použijú pri trénovaní a testovaní rozpoznávaču. V prípade nástroju Kaldi je však potrebné brať ohľad na typ a jednotlivé parametre zvukových nahrávok, ktoré si vytvárame, resp. ktoré plánujeme použiť. Kaldi si vo väčšine prípadov žiada zvukové dáta vo formáte .wav, pričom vzorkovacia frekvencia musí byť 16kHz, kódovanie 16-bit signed integer, little-endian, mono, teda jeden kanál. Pokiaľ použité zvukové nahrávky nespĺňajú tieto parametre, Kaldi na to odpovedá

množstvom chybových hlášok. Problém nesprávne formátovaných nahrávok sa dá vyriešiť pomocou multiplatformovej utility SoX (Sound eXchange), ktorá vie preformátovať existujúce súbory presne podľa uvedených požiadaviek.

Za účelom vyskúšania nástroju Kaldi sme si získali dokopy 24 nahrávok od štyroch osôb, pričom každá osoba nám poskytla 6 nahrávok. Experiment sme si nastavili tak, že každá nahrávka obsahovala tri číslice vyslovené v anglickom jazyku. Získané nahrávky boli vytvorené vo formáte .wav, no pri ich použití hlásil nástroj Kaldi chybu. Po následnej identifikácii problému sme zistili, že získané nahrávky nespĺňajú požadované parametre a tak ich bolo potrebné upraviť. Na úpravu nahrávok sme použili spomenutý nástroj SoX, ktorý preformátoval nahrávky presne tak, aby spĺňali požiadavky nástroju. Pre úpravu jednej nahrávky bol použitý nasledovný príkaz:

```
sox {file_name} -e signed-integer -b 16 -r 16000 -c 1
```

Po úprave nahrávok bolo potrebné ich rozdeliť na trénovacie a testovacie. Na trénovanie rozpoznávaču sme sa rozhodli použiť 18 nahrávok a na testovanie zvyšných 8. Nakoľko sa jedná o koncept strojového učenia, testovacia sada dát by nemala obsahovať nič z testovacích dát. To v praxi znamená, že pri rozpoznávaní hlasu je potrebné odlíšiť rečníkov, a vytvorený rozpoznávač otestovať na nahrávkach rečníka, ktorého hlas rozpoznávač ešte nepočul.

V hlavnom adresári Kaldi sme si v pod-adresári egs, ktorý je určený pre ukladanie projektov, vytvorili vlastný priečinok *digits*. V rámci tohto priečinku je potrebné si manuálne vytvoriť vlastnú adresárovú štruktúru pre uloženie jednotlivých častí projektu, ako sú dáta, skripty, akustický model, jazykový model a ďalšie. Naše nahrávky sme si uložili osobitne do adresáru *train* a *test* a to nasledovne:

- trénovacie dáta - Kaldi/egs/digits/digits_audio/train
- testovacie dáta - Kaldi/egs/digits/digits_audio/test

3.2 Vytvorenie akustického modelu

Po úspešnej akvizícii a kontrole zvukových dát je potrebné zadefinovať sadu súborov pre Kaldi, ktoré sa použijú pre vytvorenie akustického modelu. Ako už bolo spomenuté v podkapitole 2.2.1, akustický model definuje vzťahy medzi zvukovým signálom a fonémami. Na jeho vytvorenie je potrebné vytvoriť k trénovacím dátam ich prepis. Nástroj Kaldi potrebuje aj skupinu ďalších súborov, ktoré obsahujú špecifické informácie potrebné pre vytvorenie akustického modelu. V nasledujúcom zozname detailnejšie približujeme spomenuté súbory a ich význam.

- **spk2gender** slúži na určenie pohlavia pre jednotlivých rečníkov. štruktúra súboru je nasledovná s predpisom <*idRečníka*> <*pohlavie*>, pričom pohlavie je kódované značkami *m* ako *male* a f ako *female*.

```
andrej m
petra f
```

- súbor **text** definuje prepis jednotlivých nahrávok. Súbor má štruktúru <*idPrepisu*> <*prepis*>, kde je potrebné si zadefinovať identifikátor prepisu nahrávky, a zaň pripojiť samotný prepis. Vytvorený identifikátor prepisu bude pre použitý v súbore *wav.scp* pre namapovanie nahrávky na daný prepis.

```
andrej_1 one two three
petra_2 four five six
```

- **wav.scp** mapuje nahrávky na ich prepis. Štruktúra súboru je nasledovná: <*idPrepisu*> <*adresaNahravky*>. Adresa nahrávky je úplná adresa zvukovej nahrávky z množiny trénovacích dát.

```
andrej_1 Kaldi/egs/digits/digits_audio/train/andrej/audio1.wav
petra_2 Kaldi/egs/digits/digits_audio/train/petra/audio2.wav
```

- **utt2spk** mapuje prepis nahrávky na konkrétneho rečníka. Súborová štruktúra: <*idPrepisu*> <*idRečníka*>.

```
andrej_1 andrej
petra_2 petra
```

- **corpus.txt** obsahuje po riadkoch prepis zvukových nahrávok.

```
one two three
four five six
```

Všetky spomenuté súbory je potrebné uložiť do priečinku so zvukovými nahrávkami. Proces vytvárania uvedených súborov je potrebný pre trénovacie, ale aj testovacie dáta. Podľa adresárovej štruktúry na obrázku č.2 si môžme všimnúť, že súbory, ktoré sme si opísali je potrebné vložiť do adresáru *egs/wsj/s5/data/train*, prípadne *egs/wsj/s5/data/test*, podľa toho pre ktorú časť dát boli súbory vytvorené.

