Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

PLZEŇ, 2016 **ONDŘEJ VÁCHAL**

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 4.5.2016 ……..

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce Ing. Mgr. Josefu Psutkovi, Ph.D. za jeho pomoc a úsilí, které mi věnoval při tvorbě této bakalářské práce.

Anotace v češtině

Tato bakalářská práce se zabývá tématem rozpoznávání řeči. Její teoretická část je zaměřena na statistické rozpoznávání a to především na tvorbu akustických modelů pomocí Skrytých Markových modelů, které jsou k rozpoznávání využívány. Práce dále obsahuje jednoduchý návod k anotaci zvukových nahrávek a v neposlední řadě také návod pro natrénování jednoduchého monofónového modelu. Praktická část se zabývá postupem přípravy trénovacích dat a jejich autentickými opravami a následně natrénováním jednoduchých akustických modelů několika sportů a diskusí těchto výsledků.

Klíčová slova v češtině

rozpoznávání řeči, skryté Markovy modely, pravděpodobnost, HTK, trénování modelů, referenční přepis

Anotace v angličtině

This bachelor thesis deal with speech recognition. The theoretical part is focused on statistical recognition especially on hidden Markov models that are used for recognition. It also includes simple instructions how to annotate audio recordings. Finally, it also contains instructions for training simple monophone model. The practical part is primarily aimed on data preparation and modification. Further it is focused on training a few simple sports models and discussion of its results.

Klíčová slova v angličtině

speech recognition, hidden Markov models, probability, HTK, models training, reference transcritpion

Obsah

1 Úvod

Počítačové rozpoznávání mluvené řeči je předmětem zájmu výzkumných center již několik desetiletí. První úspěchy vedly k závěrům, že se v brzké době podaří vytvořit takový rozpoznávač, který zvládne bez problémů rekonstruovat to, co člověk řekl. I přes velké pokroky v této problematice je však tato myšlenka stále velmi vzdálenou, neboť rozpoznat jakéhokoliv řečníka, který mluví o libovolném tématu, je problém mnohem obtížnější, než se původně zdálo. Důvodů, proč je rozpoznávání velmi složité, je hned několik.

Hlas jednotlivého člověka se liší od hlasu ostatních osob. Každý člověk má jinou barvu hlasu, jiný přízvuk, mluví jinak rychle, je mu lépe či hůře rozumět atd. Rozpoznávací systémy lze tedy rozdělit na ty, které jsou natrénovaný pouze na jednoho nebo malou skupinu řečníků a na systémy, které jsou na řečníku nezávislé. Další problémem je, že žádný řečník nevysloví stejné slovo stejně. Řečový signál se mění, když člověk řekne stejnou promluvu potichu nebo nahlas, nebo pokud je v klidu či rozčílen. V souvislé promluvě se navíc objevuje jev koartikulace, který může změnit výslovnost začátku a konce slova v závislosti na jeho kontextu. Velkou roli v úspěšnosti rozpoznávání hraje také akustické pozadí. Řeč je mnohem snáze rozpoznatelná v naprostém klidu, než v hlučném prostředí, neboť se může stát, že některá písmena nebo i celá slova v šumu zaniknou.

Velkou roli v úspěšnosti rozpoznávání má také složitost řešené úlohy. Mnohem snazší je samozřejmě rozpoznání izolovaných slov z malého slovníku než rozpoznávání diktátu o rozsahu tisíců slov. Nejobtížnější úlohou je však rozpoznávání souvislé řeči. V tomto případě velmi záleží, zda-li se jedná o řeč čtenou, která je dopředu připravená, a nebo o řeč spontánní. Pokud mluvíme spontánně, dopouštíme se mnoha takzvaných neřečových událostí, kdy může docházet k hlasitému váhání, nakousávání jednotlivých slov, kdy řečník vyřkne pouze část jednotlivého slova, a pak se může opravit nebo slovo zopakovat. V mluveném projevu také bývá problém se spisovností jazyka, kdy většina z nás má tendenci používat hovorové výrazy, což opět ztěžuje úlohy rozpoznávání.

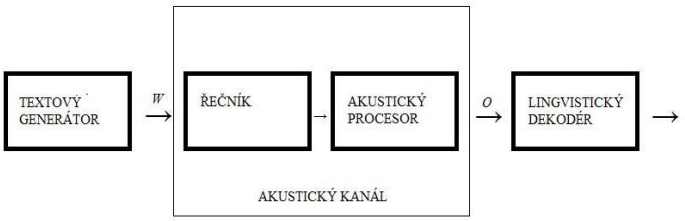
Tato bakalářské práce je zaměřena především na trénování akustických modelů. Nejprve je zmíněn návod na trénování těchto modelů pomocí sady HTK. Dále je pak uveden návod na anotaci přepisů pomocí programu Transcriber, kde se nejprve seznámíme s tímto programem a dále se naučíme, jak vlastně přepisovat. Na konci této části je zmíněn také program pro kontrolu přepisů LM Edit, který nám umožní kontrolu všech přepisů zároveň. V praktické části poté probíhá samotné natrénování akustických modelů. Nejprve je zmíněna příprava všech potřebných dat a jsou natrénovány první modely. Dále jsou pak prováděny opravy na trénovacích datech a je kladena snaha na jejich co nejlepší vylepšení. Na opravených datech jsou dále opět natrénovány akustické modely, do kterých je následně přidáno více složek pro zlepšení přesnosti rozpoznávání. Pro tyto akustické modely je nakonec provedeno rozpoznávání ne jen s obyčejnou rozpoznávací sítí, ale také s kvalitně zpracovanými jazykovými modely, které jsou poskytnuty Západočeskou univerzitou a na jejichž přípravě jsem se sám podílel. V závěru práce jsou pak provedeny křížové testy mezi jednotlivými sporty a nakonec jsou zhodnoceny veškeré výsledky, kterých bylo při rozpoznávání dosaženo.

2 Statistické rozpoznávání

2.1 Statistické přístup k rozpoznávání souvislé řeči

První metody rozpoznávání se začali objevovat v sedmdesátých letech. Slovo zde bylo zpracováno jako celek a klasifikovány do té třídy, ke které mělo nejmenší vzdálenost. Třídy zde byli jednotlivá slova ve slovníku. Klíčovým problémem tak bylo určení nejmenší vzdálenosti. Ta byla jednoduše řečeno určena na základě porovnání jednotlivých obrazů, které byly určeny pomocí nelineární transformace jednoho z obrazů. Druhé skupiny metod používají ke klasifikaci přístup založený na statistických metodách, ve kterých jsou slova a celé promluvy modelovány pomocí skrytých Markových modelů. Jednotlivá slova mohou být modelována jako celek jedním skrytým Markovým modelem, anebo jsou mnohem častěji konstruovány modely subslovních jednotek. Pod pojmem subslovní jednotka si můžeme představit například slabiku, foném, trifón atd. a promluva je modelována zřetězením těchto modelů. Pro každou subslovní jednotku jsou pak při trénování na základě trénovací množiny stanoveny parametry jednotlivých modelů a neznámá promluva je pak rozpoznána na základě toho, jaká posloupnost slov nebo subslovních jednotek generuje promluvu s největší aposteriorní pravděpodobností.

Základní schéma statistického přístupu rozpoznávání se skládá z akustického kanálu a lingvistického dekodéru. Akustický kanál obsahuje akustický procesor a řečník. Hlavním úkolem akustického procesoru je transformace řečových kmitů produkovaných řečníkem na posloupnosti vektorů příznaků a lingvistický dekodér poté překládá tyto řetězce příznaků na řetězce slov. Rozpoznávání je zde tedy definováno jako problém dekódování s maximální aposteriorní pravděpodobností.



Obrázek 1 Blokové schéma systému rozpoznávání založeném na statistickém přístupu

Předpokládejme, že $\boldsymbol{W = \{w_1\ w_2 \ldots\ w_N\}}$ je posloupnost *N* slov a $\boldsymbol{O = \{O_1\ O_2 \ldots\ O_N\}}$ je akustická informace. Akustickou informací rozumíme posloupnost vektorů příznaků odvozenou akustickým procesorem z řečového signálu, ze které pak lingvistický procesor zkouší rozpoznat vyslovená slova. Naším cílem je najít posloupnost slov $\widehat{\boldsymbol{W}}$, která maximalizuje podmíněnou pravděpodobnost $P(\boldsymbol{W|O})$. Snažíme se tedy najít nejpravděpodobnější posloupnost slov pro danou akustickou informaci. S využitím Bayesova vztahu můžeme vztah přepsat do tvaru:

$$\widehat{\boldsymbol{W}} = argmax\, P(\boldsymbol{W}|\mathbf{O}) = argmax \frac{P(\boldsymbol{W})P(\boldsymbol{O}|\mathbf{W})}{P(\boldsymbol{O})}$$

kde $P(\boldsymbol{O}|\mathbf{W})$ je pravděpodobnost, že při vyslovení slov ***W*** bude produkována posloupnost výstupních vektorů příznaků. ***P(W)*** je apriorní pravděpodobnost posloupnosti slov ***W*** a ***P(O)*** je apriorní pravděpodobnost posloupnosti výstupních vektorů. Protože pravděpodobnost ***P(O)*** není funkce ***W***, můžeme ji při hledání maxima ignorovat a výsledný vztah pro posloupnost $\widehat{\boldsymbol{W}}$ bude poté mít tvar:

$$\widehat{W} = argmax\, P(\boldsymbol{W}, \boldsymbol{O}) = argmax\, P(\boldsymbol{W})P(\boldsymbol{O}|\mathbf{W})$$

Z dané rovnice plyne, že stanovení nejlepší posloupnosti slov ***W*** lze řešit pomocí dvou oddělených pravděpodobností $P(\boldsymbol{W})\ a\ P(\boldsymbol{O}|\mathbf{W})$. Tyto posloupnosti mohou být trénovaný a modelovány nezávisle na sobě. Apriorní pravděpodobnost ***P(W)*** obsahuje informaci o jazykovém modelu a podmíněná pravděpodobnost $P(\boldsymbol{O}|\mathbf{W})$ obsahuje informaci o akustickém modelu. Tyto dvě informace musíme zjistit ještě před rozpoznáváním a to na základě řečových a jazykových dat.

Rozpoznávání tedy spočívá v nalezení posloupnosti slov $\widehat{\boldsymbol{W}}$ takové, která maximalizuje daný součin pravděpodobností přes všechny možné posloupnosti slov ***W***. Tato operace je však velmi výpočetně náročná a proto jsou v praxi používaný prohledávací a rozhodovací strategie, které tuto náročnost zredukují a to pokud možno s co nejmenšími ztrátami přesnosti při rozpoznávání.

Úloha statického rozpoznávání by se dala tedy shrnout v několika krocích.
1. Analýza řečového signálu, ze které získáme posloupností vektorů ***O***.
2. Vytvoření akustického modelu pro určení pravděpodobnosti ***P(O|W)***.
3. Vytvoření jazykového modelu pro určení pravděpodobnosti ***P(W)***.
4. Aplikace prohledávacích strategií za účelem nalezení nejpravděpodobnější posloupnosti slov.

2.2 Akustické modelování

Jak již bylo výše uvedeno, úkolem akustického modelování je tedy poskytnout co nejpřesnější odhad podmíněné pravděpodobnosti ***P(O|W)*** pro libovolnou posloupnost vektorů příznaků a pro libovolnou posloupnost slov. Akustické modely by měly splňovat několik základní vlastností.

- Flexibilita
- Přesnost
- Účinnost

Flexibilita je velice důležitá z toho důvodu, že podmínky, za kterých je rozpoznávač používán a za kterých byl natrénován, jsou často velice odlišné. Tím máme na mysli například různé řečníky, různé akustické pozadí například šum, hlasy v pozadí, odlišné tempo řeči atd. Přesností myslíme hlavně to, abychom dokázali odlišit velice podobně akusticky znějící slova, která ale mají velice odlišný lingvistický význam. Tím jsou myšlena například slova ves a les,

salám a salát atd., která mohou mnohdy znít velice podobně, avšak jejich význam je velice odlišný. A nakonec účinnost, která je velmi důležitá při nasazování rozpoznávačů v reálných aplikacích, kdy odezva systému musí být dostupná v reálném čase. Jako velmi účinný se při řešení této úlohy ukázalo být využití skrytých Markových modelů. Tyto modely byly poprvé použity již v polovině sedmdesátých let, avšak k většímu rozšíření u nich došlo až v polovině let osmdesátých.

2.2.1 Skryté Markovy modely

Modelování řeči pomocí Markových modelů vychází z představy o vytváření řeči. Při vytváření řeči člověkem si lze představit, že během krátkého časového okamžiku je artikulační ústrojí v jedné z množiny konfigurací. V tomto časovém intervalu je pak hlasovým ústrojím produkován krátký signál, který závisí právě na konfiguraci hlasového ústrojí. Tento signál může být popsán určitými charakteristikami, jež můžeme reprezentovat vhodným vektorem příznaků. Z představy o vytváření řeči vychází i konstrukce klasifikátoru založená na modelování signálu pomocí Markova procesu. Během tohoto procesu jsou generovány dvě vzájemně svázané posloupnosti náhodných proměnných a to podpůrný Markovův řetězec, který je posloupností konečného počtu stavů, a řetězec vektorů příznaků. Ten reprezentuje spektrální charakter jednotlivých mikrosegmentů signálu. Pro tyto charakteristiky jsou vytvořeny náhodné funkce, které pravděpodobnostně ohodnocují vztah charakteristik ke všem stavům.

Úloha nalezení co nejlepšího odhadu pravděpodobnosti $P(\boldsymbol{O}|\boldsymbol{W})$ je tedy v našem případě úlohou určení struktury skrytého Markova modelu a určení hodnoty jeho parametrů. Tyto parametry lze určit dvěma způsoby a to expertním odhadem na základě apriorních znalostí anebo metodou statistické indukce z množiny trénovacích dat. Pro naše potřeby využijeme obou těchto metod. Apriorní znalost využijeme k určení struktury Markova modelu a statistické indukce využijeme pro odhad parametrů.

2.2.2 Shrnutí akustického modelování

Pro akustické modelování tedy využíváme Skryté Markovy modely. Tyto modely jsou tedy stochastické procesy, které v diskrétních časových okamžicích generují posloupnost vektorů pozorování $\boldsymbol{o}$. V každém z těchto časových okamžiků změní model svůj stav s_i na základě předem daných pravděpodobností přechodu $\boldsymbol{a_{ik}}$. Tato pravděpodobnost přechodu nám tedy určuje, s jakou pravděpodobností přechází model z jednoho stavu s_i, ve kterém je v jednom časovém okamžiku, do stavu s_k, ve kterém je v následném časovém okamžiku. Pro pravděpodobnost přechodu tedy platí následný vztah.

$$a_{ik} = P(s(t+1) = s_k, s(t) = s_i$$

$s(t)$ je tedy nějaký stav modelu v časovém okamžiku t. Dále ještě předpokládáme, že pro všechny časové okamžiky s_i platí podmínka:

$$\sum_{1}^{N} a_{ik} = 1$$

Přechod z jednoho stavu do druhého způsobuje, že model generuje příznakový vektor a to podle rozdělení výstupní pravděpodobnosti $f_k(\boldsymbol{o_t})$ příslušné k tomuto stavu. Funkce rozdělení výstupní pravděpodobnosti je tedy definována vztahem:

$$f_k(\boldsymbol{o_t}) = P(\boldsymbol{o_t} \mid s(t)) = s_k$$

kde *P* je hustota pravděpodobnosti a s_k je stav v čase *t*.

