České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra radioelektroniky

Automatický přepis záznamů přednášek s využitím nástrojové sady KALDI

Diplomová práce

Bc. Josef Hůla

Magisterský program: Elektronika a komunikace
Specializace: Audiovizuální technika a zpracování signálů
Vedoucí: doc. Ing. Petr Pollák, CSc.

Praha, květen 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hůla** Jméno: **Josef** Osobní číslo: **519706**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Audiovizuální technika a zpracování signálů**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Automatický přepis záznamů přednášek s využitím nástrojové sady KALDI

Název diplomové práce anglicky:

Automated Transcription of Lecture Records Using KALDI Toolkit

Pokyny pro vypracování:

1. Seznamte se s problematikou rozpoznání řeči na bázi DNN-HMM s užším zaměřením na rozpoznávání spojité řeči v audio-video záznamech přednášek. Navrhněte vhodnou architekturu DNN-HMM systému a vytvořte vhodný jazykový model pro úlohu přepisu přednášek z oblasti zpracování signálů.
2. Implementujte přepis přednášek s využitím nástrojů KALDI a dostupných vzorových receptů pro úlohu rozpoznávání spojité řeči. Shromážděte potřebná data pro trénování akustického modelu na bázi DNN a jazykového modelu pro danou tematickou oblast. Navrhněte obecné požadavky a proceduru tvorby jazykového modelu pro novou tematickou oblast.
3. Navržený systém otestujte na evaluačních datech z dostupných řečových databází resp. na zkorigovaném přepisu části vybrané přednášky. Zaměřte se především na vliv použitého jazykového modelu pro danou tematickou oblast. Dosažené výsledky srovnejte s přepisem získaným s vybranými veřejně dostupnými systémy.
4. S optimalizovaným systémem realizujte přepisy dlouhých záznamů přednášek.

Seznam doporučené literatury:

[1] J. Psutka, L. Miller, J. Matoušek, V. Radová: Mluvíme s počítačem česky. Academia, 2006.
[2] D. Yu, L. Deng. Automatic Speech Recognition A Deep Learning Approach. Springer-Verlag London. 2015
[3] D. Povey et al, The Kaldi Speech Recognition Toolkit. In Proc. of IEEE 2011 ASRU, Hawai, US, 2011.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Petr Pollák, CSc. katedra teorie obvodů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **03.02.2024** Termín odevzdání diplomové práce: **24.05.2024**

Platnost zadání diplomové práce: **21.09.2025**

doc. Ing. Petr Pollák, CSc. podpis vedoucí(ho) práce	doc. Ing. Stanislav Vítek, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Vedoucí práce:
doc. Ing. Petr Pollák, CSc.
Katedra teorie obvodů
Fakulta elektrotechnická
České vysoké učení technické v Praze
Technická 2
160 00 Praha 6
Česká republika

Deklarace

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24.5.2024

.....
Bc. Josef Hůla

Poděkování

Rád bych zde tímto poděkoval svému vedoucímu, doc. Ing. Petru Pollákovi CSc., za jeho vedení a podporu při řešení práce. Jeho výuka ve mě vzbudila zájem o vědní obor zpracování řeči a dala nezbytné znalosti, které jsem zde mohl aplikovat.

Abstrakt

Práce popisuje implementaci funkčního rozpoznávače dlouhých nahrávek spojité řeči s použitím nástrojové sady *Kaldi* a s jazykovým modelem upraveným pro daný tématický okruh. Takový systém má možnost využít množství textových dat vzniklých v průběhu let a aplikovat je při tvorbě přepisů záznamů přednášek, jejichž nahrávání se stalo v poslední době běžnou praxí. Použitá architektura již sice nepředstavuje *state-of-the-art* přístup, ale oproti dnes upřednostňovaným *End-to-End* systémům nabízí vysokou konfigurovatelnost a interpretovatelnost parametrů. Systém využívá GMM-HMM systém pro získání dat pro natrénování DNN-HMM rozpoznávače. Jedná se o hybridní systém, využívající TDNN sítě a její faktorizované řetězové struktury.

Trénování proběho na třech databázích o celkovém rozsahu 85 hodin z nichž jedna obsahovala spontánní řeč. V rámci práce je vytvořeno několik nezbytných podpůrných částí, a to zpracování PDF souboru, segmentace záznamů, tvorba jazykového modelu a generování přepisů v různých formátech. Vytvořené rozhraní je implementováno v jazyce Bash. V části popisující implementaci je dán stručný popis fungování a návod na použití. Následně je ověřen přínos rozšíření jazykového modelu vedoucí na zvýšení přesnosti rozpoznání o 20 % oproti obecnému modelu s nejlepším výsledkem 15,62 % WER. V závěru jsou dosažené výsledky srovnány s výsledky veřejně dostupného modelu Chirp od společnosti Google. Získané přepisy jsou podány v několika formátech a umožňují aplikaci v různých multimediálních prostředích. Na závěr jsou diskutována možná další vylepšení vytvořeného systému.

Klíčová slova: automatické rozpoznání spontánní řeči, GMM-HMM, DNN-HMM, Kaldi, Bash,jazykové modely, SRILM

Abstract

This thesis describes an implementation of a functional long spontaneous speech recognition application which utilizes the *Kaldi* toolkit and a domain specific language model. Such an application has the option to exploit a significant amount of data created within several recent years and apply it for the transcription of recordings of lectures, recording of which has recently become a common practice. Although the used architecture no longer represents a *state-of-the-art* approach, it offers high configurability and interpretability of parameters compared to the *End-to-End* systems preferred today. The system uses the GMM-HMM system to obtain data for training the DNN-HMM recognizer. It is a hybrid system which makes use of a TDNN network and its factorized chain structure.