3.3 Vytvorenie jazykového modelu

Jazykový model sme si už priblížili v podkapitole 2.2.2 a vieme, že predstavuje pravdepodobnostnú distribúciu postupnosti lingvistických častí (slová, fonémy). Preto podobne ako pri akustickom modeli, tak aj pri tom jazykovom je potrebné vytvoriť sadu súborov, ktoré sa použijú pre jeho vytvorenie.

- **lexicon.txt** je prvý z kľúčových súborov, ktorý v sebe obsahuje po riadkoch slová, a ich uvedenú výslovnosť na úrovni foném. Napríklad v prípade slov *jeden* a *sedem* by súbor vyzeral nasledovne:

```
jeden je de n
sedem se de m
```

- **nonsilence_phones.txt** v sebe obsahuje všetky neznelé fonémy zo súboru *lexicon.txt*. V prípade vyššie uvedeného lexikónu by súbor vyzeral takto:

  j e
  d e
  n
  s e
  d e
  m

- **silence_phones.txt** v sebe obsahuje všetky znelé fonémy zo súboru *lexicon.txt*.

- **optional_silence.txt**

Vytvorené súbory je potrebné situovať do *egs/wsj/s5/data/local/lang*, čo opäť znázorňuje obrázok č.2.

3.4 Trénovanie rozpoznávaču

Trénovanie rozpoznávaču je realizované, keďuž máme pripravené všetky prerekvizity v podobe akustického a jazykového modelu. Scenár vytvorenia rozpoznávaču ktorý prezentujeme spočíva v spustení shell skriptu, ktorý sa postará o všetky fázy, od vytvorenia akustického modelu až po natrénovanie samotného rozpoznávaču. V nasledujúcich bodoch si popíšeme jednotlivé časti skriptu, tak ako je prezentovaný autormi nástroju Kaldi [2].

- Vytvorenie akustického modelu

- Extrakcia čŕt

- Vytvorenie jazykového modelu
- Mono trénovanie a dekódovanie
- Zarovnávanie - alingment
- Triphone trénovanie a dekódovanie

3.5 Vyhodnocovanie výsledkov

Za predpokladu úspešného vykonania skriptu, je možné dostať sa k výsledkom trénovania. Spomínaný skript vytvára súbory s výsledkami pre rôzne nastavenie rate-u a vieme tak nájsť najoptimálnejšie nastavenie parametrov. Skript vytvorí súbory s výsledkami v nasledujúcich priečinkoch:

```
Kaldi/egs/digits/exp/mono/decode/
Kaldi/egs/digits/exp/tri1/decode/
```

V týchto priečinkoch sa nachádzajú súbory s názvami *wer_<wip>_<lmwt>*, kde *wip* (word insertion penalty) je koeficient, ktorý upravuje hodnotu WER pri vkladaní slov, pričom nadobúda hodnoty 0, 0.5, 1 a *lmwt* (language model weight) je koeficient váhy jazykového modelu a nadobúda hodnoty od 7 do 17.

Po otvoreni jedného zo súborov, môžme pozorovať hodnoty metriky SER (Sentence Error Rate) a WER (Word Error Rate).

```
%WER 38.89 [ 7 / 18, 1 ins, 1 del, 5 sub ]
%SER 66.67 [ 4 / 6 ]
Scored 6 sentences, 0 not present in hyp.
```

Metrika WER vyjadruje pomer počtu vložení, vymazaní prípadne nahradení voči celkovému počtu rozpoznávaných slov. V našom prípade sme dosiahli výsledok, kedy v rozpoznanej reči došlo k jednému vloženiu, jednému vymazaniu a k piatim nahradeniam, a teda chyba bola na 7 slovách z celkového počtu 18.

V prípade metriky SER ide o vyjadrenie pomeru počtu nesprávne identifikovaných viet voči celkovému počtu viet. V našom experimente sme mali spolu 6 odlišných viet v podobe postupnosti troch anglických slov. V zobrazenom prípade boli správne rozpoznané iba dve vety.

Dosiahnuté výsledky nie sú síce príliš optimistické, no vzhľadom na veľmi malú trénovaciu sadu dát považujeme dosiahnuté výsledky za očakávané.