Rozdělení výstupní pravděpodobnosti musí při modelování řečových zvuků splňovat několik vlastností. Musí být dostatečně specifické, aby od sebe oddělilo různé zvuky, ale současně musí být dostatečně robustní, aby zahrnulo různorodost řečového signálu. Nejvíce používané rozdělení je v současné době spojité normální rozdělení se směsí hustotních funkcí. Tvar hustoty výstupní pravděpodobnosti je tvořen součtem jednotlivých hustot pravděpodobností. Každé z těchto hustot pravděpodobností je tvořená svým vlastním vektorem středních hodnot a kovarianční maticí. Parametry rozdělení výstupní hustoty jsou tedy tvořeny výše zmíněným vektorem středních hodnost, kovarianční maticí, ale také váhami jednotlivých složek. Tvar hustotní funkce je tedy dán vztahem:

$$f_k(\boldsymbol{o_t}) = \sum_{j=1}^{m} \pi_{kj} N(\boldsymbol{o_t}, \boldsymbol{u_{kj}}, \boldsymbol{\Sigma_{kj}})$$

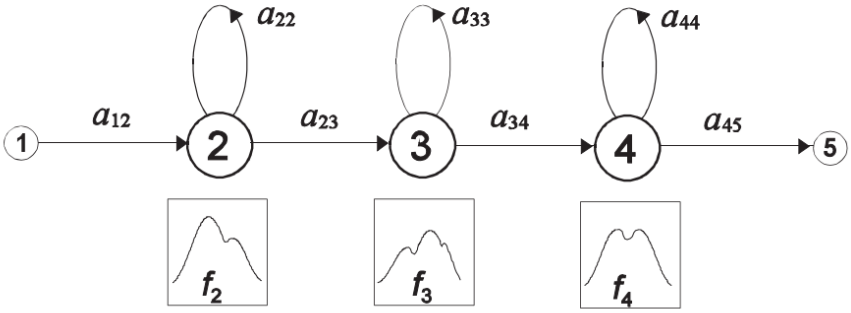
kde m je počet složek hustotní směsi našeho vektoru pozorování, π_{kj} je váha j-té složky a $N(\boldsymbol{o_t}, \boldsymbol{u_{kj}}, \boldsymbol{\Sigma_{kj}})$ je vícedimenzionální normální rozdělení s danou střední hodnotou a kovarianční maticí. Toto rozdělení si můžeme napsat jako:

$$N\left(\boldsymbol{o_t}, \boldsymbol{u_{kj}}, \boldsymbol{\Sigma_{kj}}\right) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma_{kj}})}} \exp[-0.5\left(\boldsymbol{o_t} - \boldsymbol{u_{kj}}\right)^T \boldsymbol{\Sigma_{kj}}^{-1}(\boldsymbol{o_t} - \boldsymbol{u_{kj}})]$$

kde *d* je dimenze vektoru pozorování $\boldsymbol{o_t}$. V praxi jde samozřejmě o dimenzi vektoru příznaků.

Nyní je čas podívat se na strukturu HMM. Při modelování mluvené řeči se nejčastěji používají takzvané levo-pravé Markovy modely. V současné době se pro modelování používají nejčastěji menší jednotky, než jsou slova. To mohou být například monofóny nebo trifóny. Tyto jednotky jsou odvozeny ze slov a je výhodnější modelovat je než jednotlivá slova, neboť počet slov používaných při mluvené řeči jde do řádů několika desetitisíců. Nyní si ještě pro lepší představu řekneme, co to vlastně trifón je. Trifón je jakýsi kontextově závislý foném, který bere v úvahu svůj pravý i levý kontext, což znamená pravý i levý foném. Přepis slova do fonémové nebo trifónové struktury lze ukázat na samostatně vysloveném slově *„ahoj“*, které má fonémový přepis *„ sil a h o j sil“* a trifónový přepis *„sil sil-a+h a-h+j o-j+sil sil“*. Řetězec sil vychází z anglického slova *silence* a má význam pauzy.

Strukturu monofónů respektive trifónů si můžeme vyjádřit například jednoduchým pětistavovým modelem. Jde tedy o levo-pravý model, jehož krajní stavy jsou neemitující a slouží pouze ke zřetězení do modelů celých slov popřípadě vět. Ostatní stavy tedy stav 2,3 a 4 jsou tedy stavy emitujícími a obsahují jednotlivé směsi hustotních funkcí. Trénování těchto parametrů modelu je prováděno pomocí Baum-Welchova algoritmu.



Obrázek 2 Schéma trifónu

2.2.3 Využití akustického modelu při reálném běhu systému

Při reálném běhu systému je třeba nejprve prohnat řečový signál blokem zpracování signálu. Tento blok nám převede řečový signál na posloupnost příznakových vektorů $\mathbf{o_1}$,… $\mathbf{o_t}$. Tyto příznakové vektory se dají získat několika metodami a to například metodou lineárního prediktivního kódování (LPC) či melovskou frekvenční filtrací(MFCC). Metoda lineárního prediktivního kódování je založena na metodě, kde každý z vektorů příznaků odpovídá parametrům matematického modelu hlasového traktu pro jeden mikrosegment řeči. Každý mikrosegment je dlouhý pouze několik milisekund a to přibližně 30, avšak výpočet parametrů se provádí s periodou 10 milisekund. Proces tedy probíhá tak, že postupně zpracováváme jednotlivé 30 milisekundové mikrosegmenty každý s 10 milisekundovým krokem.

Akustický model je pak využíván k určení jednotlivých pravděpodobností a to v každém mikrosegmentu pro všechny složky Gaussovského rozdělení. Tento výpočet nám v systému provádí takzvaný labeler. Matematicky lze jeho funkci pro výpočet pravděpodobnosti $f_k(.)$ jednoho vektoru příznaků popsat jako:

$$f_k(\boldsymbol{o_t}) = \sum_{j=1}^{m} \pi_{kj} N(\boldsymbol{o_t}, \boldsymbol{u_{kj}}, \boldsymbol{\Sigma_{kj}}) = \sum_{j=1}^{m} \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma_{kj}})}} \exp[-0.5(\boldsymbol{o_t} - \boldsymbol{u_{kj}})^T \boldsymbol{\Sigma_{kj}}^{-1} (\boldsymbol{o_t} - \boldsymbol{u_{kj}})]$$

kde většinu proměnných už bychom měli znát, ale pro jistotu si je zopakujeme. *k* značí index stavu, *m* je počet složek normální rozdělení, π_{kj} je váhová konstanta j-té složky normálního rozdělení k-tého stavu, $\boldsymbol{\Sigma_{kj}}$ značí kovarianční matici a $\boldsymbol{u_{kj}}$ střední hodnotu. Dá se tedy říct, že labeler nám počítá, s jakou pravděpodobností nějaký stav *k* vytváří příznakový vektor $\mathbf{o_t}$.

Rozpoznávání je tedy založeno na představě, že ke každé posloupnosti slov ***W*** náleží jeden skrytý zřetězený Markovův model *M*. Úkolem tohoto modelu je ocenit pravděpodobnost, že byla vyslovena daná posloupnost slov. Algoritmus, který nám vypočte pravděpodobnost generování posloupností výstupních vektorů ***O*** modelem *M* se nazývá forward-backward algoritmus.

Chceme-li rozpoznávat řeč v reálném čase, narazíme však na problém. Při rozpoznávání řeči se využívá až 100 tisíc Gaussovských hustotních funkcí při dimenzi

příznakového vektoru několik desítek. To vede k obrovským výpočetním zátěžím a problém je třeba nějak zjednodušit. Výpočetní náročnost tedy můžeme zjednodušit například minimalizací dimenze prostoru příznaků a nebo použitím pouze diagonálních kovariančních matic. Použití těchto metod samozřejmě znamená značné snížení výpočetních nároků, avšak také znamená snížení přesnosti rozpoznávání.

2.3 Jazykové modelování

Jazykový model je společně s blokem akustického modelu a akustické analýzy důležitou částí při rozpoznávání mluvené řeči. Účelem jazykového modelu je poskytnout co nejrychleji a nejpřesněji odhad apriorní pravděpodobnosti P(**W**) pro jakoukoliv posloupnost slov **W**. Tuto pravděpodobnost spočteme ze vzorce:

$$P(\boldsymbol{W}) = \sum_{k=1}^{K} P(\boldsymbol{w_k} | \boldsymbol{w_{k-1}} \dots \boldsymbol{w_1}).$$

Budeme-li dále uvažovat, že systémy rozpoznávání řeči pracují většinou s obrovsky rozsáhlými slovníky, pak není možné, abychom tyto pravděpodobnosti mohli dostatečně robustně odhadnout. Většinou se tedy pracuje s aproximací tohoto odhadu, kdy dojde k redukci odhadovaných parametrů. Nejčastěji používanou metodou je stanovení ekvivalentních tříd slov podle slovní historie. Jinak řečeno všechny historie slov, který se shodují v *n-1* slovech jsou zařazeny do stejné třídy

$$P(\boldsymbol{W}) \approx \sum_{k=1}^{K} P(\boldsymbol{w_k} | \boldsymbol{w_{k-1}} \dots \boldsymbol{w_{k-n+1}}).$$

Takovým to modelům říkám n-gramové modely. V praxi jsou pak nejčastěji využívané bigramové nebo trigramové jazykové modely. U bigramových jazykových modelů je *n* rovno dvěma a u trigramových je *n* rovno třem.

I přes velkou redukci odhadovaných parametrů při rozpoznávání souvislé řeči je však jejich počet enormní. Pro představu pro slovník s *V* položkami existuje stále V^n n-gramových statistik, které je potřeba odhadnout. Nejpřirozenějším způsobem je odhadnout n-gramové modely na základě relativních četností příslušných jevů v trénovacích datech. Avšak uvažujeme-li například slovník s $V = 10^5$ položkami, bude stejně potřeba odhadnout 10^{10} různých bigramů. Je očividné, že většina těchto bigramů se neobjeví v rozpoznávaném textu. Abychom nemuseli každému z těchto bigramů přidělit nulovou pravděpodobnost, bylo navrženo několik důmyslných postupů, které mají za úkol soubor odhadovaných pravděpodobností vyhladit. K tomu jsou využívány takzvané úsporné, interpolační nebo diskontní schémata. Pouštět se do podrobnější analýzy této problématiky není předmětem této práce.

Slovník také přiřazuje ke každému slovu, které je použito při úloze rozpoznávání a je obsaženo ve slovníku, jeho výslovnostní podobu. Pro každý jazyk tedy i pro náš český je

definována fonetická abeceda. Ta je pak použita pro fonetickou transkripci každého slova. Fonetická transkripce pak může obsahovat pro každé slovo i několik variant.

3 Trénování akustických modelů pomocí HTK

3.1 HTK

HTK z anglického *Hidden Markov Model Toolkit* slouží k definici, trénování a rozpoznávání pomocí skrytých Markových modelů. Dále obsahuje pomůcky pro parametrizaci řečových signálů, vyhodnocení výsledků rozpoznávání, práci s výslovnostními slovníky atd. HTK je napsán v jazyce C a je možné ho pro nekomerční využití stáhnout na webu.

3.2 Příprava k trénování

3.2.1 Co vše potřebujeme před tím, než začneme pracovat HTK

- Nahrané trénovací promluvy. To znamená zvukovou stopu všech promluv, na kterých chceme náš model trénovat. Tyto soubory umístíme do složky wav.
- Seznam trénovačích promluv pro parametrizaci ve formátu:
  /wav/veta001.wav /htk/veta001.htk
  /wav/veta002.wav /htk/veta002.htk
  kde v levém sloupci jsou názvy souborů, které chceme zparametrizovat, a v pravém sloupci jsou názvy souborů po parametrizaci. Označení /wav/ respektive /htk/ značí, že dané soubory budou načítány respektive ukládány z těchto složek. Soubory v levém sloupci už tedy dopředu existují a jsou to jednotlivé nařezané promluvy. Soubory v pravém sloupci budou teprve vytvořeny při parametrizaci dat. Zde je nutné vždy mít dopředu vytvořenou složku s názvem htk, jinak program zahlásí chybu, ale k tomu se ještě dostaneme později.
- Seznam testovacích a trénovacích promluv *test.scp a train.scp* ve formátu:
  /htk/veta001.htk
  /htk/veta002.htk
  kde v souboru *train.scp* je odkaz na všechny promluvy, na kterých bude náš model natrénován a ve složce *test.scp* naopak odkaz na promluvy, na kterých bude následně model otestován.
- Soubor s přepisem všech promluv na úrovni slov *words.mlf* a to jak testovacích tak trénovacích ve formátu:
  #!MLF!#
  "/veta001.lab"*
  dobrý
  den
  přeji
  .
  "/veta002.lab"*
  také
  vás
  zdravím.

….

kde první řádek tohoto souboru obsahuje řetězec #!MLF!# . Dále je pak vždy v uvozovkách uveden název věty, který se velmi podobá výše uvedeným parametrizovaným souborům. To z toho důvodu, že kdykoliv pracuje HTK s nahrávkou a potřebuje její přepis, hledá ji v souboru se stejným jménem, jako je název dané nahrávky akorát s jinou koncovkou a to *.lab*. Zpracovávali tedy HTK například nahrávku */htk/veta001.htk*, bude hledat její transkripci pod tímto názvem */htk/veta001.lab*. Toto značení je tedy pro HTK velmi důležité a při vytvoření *words.mlf* je třeba toto brát v potaz.

Dále vidíme, že každé slovo je na svém vlastním řádku a každá jednotlivá věta je ukončena tečkou opět na vlastním řádku. I tuto konvenci je třeba striktně dodržet, protože zapomene-li byť jedinou větu ukončit tečkou, HTK při trénování opět zahlásí chybu.

- Slovník výslovností *dict.txt* ve formátu:
  *ahoj a h o j _sp_*
  *absence a p s e n c e _sp_*
  ….
  Kde v levém sloupci jsou všechna slova vyskytující se v referenčním přepisu (*words.mlf*) a v pravém sloupci je jejich fonetická transkripce vyjádřená posloupností fónu z české abecedy.

  Z tohoto důvodu musíme mít dále vytvořeny soubory monophones0 a monophones1, které obsahují všechny fóny používané při fonetické transkripci. Rozdíl mezi těmito soubory je pouze v tom, že soubor monophones1 obsahuje navíc symbol krátké pauzy _sp_, který je do trénování přidán v průběhu, jak si ukážeme později. Je velmi důležité, aby tyto soubory obsahovali všechny fóny, které budou při fonetické transkripci využity, jinak se opět setkáme s chybou při trénování.

3.2.2 Vytváření souborů s přepisem na úrovni fonémů

Náš systém bude založený na modelování malých jednotek nazývanými monofóny. Modelování lze provádět také pomocí větších seskupení například trifónů, se kterými lze dosáhnout lepších výsledků, avšak trénování pomocí monofónů je jednodušší a pro naše případy postačující. Každý monofón bude reprezentován svým vlastním skrytým Markovým modelem (HMM). Jednotlivé monofóny tedy zastupují příslušné HMM. Proto kromě transkripce na úrovni slov, kterou máme v souboru *words.mlf* je třeba vytvořit také transkripci na úrovni monofónů. Tento přepis vytvoříme tedy již z dříve připravených words.mlf a dict.txt a pomocí programu HLEd z balíku HTK.

*HLEd –l * –d dict_sp.txt –i phones0.mlf mkphones0.led words.mlf*

*(HLEd –l * –d dict_sp.txt –i phones1.mlf mkphones1.led words.mlf)*

Parametry programu:
*-l** - Způsobuje, že do jmen souborů ve words.mlf je vložen symbol * na místo jeho cesty.
-d dict - Načte slovník výslovností ze soubory *dict_sp.txt*

–i phones0.mlf - Za parametrem –i je jméno výstupního souboru na úrovni monofónů.
mkphones0.led - Soubor s příkazy, které program HLEd vykoná.

Zde bych rád upozornil na první problém, na který můžeme narazit. Pokud jsme nějak upravovali slova ve výslovnostním slovníku, musíme naprosto stejně upravit slova i ve *words.mlf*. Pokud tak neučiníme, program nám hodí chybu, že dané slovo nemohl najít. Na tuto chybu jsem narazil při trénování mnohokrát a je třeba být tedy při úpravách opatrný.

Do soubory *mkphones0.led* vložíme tedy 3 příkazy, které program HLEd provede. Na první řádku vložíme příkaz *EX*, který způsobí, že každé slovo z referenčního přepisu je nahrazeno fonetickou transkripcí z výslovnostního slovníku. Na druhý řádek vložíme příkaz *IS _sil_ _sil_*. Ten vloží fón *_sil_* (z anglického *silence*-pauza, v našem případě dlouhá) na začátek a konec každé věty. Na poslední řádek vložíme příkaz *DE _sp_,* který odstraní všechny krátké pauzy, protože v první fázi trénování je nebudeme potřebovat.