The training took place on the basis of three databases with a total scope of 85 hours, one of which contained spontaneous speech. Several necessary supporting parts are created within the work, namely PDF file processing, recording segmentation, language model formation and transcription generation in various formats. The created interface is implemented in the Bash language. In the section describing the implementation, a brief description of the operation and instructions for use are given. Subsequently, the benefit of expanding the language model is verified, leading to an increase in recognition accuracy by 20 % compared to the general model with the best result of 15.62 % WER. Finally, the achieved results are compared with the results of the publicly available Chirp model from Google. The obtained transcripts are provided in several formats and allow application in various multimedia environments. Finally, possible further improvements of the created system are discussed.

Keywords: Automatic Spontaneous Speech Recognition, GMM-HMM, DNN-HMM, Kaldi, Bash, Language Models

Obsah

Seznam tabulek

Seznam obrázků

Seznam zkratek

ASR	Automatic Speech Recognition (automatické rozpoznávání řeči)
CLM	Conventional Language Model (konvenční jazykový model)
CMVN	Cepstral Mean and Variance Normalization (normalizace střední hodnoty a variance kepstra)
DFT	Discrete Fourier Transform (diskrétní Fourierova transformace)
DNN	Deep Neural Network (hluboká neuronová síť)
DPD	Parallel Distributed Processing (distribuované paralelní výpočty)
EM	Expectation Maximization (algoritmus maximální věrohodnosti)
FLM	Forward Language Modelling (dopředný jazykový model)
FSM	Finite State Machine (konečný automat)
GMM	Gaussian Mixture Models (Gaussovské smíšené modely)
HMM	Hidden Markov Models (skryté Markovovy modely)
LDA	Linear Discriminant Analysis (lineární diskriminativní analýza)
LF-MMI	Lattice-Free Maximal Mutual Information (bezmřížková maximální vzájemná informace)
LLM	Large Language Model (velký jazykový model)

LMW	Language Model Weight (váha jazykového modelu)
LVCSR	Large Vocabulary Continuous Speech Recognition (rozpoznání spontánní řeči s velkým slovníkem)
MFCC	Mel-Frequency Cepstral Coefficients (mel-frekvenční kepstrální koeficienty)
MLE	Maximum Likelihood Estimation (metoda maximální věrohodnosti)
MLLR	Maximum Likelihood Linear Regression (maximálně věrohodná lineární regrese)
MLLT	Maximum Likelihood Liner Transform (maximálně věrohodná lineární tranformace)
MMI	Maximum Mutual Information (maximální vzájemná informace)
NLP	Natural Language Processing (zpracování přirozeného jazyka)
OOV	Out of Vocabulary Word (výraz mimo slovník)
PCA	Principal Component Analysis (analýza hlavních komponent)
PLM	Pre-trained Language Model (předtrénovaný jazykový model)
SAT	Speaker Adaptive Training (Trénování přizpůsobené na mluvčího)
SGD	Stochastic Gradient Descent (stochastický gradientní sestup)
STT	Speech to Text (konverze řeč-text)
SVD	Singular Value Decomposition (rozklad na singulární hodnoty)
TDNN	Time Delay Neural Network (hluboká neuronová síť se zpožďovacími linkami)
UBM	Universal Background Model (univerzální model řečníka)

USM	Universal Speech Model (univerzální řečový model)
VTLN	Vocal Tract Length Normalization (normalizace na délku vokálního traktu)
WER	Word Error Rate (míra slovní chybovosti)
WFST	Weighted Finite State Transducer (váhovaný konečný transduktor)
WSJ	Wall Street Journal

Kapitola 1

Úvod

Lidská řeč a její psaná forma jsou bez debat nejefektivnějším způsobem přenosu informace mezi lidmi a tento fakt zatím nezměnil ani překotný vývoj moderních technologií, které jsou pak nástrojem k záznamu, uchování, přenosu a v poslední dekádě čím dál častěji i k interpretaci těchto informací. Kromě interpretace, kterou zaštiťují veřejně dostupné multimodální velké jazykové modely jakým je například ChatGPT od Open AI, dosahují nástroje v ostatních jmenovaných odvětvích schopností a možností lidí samotných, a to především díky desítkám let vývoje. Zde mějme na mysli záznam a reprodukci zvuku, obrazu a textu ve vysoké kvalitě, technologie cloudových úložišť, stále se zvyšující přenosové rychlosti bezdrátových komunikačních sítí.

Je zde však jedno odvětví, jehož výsledky nejsou srovantelné s výkony podávanými dětmi v základní škole a to i přes více než šedesát let vývoje a nasazení nejmodernějších technik. Tím odvětvím je konverze mluvené řeči do jeho psané podoby (STT), konkrétně pak automatické rozpoznání řeči (ASR). Cílem rozpoznání řeči je získání textu, který s nejvyšší pravděpodobností odpovídá vstupní nahrávce. Uplatnění takového sytému jsou dnes nedozírná, mimo jiné hlasové ovládání všemožných přístrojů, tvorba textových záznamů konferencí, debat apod., překlady řeči v reálném čase nebo dokonce konverzace s modely umělé inteligence.