4 Analýza problému

V tejto kapitole sa zameriame na špecifikáciu experimentálneho rozpoznávaču hlasovej reči. Experimentálny rozpoznávač je vytváraný v spolupráci s Transparency International Slovensko (TISK). Vytvorený rozpoznávač by mal byť v TISK používaný za účelom prepisu nahrávky do textovej podoby.

V úvodnej fáze analýzy sa zameriava na špecifikáciu problému – definícia vlastností rozpoznávaču, jeho použitie, prostredie nahrávok a pod.

4.1 Charakteristika nahrávok a prostredia

Na základe poskytnutých údajov a informácii od TISK vieme, že nahrávky, ktoré majú byť našim rozpoznávačom prepísané do textovej podoby sú vytvárané na rôznych interných akciách organizácie. Nahrávanie prebieha zvyčajne tak, že je nahrávaný len zvuk buď diktafónom alebo mobilným zariadením. Jednotlivý rečníci vo väčšine prípadov nemávajú k dispozícii vlastný mikrofón a preto sú ich línie v nahrávkach spojené aj s okolitým šumom. Nahrávané diskusie a stretnutia sú zvyčajne v interiéri, no v poslednom období z dôvodu pandemickej situácie sa niektoré nahrávky tvoria v exteriéri alebo prostredníctvom aplikácie, čo do iste miery mení vlastnosti prostredia. TISK nám poskytlo aj nahrávky vytvorené v poslednom období, v ktorých je zaznamenaný hlas moderátora a hosťa, pričom hosťov hlas je nahrávaný z aplikácie, a teda dochádza k zásadnej zmene kvality hlasu.

5 Experimenty a ich výsledky

V tejto časti sa zameriame na popis vykonaných experimentov s použitím nástroju Kaldi. Cieľom našich experimentov bolo oboznámiť sa s týmto nástrojom a identifikovať prípadné problémy pred samotným návrhom riešenia. Pre jednotlivé experimenty popisujeme použitý postup, informácie o dátach a na konci aj samotné výsledky a zistenia vyplývajúce z experimentu.

5.1 Experiment s poskytnutými nahrávkami

V rámci spolupráce s TISK nám bolo poskytnutých niekoľko zvukových nahrávok a prepisov, na ktorých by mal vedieť rozpoznávač dobre fungovať. Rozhodli sme sa preto pre vykonanie tohto experimentu, ktorý spočíval v analýze poskytnutých nahrávok, ich spracovaní a vo vytvorení nového projektu v nástroji Kaldi, a následnom natrénovaní rozpoznávaču.

5.1.1 Analýza a spracovanie nahrávok

K dispozícii sme mali tri nahrávky, v ktorých boli zaznamenané rozhovory dvoch osôb. Rozhovor v prvej nahrávke bol medzi mužom a ženou a vo zvyšných dvoch bol rozhovor medzi dvomi ženami. V oboch prípadoch išlo o nahrávky, ktoré boli nahrávané na diktafón, pričom hlas jedného rečníka bol nahrávaný priamo a hlas druhého rečníka bol zachytený z iného komunikačného zariadenia.

Po vypočutí nahrávok sme zistili, že kvalita hlasu rečníkov sa významne líšila, práve z dôvodu že hlas vzdialeného rečníka bol ovplyvnený kvalitou prenosu. Pre natrénovanie modelu sme však potrebovali použiť také nahrávky, ku ktorým je možné vytvoriť ich presný prepis. Z tohto dôvodu sme museli nahrávky ručne zostrihať a priradiť k nim ich úplný prepis. Od TISK sme dostali aj prepisy poskyt-

nutých nahrávok, avšak tieto prepisy už prešli úpravou niektorým z pracovníkov. V prepisoch bol upravený slovosled, niektoré slová boli pridané a niektoré zas odstránené. Napriek týmto zmenám nám prepis veľmi pomohol, no bolo potrebné venovať čas úprave častí, ktoré sme sa rozhodli použiť.

Pri spravovaní a zostrihaní nahrávok sme boli nútený veľkú časť dát zahodiť a vybrali sme len tie pasáže, ku ktorým bolo možné vytvoriť presný prepis. Pri výbere pasáží sme snažili vyhnúť častiam, v ktorých dochádzalo k prekrytiu reči dvoch rečníkov a v ktorých rečníci využívajú výplnkové slová – expletíva. V nasledujúcej tabuľke č.1 zhŕňame, dĺžku stôp jednotlivých nahrávok pred a po ich spracovaní.

Tabuľka 1: Tabuľka dĺžok zvukových stôp pred a po spracovaní.