Jakmile budeme mít hrubé modely přibližně natrénovány, přidáme zpět i model krátké pauzy *_sp_,* který odvodíme z modelu dlouhé pauzy *_sil_*. Z tohoto důvodu provedeme opět stejný příkaz s tím rozdílem, že soubor *mkphones1* již bude obsahovat pouze první dvě řádky. Nové soubory *phones0.mlf* a *phones1.mlf* budou poté vypadat takto:

phones0.mlf
#!MLF!#
"/veta001.lab"*
*_sil_*
a
h
o
j
j
a
k
s
e
m
A
S
t
y
j
o
*_sil_*

phones1.mlf
#!MLF!#
"/veta001.lab"*
*_sil_*
a
h
o
j
*_sp_*
j
a
k
*_sp_*
s
e
*_sp_*
m
a
s
….

3.2.3 Parametrizace řečových událostí

Pod pojmem parametrizace si představme převod zvukové nahrávky na posloupnost vektorů parametrů. Nahrávky by bylo možné parametrizovat i při trénování, ale parametrizaci by bylo nutno provádět při každém trénovacím cyklu. Takhle stačí parametrizovat pouze jednou a díky tomu si ušetříme spoustu času. Před spuštěním kódu je nutné mít dopředu vytvořený adresář s názvem HTK, do kterého se nám parametrizované soubory vytvoří.

HCopy -T 1 -C CF_param.mfc -S param.scp

Kde:
-*S param.scp*- Seznam veškerých souborů pro parametrizaci.
-*C CF.mfc*- Určuje co má přesně HCopy se soubory udělat (vstupní a výstupní formát).

V našem případě je soubor *CF.mfc* již vytvořen, takže se jím nebudeme příliš zabývat. Parametrizovat se bude do MFCC 13 koeficientu + delta + delta delta koeficienty. Po parametrizaci dostaneme pro každý mikrosegment vektor o velikost 39. Pod pojmem mikrosegment si představme časový úsek v řádech několika mikrosekund.

3.3 Tvorba monofónních modelů

Nyní přijde na řadu část, ve které budou jednotlivé fóny české abecedy reprezentovány každý jedním skrytým Markovým modelem.

3.3.1 Definice HMM

Na začátku si definujeme, jak budou jednotlivé Markovské modely vypadat. Každý model bude mít 5 stavů. Tyto stavy si můžeme představit jako 5 teček vedle sebe, při čemž krajní tečky jsou takzvané neemitující stavy. To znamená, že negenerují žádné vektory parametrů a slouží pouze ke spojení jednotlivých modelů. Jsou také v topologii automaticky obsaženy, takže není potřeba je nějak definovat. Obecný formát jednotlivého modelu se z konvence ukládá do souboru s názvem *proto.* V praxi pak Markův model v HTK například pro písmeno A vypadá následovně.

~h "A"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 39
-5.260925e+000 -6.052345e+000 -1.219461e+000 -2.440739e+000……
<VARIANCE> 39 2.208805e+001 7.596239e+000 4.013688e+000……
<GCONST> 7.698160e+000

<STATE> 3
<MEAN> 39
-5.260925e+000 -6.052345e+000 -1.219461e+000……

<VARIANCE> 39
2.208805e+001 7.596239e+000 4.013688e+000 2.436403e+000…….
<GCONST> 7.698160e+000
<STATE> 4
<MEAN> 39
-5.260925e+000 -6.052345e+000 -1.219461e+000
<VARIANCE> 39
2.208805e+001 7.596239e+000 4.013688e+000……
<GCONST> 7.698160e+000
<TRANSP> 5
0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
0.000000e+000 6.000000e-001 4.000000e-001 0.000000e+000 0.000000e+000
0.000000e+000 0.000000e+000 6.000000e-001 4.000000e-001 0.000000e+000
0.000000e+000 0.000000e+000 0.000000e+000 7.000000e-001 3.000000e-001
0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>

V obecném tvaru jednotlivého modelu (*proto)* bychom ještě na první řádce našli tento kód.

<VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC>

Kde parametr *VECSIZE* určuje délku jednotlivých vektorů parametrů. Asi nebude žádným překvapením, že tato délka bude 39. Dále určuje jejich typ *MFCC_D_A_0*. To znamená, že používáme melovské kepstrální koeficienty s nultým koeficientem, delta koeficienty plus akcelerační delta-delta koeficienty. Z toho také vyplývá naše délka jednotlivých vektorů, neboť máme 3x13 melovských kepstrálních koeficientů.

Vrátíme-li se k výše uvedenému modelu, tak za řetězcem ~h je vždy v uvozovkách uvedeno jméno jednotlivého fónu. *Numstates* nám uvádí počet stavů každého modelu. Dále vždy máme uvedené jednotlivé stavy od dvou do čtyř a pro každý z nich vektor středních hodnot a kovarianční matici. Nakonec máme ještě parametr *TransP,* který představuje informaci o velikosti matici přechodů a poté vlastní matici přechodu. V matici je především důležité, zda-li je jednotlivá hodnota nulová či nenulová. Je-li hodnota $a_{jk} > 0$, pak je definován přechod ze stavu *i* do stavu *j.* Pokud naopak je hodnota nulová, pak přechod těchto stavů definován není.

Vrátíme-li se nyní k samotnému trénování střední hodnota a kovariance prototypu jsou přepočítána pomocí HCompV. Tento program spočte celkovou střední hodnotu a kovarianci ze všech trénovacích dat a nastaví všechna Gaussova rozdělení v modelu na tyto hodnoty.

HCompV –C CF.mfc –f 0.01 –m –S train.scp –M hmm0 proto

Parametry programu:
-*C.CF.mfc* – Konfigurační soubor.
-*f 0.01* – Způsobí vytvoření marka vFloor1, které obsahuje dolní mez kovariance.

-m – Díky tomuto parametru se přepočte ne jen kovariance, ale i střední hodnota.
-S train.scp – Seznam trénovacích parametrizovaných nahrávek.
-M hmm0 – Jméno výstupního adresáře.

Výstupní adresář *hmm0* je nutné před spuštěním programu vytvořit. Před samotným trénováním je dobré si vytvořit těchto adresářů několik, abychom se tím později nemuseli obtěžovat. Budeme-li se držet předepsané konvence, je dobré vytvořit adresáře hmm0 až hmm9 a poté ještě *hmmA* a *hmmB* pro finální výsledky.

Nyní musíme v tomto adresáři vytvořit takzvaný *Master Macro File* (MMF, který bude obsahovat definice pro jednotlivé monofóny. Na začátku mají všechny monofóny stejné parametry, vypočtené z předchozího kroku. Požadovaný MMF tedy vytvoříme tak, že pro každý monofón uděláme kopii modelu proto a pouze přejmenujeme jeho jméno za parametrem *~h* podle jednotlivých monofónů. Toto by šlo udělat ručně, avšak dělat to pro všechny monofóny by bylo poměrně zdlouhavé. Je proto vytvořený skript, který nám práci ulehčí. Vstupem tohoto programu je soubor monophones0, který obsahuje všechny monofóny, kromě krátké pauzu, který zatím při trénování nepoužíváme a dále soubor s modelem proto.

MakeMMF proto monophones0 vFloors models

Parametr *proto* a *monophones0* je již vysvětlen výše. Parametr *vFloors* značí soubor, co vynikne jako meziprodukt v předchozím kroku a parametr *models* značí výstupní soubor, kam se jednotlivé modely monofónů uloží. Při spuštění tohoto programu musíme být ve složce *hmm0.*

Teď máme vytvořené modely pro jednotlivé monofóny a přistoupíme k části trénování. Trénování se provádí pomocí programu HERest, který je postaven na Baum-Welchově algoritmu. Před spuštěním programu se vrátíme o jednu složku výše. Budeme tedy ve složce, která obsahuje adresář hmm0 s jednotlivými modely.

HERest -T 1 -C CF.mfc -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm0/MODELS -M hmm1 monophones0

Parametry programu:
-I phones0.mlf – Soubor s monofónní transkripcí, který jsme vytvořili na začátku.
-t 250.0 150.0 1000.0 – Nastaví práh prořezávání Baum-Welchova algoritmu.
-H hmm0/MODELS – Vstupní soubor s modely, které budou reestimovány.
-M hmm1 – Výstupní adresář, kam budou uložené reestimované modely.

Ostatní parametry již byly vysvětleny několikrát, tak jen při jistotu si připomeňme, že výstupní adresář *hmm1* musí být vytvořený již dopředu. Pokud si shrneme fungování programu do několika vět, bude to vypadat asi takto. Všechny modely, které obsahuje soubor *monophones0* se hledají v adresáři *hmm0/MODELS*. Po nalezení jsou vtáhnuté do programu a reestimovány na základě trénovacích dat. Reestimované modely jsou poté uloženy do soubor MODELS avšak tentokráte do adresáře *hmm1*.

V tomto kroku bych rád upozornil na několik problémů, které mohou nastat. Pokud některý model ze souboru *monophones0* v souboru *MODELS* není, program zahlásí chybu a skončí. Dále se může stát, že pokud máme někde v předpřipravených datech nějakou formátovou chybu například uvozovky, tečku, špatnou mezeru, objeví se nám problém právě v tomto stádiu a tuto chybu je občas poměrně těžké naleznout. Další problém může nastat, pokud si vytváříme seznam monofónů sami a zapomene na nějaký. Pokud se však přes tento krok trénování dostaneme, máme z větší části vyhráno, neboť většina problémů je objevena právě v tomto kroku nebo před tím.

Pro zlepšení přesnosti je třeba reestimaci provést více než jednou. Pustíme tedy předchozí program ještě několikrát pouze s drobnými změnami.

HERest -T 1 -C CF.mfc -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm1/MODELS -M hmm2 monophones0

HERest -T 1 -C CF.mfc -I phones0.mlf -t 250.0 150.0 1000.0-S train.scp -H hmm2/MODELS -M hmm3 monophones0

HERest -T 1 -C CF.mfc -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm3/MODELS -M hmm4 monophones0

Reestimaci tedy provedeme celkem čtyřikrát pro co nejlepší přepočet parametrů modelu. Nabízí se zde otázka, zda-li by ještě několik dalších reestimací nezlepšilo dále výpočet našich parametrů, avšak již při čtvrté reestimaci dochází obvykle pouze k drobným změnám a pouštět tedy program znovu by již dále nemělo smysl. Jak je vidět, daný program má poměrně hodně parametrů, které při psaní z příkazového řádku je velice jednoduché splést. Stačí někde zapomenout na mezeru či pomlčku a okamžitě nám nastane velký problém. Proto je dobré si vytvořit nějaký skript, do kterého dané kódy napíšeme pouze jednou a pak už budeme spouštět pouze ten. Samozřejmě pokud pak nastane při trénování chyba, je těžší ji při použití skriptu najít. Avšak v těchto krocích by k chybě už dojít nemělo.

3.3.2 Úprava modelu pauz

Pro zatím jsme během našeho trénování používali pouze model dlouhé pauzy _sil_. Pokud se nad tím však zamyslíme, bylo by dobré umět rozlišit krátkou pauzu od dlouhé a také odchytit všelijaké šumy, které se v nahrávce vyskytnou. Změníme tedy topologii modelu pauzy. Přidáme tedy přechody ze stavu 2 do stavu 4 a stejně tak i opačně ze stavu 4 do stavu 2. Dále se vytvoříme také model krátké pauzy. Ten bude mít pouze jeden emitující stav svázaný s prostředním stavem modelu dlouhé pauzy. Model krátké pauzy, který budeme značit jako *_sp_*, bude tedy s modelem dlouhé pauzy sdílet některé parametry.

Nyní si ukážeme jednoduchý postup pro úpravu modelu *_sil_* a vytvoření modelu *_sp_*.

1. Při poslední reestimaci jsme náš poslední model uložili do složky *hmm4*. Do té se tedy dostaneme a otevřeme si soubor *models*. Ten můžeme otevřít v jakémkoliv textovém editoru. Zde mohu doporučit použít textový editor *PSPad*, který po vyzkoušení několika editorů se mi ukázal jako nejlepší.

2. Model krátké pauzy nyní vytvoříme tak, že si zkopírujeme prostřední stav modelu dlouhé pauzy *_sil_* a část jeho matice přechodu. Tato úprava zabere pouze několik desítek vteřin a tak je zbytečné na ni vytvářet nějaký skriptík. Zmiňovanou úpravu si nejlépe ukážeme na praktickém příkladu.

   Jestliže vypadá model dlouhé pauzy _sil_ takto:

   ~h "_sil_"
   <BeginHMM>
   <NumStates> 5
   <State> 2
   …
   <State> 3
   <Mean> 39
   -4.74 2.88 –1.03 …
   <Variance> 39
   1.11 2.85 1.92 …
   <GConst> 1.01
   …
   <TransP> 5
   0.00 1.00 0.00 0.00 0.00
   0.00 0.67 0.33 0.00 0.00
   0.00 0.00 0.84 0.16 0.00
   0.00 0.00 0.00 0.95 0.05
   0.00 0.00 0.00 0.00 0.00
   <EndHMM>

   Bude model krátké pauzy vypadat následovně:

   ~h "_sp_"
   <BeginHMM>
   <NumStates> 3
   <State> 2
   <Mean> 39
   -4.74 2.88 –1.03 …
   <Variance> 39
   1.11 2.85 1.92 …
   <GConst> 1.01
   <TransP> 3
   0.00 1.00 0.00
   0.00 0.84 0.16
   0.00 0.00 0.00
   <EndHMM>

   Shrneme-li ještě jednou naše úpravy, tak jsme upravili počet stavů z 5 na 3. Dále jsme pro náš prostřední stav číslo 2 použili parametry z modelu dlouhé pauzy ze stavu číslo

4. A nakonec jsme použili část prvního, třetího a pátého řádku matice přechodu dlouhé pauzy.

3. Použijeme program HHEd, který nám poupraví soubory s HMM.

HHEd –T 1 –H hmm4/MODELS sil.hed monophones1

Parametry:
hmm4/MODELS – Vstupní soubor.
monophones1 – Zde pozor, používáme již *monophones1* s modelem *_sp_.*
sil.hed – Soubor obsahující příkazy, které má HHEd vykonat.

Soubor *sil.hed* si musíme před spuštěním programu vytvořit a bude obsahovat následující příkazy.

AT 2 4 0.2 {_sil_.transP} – Přidá nám do modelu _sil_ přechod ze stavu 2 do stavu 4. Číslo 0.2 značí pravděpodobnost tohoto přechodu.

AT 4 2 0.2 {_sil_.transP} – Stejný příkaz jako minule pouze jdeme ze stavu 4 do stavu 2.

AT 1 3 0.3 {_sp_.transP} – Do modelu _sp_ přidá přechod ze stavu 1 do stavu 3. Oba tyto stavy jsou neemitující, neobsahují tedy žádné parametry.

TI silst {_sil_.state[3],_sp_.state[2]} – Spojí nám třetí stav modelu *_sil_* s druhým stavem modelu *_sp_*. Konečný stav bude v souboru *MODELS* ve složce *hmm4* uložen jako *silst*.

Po výše uvedených úpravách máme konečně modely pauz tak, jak jsme je potřebovali a nyní je na čase opět provést reestimaci parametrů, neboť úprava modelu pauz má samozřejmě velký vliv na tyto parametry.

HERest -T 1 -C CF.mfc -I phones1.mlf -t 250.0 150.0 1000.0-S train.scp -H hmm4/MODELS -M hmm5 monophones1

HERest -T 1 -C CF.mfc -I phones1.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm5/MODELS -M hmm6 monophones1

HERest -T 1 -C CF.mfc -I phones1.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm6/MODELS -M hmm7 monophones1

3.3.3 Přerovnání trénovacích dat

K přerovnání stejně jako k samotnému trénování budeme používat program HVite. Z názvu HVite lze odvodit, že program využívá Viterbiho algoritmus. Přerovnání spočívá přibližně v tom, že program vytvoří transkripci na úrovni monofónů a pokud je ve slovníku více variant výslovnosti, vybere podle dosud natrénovaných modelů tu nejlepší. Před přerovnáním je dobré si nejprve upravit slovník výslovností. To uděláme tak, že každé slovo

ve slovníku zdvojíme. Tím je myšleno, že na jedné řádce bude vždy slovo končit krátkou pauzou _sp_ a na další dlouhou pauzu _sil_. Díky tomu si pak bude moct program skutečně vybrat, která pauza vyhovuje lépe.