Naprostými průkopníky byli v padesátých letech inženýři v Bell Laboratories, kteří jako první sestrojili přístroj, který byl schopný rozpoznat vyslovené číslice. Tento přístroj byl společností IBM o deset let později rozšířen o dalších šest příkazů [1] a jeho princip spočíval jako u jeho předchůdce v měření spektrální vzdálenosti. V osmdesátých letech navrhl F. Jelinek využít modelování založené na skrytých Markovových modelech (HMM) [2], které se od té doby hojně využívá za pomoci teorie konečných automatů (FSM). Dalším výrazným pokrokem bylo zapojení umělých neuronových sítí v letech devadesátých, které díky nové technologii paraleních výpočtů (DPD) přineslo po několika pokusech žádané výsledky v podobě systému SPHINX od CMU [3] nebo BYBLOS od BBN [4]. Tyto

aplikace již zahrnovaly slovníky v řádech tisíců až desetitisíců slov a pokud tuto podmínku splňují společně se schopností pracovat se spojitou přirozenou řečí, spadají do skupiny LVCSR systémů.

Dnes již prakticky všechny přelomové systémy využívají moderních metod strojového učení. Dobrým příkladem je open-source projekt Whisper od OpenAI [5]. Jedná se o enkodér-dekodér architekturu s transformery [6], mající za vstup mel-frekvenční spektrogram v logaritmické míře, trénovanou na různých datech o celkové délce 680 tisíc hodin. Jeho konkurentem je pak Chirp od společnosti Google, jehož hlavní předností je podpora pro více než sto jazyků. Postaven na architektuře konvolučního transformeru [7], jedná se o USM, univerzální řečový model, jehož trénování je rozděleno do tří fází: bez učitele na 12 milionech hodin záznamu pokrývajících přes 300 jazyků, bez učitele na kombinaci zvukových záznamů a textu nesouvisejícího se záznamem obsahujícího 28 bilionů vět z 1140 jazyků a nakonec učení s učitelem. Tento model se může použít buďto k obecnému ASR, nebo být dále upřesněn podle jazyka či domény. Ze zveřejněných výsledků se lze dočíst, že pro češtinu dosahuje poslední model Whisper 9% WER. Pro Chirp se výsledek pouze pro český jazyk nepodařilo dohledat.

Ač se zadání předložené práce může zdát přímočaré, není tomu zdaleka tak. Pro pochopení složitosti dané úlohy si nejprve utvořme představu o zpracovávaných datech, tedy řeči. Aplikaci ASR si můžeme s notnou dávkou abstrakce představit jako dekodér v systému, kde na vstupu i výstupu je text a kde kodér představuje člověk. Ač o způsobu formulace myšlenek bychom mohli vést dlouhé spory, je nasnadě, že náš mozek v konečném důsledku formuluje naše myšlenky do slov. V mozku se následně rozběhnou psycholingvistické procesy, které slovům přiřadí jejich podobu z jednotlivých fonémů a toto fonologické kódování je následně převedeno povely pro zhruba stovku příčně pruhovaných svalů počínaje břišní dutinou, přes hlasivky a konče svaly v dutině ústní, které jsou schopny generovat až 14 fonémů za sekundu [8]. Na přesnost rozpoznání má vliv jakákoliv výrazná odchylka ve verbálních schopnostech uživatele rozpoznávacího systému. Mimo jiné má také podstatný vliv kulturní pozadí mluvčího, které ovlivňuje jeho jazykové schopnosti, dialekt a které je do charakteru řeči mnohem složitěji zakódováno, kupříkladu oproti obrazové informaci [9]. Dále mohou srozumitelnost řeči ovlivnit onemocnění dýchacích cest a nemoci, jejichž příznaky se projevují v oblasti nazálních dutiny. Konečným prvkem, který ovlivňuje kvalitu zpracovávaných dat, je způsob záznamu a převodu řeči do digitální podoby. Jelikož se záznamy tvoří prakticky pouze jako záznam akustického vlnění pomocí mikrofonů, jsou v nahrávkách často přítomny rušivé elementy, souhrnně označovány jako šum.

Tato práce je členěna do následujících kapitol. V druhé kapitole je nastíněno jaké jsou možnosti postupu s cílem rozpoznání řeči v českém jazyce se specifickým slovníkem a je

položen teoretický základ metod použitých v části praktické. Ve třetí kapitole jsou podrobněji představeny jednotlivé kroky implementovaného systému rozpoznávání spontánní řeči. V následující čtvrté kapitole je popsáno konkrétní nastavení použité při vyhodnocení a porovnání výsledků uvedených v kapitole páté. V závěrečné šesté kapitole je práce shrnuta a jsou nastíněna další možná rozšíření. V praktické části jsou získané znalosti aplikovány za použití nástroje Kaldi a nástrojů vzniklých v rámci prací v Laboratoři zpracování řeči při Fakultě Elektrotechnické ČVUT v Praze.