ID nahrávky	**Pred spracovaním**	**Po spracovaní**
1	00:16:31	00:01:47
2	00:12:48	00:01:29
3	00:12:51	00:01:54

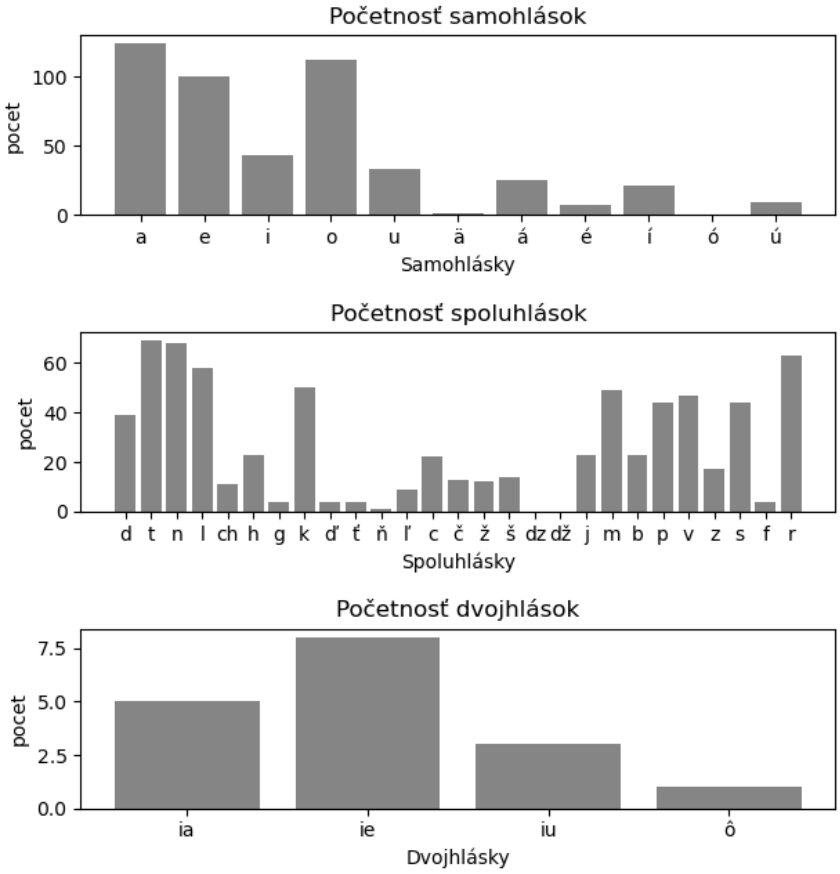
V rámci analýzy nahrávok boli identifikovaný štyria rečníci, pričom dvaja rečníci boli prítomný aj v nahrávke č.2 aj v nahrávke č.3. Zostrihané nahrávky sme si vhodne pomenovali a pre každú z nich sme si pripravili spomínaný prepis. Poslednou časťou prípravy zvukových dát bola úprava ich parametrov. Tak ako bolo spomenuté v podkapitole 3.1, Kaldi potrebuje mať audio súbory v konkrétnom formáte a s príslušnými parametrami. Preto sme zostrihané nahrávky ešte nechali spracovať vytvoreným skriptom, ktorý nahrávky upravil podľa potrieb s použitím utility SOX. Skript na obrázku č.3 zoberie po spustení všetky súbory s koncovou *.wav*, vypíše názov spracovávaného súboru a tento súbor upraví pomocou príkazu *SOX*. Upravenú nahrávku uloží pod rovnakým názvom, no do názvu ešte pridá príznak *new*. Príkaz skriptu zmení prepínačom *-r* vzorkovaciu frekvenciu na *16kHz*, prepínačom *-e* nastaví kódovanie na *signed-integer* a počet kanálov sa nastaví na

1=mono pomocou prepínaču *-c*.

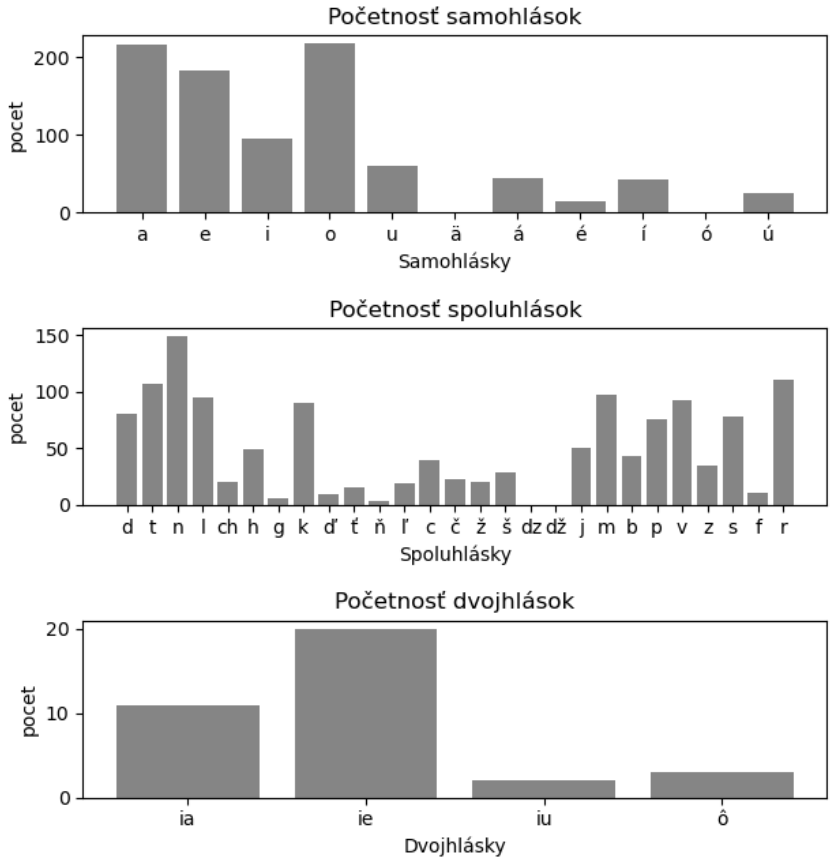
```
#!/bin/bash

for i in *.wav;
do
echo $i; sox "$i" -e signed-integer -b 16 -r 16000 -c 1 ${i%%.wav}.new.wav;
done
```