Nový zdvojený slovník bude vypadat nějak takto:

ahoj	*a h o j _sp_*
ahoj	*a h o j _sil_*
anténa	*a n t E n a _sp_*
anténa	*a n t E n a _sil_*

Program na následné přerovnání pak následovně:

*HVite –T 1 -l * –y lab -o SWT -b _SIL_ -C CF.mfc -m -a -H hmm7/MODELS -i aligned.mlf -t 250.0 -I words.mlf -S train.scp dict.txt monophones1*

Kde:
*-l ** - Způsobí, že do jmén souborů v souboru *MLF* je vložen symbol * místo skutečně cesty. Jména souborů jsou jen pro upřesnění řetězce v uvozovkách na začátku každé věty.
-y lab –Jména souborů budou mít koncovku *.lab*.
-o SWT – Formát výstupního souboru. Při tomto nastavení se nám nevypíše výpis skóre, celých slov a časových údajů a jsou vypisovány pouze monofóny a výstupní soubor je ve stejném formátu jako phones1.mlf.
*-b _SIL_* - Vkládá slovo *_sil_* na začátek a konec každé promluvy.
-C CF.mfc – Konfigurační soubor, stále stejný.
-a – Tento parametr říká programu, zda-li má provést rozpoznání nebo přerovnání. V tomto případě provede přerovnání.
-m – Parametr *m* nám zajistí výstup na úrovni fonémů.
-i aligned.mlf –Výstupní soubor s novou monofónní transkripcí.
-t 250.0 – Prořezávací práh.
dict.txt – Nový zdvojený slovník výslovností.

Výsledkem přerovnání bude tedy soubor *aligned.mlf*, který obsahuje nové monofónní transkripce promluv. Tento soubor může vypadat následovně:

"/text0veta17.lab"*
*_sil_*
t
e
n
*_sp_*
z
A
p
a
s
*_sil_*

m
U
Z
e

Vidíme tedy, že dochází ke kombinaci krátkých a dlouhých pauz podle toho, která více vyhovuje. Také se ale může stát, že přerovnat některé věty se nepodaří. To je většinou z důvodu, že jsou věty špatně přepsané, nebo s velkým ruchem nebo se také může stát, že většinu nahrávky mluví jeden řečník a pár vět, které pak pronese někdo jiný, obvykle také přerovnáním neprojdou. Z tohoto důvodu musíme nyní upravit náš seznam trénovacích vět train.scp a vyřadit z něj věty, které neprošli přerovnáním. K tomuto účelu opět máme program, který práci udělá za nás.

CreateAligned.exe aligned.mlf train.scp aligned.scp ne.scp

Parametry program:
aligned.mlf – Výstupní soubor po přerovnání.
train.scp – Vstupní soubor trénovacích vět.
aligned.scp – Výstupní soubor trénovacích vět.
ne.scp - Soubor obsahující věty, které neprošli přerovnáním.

Po tomto kroku nám tedy vypadnou věty, které nám trénování tak trochu kazí a nyní je opět potřeba provést reestimaci. Ještě před tím bych upozornil na soubor ne.scp. Ten je dobré vždy po jeho vygenerování otevřít a prozkoumat. Měl by vždy obsahovat několik vět, které tedy neprošli zarovnáním. Ale také se může stát, že někde při přípravě dat došlo k závažnějšímu problému, například nám někde vypadla nějaká věta a pak nahrávky a jejich přepisy na sebe samozřejmě nesedí. Pak se nám tedy v tomto souboru může objevit velké množství vět a je dobré chybu odstranit a začít s trénováním od začátku.

Nyní tedy provedeme již výše zmíněnou reestimaci. Tentokrát však už použijeme nový seznam trénovacích promluv a nový monofónní přepis.

HERest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0-S aligned.scp -H hmm7/MODELS -M hmm8 monophones1

HERest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0-S aligned.scp -H hmm8/MODELS -M hmm9 monophones1

HERest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0 -S aligned.scp -H hmm9/MODELS -M hmmA monophones1

HERest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0 -S aligned.scp -H hmmA/MODELS -M hmmB monophones1

Výsledné modely máme tedy ve složce *hmmB* a můžeme se pustit k samotnému trénování. Speciálně pro tyto reestimace ještě jednou doporučuji vytvořit si skriptík, do kterého napíšeme tyto kódy, neboť psát je při každém trénování do příkazového řádku je zbytečná ztráta času.

3.3.4 Přidávání složek

Nyní jsme se dostali do stavu, kdy máme akustické modely natrénované co nejlépe. Chceme-li dále zlepšovat jeho přesnost, musíme do akustického modelu přidat více složek. K tomu opět použijeme nástroje ze sady HTK. Příkaz pro složkování poté vypadá následovně:

md hmm_2_0
HHEd -T 1 -A -C CF.mfc -H hmm_1_4/models -M hmm_2_0 add_next.hed
monophones1 > hmm_2_0\log
md hmm_2_1
Herest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0 -S
aligned.scp -s hmm_2_1\stats -H hmm_2_0\models -M hmm_2_1 monophones1 >
hmm_2_1\log
md hmm_2_2
Herest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0 -S
aligned.scp -s hmm_2_2\stats -H hmm_2_1\models -M hmm_2_2 monophones1 >
hmm_2_2\log
md hmm_2_3
Herest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0 -S
aligned.scp -s hmm_2_3\stats -H hmm_2_2\models -M hmm_2_3 monophones1 >
hmm_2_3\log
md hmm_2_4
Herest -T 1 -C CF.mfc -I aligned.mlf -t 250.0 150.0 1000.0 -S
aligned.scp -s hmm_2_4\stats -H hmm_2_3\models -M hmm_2_4 monophones1 >
hmm_2_4\log

kde nejprve si opět vytvoříme složky pro ukládání modelů. K tomu nám slouží příkazy *md hmm0, hmm1…* Dále přidáme pomocí programu HHEd složku do akustického modelu. To nám značí parametr *add_next.hed.* A nakonec provedeme čtyřikrát již dobře známou reestimaci modelů. Výsledný model máme uložený ve složce hmm_2_4 a z toho pak provedeme výsledné rozpoznávání. Po přidání složek dojde při rozpoznávání k poměrně velkým zlepšením. Složky přidáváme tak dlouho, dokud se úspěšnost při rozpoznávání nepřestane zvětšovat

3.4 Rozpoznávání

Pro zatím jsme natrénovali jednoduchý monofónový model a nyní je na čase přistoupit k rozpoznávání. Rozpoznávat budeme z promluv, které jsme doposud nepoužili pro natrénování modelů, ale které máme zařazené v souboru *param.scp*. Tyto věty jsou tedy zparametrizované. Pro rozpoznání použijeme již výše popsaný program HVite, který se dá tedy použít pro rozpoznání i prořezávání a v tomto případě ho použijeme pro rozpoznání.

3.4.1 Použití rozpoznávací sítě

Ještě před spuštěním programu si musíme připravit poslední věc a to je rozpoznávací síť v programu označenou jako parametr *wdnet*. V nejjednodušším případě ji připravíme tak, že do ní dáme pouze slova, která obsahují dané neznámé promluvy. Síť nám tedy bude říkat,

s jakou pravděpodobností může dané slovo být po jiném. Máme-li k dispozici jazykový model k problematice, kterou rozpoznáváme, rozpoznávací síť nepoužíváme. Jazykový model nám velmi stručně řečeno udává pravděpodobnosti různých slov a pravděpodobnosti slov, které mohou za těmito slovy následovat, avšak této problematice se zde hlouběji věnovat nebudeme.

*HVite -C CF.mfc -H hmmB/models -S test.scp -i vysledek_all.txt -l * -p -60.0 -w wdnet dict.txt monophones1*

Parametry:
hmmB/models – Složka s výslednými modely.
-S test.scp – Seznam testovacích vět.
-i výsledek_all.txt – Výsledek, který nám dá rozpoznávač.
*-l **- Způsobí, že do jmén souborů v MLF je vložen symbol * místo cesty k souboru.
-p – 60.0 – Tímto parametrem volíme volbu penalizace, čímž můžeme změnit váhu rozpoznávání krátkých slov.
-w wdnet – Rozpoznávací síť.
dict.txt – Zdvojený výslovnostní slovník.
monophones1 – Seznam všech monofónů.

Nyní tedy máme výsledky, které nám vyhodnotil rozpoznávač. Pro jejich zpřehlednění ještě použijeme program Hresults. Ten nám určí, s jakou úspěšností jsme rozpoznali jednotlivé věty. Abychom tuto úspěšnost určili, musíme také znát, co ve větách bylo řečeno. To my samozřejmě známe, neboť všechny tyto věty máme v souboru *words.mlf*. Zde bych také rád upozornil na jednu věc. Pokud budeme s výsledky dále pracovat, doporučuji zkontrolovat testovací sadu vět, tedy zda nám sedí přepsaný text s tím, co bylo skutečně řečeno. Pokud totiž vycházíme z přepisu, který psal člověk, je velmi pravděpodobně, že se tam nějaká chyba objeví.

Pro finální výsledky tedy použijeme program Hresults.

Hresults -f -I words.mlf monophones1 vysledek_all.txt > vysledek.txt

-I words.mlf – Náš referenční soubor, ve kterém jsou přepisy testovacích i trénovacích nahrávek.
monophones1 – Seznam všech monofónů.
výsledek_all.txt – Výsledky, které nám dal rozpoznávač.
výsledek.txt – Výsledný soubor, kde máme uvedeno, co bylo jak rozpoznáno.

Zde je ukázka, jak například mohou vypadat finální výsledky.

---------- Sentence Scores ----------

====== HTK Results Analysis ======

Date: Thu Feb 25 20:10:11 2016

Ref : words.mlf

Rec : vysledek_all.txt

---------- File Results ----------

text0veta1.rec: 57.89(52.63) [H= 11, D= 2, S= 6, I= 1, N= 19]

text0veta2.rec: 84.21(68.42) [H= 16, D= 1, S= 2, I= 3, N= 19]

text0veta3.rec: 33.33(22.22) [H= 3, D= 1, S= 5, I= 1, N= 9]

text0veta4.rec: 46.67(46.67) [H= 7, D= 3, S= 5, I= 0, N= 15]

…

SENT: %Correct=0.00 [H=0, S=74, N=74]

WORD: %Corr=62.83, Acc=57.13 [H=727, D=124, S=306, I=66, N=1157]

Z těchto výsledků se dá vyčíst mnoho a popřípadě upravit ještě nějaké parametry při rozpoznávání. Pro nás asi klíčové výsledky jsou na posledních dvou řádkách, kde vidíme, že jsme rozpoznávali 1157 slov s úspěšností okolo 57 procent.

3.4.2 Použití jazykového modelu

V předchozím kroku jsme tedy použili nejjednodušší rozpoznávací síť, která obsahovala pouze slova z testovacích promluv. Nyní pro rozpoznávání použijeme jazykový model. Jazykový model nám stručně řečeno určuje pravděpodobnost pro libovolnou posloupnost slov. Jedná se o obrovské soubory, které obsahují sta tisíce slov. Jazykový model můžeme mít univerzální, který můžeme použít pro jakékoliv promluvy, ale také můžeme mít jazykový model pro konkrétní oblast rozpoznávání. Rozpoznáváme-li tedy například fotbal, určitě dosáhneme lepší úspěšnosti při použití jazykového modelu na fotbal než při použití jakéhokoliv jiného modelu.

Recognition.exe -ini XXX.INI -result WORD.mlf -list list.txt -htk –wip 2.0 –lmw 10.0

HResults -f -I reference.mlf reference.mlf WORD.mlf > vysledek.txt

Kde:
XXX.INI- Zde jsou definovány parametry rozpoznávání (váha jazykového modelu, hloubka prořezávání…)
WORD.mlf- Soubor kam se uloží to, co bylo rozpoznáno.
list.txt – Seznam rozpoznávaných vět.

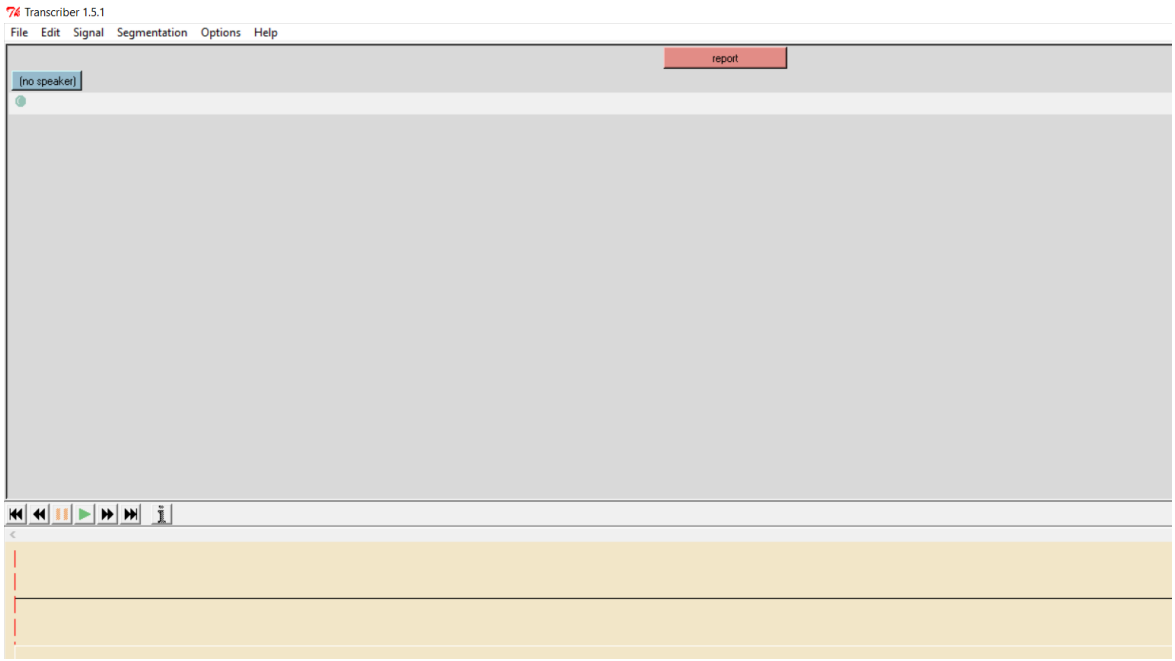
wip 2.0 – Penalta krátkých slov.
lmw 10.0 – Váha jazykového modelu.
reference.mlf – Tento soubor je roven výše používanému words.mlf. Obsahuje tedy referenční přepis toho, co zrovna rozpoznáváme.
vysledek.txt- Výsledný soubor, kde máme uvedeno, co bylo jak rozpoznáno. Obsahuje tedy to samé co *vysledek.txt* v kroku 3.4.1.

4 Návod k přepisu

V předchozí části jsme si ukázali, jak může probíhat trénování monofónového modelu. Na začátku jsme vycházeli z nějakého přepisu, ze kterého jsme pak dělali výslovnostní slovník, *words.mlf* a podobně. Nyní si ukážeme, jak mohou tyto přepisy vznikat.

4.1 Přepisování pomocí programu Transcriber

K přepisování nahrávek by se jistě našlo mnoho vhodných programů. My si k tomu zvolíme program s názvem Transcriber. Tento program je volně ke stažení na internetu. K dispozici je také několik verzí, ta nejaktuálnější je Transcriber 1.5.1. Program tedy stáhneme a pomocí jednoduchého postupu nainstalujeme. Poté se nám na ploše objeví červená ikonka tohoto programu. Tu tedy spustíme a otevře se nám samotný program, který vypadá následovně.



Obrázek 3 Program Transcriber

V dolní části programu se nám bude zobrazovat zvuková stopa nahrávky, v prostřední části náš přepsaný text a v horní části je několik možností programu.

Vzhledem k tomu, že budeme přepisovat český jazyk, musíme ještě do programu nahrát konfigurační soubor. Ten nahrajeme kliknutím na možnost *Options,* poté *Load configuration file* a pak už jen vybereme konfigurační soubor.