Kapitola 2

Základy rozpoznávání spojité řeči

Rozpoznávání spojité řeči se sestává z několika bloků. V následujících kapitolách jsou postupně jednotlivé bloky popsány v pořadí, v jakém se přirozeně nacházejí ve struktuře takové aplikace, viz. diagram 2.1. Cílem aplikace je získání posloupnosti slov $\hat{\mathbf{w}}$, která má pro daný vstup maximální aposteriorní pravděpodobnost [10]. Tento vztah lze za znalosti posloupnosti příznakových vektorů $\mathbf{c} = c_1, c_2, \ldots c_N$ vyjádřit pomocí vztahu

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, p(\mathbf{w}|\mathbf{c}) \ , \tag{2.1}$$

a jelikož tento nelze přímo aplikovat, pomocí Bayesova pravidla dostáváme

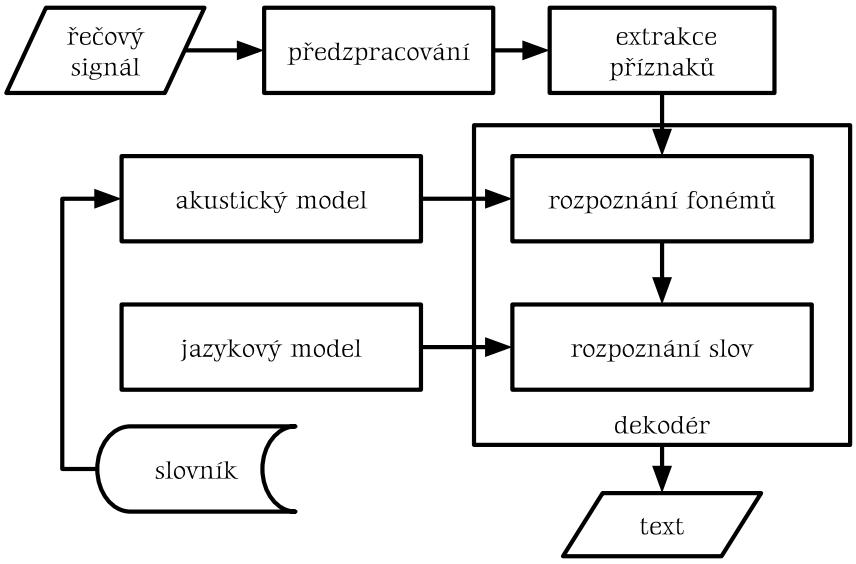
$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{p(\mathbf{c}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{c})} \ , \tag{2.2}$$

kde pravděpodobnost $p(\mathbf{c}|\mathbf{w})$ stanoví akustický model jako pravděpodobnost pozorování daných příznaků pro posloupnost slov $\mathbf{w}$ a pravděpodobnost $p(\mathbf{w})$ je určena jazykovým modelem jako pravděpodobnost výskytu posloupnosti slov $\mathbf{w}$. Jelikož hledání maxima daného výrazu se provádí pro fixní hodnotu posloupnosti příznakových vektorů, je tudíž výraz (2.2) možné zjednodušit na

$$|\hat{\mathbf{w}}| = \underset{\mathbf{w}}{\operatorname{argmax}}\, p(\mathbf{c}|\mathbf{w})p(\mathbf{w}) \ . \tag{2.3}$$

2.1 Extrakce a zpracování příznaků

Lidský hlas přenáší většinu informace v relativně úzkém pásmu slyšitelných frekvencí. Různé zdroje uvádí různé hranice, ale většina uvádí spodní hranici mezi 100 až 200 Hz, horní hranice je pak spornější a v případě reprodukované produkce se více odvíjí od kvality nahrávky, resp. dynamického rozsahu, ale uvádí se mezi 8 až 10 kHz. Hladina akustické

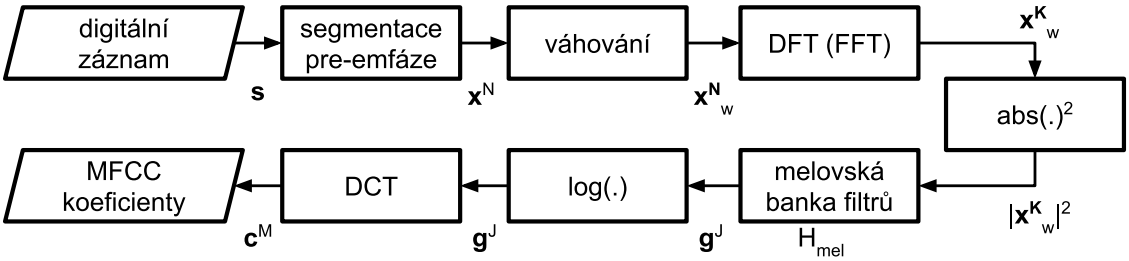


Obrázek 2.1: Bloková struktura ASR systému

energie řeči však počíná kolem 2,5 až 3 kHz výrazně klesat. Ztrátě informace na vyšších frekvencích zabraňuje před extrakcí příznaků aplikace preemfáze, kdy se zvýší energie signálu na vysokých frekvencích. Zároveň se často sníží vzorkovací frekvence, obvykle na hodnotu 16 kHz, což výrazně sníží výpočetní a paměťové nároky. Mezi další možnosti patří redukce šumu a zkreslení. Po prvotním zpracování je signál rozdělen na rámce, s délkou v nižších desítkách milisekund, s určitou délkou kroku, obvykle 10 milisekund, a rámce jsou váhovány, aby se předešlo spektrálnímu prosakování. Délka těchto rámců je navržena tak, aby dávaly dostatečné rozlišení ve frekvenční oblasti ale zároveň nebyly delší než jeden fón, základní stavební jednotka mluvené řeči. Pro každý takový rámec se pak získají příznaky. Tento proces se nazývá parametrizace signálu. Existuje odvětví rozpoznávačů, rychle se rozvíjející, které příznaky neextrahují a rozpoznávací algoritmy pak spouští přímo na surových datech. Tyto systémy se nazývají *End-to-End* a stojí na principu klasifikace vzorů v datech [11]. Tyto systémy se však od námi prezentovaných dosti liší a proto o nich není v této práci již dále pojednáno.