Obrázok 3: Skript pre úpravu nahrávok s použitím utility SOX

Pred zahájením vytvárania projektu nás ešte zaujímalo, aké je v dátach pokrytie jednotlivých hlások, čo by mohlo vo výsledku ovplyvniť trénovanie, pri ktorom sa využívajú HMM. Za týmto účelom sme si vytvorili program v jazyku Python, ktorý zo vstupného súboru obsahujúceho všetky použité slová vytvoril histogramy pre samohlásky, spoluhlásky a dvojhlásky.



Obrázok 4: Histogramy pokrytia hlások v počiatočnom datasete



Obrázok 5: Histogramy pokrytia hlások vo zväčšenom datasete

V ďalších fázach sme pracujeme s dvomi datasetmi, kde jeden obsahuje len nahrávky č.1 a č.2 a druhý dataset obsahuje všetky 3 nahrávky z tabuľky č.1. Na predošlých obrázkoch č.4 a č.5 sa nachádzajú vykreslené histogramy pokrytia hlások podľa druhov pre počiatočný a pre zväčšený dataset. Dataset v oboch prípadoch neobsahuje spoluhlásky *dz* a *dž*, samohlásku *ó* a samohláska *ä* je zastúpená len jedným výskytom. Tieto hlásky sú zo štatistického hľadiska v reči používané menej a ich malé, prípadne žiadne zastúpenie nie je prekvapivé aj vzhľadom na malú veľkosť datasetu. Pri vytváraní datasetu pre finálny rozpoznávač však bude potrebné pokryť tieto hlásky.

5.1.2 Vytvorenie projektu v Kaldi

Pre tento experiment sme sa rozhodli pre použitie postupu natrénovania rozpoznávaču, ktorý vytvorili autori nástroju Kaldi pre začiatočníkov [2]. Samotné vytvorenie projektu sme realizovali podľa popísaného postupu z kapitoly 3. V priečinku *egs* sme si vytvorili vlastný priečinok pre náš projekt. V rámci neho sme si zadefinovali adresár *data_audio* pre zvukové nahrávky, ktoré sme ešte rozdelili na trénovacie a testovacie v pomere 80:20 a uložili ich do príslušných priečinkov *train* a *test*. V týchto priečinkoch je vhodné vytvoriť ďalšie adresáre pre každého rečníka a povkladať do nich príslušné nahrávky. Pri veľkom množstve dát a rečníkov budeme mať k dispozícii prehľadnejšiu štruktúru, s ktorou sa bude ľahšie pracovať pri možnom pridávaní alebo úprave dát.

Ďalší adresár *data* sme vytvorili pre uloženie súborov, ktoré definujú jazykový a akustický model. Súbory pre akustický model je potrebné vytvoriť pre trénovaciu aj pre testovaciu dátovú sadu. Na nasledujúcich obrázkoch poskytujeme pohľad na štruktúru vytvorených súborov pre zadefinovanie akustického modelu. Na obrázku č.6 sa nachádza obsah súboru *spk2gender* v ktorom sú zadefinovaný štyria rečníci tak ako boli identifikovaný vo fáze prípravy dát. Identifikačné názvy rečníkov boli vytvorené podľa našich potrieb a k nim prislúcha označenie pohlavia *f* (female) alebo *m* (male).

```
moderator1 f
moderator2 f
recnik1 m
recnik2 f
```

Obrázok 6: Obsah a štruktúra súboru spk2gender

Na obrázku č.7 sa nachádza ukážka súboru *utt2spk*, v ktorom sme si zadefinovali identifikačné názvy pre súvislé úseky reči a k nim sme priradili identifikátor rečníka, ktorý daný úsek reči vyslovil.