V dalším kroku si vyplníme v programu své jméno, aby bylo poznat, že jsme přepis napsali my. Klikneme tedy na možnost *File*, poté *Edit episode attributes* a vyjede nám malé okénko, kde do druhé řádky napíšeme své jméno.

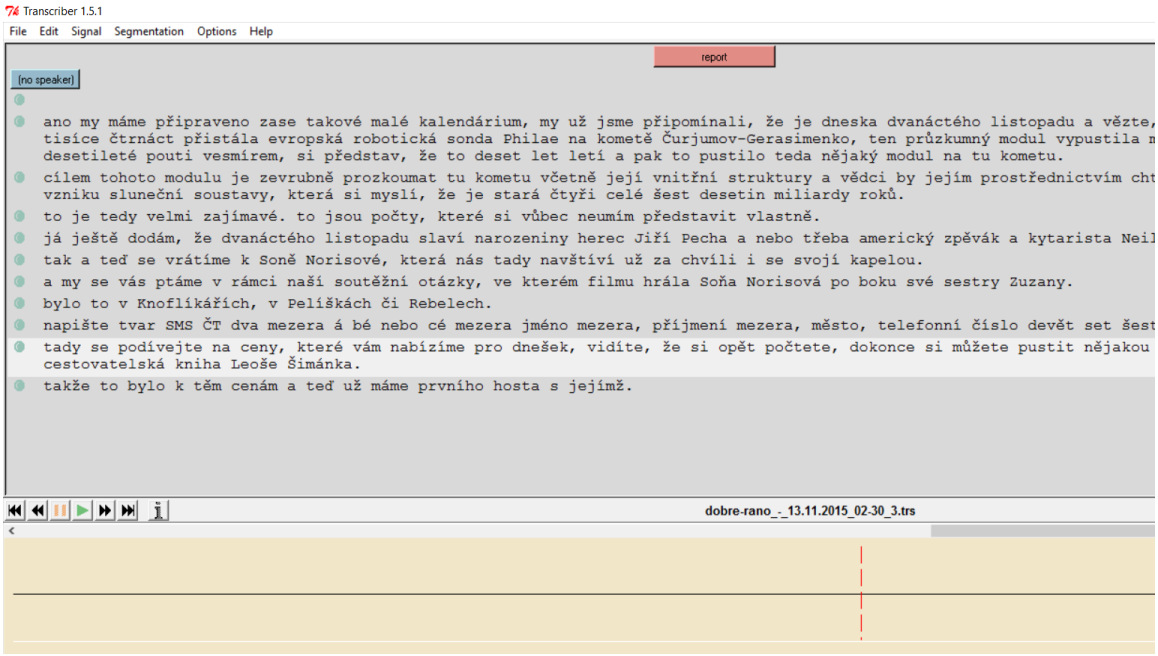
Nyní máme náš přepis podepsaný a můžeme se pustit do vytvoření nového přepisu. Klikneme na možnost *File* a vytvoříme nový přepis následným výběrem možnosti *New trans.* Poté na nás opět vyskočí okénko. To po nás chce, abychom mu vybrali zvukový soubor, který budeme chtít přepsat. Najdeme tedy soubor na disku a výběr potvrdíme. Pokud jsme udělali

vše správně, zobrazí se nám v dolní části zvuková stopa našeho přepisu. Dále je dobré si nový přepis uložit. To uděláme tak, že opět klikneme na možnost *File*, dále na *Save as* a vybereme místo na disku, kam soubor uložíme. Poté vždy když něco napíšeme, můžeme soubor uložit opět v této sekci kliknutím na možnost *Save,* avšak mnohem rychlejší možnost a také ta, kterou budeme dále používat je pomocí klávesové zkratky *Ctrl + S.*

4.2 Jak přepisovat

V předchozím kroku jsme si tedy vytvořili a uložili nový soubor, který budeme chtít přepisovat. Teď je tedy potřeba říci, jak vlastně budeme soubory přepisovat. Přepisy budeme psát podobně, jako kdybychom psali normální text avšak s pár rozdíly. Ty si zde v několika krocích ukážeme společně se základním ovládáním programu.

- Chceme-li pustit danou nahrávku a začít přepisovat, můžeme tak udělat buďto pomocí ikonky *play* těsně nad zvukovou stopou anebo mnohem praktičtější je používat klávesu Tabulátoru. Ta nám při jejím stisknutí nahrávku buď pustí, nebo zastaví. Ve zvukové stopě také můžeme pohybovat pomocí myše, když levým tlačítkem klineme na libovolné místo, kam se chceme dostat.
- Jednotlivé věty budeme rozdělovat do segmentů. Segment v programu Transcriber uděláme pomocí tlačítka *Enter*. Naopak chceme-li segment smazat, stiskneme klávesy *Ctrl + Enter*.
- Pokud by se stalo, že v jednom segmentu mluví řečník dlouho a text už je dlouhý přes 3 a více řádek, musíme danou větu rozdělit uměle. Vybereme si tedy vhodnou část, například místo, kde větu rozděluje čárka a tam daný segment rozdělíme do dvou.
- Pokud se stane, že řečník dlouhou dobu nemluví, nebo mu nerozumíme, označíme daný úsek takzvaným prázdným segmentem. Pod pojmem prázdný segment si představme segment, který je odněkud někam, avšak neobsahuje žádný přepsaný text. (viz obrázek 4)
- Narazíme-li na úsek, kde mluví dva řečníci přes sebe, tak napíšeme nejprve promluvu, co řekl jeden řečník, poté ukončíme tečkou a za tu napíšeme promluvu druhého řečníka a opět ukončíme tečkou, která nám zároveň značí konec segmentu.
  - Nespisovné koncovky se snažíme zespisovnit. Žádné větší úpravy však neděláme a jinak přepisujeme vždy to, co řečník řekl.
  - Číslovky píšeme vždy za pomocí slov. Čísla používáme pouze při určování pádů jednotlivých jmén (viz dále).
  - Nejedná-li se o jméno, budeme psát slovo vždy malým písmenem (i na začátku věty).
  - Stejně jako v normálně psaném textu bude i zde každá věta, v našem případě segment, končit tečkou.
  - Námi psaný přepis tedy může vypadat nějak takto.



Obrázek 4 Program Transcriber při praktickém použití

Zde krásně vidíme, jak jsou jednotlivé věty rozděleny do segmentů. Hned na začátku přepisu byla přibližně dvacetivteřinová část, kde se nemluvilo a hrála pouze znělka pořadu a proto je daný segment udělaný jako prázdný segment.

4.3 Značení jmén

Pokud děláme pouze jednoduchý přepis, kde nám jde pouze o přepsaný text, tak známe již vše potřebné. Může se ale stát, že budeme psát například sportovní přepis a budeme z něj chtít dostat i další informace, jako například jména sportovců, sportovišť a mnoho dalších tříd. Jména do závorek společně s pádem dáváme kvůli dalšímu využití pro jazykové modelování. Jména si můžeme rozdělit do tří základních kategorií.

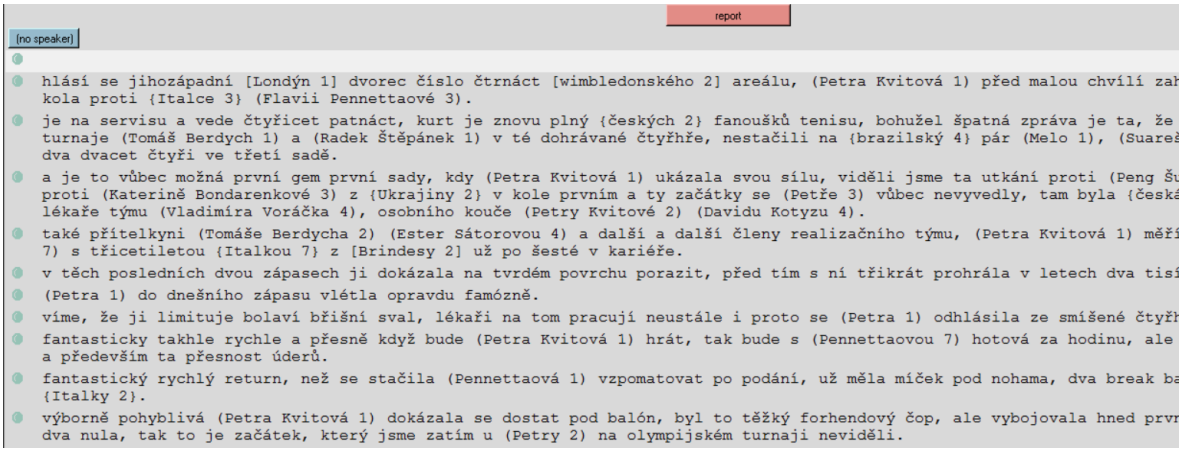
1. Jména sportovců – Do této kategorie budeme řadit jména sportovců, trenérů a všech osob, které budou v daném přepisu řečeny a mají co dočinění s daným sportem. Tato jména budeme dávat do kulatých závorek.
2. Jména klubů – Zde budou spadat jména veškerých klubů, klubových příslušníků ale také státních národností. Tuto kategorii budeme dávat do složených závorek.
3. Jména sportovišť – Tato kategorie bude asi nejméně používaná. Budou nám do ní spadat jména sportovišť, kde se daná událost koná. Značit tato jména budeme pomocí hranatých závorek.

Každé z těchto jmén tedy vždy vložíme do některé ze tří závorek. Spolu se jménem bude v závorce uvádět také pád, ve kterém bylo dané jméno řečeno. Nyní by bylo dobré si asi ukázat několik příkladů:

1. Jména sportovců:
   (Cristiano Ronaldo 1) přihrál (Benzemovi 3) do nadějné situace.
   Trenér (Koubek 1) je velice nespokojený s výkonem (Pavla Královce 2).

2. Jména klubů:
   {Sparta Praha 1} poráží v dnešním zápase {Plzeň 4} vysoko čtyři nula.
   {čeští 1} fanoušci ukázali, že dokáží fandit mnohem lépe než {Švédi 1}.
3. Jména sportovišť:
   Vítáme vás v [Holešovicích 6], kde za malou chvíli začne utkání.
   Olympiáda v [Londýně 6] je mnohem lepší než ta v [Pekingu 6] před čtyřmi lety.

V přepisu bude značení jmén vypadat asi takto.



Obrázek 5 Program Tranciber při ukázce značení jmén

4.4 Značení řečníků

Nyní už tedy umíme přepsat jak jednoduchý text, tak i nějaké například sportovní klání. V tomto kroku si ještě předvedeme, jak navíc označit, kdo co řekl.

Nejprve si ukážeme jak vytvořit prvního mluvčího. Při spuštění Transcriberu se nám vlevo nahoře objeví modrý obdélníček s nápisem *no speaker*. Klikneme tedy na něj a vyskočí nám okno, ve kterém se tvoří jednotliví mluvčí. Dále klikneme na možnost *Create speaker*, zadáme jméno mluvčího a potvrdíme tlačíkem *OK* v levém dolním rohu. Občas se může stát, že tlačítko *OK* pro potvrzení nemusí být vidět. To z toho důvodu, že vyskakovací okno je zmenšené. Zvětšíme ho tedy tak, že najedeme na pravý dolní roh a okno roztáhneme.

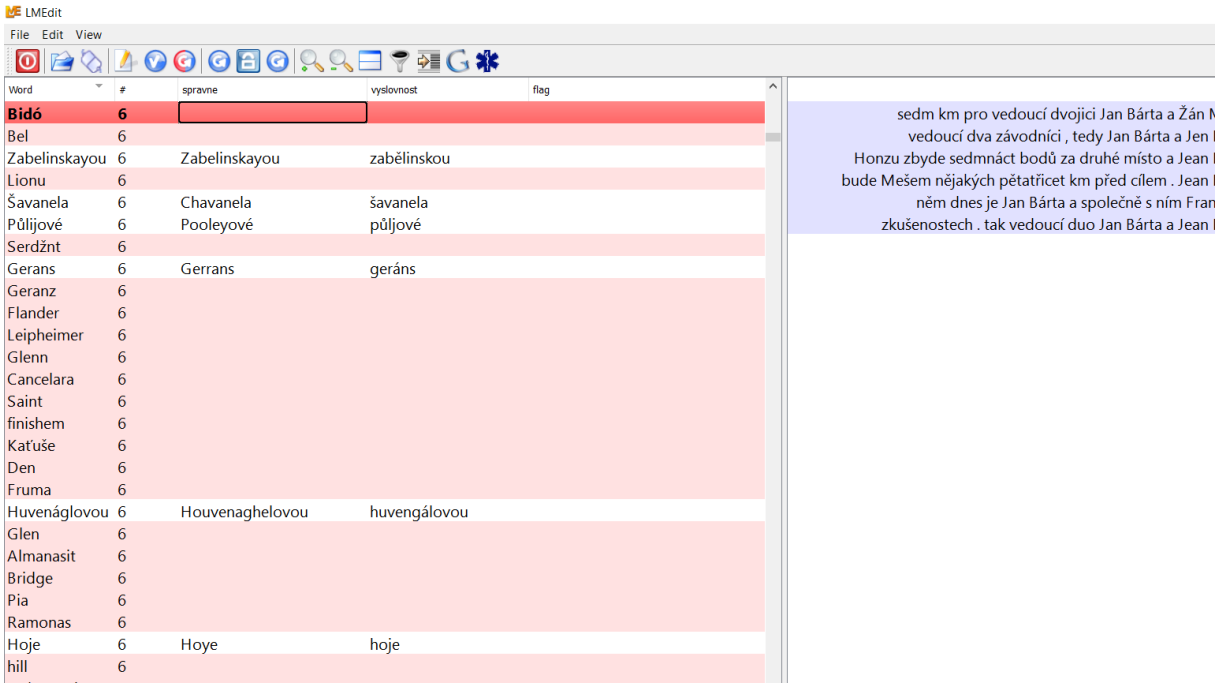
Chceme-li dále přidávat řečníky v průběhu přepisu, můžeme to dělat pomocí dvou možností. Buď to klikneme vždy na modrý obdélníček, který nám značí řečníka, a vybereme, popřípadě vytvoříme správného řečníka anebo se nám opět nabízí druhá možnost a to je použití klávesové zkratky *Ctrl + t*. Také se nám může stát, že dva řečníci budou mluvit přes sebe. To vyřešíme tak, že při vybírání řečníku zaškrtneme možnost *Overlapping speech* a pak už jen určíme, co který řečník řekl.

Dále se může stát, že budeme chtít nějaký segment označit jako neřečový, čili že v něm nemluví žádný řečník nebo mu není rozumět. Toho docílíme tak, že opět použijeme například klávesovou zkratku *Ctrl + t* a přibližně uprostřed vyskakovacího okna zaškrtneme možnost *No speaker.*

Program Transcriber nám nabízí ještě celou řadu dalších možností. Avšak pro naše potřeby si vystačíme s tím, co jsme si uvedli. Za zmínku stojí ještě možná jedna funkce a to je nastavení různých klávesových zkratek. To se nám bude hodit především při psaní jmén, kdy nemusíme vždy složitě psát například složené závorky, ale napíšeme pouze klávesovou zkratku například *Ctrl + L* a vyskočí nám hotová složená závorka. Tyto zkratky nastavíme tak, že klikneme na možnost *Option* a dále na *Bindings*. Poté nám vyskočí opět malé okno s několika možnostmi. Chceme-li vytvořit novou zkratku, klikneme na možnost *New*. Na první řádek poté zadáme klávesovou zkratku, kterou chceme použít a na druhý řádek napíšeme řetězec, který chceme při stisknutí této klávesové zkratky napsat. Poté vše už jen potvrdíme tlačítkem *OK* a můžeme hned vyzkoušet, že při stisknutí nově nadefinované zkratky nám vyskočí námi chtěný řetězec.

4.5 Kontrola

Po přepsání většího množství přepisů je dobré provést nějakou hromadnou kontrolu. K tomu využijeme program LM Edit. Nejprve projedeme všechny přepisy nějakým nástrojem, který nám vyhodnotí, jaká slova jsou pro něj neznámá. Tento nástroj si můžeme představit jako slovník, který obsahuje například půl milionu českých slov. S tímto slovníkem porovnáme jednotlivé přepisy a ten nám pak vyhodí slova, která nezná. To jsou obvykle slova, která obsahují překlepy nebo neznámá jména, která jsou potřeba zkontrolovat atd. Soubor neznámých slov pak načteme do programu LMEdit.