2.1.1 Mel-frekvenční kepstrální koeficienty

Vlastností příznakového vektoru by měl být co nejvíce diskriminativní popis pro jednotlivé hlásky. Během let se experimentovalo s mnohými, z nichž nejpodstatnější jsou mel-frekvenční kepstrální koeficienty [12] které jsou díky jejich výsledkům a přímočarosti implementace nejpoužívanějšími příznaky, mimo jiné jsou součástí základního nastavení rozpoznávačů Kaldi [13] nebo Sphinx [14]. Nejprve se získají spektrální koeficienty pomocí



Obrázek 2.2: Bloková struktura extrakce MFCC koeficientů

diskrétní Fourierovy Transformace (DFT). Váhováním vzorků rámce se zamezí spektrálnímu prosakování při spektrální analýze. Získané amplitudové spektrum je filtrováno pomocí banky filtrů, jejíž střední frekvence jednotlivých pásem odpovídají melovské stupnici a aproximuje tak psychoakustické vlastnosti lidského slyšení. Energie v jednotlivých pásmech jsou následně převedeny logaritmickou mírou, což opět odpovídá vlastnostem lidského ucha, konkrétně vnímání hlasitosti podle Weber-Fechnerova zákona [15]. Kepstrální koeficienty získáme pomocí

$$\mathbf{c}[m] = \sqrt{\frac{2}{J}} \sum_{j=1}^{J} \mathbf{g}[j] \cos\left(\frac{\pi n}{J}(j - 0.5)\right) \ , \tag{2.4}$$

kde $m = 1, 2, \ldots, M$, $j = 1, 2, \ldots J$, $\mathbf{c}[m]$ jsou kepstrální koeficienty a $\mathbf{g}[j]$ představuje logaritmus energie *j-tého* pásma banky filtrů o celkovém počtu filtrů J daný výrazem

$$\mathbf{g}[j] = \log\left(\sum_{k=0}^{N/2} |\mathbf{x}_w[k]|^2 H_{mel,j}[k]\right) \ , \tag{2.5}$$

kde $j = 1, 2 \ldots, J$ a $\mathbf{x}$ je vektor spektrálních koeficientů. Přínos jednotlivých kepstrálních koeficientů klesá, a tak se v praxi pro aplikaci s použitím HMM často používá s uspokojivými výsledky prvních 12, či 13 se započítáním nultého, koeficientů [16]. Pro aplikace vyžadující více dat pro přesnější fungování se jich používá více. Jelikož vektor MFCC koeficientů $\mathbf{g}$ nepopisuje časovou dynamiku ve spektru [17] a díky vlastnosti podmíněné nezávislosti jednotlivých stavů Markovova modelu (sekce 2.2.1) [10] není vztah mezi jednotlivými realizacemi modelován, bývá vektor MFFC koeficientů doplněn regresními koeficienty prvního a druhého řádu označovanými *delta* a *delta-delta* koeficienty. Tyto koeficienty jsou pak dány pro vektor kepstrálních koeficientů $\mathbf{g}$ rámce n a šířku kontextu m jako

$$\Delta\mathbf{g}_n = \frac{\sum_{i=1}^{n} w_i(\mathbf{g}_{n+i} - \mathbf{g}_{n-1})}{2\sum_{i=1}^{n} w_i^2} \ , \tag{2.6}$$

kde w_i jsou váhy a *delta-delta* koeficienty jsou získány identickým postupem jako *delta* koeficienty. Výsledný příznakový vektor C je pak

$$\mathbf{c_n} = [\mathbf{c}_n^T \Delta\mathbf{c}_n^T \Delta^2\mathbf{c}_n^T]^T . \tag{2.7}$$

2.1.2 Zpracování příznakových vektorů

Jedním z aspektů které je nutné brát v potaz je rozličnost ve výšce posazení hlasu a formantových oblastí daných fyziologií vokálního traktu mluvčích. Ač se o procesu normalizace těchto parametrů v rámci lidského vnímání řeči vedou spory [18], přínos pro strojové rozpoznání je nepopiratelný. Většina přístupů vedoucích k normalizaci lze aplikovat třemi způsoby, globálně, adaptací na mluvčího či adaptací na promluvu samotnou, kdy každá jmenovaná varianta specifikuje rámec, ve kterém se vyhodnocují parametry pro normalizaci. Nejčastěji se lze setkat s normalizací střední hodnoty a variance kepstrálních koeficientů (CMVN), která, jak název napovídá, provádí normalizaci střední hodnoty, obvykle na nulu, a variace kepstrálních koeficientů na hodnotu jedna. Pokročilejší technikou je pak normalizace podle délky vokálního traktu (VTLN) [18], jejíž různé varianty spočívají v získání jediného koeficientu z informace o formantových oblastech pomocí něhož se posunou střední frekvence melovské banky filtrů, čímž se docílí nižší variance kepstrálních koeficientů mezi mluvčími.