Úsek reči predstavuje postupnosť slov v jednej nahrávke, v ktorej je zachytený hlas len jedného rečníka.

```
moderator1-01 moderator1
moderator1-02 moderator1
moderator1-03 moderator1
moderator1-04 moderator1
moderator1-05 moderator1
moderator2-01 moderator2
moderator2-02 moderator2
moderator2-03 moderator2
moderator2-04 moderator2
```

Obrázok 7: Ukážka obsahu a štruktúry súboru utt2spk

V súbore *wav.scp* sme priradili pre jednotlivé úseky reči úplnú cestu k príslušnej *.wav* nahrávke. V uvedených cestách k nahrávkam môžme pozorovať, vyššie popísanú súborovú štruktúru. Z cesty vieme vyčítať, že napríklad nahrávka *exp-1-f-moderator1.wav* patrí do trénovacej množiny a hlas v nahrávke patrí rečníkovi pod identitou *moderator1*.

```
moderator1-01 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator1/exp-1-f-moderator1.wav
moderator1-02 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator1/exp-2-f-moderator1.wav
moderator1-03 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator1/exp-3-f-moderator1.wav
moderator1-04 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator1/exp-4-f-moderator1.wav
moderator1-05 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator1/exp-5-f-moderator1.wav
moderator2-01 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator2/exp-1-f-moderator2.wav
moderator2-02 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator2/exp-2-f-moderator2.wav
moderator2-03 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator2/exp-3-f-moderator2.wav
moderator2-04 /home/andrej/Desktop/Kaldi/kaldi/egs/experiment/data_audio/train/moderator2/exp-4-f-moderator2.wav
```

Obrázok 8: Ukážka obsahu a štruktúry súboru wav.scp

Ďalším súborom pre akustický model je súbor *text*, ktorý je zobrazený na obrázku č.9. Definujeme v ňom jednotlivé úseky reči, kde pre každý úsek priradíme presný prepis reči, ktorý sme si pripravili pri spracovaní nahrávok.

```
moderator1-01 dobre pamätáš si ešte na prvú úlohu ktorá bola s takým zadaní keď si nastúpil do transparency
moderator1-02 takže vlastne v spolupráci s transparency si pracoval na dvoch rebríčkoch transparentnosti nemocníc
moderator1-03 keďže si pracoval s tými dátami a to bol v podstatne taký veľký súbor tých dát našiel si v nich niečo
moderator1-04 ako dlho trvala celá tvoja stáž
moderator1-05 sú ešte nejaké iné úlohy ktoré osobne si mal preferenciu
moderator2-01 čo ťa motivovalo aby si sa stala dobrovoľníčkou pre transparency
moderator2-02 a prečo si si vybrala práve transparency ako organizáciu
moderator2-03 a keď si spomínala tie výsledky ktoré si registrovala naše predtým ako si sa k nám pridala do tímu je
moderator2-04 na akých dobrovoľníckych úlohách si spolupracovala
```

Obrázok 9: Ukážka obsahu a štruktúry súboru text

Doposiaľ uvedené súbory musia spĺňať ešte jednu vlastnosť, ktorou je lexikografické usporiadanie riadkov. Ak by táto vlastnosť nebola splnená, Kaldi by pri procese trénovania zahlásil chybu a trénovanie by sa tak skončilo neúspešne.

Ďalším súborom ktorý je potrebné vytvoriť je *corpus.txt*. V tomto súbore sa nachádza jazykový korpus, v ktorom sú obsiahnuté všetky textové pasáže trénovacích aj testovacích dát. Súbor ma po riadkoch uložené jednotlivé pasáže tak ako vidno na obrázku č.9 ale bez akýchkoľvek identifikátorov. Tento súbor je potrebné umiestniť do nového priečinku *local*, ktorý vytvoríme v kmeňovom adresári nášho projektu. V prípade tohto súboru už nie je potrebné jeho lexikografické usporiadanie.

Druhou časťou vytvárania projektu bolo pripravenie súborov pre natrénovanie jazykového modelu. Všetky nasledujúce súbory ktoré si teraz popíšeme sú v našom projektu uložené v priečinku *dict*, ktorý nájdeme na ceste:

- *experiment/data/local/dict*

Prvým súborom pre jazykový model je *lexicon.txt*. Tento súbor je určený pre uloženie výslovnosti každého slova z vytvoreného korpusu (*corpus.txt*), čo je dosiahnuté rozbitím každého slova na malé jednotky, v našom prípade fonémy. Pre vytvorenie lexikónu bolo preto potrebné zvoliť spôsob, ako fonémy vhodne reprezentovať. Po analýze možných riešení sme sa rozhodli použiť medzinárodnú fonetickú abecedu IPA (International Phonetic Alphabet). Pri hľadaní existujúcich implementačných riešení prevodu slov na fonémy sme našli riešenie, ktoré vytvoril L.Kyjánek vo svojej práci. Ide v princípe o skript v jazyku Python, v ktorom sú implementované pravidlá abecedy IPA a samotný mechanizmus prevodu slova na fonémy [6][7]. Toto riešenie sme zakomponovali do vlastného programu, pomocou ktorého sme si vygenerovali výsledný lexikón. Na obrázku č.10 sa nachádza ukážka z vytvoreného lexikónu. Vytvorenie lexikónu si vyžaduje splniť niekoľko podmienok. Opäť platí, že riadky v súbore musia byť lexikograficky usporiadané a nesmú sa vyskytovať duplicitne. V lexikóne vieme zadefinovať rôznu výslovnosť

pre to isté slovo, ale táto sa musí líšiť na úrovni priradených foném.

```
druhou d r u ɦ ɔu̯
dva d v a
dvoch d v ɔ x
emu ɛ m u
ešte ɛ ʃ c ɛ
falošné f a l ɔ ʃ n ɛː
funguje f u ŋ ɡ u j ɛ
hľadaní ɦ ʎ a d a ɲ iː
hlásila ɦ l aː s ɪ l a
```