Obrázek 6 Program LMEdit

Na obrázku 6 můžeme vidět, jak program LMEdit vypadá. Úplně vlevo je vždy neznámé slovo. Dále je pak sloupec obsahující číslo, kolikrát se dané slovo vyskytlo špatně. Poté následují sloupce pro správný tvar slova a jeho výslovnost. V pravé části programu pak je zobrazen kontext, ve kterém se slovo vyskytuje. Detailnější popis jak správně pracovat

s LMEditem je samozřejmě mnohem obsáhlejší, ale to není předmětem této bakalářské práce.

5 Praktická část

Praktická část této práce spočívá v natrénování akustických modelů pro několik sportů. Dále je pak prováděna snaha o zlepšení těchto modelů pomocí různých modifikací dat a na konci práce jsou vyhodnocovány a porovnávány jednotlivé výsledky.

5.1 Prvotní natrénování akustických modelů

Celá praktická část se točí okolo natrénování akustických modelů pro několik sportů. Z velké množiny sportů byl pro trénování vybrán basketball, házená, alpské lyžování, klasické lyžování, golf a motorismus. Tyto sporty nebyly vybrány pouze náhodou. Basketball a házená byli vybrány z důvodu jisté podobnosti sportů. Jedná se totiž o míčové hry hrané uvnitř přibližně se stejným akustickým pozadím a podobnější slovní zásobou než třeba plavání a hokej. Stejně tak bylo vybráno i alpské a klasické lyžování, kde je také jistá podobnost. A nakonec byly vybrány dva sporty, které nemají žádnou podobnost s ostatními ani mezi sebou a to golf a motorismus. Úkolem bylo tyto sporty tedy natrénovat a poté provést křížové testy jednotlivých sportů, abychom viděli, jestli podobnost mezi sporty opravdu platí nebo ne.

Na začátek tedy bylo potřeba pro každý sport sehnat jeho zvukovou stopu a jeho přepis. Veškeré tyto sporty byly přepsány v rámci projektu Fakulty kybernetiky na Západočeské univerzitě s Českou televizí, takže data bylo odkud brát. Od každého sportu bylo potřeba alespoň několik hodin přepisu, aby se dal akustický model natrénovat kvalitně.

5.1.1 Formát přepisů

Trénování akustických modelů tedy budeme provádět z přepisů. Libovolný přepis lze otevřít v jakémkoliv textovém editoru a může vypadat například takto:

<?xml version="1.0" encoding="CP1250"?>
<!DOCTYPE Trans SYSTEM "trans-14.dtd">
<Trans scribe="Ondřej Váchal" audio_filename="Basketball-zeny-CZE-AUS" version="9" version_date="110709">
<Episode program="" air_date="">
<Section type="report" startTime="0" endTime="2613.050">
<Turn startTime="0" endTime="2613.050">
<Sync time="0"/>
dobrý den z haly ve [Wukesongu 6] vás zdravíme spolu s (Jaroslavem Kovářem 7). dobrý den.
<Sync time="13.941"/>
před {českmými 7} basketbalisktami je čtvrtfinále olympijských her v [Pekingu 6], jejich soupeřkami budou mistryně světa z {Austrálie 2} soupeř tedy nejtěžší skoro možný kromě basketbalistek {Spojených států 2}, tady už je všechno připraveno k rozskoku.
<Sync time="32.503"/>

ve středovém kruhu je (Petra Kulichová 1) spolu s ní (Lauren Jacksonová 1), takže pojďme k sestavám.
<Sync time="40.306"/>
{český 1} tým, který vybojoval rozskok, hraje v té obvyklé základní sestavě, čtyři (Veselá 1), pět (Večeřová 1), devět (Machová 1), třináct (Kulichová 1) a patnáct (Vítečková 1).
<Sync time="51.948"/>
sestava {Australanek 2} je následující se sedmičkou (Taylorová 1), osmičku má (Batkovičová 1), desítku ústřední rozehrávačka mistryň světa (Harrowerová 1), dvanáctku (Snellová 1) a patnáctku kapitánku (Jacksonová 1).
<Sync time="66.516"/>
(Jardo 5) abychom uvedli fanoušky a diváky do tohoto utkání, jak už jsem říkal na úvod, našimi soupeřkami jsou mistryně světa.
…
</Turn>
</Section>
</Episode>
</Trans>

Na prvních dvou řádkách je uvedeno použité kódování a verze XML. Na třetí řádce v atributu *Trans scribe* je pak uvedeno jméno přepisovače a název souboru. V tomto případě jsem pod jménem přepisovače uveden já osobně, neboť na tvorbě přepisů jsem se několik let podílel. Poté je uvedena délka přepisů v sekundách a následně je pak již samotný přepis, který je vždy oddělen časovými značkami, které značí začátek a konec dané věty. Konec přepisu pak poznáme podle ukončení jednotlivých XML atributů.

5.1.2 Nařezání zvukové stopy

Jako první krok při přípravě dat si vytvoříme skript, kterým nařežeme zvukovou stopu podle jednotlivých vět. Časové značky, podle kterých budeme jednotlivé zvukové stopy řezat, získáme z přepisu, kde každá věta je označená začátkem a koncem a je u ní udávána tato časová hodnota v sekundách. V praxi může věta v přepisu vypadat například takto:

> *<Sync time="9.457"/>*
> *ještě jednou dobré ráno, vítejte u zpráv.*
> *<Sync time="12.095"/>*

a my tedy budeme chtít vždy získat z přepisu začátek a konec věty a k tomu si každou větu musíme nějak unikátně pojmenovat. K nařezání souboru použijeme program *WaveCutter*. Data do něj budeme posílat v tomto formátu:

Vstup=Klasicke_lyzovani-skiatlon_muzi_1.wav
0.000 21.261 text0veta1.wav
21.261 30.185 text0veta2.wav
30.185 38.301 text0veta3.wav
38.301 49.727 text0veta4.wav

kde na prvním řádku je uveden název souboru, který obsahuje zvukovou stopu, a na dalších řádkách jsou pak jednotlivé věty s jejich časovými značkami. Do příkazového řádku pak vložíme následující kód:

WaveCutter.exe casy1.txt ./ ./

casy1.txt-soubor, ve kterém jsou časové značky a názvy jednotlivých vět.
./ ./ - použijeme-li tento parametr, nařezané soubory se nám uloží do složky, ve které právě jsme v příkazovém řádku.

Před spuštěním programu na nařezání musíme mít tedy v dané složce uložený zvukový soubor, který chceme nařezat a soubor s časovými značkami. Výstupem programu jsou tedy námi chtěné zvukové soubory nařezané podle vět, které budeme dále zpracovávat.

5.1.3 Vytvoření referenčního přepisu a výslovnostního slovníku

Pro vytvoření souboru referenčního přepisu *words.mlf* dostaneme z našich přepisů pouze ta data, která obsahují samotné promluvy. To uděláme tak, že si vytvoříme program, který nám načte vždy pouze řádky, které obsahují přepsané promluvy. Ty pak uložíme do nového souboru, kde vždy každá věta bude na samostatném řádku. Z věty je před uložením potřeba odstranit všechny čárky, tečky, ale také označení jednotlivých jmén společně s jejich pády. Soubor s větami, ze kterého budeme dále vycházet, vypadá například takto:

v motocyklové kategorii si drží vedení Marc Coma
dnes už se cítím mnohem lépe a mohu pořádně zabojovat
pod mírným tlakem byl Španěl Villadoms který hájí zatím třetí příčku
budu bojovat o třetí místo uvidíme jak to dopadne
Španěl Barreda Bort musí jet na maximum aby stáhl náskok vedoucího Comi

Máme-li vytvořený tento soubor, můžeme přejít k dalšímu kroku. Z tohoto souboru chceme vytvořit výslovnostní slovník a referenční přepis. To uděláme tak, že pomocí jednoduchého skriptu pošleme soubor na internetovou stránku *https://services.speechtech.cz/welcome/utils/text* . Ta nám následně vrátí tři soubory. Dva soubory se týkají výslovnostního slovníku a mají koncovku *check* a *ok.* Soubor s koncovkou *ok* obsahuje všechny výslovnosti, se kterými si program věděl rady a dále nejsou potřeba kontrolovat. Soubor s koncovkou *check* obsahuje naopak slova, se kterými si program nevěděl rady a vytvořil jejich výslovnost na základě pravidel českého jazyka a je nutné tyto soubory zkontrolovat. Soubory týkající se výslovnostního slovníku mají tento formát:

```
lamy            l a m i
sami            s a m i
maty            m a t i
od              o t
od              o d
hanu            h a n u
alp             a l p
alp             a l b
devon           d e v o n
nevel           n e v e l
```

kde vidíme, že v levém sloupci jsou vždy jednotlivá slova a v pravém pak jejich výslovnosti. Třetí ze souborů, který nám vygenerovala internetová stránka, pak obsahuje naše věty oddělené tímto znakem |_ .

Nyní již máme vše potřebné k vytvoření finálních slovníků výslovností a referenčního přepisu. Nejprve se tedy pusťme do vytvoření slovníku *dict_sp.txt*. Poté co jsme zkontrolovali jednotlivá slova v souboru s koncovkou *check*, zkopírujeme je a vložíme je všechny na konec do souboru *ok*. Dále si vytvoříme jednoduchý skript, do kterého tento soubor načteme a na konec každé řádky vložíme označení pauzy _sp_. Na první tři řádky pak ještě vložíme potřebný řetězec (viz níže) a výsledný slovník bude vypadat následovně:

*_END_ [] _sil_*
*_SIL_ [] _sil_*
*_START_ [] _sil_*
*plus p l u s _sp_*
*a a _sp_*
*a A _sp_*
*se s e _sp_*
*na n a _sp_*
*v v E _sp_*
*v f _sp_*
…..

Stejným způsobem vytvoříme i slovník *dict.txt*. Ten bude navíc obsahovat slova také s pauzu *_sil_,* aby si program při trénování mohl vybrat, kterou s výslovnosti použijeme. Na jedné řádce tedy vždy bude slovo s pauzou *_sp_* a na další přesně to samé akorát s pauzou *_sil_*. Tím pádem máme vytvořené výslovnostní slovníky potřebné pro trénování a můžeme se pustit k vytvoření referenčního přepisu *words.mlf*.

Vytvoříme si tedy opět jednoduchý skript, který nám načte poslední soubor vygenerovaný internetovou stránkou a to soubor s koncovkou *.text*. Z něj budeme postupně načítat jednotlivé věty a vždy když se objeví znak |_, tak víme, že nám začíná nová věta. Výstup našeho skriptu bude již soubor *words.mlf,* kde na první řádku vložíme řetězec #!MLF!# a dále pak budou následovat vždy názvy vět a poté na každé řádce jednotlivé slovo. Každá věta bude zde končit tečkou, která také bude na samostatné řádce. Formát souboru *words.mlf* máme v návodu k trénování HTK v bodě 3.2.1 a jediné, na co si zde musíme dát pozor, aby nám pojmenování jednotlivých vět sedělo s tím, jak jsme si věty pojmenovali při jejich nařezání. Není úplně od věci si náhodně několik nahrávek pustit a zkontrolovat, zda to, co v nich je řečen, opravdu sedí s tím, co je pod danou větou i v referenčním přepisu *words.mlf*.

5.1.4 Vytvoření seznamů train.scp, test.scp a param.scp

Poslední, co nám ještě k trénování schází, jsou seznamy trénovacích a testovacích vět a seznam promluv pro parametrizaci. Formát těchto souborů máme opět v návodu na trénování a je velmi důležité se ho držet. Jediné, na co je možná potřeba upozornit je, aby věty, které jsou v trénovacích a testovacích seznamech, byly odlišné. To znamená, máme-li

celkem 2000 vět, tak do trénovacího seznamu dáme například prvních 1900 vět a testovací seznam pak bude obsahovat posledních 100 vět.

5.1.5 První výsledky

Po připravení všech potřebných souborů pro jednotlivé sporty se můžeme pustit do trénování. Trénování provedeme postupně dle návodu pro každý sport zvlášť.

Po natrénování bylo tedy dosaženo prvních výsledků pro každý sport a bylo také dosaženo určité úspěšnosti při rozpoznávání. Pokud bychom chtěli tuto úspěšnost zlepšit, dosáhli bychom tak postupným přidáváním složek na konci trénovacího cyklu, avšak pro zatím se spokojíme s jednosložkovým akustickým modelem. Pokud si uvedeme nějaký příklad, tak například na házené bylo dosaženo úspěšnosti přibližně 40 procent. Pokud bychom přidávali složky, tak by naše úspěšnost na házené skončila někde okolo 60 procent. Ze všeho nejlépe dopadl při prvním rozpoznáváním motorismus, neboť data, která byla pro něj použita, byla namluvena pouze jedním řečníkem a navíc ve studiu a tudíž bez nějakého většího akustického ruchu. Ostatní sporty skončili někdy mezi 30-45 procenty. Pro otestování akustických modelů bylo vybráno vždy přibližně 100 vět z jednotlivých přepisů, které zhruba tisíc až dva tisíce slov.

Název sportu	**Alpské lyžování**	**Klasické lyžování**	**Basketball**	**Házená**	**Motorismus**	**Golf**
Prvotní rozpoznání	32.5%	27.0%	24.9%	33.1%	61.3%	26.5%
Počet testovacích slov	1986	2306	1457	1214	1179	1543

5.2 Oprava dat

V předchozím kroku bylo tedy dosaženo při trénování nějaké úspěšnosti. Naší snahou samozřejmě je, aby tato úspěšnost byla co nejlepší. Proto bude dále provedeno několik úprav, které nám pomohou dosáhnout lepší úspěšnosti.

5.2.1 Náhrady v přepisu

Vzhledem k tomu, že celé naše trénování vychází z přepisu, tak hlavní informace, kterou se můžeme pokoušet vylepšit je o tom, co vlastně bylo přepsáno. V přepisech se samozřejmě vyskytují některé překlepy. Ty nám pak trénování samozřejmě kazí naše modely a je tedy vhodné je nějakým způsobem odstranit. Pro každý sport nejprve použijeme univerzální slovník, kde je několik desítek tisíc náhrad a dále pak konkrétní slovník pro daný sport. Pro lepší představu slovník náhrad vypadá následovně:

abecdu abecedu
abch abych
abmice ambice

abolbovat absolvovat
Aborginies Aborigines
aboriginsů aboriginců

kde v levém sloupci je vždy chybné slovo (většinou překlep), které se může někde v přepisu vyskytovat a v pravém sloupci je pak slovo, kterým chybné slovo nahradíme. K provedení těchto náhrad je zapotřebí si opět udělat nějaký skript, kde postupně projedeme v přepisu slovo od slova a vždy zkontrolujeme, jestli se náhodou nějaký překlep ze slovníku neobjeví i v přepisu. V takovém případě překlep nahradíme za slovo ve druhém sloupečku. Provést náhrady vždy slovo za slovo je jednoduché. Problém pak nastane, chceme-li provést náhradu *m* slov za *n* slov. V tomto případě byly tyto náhrady provedeny tak, že byly postupně zkontrolovány všechny dvojice, trojice a čtveřice slov v přepisu, zda-li nebude nalezena shoda ve slovníku náhrad. Pro větší řetězce slov nebylo potřeba náhrady provádět, neboť takové už se ve slovníku nevyskytovali a akorát by se tím zvětšila výpočetní náročnost. Pro představu máme-li slovník o 50 000 slovech a přepis o 20 000 slovech a chceme jej slovo po slovu postupně projet a zkontrolovat, tak už to jsou miliony a miliony výpočtů. A provádíme-li to dále i pro všechny dvojice, trojice, čtveřice atd. slov, tak se výpočetní náročnost samozřejmě dále navyšuje.