Jelikož jsou z podstaty jejich vzniku a odvození prvky příznakového vektoru vzájmně závislé lze na ně úspěšně aplikovat metody na dekorelaci a redukci dimenzionality. Tento účel plní při tvorbě MFCC koeficientů krok diskrétní kosinové transformace, viz. obr. 2.2. Avšak informace o temporálním chování signálu lze získat i jinými přístupy. Kontext mezi jednotlivými rámci lze získat přidáním regresivních koeficientů vyšších řádů do rovnice (2.7) nebo řazením statických příznaků za sebe do vektoru

$$\tilde{\mathbf{c}}_n = [\mathbf{c}_{n-m}^T \mathbf{c}_{n-m+1}^T \dots \mathbf{c}_n^T \mathbf{c}_{n+1}^T \dots \mathbf{c}_{n+m}^T]^T , \tag{2.8}$$

kde m je opět hloubka kontextu. Problém nárůstu dimenzionality lze řešit následnou aplikací redukce dimenzionality formálně definovanou pomocí lineární transformace $\mathbf{c}_n^* = \mathbf{A}\tilde{\mathbf{c}}_n$ [10], kde $\mathbf{A}$ představuje transformační matici. Volba transformace záleží mimo jiné na tom, zda chceme zároveň optimalizovat diskriminaci mezi třídami danými akustickým modelem či nikoliv. Analýza hlavních komponent (PCA) nebere v potaz separabilitu mezi třídami, a proto se lze častěji setkat s lineární diskriminativní analýzou (LDA) nebo její variantou využívající heteroskedasticity kovariančních matic jednotlivých tříd.

2.2 Akustický model

Jak jsme si uvedli v úvodu kapitoly, akustický model ve výrazu (2.3) představuje člen $p(\mathbf{c}|\mathbf{w})$. Akustický model se tedy snaží najít sekvenci fonémů, základních lingvistických jednotek schopných rozlišit od sebe jednotlivá slova v daném jazyce, které maximalizují aposteriorní pravděpodobnost pozorování posloupnosti příznakových vektorů. Pozorovaná produkce akustické reprezentace fonému je přímým důsledkem stavu vokálního traktu který nemůžeme přímo stanovit a s dávkou abstrakce ho můžeme považovat za diskrétní. Pak můžeme takový systém definovat jako systém se skrytými stavy generující v čase jistá pozorování.

2.2.1 Rozpoznávání na bázi skrytých Markovových modelů

Standardním principem modelování výslovnostní stránky rozpoznávače jsou skryté Markovovy modely, vycházející principiálně z homogenních Markovových řetězců. Markovův řetězec je definován jako diskrétní náhodný proces, posloupnost diskrétních stavů $\mathbf{q}$, z množiny všech možných posloupností $\mathbb{Q}$. Stavy jsou definovány jako s_n z diskrétní konečné množiny stavů $\mathbb{S}$ o mohutnosti I, tedy

$$s_n \in \{s_i; \quad i = 1, 2, \ldots, I\} \; . \tag{2.9}$$

Základním vlastností Markovových řetězců je *Markovův předpoklad*, že pravděpodobnost přechodů mezi stavy s_i a s_j, a_{ij}, je dána pouze předchozím stavem, čímž dostáváme vztah pro Markovův řetězec prvního řádu

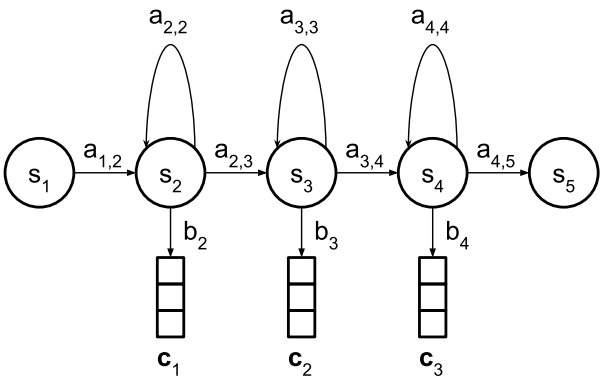
$$p(q_n = s_j | q_{n-1} = s_i) \doteq a_{i,j} \tag{2.10}$$

a pro pravděpodobnost výskytu posloupnosti proměnných $p(q_1, q_2, \ldots, q_N)$ o délce N dostáváme výraz

$$p(q_1, q_2, \ldots, q_N) = p(q_1) \prod_{n=2}^{N} p(q_n | q_{n-1}) \; . \tag{2.11}$$

Skrytý Markovův model pak dostáváme tak, že stavy charakterizujeme jako náhodné proměnné, tedy že pozorované příznaky jsou realizací pravděpodobnostní funkce stavu. Stavy charakterizujeme pravděpodobností pozorování stavu $b_n = p(q_n|c_i) \quad i = 1, 2, \ldots, I$ a použitím (2.3) a (2.11) dostáváme sekvenci $\mathbf{q}$ s nejvyšší pravděpodobností realizace za pozorování sekvence příznaků $\mathbf{c}$

$$\mathbf{q} = \underset{\mathbb{Q}}{\operatorname{argmax}}\, p(\mathbf{q}, \mathbf{c}) = \underset{\mathbb{Q}}{\operatorname{argmax}} \prod_{i=1}^{I} p(q_i|q_{i-1}) p(c_i|q_i) \; , \tag{2.12}$$