Obrázok 10: Ukážka obsahu a štruktúry súboru lexikon.txt

Poslednými dvomi súbormi sú *silence_phones.txt* a *non_silence_phones.txt*. Na obrázku č.11 sa nachádza ukážka súboru *nonsilence_phones.txt*, kde sa nachádzajú všetky znejúce fonémy.

```
a
aː
æ
b
c
d͡
d͡z
d͡ʒ
ɛ
ɛː
```

Obrázok 11: Ukážka obsahu a štruktúry súboru nonsilence_phones.txt

Na obrázku č.12 sa nachádza ukážka súboru *silence_phones.txt*, kde môžme pozorovať fonémy, ktoré reprezentujú ticho alebo pauzu v reči.

```
sil
spn
```

Obrázok 12: Ukážka obsahu a štruktúry súboru silence_phones.txt

Pri týchto dvoch súboroch je nutnosťou, aby každá fonéma použitá v lexikóne bola definovaná v jednom z nich. Ak táto podmienka nie je splnená, trénovací skript túto skutočnosť deteguje a vypíše príslušnú chybovú hlášku.

Poslednou fázou prípravy projektu je skopírovanie pomocných utilít a programov do nášho projektu. Ako prvé je potrebné vloženie adresárov *steps* a *utils* z lokality *kaldi/egs/wsj/s5* do koreňového adresáru nášho projektu. V skopírovaných adresároch sa nachádzajú rôzne skripty a pomocné programy, ktoré sú využívané počas trénovania modelov. Druhým krokom je skopírovanie skórovacieho skriptu, ktorý vyhodnotí metriky WER a SER natrénovaných modelov. Ide o skript *score.sh*, ktorý sa nájdeme na tejto ceste kaldi/egs/voxforge/s5/local. V kmeňovom adresári nášho projektu si vytvoríme priečinok *local* a do neho vložíme spomínaný skript *score.sh*.

Posledným krokom je vytvorenie troch skriptov, *cmd.sh*, *path.sh* a *run.sh*, ktoré zabezpečia celý trénovací proces. Trénovanie sa vykoná spustením skriptu *run.sh*, v rámci ktorého dochádza k natrénovaniu dvoch modelov a to Mono (monofón model) a TR1 (trifónový model). Obsah týchto skriptov vzhľadom na ich komplexnosť neuvádzame, avšak k dispozícii sú na stránke zdroju a v prílohe tejto práce [2]. Po úspešnom zvládnutí všetkých predchádzajúcich krokov je projekt pripravený a je možné zahájiť trénovanie modelov.

5.1.3 Trénovanie a výsledky

Náš experiment sme z počiatku začali realizovať len s použitím prvých dvoch nahrávok z tabuľky č.1, a neskôr sme k nim pridali aj poslednú tretiu nahrávku. V tejto podkapitole sa preto pozrieme na výsledky trénovania pri použití prvých dvoch nahrávok a porovnáme ich s výsledkami trénovania pri použití všetkých nahrávok. Úspešnosť modelov si zároveň porovnáme voči trénovacím aj testovacím dátam, aby sme vedeli vyhodnotiť, či vytvorené modely nie sú pretrénované a akú budú mať tieto modely úspešnosť na dátach, ktoré predtým nevideli.

Pripomíname, že počiatočný a zväčšený dataset boli rozdelené na trénovaciu a testovaciu sadu v pomere 80:20. Prevspustením trénovania sme ešte navýšili počat

iterácii trénovania oboch modelov. V skripte */steps/train_mono.sh*, ktorý slúži pre natrénovanie modelu Mono sme zväčšili počet iterácií zo 40 na 80 a v skripte */steps/train_deltas.sh* pre trénovanie modelu TR1 sme upravili počet iterácií z 35 na 70.

Pre každý zo scenárov sme museli upraviť popísané súbory z predchádzajúcej podkapitoly a natrénovať modely Mono a TR1. Po úspešnom zbehnutí skriptu *run.sh* sme sa pozreli na dosiahnuté výsledky a metriky natrénovaných modelov, ktorých výsledky sa nachádzajú v týchto adresároch:

- /exp/mono/decode/scoring_kaldi
- /exp/tr1/decode/scoring_kaldi

V obidvoch priečinkoch sa nachádza súbor *best _wer*, ktorý obsahuje najlepší dosiahnutý výsledok. V nasledujúcej tabuľke č.2 uvádzame najlepšie dosiahnuté hodnoty metriky WER oboch modelov pri použití počiatočného aj zväčšeného datasetu, pričom na validáciu bol použitý trénovací dataset.