Další náhrady v přepisu provedeme pomocí slovníku pro konkrétní sport, který jsme dostali z výše zmíněného programu LM edit. Ten je oproti předchozímu slovníku v trochu horším formátu. První sloupec ve slovníku obsahuje slova, která byla pro program neznámá. To znamená, že může obsahovat jak překlepy, tak také například jména nebo cizí výrazy. Druhý sloupec pak může a nebo také nemusí obsahovat správný tvar slov, třetí sloupec pak obsahuje výslovnost a čtvrtý ještě četnost slov. Pro lepší představu si opět uveďme příklad.

vlemi velmi 1
Borelou 1
čtvrtýho čtvrtého čtvrtého; čtvrtýho 1
presenty presenty preznty 1

Pro provedení náhrad v přepisu potřebujeme tvary pouze v prvním a druhém sloupci, proto je dobré si vytvořit opět krátký skript, do kterého si načteme pouze první dva sloupce a to vždy pouze v případě, obsahuje-li druhý sloupec nějakou náhradu. Pokud tomu tak není, slovo obvykle nebylo ve slovníku opraveno a nemůžeme ho tedy něčím nahradit. Vytvoříme si tedy nový slovník pouze s prvníma dvěma sloupci a ten pak naprosto stejným způsobem jako univerzální slovník náhrad projedeme slovo po slovu a provedeme náhrady v přepisu. Po provedení těchto náhrad ve všech přepisech a následném natrénování akustických modelů, se nám zlepší procentuální úspěšnost na jednotlivých sportech o několik procent.

5.2.2 Přidání výslovností

V předchozím kroku jsme pro úpravu přepisů použili dva typy slovníků. Jak si lze z výše uvedeného příkladu všimnout, druhý slovník pro konkrétní sport obsahuje také kromě samotných náhrad slov také jejich výslovnosti. Je tedy dobré toho využít. Vytvoříme si skript, který nám načte a následně zapíše pouze třetí sloupec slov tedy sloupec s výslovnostmi. Ten poté pošleme pomocí programu na internetovou adresu, jako když vytváříme slovník

výslovností. Program nám poté vrátí opět tři soubory, ze kterých tentokrát použijeme pouze dva týkající se výslovností. Tentokrát nebudeme výslovnosti nějak kontrolovat, ale oba soubory vložíme do jednoho. Jednotlivé řádky v souboru vypadají takto:

lends *l e n d z*
vistler *v i s t l e r*
lens *l e n s*
lens *l e n z*
flets *f l e c*
…..

kde v levém sloupci jsou výslovnosti ze slovníku, jak je napsal uživatel a v pravém pak výslovnosti z internetové stránky, které se nám hodí při rozpoznávání. Dále je ještě potřeba dostat tyto dvojice do stejného tvaru, jako jsou pak ve výslovnostním slovníku. Uděláme si tedy další skriptík, ve kterém projedeme první sloupec z tohoto souboru a třetí sloupec ze slovníku daného sportu a při shodě zapíšeme na výstup druhý sloupec z tohoto souboru tedy sloupec výslovností pro trénování spolu se správným tvarem slovesa ze slovníku sportu. Pro lepší představivost si ukážeme, jak bude vypadat výstup tohoto programu:

Lands *l e n d z*
Whistler *v i s t l e r*
Lands *l e n s*
…..
Zde krásně vidíme, že máme konečně správné dvojice, které potřebujeme do slovníku výslovností. Nakonec ještě máme dvě možnosti, jak tyto výslovnosti do slovníku výslovností přidáme. První možnost je, že výslovnosti jednotlivých slov nahradíme těmi, co jsme si vytvořili. Druhá možnost je, že výslovnosti pouze přidáme na konec a program si pak sám vybere, která je lepší. Druhá možnost přidání výslovností se nakonec ukázala být jako lepší, i když rozdíly nebyly nějak velké.

Po přidání těchto nových výslovností do slovníků výslovností *dict.txt* a *dict_sp.txt* tedy opět provedeme trénování. U jednotlivých sportů opět došlo k mírnému zlepšení, i když ne k tak velkému, jako při provádění náhrad v přepisu.

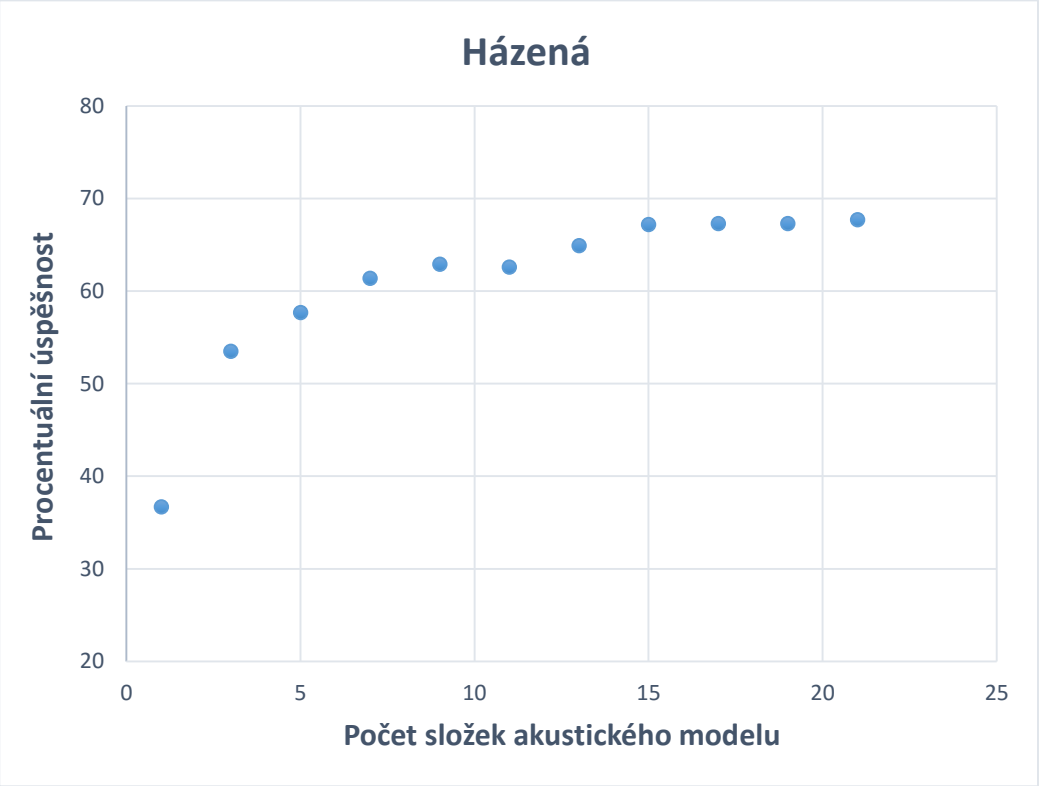
Výsledky po jednotlivých úpravách najdeme v této tabulce:

Název sportu	Alpské lyžování	Klasické lyžování	Basketball	Házená	Motorismus	Golf
Prvotní rozpoznání	32.5%	27.0%	24.9%	33.1%	61.3%	26.5%
Náhrady v přepisech	33.9%	28.1%	25.3%	35.9%	62.1%	26.9%
Přidání výslovností	34.8%	28.3%	26.0%	36.7%	62.1%	27.0%

5.3 Přidání složek do akustického modelu

V předchozím kroku jsme vylepšili trénovací data provedením náhrad a přidáním výslovností. Dosáhli jsme tedy v rámci možností co nejlepšího akustického modelu. Chceme-li dále vylepšit úspěšnost rozpoznávání, přidáme do akustického modelu více složek. K přidání složek opět použijeme nástroje se sady HTK. Složky přidáváme pomocí výše napsaného návodu (viz 3.3.4).

Výsledky trénování například na házené po přidání složek máme na následujícím grafu:



Obrázek 7 Vývoj úspěšnosti rozpoznávání při přidávání složek

Všechny sporty nám pak zobrazuje tato tabulka:

Počet složek	Název sportu	Alpské lyžování	Klasické lyžování	Basketball	Házená	Motorismus	Golf
	1.	34.8%	28.3%	26.0%	36.7%	62.1%	27.0%
	3.	45.1%	32.6%	42.7%	53.6%	65.7%	38.9%
	5.	50.0%	34.9%	47.8%	57.7%	70.1%	43.8%
	7.	51.6%	36.7%	51.3%	61.4%	71.0%	47.7%
	9.	52.4%	37.1%	52.1%	62.3%	73.7%	49.0%
	11.	53.9%	37.8%	53.7%	62.6%	75.0%	49.8%

	13.	54.9%	38.3%	54.7%	64.9%	75.9%	50.7%
	17.	55.7%	39.5%	56.6%	67.3%	76.0%	52.1%
	21.	55.5%	40.3%	56.9%	67.7%	77.6%	53.9%

kde krásně můžeme vidět, jak při přidání prvních několika složek úspěšnost vylétla rychle nahoru a následně okolo patnácté složky se téměř zastavila lehce pod sedmdesáti procenty.

5.4 Rozpoznávání s jazykovým modelem

Nyní jsme se dostali do stavu, kde máme všechny modely co nejlépe natrénované, a nasložkované na přibližně 20 složkách. Chceme-li úspěšnost při rozpoznávání dále zlepšit, je třeba se zaměřit druhou část modelování a to tedy na jazykové modelování. Pro tyto účely byly jazykové modely poskytnuty opět Fakultou kybernetiky na Západočeské univerzitě. Tyto modely vznikly také z přepisů, ale jejich příprava je nad rámec této práce. Tyto jazykové modely jsou trigramové a obsahují přibližně 500 000 slov.

Máme-li tedy k dispozici jazykové modely, budeme rozpoznávat s nimi. Výpočetní náročnost rozpoznávání se nám při jejich použití mnohonásobně zvětší a nemáme-li v počítači dostatek operační a fyzické paměti, můžeme zde narazit na problém. Rozpoznávání s jazykovým modelem provedeme dle návodu 3.4.2. Po rozpoznání všech sportů se úspěšnost rapidně zvětší. To především díky tomu, že poskytnuté jazykové modely jsou na velmi vysoké úrovni a díky tomu se v motorismu podařilo dosáhnout úspěšnosti okolo 90 procent. Vezmeme-li v potaz, že se zaprvé jedná o přepisy psané člověkem, které byly kontrolované pouze za pomocí slovníků a zadruhé že používáme pouze jednoduchý monofónový akustický model, je tato úspěšnost velice zajímavá.

5.5 Výsledky rozpoznávání

Výsledky rozpoznávání všech sportů spolu s postupnými úpravami můžeme vidět v následující tabulce.

Název sportu	Alpské lyžování	Klasické lyžování	Basketball	Házená	Motorismus	Golf
Prvotní rozpoznání	32.5%	27.0%	24.9%	33.1%	61.3%	26.5%
Náhrady v přepisech	33.9%	28.1%	25.3%	35.9%	62.1%	26.9%
Přidání výslovností	34.8%	28.3%	26.0%	36.7%	62.1%	27.0%
Po přidání 3 složek	45.1%	32.6%	42.7%	53.6%	65.7%	38.9%
Po přidání 21 složek	55.5%	40.3%	56.9%	67.7%	77.6%	53.9%
Použití jazykového modelu	72.1%	78.9%	66.9%	79.0%	89.4%	68.8%

Z tabulky krásně můžeme vidět, že ze všeho nejlépe dopadl motorismus, ale to hlavně z toho důvodu, že soubory, ze kterých byl natrénován, byli dopředu vybrané, aby trénování vyšlo co nejlépe, jak je již zmíněno dříve. Největším překvapením je sport házená, která i přes poměrně velký akustický ruch v pozadí nedopadla nejhůř. Největším skokanem se ukázalo být klasické lyžování, které se po aplikaci jazykového modelu zlepšilo ze 40 procent téměř na dvojnásobek.

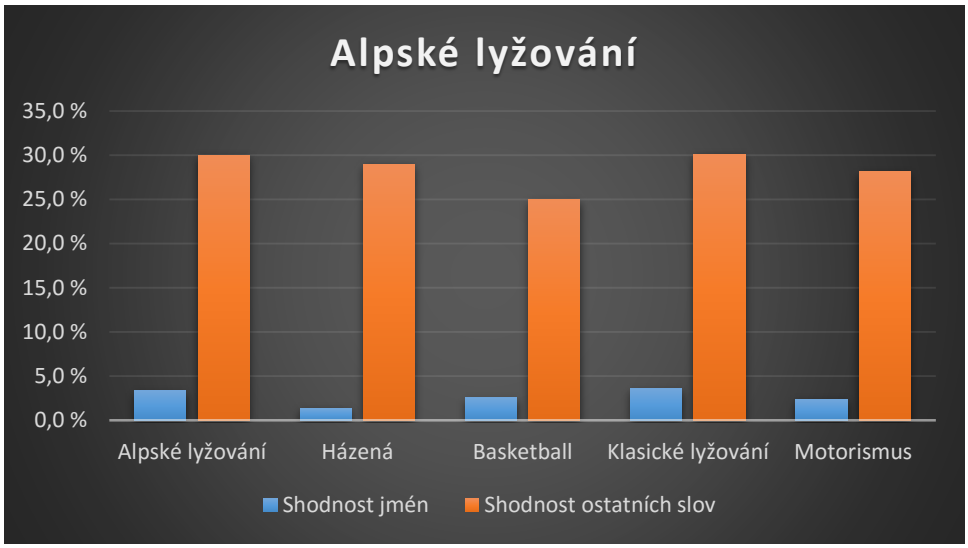
5.6 Analýza jednotlivých sportů

Analýza jednotlivých sportů je uvedena v následující tabulce:

Sport	Alpské lyžování	Basketball	Házená	Klasické lyžování	Golf	Motorismus
Počet řečníků	3	4	3	3	2	1
Počet slov za minutu	111	81	89	109	73	101
Jmén ve slovní zásobě(%)	15.3%	15.2%	14.6%	14.1%	7.9%	9.7%
Množství použitých slov	6250	4750	5770	5562	6190	4912
Největší shoda slov s	Klasické lyžování (31%)	Házená (37%)	Golf (4%)	Alpské lyžování (35%)	Alpské lyžování (30%)	Golf (35%)
Největší shod jmén s	Klasické lyžování (8%)	Házená (5%)	Basketball (4%)	Alpské lyžování (10%)	Klasické lyžování (4%)	Klasické lyžování (5%)

Nyní se ještě podíváme na všechny sporty trochu podrobněji.

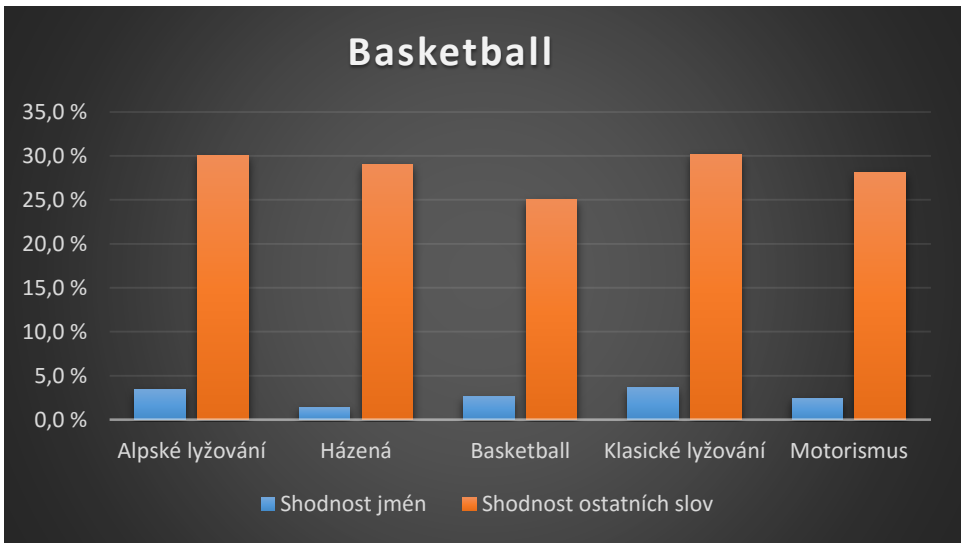
Alpské a klasické lyžování: U těchto dvou sportů bylo očekáváno, že jejich výsledky by měli být nejpodobnější. Co se týče výsledků rozpoznávání, tak oba sporty se pohybovali přibližně okolo hranice 75%. Z hlediska slovní zásoby se tato shoda také potvrdila a to jak u běžných slov tak u jmén.