Obrázek 2.3: Třístavový skrytý Markovův model

kde b_n jsou pravděpodobnostní rozdělení pozorování stavu $q_n = s_n$. Pomocí Markovových řetězců se modelují jednotlivé fonémy a identifikace fonému je realizována jako pravděpodobnost průchodu daným řetězcem za pozorování sekvence příznakových vektorů $\mathbf{c}$, což vyžaduje úpravu modelování pozorování představenou v následující kapitole. Na obrázku 2.3 můžeme vidět třístavový skrytý Markovův model. Stavy s_1 a s_5 mají pouze pravděpodobnosti přechodu do následujícího stavu protože slouží k napojování jednotlivých řetězců za účelem tvorby slov, resp. jejich mluvené formy, a z téhož důvodu jim není přiřazena příslušnost k pozorování, nazýváme je "*neemitující stavy*". K získání parametrů a a b se nejčastěji používá Baum-Welchova algoritmu [19], speciálního případu algoritmu maximální věrohodnosti (EM), a pro následné určení pravděpodobnosti průchodu daným modelem a nalezení optimální sekvence na základě předložených pozorování Viterbiho algoritmu [10].

2.2.2 Reprezentace modely Gaussovských směsí

Jelikož jsou fonémy modelovány příznakovým vektorem, volí se jako jejich reprezentace vícerozměrné normální rozdělení $b_n(\mathbf{c}_n) = \mathcal{N}(\mathbf{c}; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ kde $\boldsymbol{\mu}$ je vektor středního hodnot a $\boldsymbol{\Sigma}$ je kovarianční matice, často omezena na diagonální. Pro přiřazení do skupiny, daného fonému, slouží modely Gaussovských směsí (GMM) [20]

$$p(\mathbf{s_i}|\mathbf{c}) = \sum_{i=1}^{N} w_i \mathcal{N}(s_i, \mu_i, \Sigma_i) \qquad \text{a platí } \sum_{i=1}^{N} w_i = 1 \,. \tag{2.13}$$

Tuto architekturu nazýváme GMM-HMM, v některé literatuře se lze setkat z termínem *Continuous Density HMM* (CDHMM). Základní implementace GMM-HMM operuje s fonémy bez temporálního kontextu, tzv. monofóny. Jejich HMM řetězec může být jednostavový a jeho realizace modelována jedinou Gaussovskou distribucí. Avšak z realizace lidské řeči je patrné, že procesy ve vokálním traktu generující jednotlivé fonémy se na-

vzájem ovlivňují, a tak jeden foném může mít hned několik *fyzických* realizací, hlásek, v kontextu. K tomuto účelu slouží shlukování pomocí *rozhodovacích stromů*. Pro každý monofón se vytvoří v závislosti na referenci trénovacích dat či uměle množina trifónů, kombinací tří monofónů. Každá množina se následně reprezentuje pomocí binárního stromu, kde probíhá dělení do podskupin na základě heuristické logiky. Z výsledných skupin na konečných uzlech stromu se vytvoří trifony, pro které jsou přepočteny četnosti z monofónů a zároveň jsou obdobným postupem jako u monofónů získány parametry skrytých Markovových modelů, které jsou v tomto případě již třístavové.

2.2.3 Adaptace modelu

Pokud máme k dipozici vyhodnocený akustický model včetně přepisu a zarovnání (viz. sekce 2.5.1), je možné využít afinních lineárních transformací na adaptaci parametrů akustického modelu pro daného mluvčího. Tyto transformace se od obecné definice afinní transformace $\mathbf{x} \rightarrow \mathbf{A}\mathbf{x} + \mathbf{b}$ liší tím, že k vektoru příznaků se před transformací uměle přidává prvek o hodnotě 1. Metoda využívající pro získání transformační matice algoritmus maximální věrohodnosti (MLLR/MLLT) je jednou z používaných pro svoji univerzálnost a možnost přípravy z jiných dat, než jsou data trénovací. Princip spočívá v aplikaci jedné či více transformačních matic, pomocí kterých se normalizují střední hodnoty a variance Gaussovských směsí, v jedné variantě odlišnými maticemi a v druhé maticí stejnou pro jednotlivé třídu stavů. Značného snížení výpočetní náročnosti lze dosáhnout využitím pouze diagonálních transformačních matic. Existuje i její varianta, kde se transformace provádí přímo na vstupních příznacích (fMLLR).

2.2.4 Rozpoznávání na bázi hlubokých neuronových sítí

Jedním ze základních nedostatků akustického modelování čistě pomocí GMM-HMM je omezená schopnost efektivně modelovat nelinearity ve vícedimenzionálních datech [21]. Právě tak by se dala data v aplikaci ASR definovat, jelikož zpracovávané příznaky vznikají modulací malého počtu parametrů v jinak dynamickém a nelineárním systému [22] kterým vokální trakt je. Tento problém částečně řeší metody popsané v sekcích 2.1.2 a 2.2.3, avšak i zde dochází zpravidla pouze k lineární transformaci dat. V posledních letech se proto díky dostupnosti stále vyššího výpočetního výkonu uplatňují implementace využívající učení hlubokých neuronových sítí (DNN), ať už jako pouze jako krok k získání akustického modelu, takzvané DNN-HMM systémy, či jako kompletní nástroj na řešení daného úkolu, tzv. *End-to-End* systémy mající za vstup neuronové sítě spektrogram a za výstup finální přepis. Podle způsobu aplikace a účelu v rámci zpracování mluveného slova

existují několikeré architektury neuronových sítí.