Tabuľka 2: Dosiahnuté výsledky trénovania na trénovacích dátach

Model	**Počiatočný datatset**	**Zväčšený dataset**
	WER [%]	**WER [%]**
Mono	28.27	14.67
TR1	17.56	10.46

V nasledujúcej tabuľke č.3 uvádzame hodnoty metriky WER natrénovaných modelov voči testovaciemu datasetu.

Tabuľka 3: Dosiahnuté výsledky trénovania testovacích dátach

Model	**Počiatočný datatset**	**Zväčšený dataset**
	WER [%]	**WER [%]**
Mono	87.22	29.55
TR1	87.97	32.95

Z výsledkov v tabuľke č.2 môžeme vidieť, že ani v jednom prípade nedošlo k pretrénovaniu modelu. V prípade použitia počiatočného datasetu je metrika WER už celkom vysoká, čo sa nakoniec ukázalo aj pri validácii na testovacích dátach (tabuľka č.3). Pri použití zväčšeného datasetu došlo k zásadnému zlepšeniu metriky WER aj na trénovacích, ale hlavne na testovacích dátach. Hodnoty WER okolo 30% sú stále značne vysoké, no dôležitým pozorovaním je pre nás skutočnosť, že zväčšenie datasetu význame pomohlo k vylepšeniu úspešnosti modelov.

Výsledky a prácu na tomto experimente považujeme za pozitívnu a veľmi prínosnú. Vyskúšali sme si proces analýzy a prípravy dát, počas ktorého sme potrebovali vyriešiť akým spôsobom efektívne spracovať nahrávku a jej prepis a optimalizovali sme si proces spracovania lexikónu z jazykového korpusu. Podarilo sa nám preniknúť do hĺbky nástroju Kaldi a vďaka analyzovaniu rôznych skriptov a implementácii sme lepšie pochopili ako funguje. Počas experimentu sme sa dozvedeli, aké požiadavky kladie toolkit na nahrávky s ktorými pracuje a našli sme spôsob, ako nahrávky rýchlo a jednoducho transformovať do požadovanej podoby. Nakoniec sa nám podarilo úspešne natrénovať modely Mono a TR1, a znížiť WER zväčšením datasetu a úpravou niektorých parametrov. V rámci tohto experimentu boli na trénovanie modelov využité len HMM, ktoré vcelku zafungovali aj na malý dataset. Našim ďalším krokom ja vykonanie experimentu, v rámci ktorého by sme okrem HMM využili aj hlboké neurónové siete DNN.

5.2 Experiment s použitím postupu pre HMM-DNN

6 Opis riešenia

6.1 Návrh riešenia

6.2 Vytvorenie rozpoznávaču

6.3 Vylepšovanie modelov

6.4 Overenie riešenia

6.5 Zhodnotenie výsledkov

7 Záver

Literatúra

[1] Cmusphinx. https://cmusphinx.github.io/wiki/tutorialconcepts/. Navštívené: 15. december 2020.

[2] Kaldi for dummies tutorial. https://kaldi-asr.org/doc/kaldi_for_dummies.html, . Navštívené: 31.marca 2021.

[3] How to start with kaldi and speech recognition. https://towardsdatascience.com/how-to-start-with-kaldi-and-speech-recognition-a9b7670ffff6, . Navštívené: 15.decembra 2020.

[4] E. Chodroff. Kaldi tutorial. https://eleanorchodroff.com/tutorial/kaldi/index.html, 11 2018. Navštívené: 31.marca 2021.

[5] B. Juang and L. Rabiner. Automatic speech recognition - a brief history of the technology development. 01 2005.

[6] L. Kyjánek. Automatic phonetic transcription of the czech, slovak and polish languages, 09 2019. URL https://github.com/lukyjanek/phonetic-transcription.

[7] L. Kyjánek and J. Haviger. The measurement of mutual intelligibility between west-slavic languages. *Journal of Quantitative Linguistics*, 26(3):205–230, 2019. doi: 10.1080/09296174.2018.1464546. URL https://doi.org/10.1080/09296174.2018.1464546.

[8] J. Li, l. Deng, R. Haeb-Umbach, and Y. Gong. *Fundamentals of speech recognition*, pages 9–40. 12 2016. ISBN 9780128023983. doi: 10.1016/B978-0-12-802398-3.00002-7.

[9] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Vesel. The kaldi speech recognition toolkit. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 01 2011.

[10] D. Povey, X. Zhang, D. Povey, X. Zhang, and S. Khudanpur. S.: Parallel training of deep neural networks with natural gradient and parameter averaging, 2014.

[11] K. Veselý, A. Ghoshal, L. Burget, and D. Povey. Sequence-discriminative training of deep neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2345–2349, 01 2013.

Zoznam obrázkov