Obrázek 8 Shodnost slovní zásoby s alspkým lyžováním

Z grafu lze krásně vidět, že především shodnost jmén byla mezi těmito sporty zdaleka největší. Co se týče tempa řeči, tak i zde byla shoda u těchto sportů velice podobná. Co je velmi zajímavé, že u těchto dvou sportů bylo tempo řeči zdaleka největší, což může být nečekané.

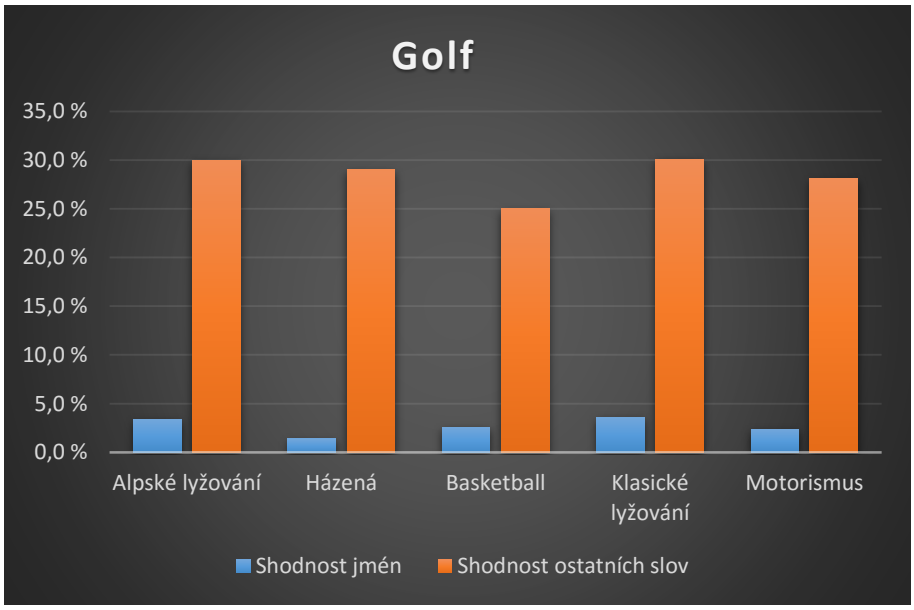
Basketball a házená: Další dvojice sportů, u které byly předpokládány podobné výsledky, je basketball a házená. Z hlediska výsledků rozpoznávání se tento předpoklad příliš nepotvrdil. Ačkoliv bylo množství řečníků podobné, tak především nahrávky basketballu obsahovali velké akustické rušení v pozadí a zřejmě z toho důvodu dosáhl basketball nejhorších výsledků.



Obrázek 9 Shodnost slovní zásoby s basketballem

Z hlediska běžné slovní zásoby si tyto sporty byly zdaleka nejpodobnější. Co se týče jmén tak zde byla velká podoba ještě společně s alpským lyžováním. Tempo řeči u obou sportů bylo velmi podobné a jednalo se o sporty, kde se mluvilo méně.

Motorismus a golf: Tyto dva sporty byly vybrány z opačného důvodu než předchozí dvojice. Neočekávali jsme zde žádnou shodu a naopak jsme chtěli potvrdit, že jak golf, tak motorismus nebudou příliš shodné s žádným z jiných sportů. Výsledky rozpoznávání u golfu nebyly příliš vysoké jen někde okolo 70%. To bylo opět způsobeno ruchem v pozadí, i když nebyl tak velký jako u basketballu a dále také častým měněním výšky hlasu řečníka (např. časté šeptání v dramatických situacích). Naopak motorismus dosáhl při rozpoznávání téměř 90%. Takto dobré výsledky byly dosaženy díky tomu, že velkou většinu nahrávek namluvil pouze jeden řečník a navíc ještě v klidném studiovém prostředí. Jediné co možná trochu kazilo úspěšnost u motorismu, byla občasná hudba v pozadí. Z hlediska slovní zásoby se tyto sporty žádnému jinému příliš nepodobali, což jsme přesně předpokládali.



Obrázek 10 Shodnost slovní zásoby s golfem

Tempo řeči obou sportů bylo také velmi rozdílné. Motorismus byl z tohoto hlediska někde uprostřed a naopak v golfu se mluvilo zdaleka nejpomaleji. To z toho důvodu, že v golfu mluvili moderátoři především, když hráči zahrávali míčky, jinak když se nic nedělo, tak bylo povětšinou ticho.

5.7 Křížové testy

V posledním kroku praktické části je potřeba provést ještě křížové testy mezi jednotlivými sporty. Pod pojmem křížové testy si představme to, že vezmeme natrénované akustické modely jednoho sportu a na nich vyzkoušíme, s jakou úspěšností budou rozpoznány slova z jiného sportu. Vezmeme tedy například akustické a jazykové modely z basketballu a na nich zkusíme rozpoznat parametrizované nahrávky například z házené.

Zde opět předpokládáme, že u obou lyžování a stejně tak u míčových sportů by mělo dojít při křížových testech k nejlepším výsledkům.

Výsledky křížových testů máme opět zaznamenány v tabulce:

Sport	Alpské lyžování	Basketball	Házená	Klasické lyžování	Golf	Motorismus
Modely alp. lyžování	72.1%	37.8%	45.7%	40.2%	6.2%	4.2%
Modely basketballu	31.7%	66.9%	32.4%	17.8%	3.6%	4.0%
Modely házené	34.9%	30.7%	79.0%	24.1%	3.2%	2.8%
Modely klas. lyžování	41.7%	27.6%	42.0%	78.9%	6.7%	11.9%
Modely golfu	8.0%	8.0%	6.2%	4.6%	68.1%	29.3%
Modely motorismu	4.5%	14.3%	3.9%	8.9%	22.0%	89.4%

kde můžeme vidět, že některé výsledky dopadly podle očekávání a některé naopak nečekaně. Alpské a klasické lyžování stejně jako z hlediska slovní zásoby dopadlo i při křížových testech nejpodobněji. U těchto dvou sportů byly výsledky pěkné a jednoznačné, což se bohužel už nedá říci o házené a basketballu. Zde jsme očekávali největší shodu mezi těmito sporty, avšak oba sporty se nejlépe shodovaly s alpským lyžováním. Naopak mezi posledními dvěma sporty motorismu a golfu nám nastala neočekávaná shoda. Pokud byly tyto dva sporty testovány na jiných jazykových a akustických modelech, dopadly vždy suverénně nejhůře. Avšak při křížových testech mezi sebou dopadly naopak s přehledem nejlépe.

6 Závěr

Cílem této bakalářské práce bylo seznámit se základními principy rozpoznávání řeči a následně natrénovat několik jednoduchých akustických modelů. Práce nejprve rozebírá základy rozpoznávání řeči, jejich stručnou historii a také úskalí, na která je možné při rozpoznání narazit. Dále se zaměřuje především na statistický způsob pro rozpoznávání řeči. Zde je hlavním tématem především tvorba akustických modelů pomocí Skrytých Markových modelů. Práce se v této části také zaobírá postupem pro natrénování akustických modelů od zpracování akustického signálu po vyčíslení jednotlivých pravděpodobností všech parametrů.

V další části práce obsahuje návod pro trénování akustických modelů pomocí sady HTK. Zde jsou postupně dopodrobna vysvětleny jednotlivé kroky a problémy, se kterými se můžeme při trénování setkat. Nejprve je v návodu uvedeno, co vše musíme připravit před samotným trénováním. Dále je uveden postup pro parametrizaci nahrávek pomocí MFCC. Poté jsou vytvořeny monofónní modely, které jsou několika způsoby postupně vylepšovány. Na konci návodu práce přistupuje k samotnému rozpoznávání. Zde jsou uvedeny dvě možnosti, jak finální rozpoznávání provést a to buď pomocí jednoduché rozpoznávací sítě anebo s využitím jazykového modelu.

Dále se práce zaměřuje na návod k přepisu pomocí programu Transcriber. Opět zde najdeme jednoduché seznámení s programem, návod jak v programu pracovat a také návod, jak pak samotné nahrávky přepisovat. Dále je zmíněno více způsobů, jak můžeme přepisovat a to buď přepis pouze jednoduchého textu anebo přepis s oddělováním řečníků a značením jmén do kategorií s pádem, což je pak dále využito pro tvorbu jazykového modelu. Nakonec této části je ještě zmíněn program pro kontrolu přepisů LMEdit, ve kterém je po přepsání odchycena drtivá většina chyb a přidána výslovnost ke jménům a slovům, která se vyslovují jinak, než se píší.

Praktická část začíná přípravou dat pro natrénování akustických modelů. Nejprve jsou nařezány jednotlivé nahrávky, dále je vytvořen referenční přepis a seznamy testovacích a trénovacích vět. Z těchto dat jsou natrénovány prvotní akustické modely a ty jsou následně vylepšovány. Nejprve jsou na každý přepis použity náhrady z univerzálního slovníku a dále pak ze slovníků pro konkrétní sporty, které byly vytvořeny v programu LMEdit. Pod pojmem náhrada si představme vždy nahrazení špatně přepsaného slova za správný tvar. Po těchto úpravách je opět provedeno natrénování akustických modelů a je dosaženo mírného zlepšení. Dále jsou pak do výslovnostního slovníku přidány výslovnosti ze slovníku vytvořeného v LMEditu. Nejprve byly jednotlivé výslovnosti nahrazeny za tyto nově vytvořené, avšak ukázalo se, že je lepší výslovnosti přidat a program při trénování si pak sám vybere, která výslovnost mu vyhovuje lépe.

Po aplikaci všech vylepšení na jednotlivé sporty a natrénování jednotlivých sportů máme tedy nejlepší možné jednosložkové monofónní modely. Ty jsou dále vylepšení přidáváním více složek do akustického modelu. Zde stejně jako při celém trénování se postupuje podle návodu 3 Trénování akustických modelů pomocí HTK. Složky jsou přidávány tak dlouho, dokud se nepřestane zlepšovat úspěšnost při rozpoznávání. Do této chvíle byla při rozpoznávání použita z jazykového hlediska pouze 0-gramová rozpoznávací síť. Pro další

rozpoznání již použijeme jazykové modely poskytnuté Katedrou kybernetiky, které nám naše výsledky posunou opět o úroveň výše.

V poslední části je provedena analýza jednotlivých sportů a diskuse výsledků. Bylo očekáváno, že při trénování bude existovat jistá podobnost mezi míčovými sporty házenou a basketballem a mezi lyžařskými sporty alpským a klasickým lyžováním. Ta se nakonec prokázala a to především z jazykového hlediska ale částečně také i z hlediska úspěšnosti při trénování. Dále byly také diskutovány poslední dva sporty a to golf a motorismus, mezi kterými jsme naopak žádnou shodu neočekávali a také se ani žádná neukázala. Na motorismu ale bylo při rozpoznávání dosaženo nejlepších výsledků téměř 90 procent a to díky studiové kvalitě nahrávek a také díky tomu, že motorismus byl jako jediný sport namluven pouze jedním řečníkem.

7 Seznam literatury

1. Psutka, J., Müller, L., Matoušek, J. a Radová, V. : Mluvíme s počítačem česky, Academia, Praha, 2006.
2. Psutka, J.: Komunikace s počítačem mluvenou řečí. Academia, Praha, 1995.
3. Ucnk.ff.cuni [online]. 2009 [cit 27.2.2012]. Dostupné na World Wide Web: <http://ucnk.ff.cuni.cz/oral/>
4. Young, S. et al., The HTK book (for HTK version 3.4). Cambridge University Press, 2006.
5. Python.org [online]. 2008 [cit 3.12.2008]. Dostupné na World Wide Web: <https://www.python.org/download/releases/3.0/>
6. Cs.wikipedia.com [online]. 2006 [31.5.2016]. Dostupné na World Wide Web: < https://cs.wikipedia.org/wiki/Extensible_Markup_Language>
7. Kky.zcu.cz [online]. 2011 [7.1. 2011]. Dostupné na World Wide Web: <http://www.kky.zcu.cz/cs/sw/jmzw>

Přílohy:

Příloha 1 – Referenční přepis words.mlf

#!MLF!#
"*/text0veta1.lab"
krásný
dobrý
den
vám
přeji
při
sledování
dalšího
letošního
dakarského
souhrnu
v
motocyklové
kategorii
nadále
kraluje
Španěl
Marc
Coma
.
"*/text0veta2.lab"
jeho
krajan
Nani_Roma
sice
zůstává
i
nadále
na
čele
průběžného
pořadí
automobilové
kategorie
ale
jeho
náskok
se

postupně
ztenčuje
.
"*/text0veta3.lab"
I
dnes
se
účastníci
Dakarské_Rallye
setkávali
s
hlubokými
dunami
.

Příloha 2 – Výslovnostní slovník

END [] _sil_
SIL [] _sil_
START [] _sil_
a a _sil_
plus p l u s _sil_
plus p l u s _sp_
a a _sp_
a A _sil_
a A _sp_
na n a _sil_
na n a _sp_
se s e _sil_
se s e _sp_
v v E _sil_
v v E _sp_
v f _sil_
v f _sp_
v v _sil_
v v _sp_
to t o _sil_
to t o _sp_
je j e _sil_
je j e _sp_
ale a l e _sil_
ale a l e _sp_
že Z e _sil_
že Z e _sp_

už	u S _sil_
už	u S _sp_
už	u Z _sil_
už	u Z _sp_
jsme	s m e _sil_
jsme	s m e _sp_
jsme	j s m e _sil_
jsme	j s m e _sp_

Příloha 3 – Ukázka chyb při rozpoznávání

Co mělo být rozpoznáno:

Dosáhnout jejich dna bylo někdy nemožné, ale na dno svých sil závodníci sáhli mnohokrát.

Co se rozpoznalo:

Doáhnout jich na to někdy nemožné ale naplno svých sil si závodníci sáhli mnohokrát.

Co mělo být rozpoznáno:

Dakarská_Rallye je závodem, ve kterém se prosazují sportovci mnoha světových zemí.

Co bylo rozpoznáno:

Dakarská_Rallye závodem, ve kterém se prosazují sportovci mnoha světových zemí.

Příloha 4 – Přepis z programu Transcriber

<?xml version="1.0" encoding="CP1250"?>
<!DOCTYPE Trans SYSTEM "trans-14.dtd">
<Trans scribe="Ondřej Váchal" audio_filename="basketbal_muzi_ESP_RUS6" version="5" version_date="121029">
<Episode>
<Section type="report" startTime="0" endTime="2854.319">
<Turn startTime="0" endTime="2854.319">
<Sync time="0"/>
opět fauloval (Kirilenko 1) a má třetí osobní chybu, vezmeme li v úvahu, že se hraje teprve třetí minuta třetí čtvrtiny, tak samozřejmě to je také nepříjemná zpráva podobně jako v té druhé čtvrtině si udělal tři fauly (Rudy Fernandez 1).
<Sync time="29.338"/>
a {Rusové 1} změnili. odešel (Ponkrashov 1), přišel (Fridzon 1).
<Sync time="35.272"/>
{Rusové 1} změnili obranný systém, opět jsou v té klasické zónové obraně v rozestavění dva

tři.
<Sync time="44.735"/>
teď to byla série {ruských 2} faulů nebo faulů {ruských 2} hráčů v rychlém sledu za sebou.
<Sync time="55.978"/>
(Fridzonův 1) faul poslal (Juana Carlose Navarra 4) opět na čáru trestného hodu.
<Sync time="81.344"/>
(Juan Carlos Navarro 1) snížil na dvacet devět třicet pět.
<Sync time="106.867"/>
rozhodčí (La Monika 1) odpískal faul (Marca Gasola 2).
<Sync time="117.492"/>
{Španělé 1} už se dopustili dvou faulů v dosavadním průběhu třetí čtvrtiny, {Rusové 1} mají tři.
<Sync time="130.512"/>
krásná přihrávka na (Shveda 4), který střílel úplně volný a skvělý, skvělý doskok (Saši Kauna 2).
<Sync time="139.351"/>
(Fridzon 1), (Fridzon 1) dává trojku vedení třicet osm dvacet devět pro {Rusy 4}.