2.2.5 Architektura neuronové sítě

Pojem hluboká neuronová síť staví na perceptronu, algoritmu, který provádí diskrétní klasifikaci na základě vícedimenzionálních vstupních dat, s důrazem na fakt, že pravidlo pro rozhodnutí není heuristické nýbrž dané pomocí parametrů sítě daných trénovacím algoritmem. Kromě klasifikace existují další úlohy DNN, pro aplikaci rozpoznání řeči je však zásadní klasifikace, a dále pak získání příznaků z dat a predikce [23]. Úlohou klasifikace je určení aposteriorní pravděpodobnosti pozorování dané třídy na základě vstupních dat. Třídy musí být specifikovány předem a jejich počet musí běžně souhlasit s počtem uzlů ve výstupní vrstvě. Základní architektura sítě se sestává ze vstupní, nulté, vrstvy, množství skrytých vrstev a vrstvy výstupní. Skryté vrstvy $l = 1, 2, \ldots, L$ se skládají z N_l uzlů, jejichž vstupy, souhrnně excitační vektor $\mathbf{z}$

$$\mathbf{z_l} = \mathbf{W_l}\mathbf{v}_{l-1} + \mathbf{b_l} \ , \tag{2.14}$$

jsou váhované součty výstupů v všech uzlů ve vrstvě předchozí, značené jako aktivační vektor $\mathbf{v}$. Vrstva v DNN s těmito vstupními vazbami se nazývá *afinní*. Pokud by byly všechny členy b nulové, jednalo by se o lineární vrstvu. $\mathbf{W}$ je matice vah a $\mathbf{b_l}$ je vektor odchylek. Aplikací aktivační funkce f dostaneme pro vrstvu l vektor výstupních hodnot uzlů, aktivační vektor, o délce dané počtem uzlů

$$\mathbf{v}_l = f(z_l) = f(\mathbf{W}_l v_{l-1} + b_l) \qquad pro \quad 0 < l < L \ . \tag{2.15}$$

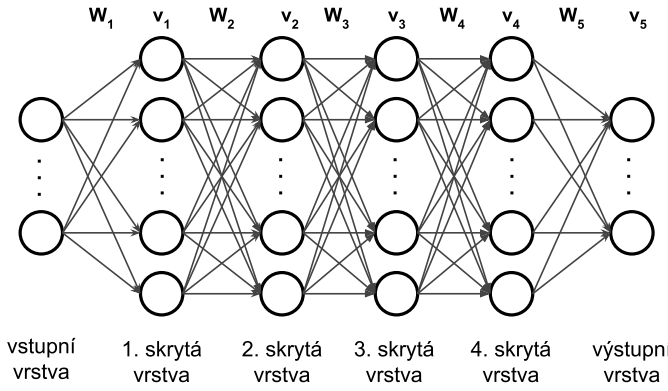
Na výstupu každého uzlu je pak aktivační funkce vstupu daného uzlu. Aktivačních funkcí je řada, ale nejběžněji se lze setkat s logistickou funkcí

$$\sigma(z) = \frac{1}{1 + e^{-z}} \ , \tag{2.16}$$

či její symetrickou verzí

$$\tanh z = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \ . \tag{2.17}$$

Další populární aktivační funkce používaná v kontextu zpracování řeči je funkce *Rectified Linear Unit* (ReLU), $f(z) = \max(0, z)$ a její mnohé varianty, především pak *maxout*, $f(z) = \max_i z_i$, kde i značí index skupiny výstupů z předchozí vrstvy [24]. Pro účely regrese je výraz (2.15) dostačující, ale pro účely klasifikace tříd, kdy jednotlivé výstupní



Obrázek 2.4: Ukázka struktury neuronové sítě se 4 skrytými vrstvami

uzly představují aposteriorní pravděpodobnosti tříd, nesplňuje podmínku

$$\sum_{i=1}^{N_L} v_L[i] = 1 \ , \tag{2.18}$$

a proto se musí excitační vektor výstupní vrstvy normovat softmax funkcí

$$v_L[i] = \frac{e^{z_L[i]}}{\sum_{j=1}^{N_L} e^{z_L[j]}} \ . \tag{2.19}$$

2.2.6 Trénování

Koeficienty vah $\mathbf{W_l}$ a odchylky $\mathbf{b_l}$ jednotlivých vrstev l nejsou dopředu známy, souhrnně se nazývají *latentní parametry*. Ty jsou postupně zpřesněny pomocí algoritmu zpětné propagace získaného derivací účelové funkce podle koeficientů vah, která je mírou rozdílu mezi požadovaným výstupem a výstupem získaným z aktuálních hodnot latentních parametrů [21]. Při použití funkce (2.19) pro klasifikaci vyplyne jako účelová funkce $\mathcal{J}$ křížová entropie mezi požadovaným výstupem $d[j]$ a získanou hodnoutou na výstupu uzlu $v_L[j]$

$$\mathcal{J}_{CE} = -\sum_{j}^{N_L} d[j] \log \, v_L[j] \ . \tag{2.20}$$

Jestliže však při trénování víme, že pravděpodobnost správné třídy je rovna jedné a ostatní jsou pak nulové, dostaneme zjednodušenou formu $\mathcal{J}_{CE} = -\log \, v_L[c]$ kde c je index pro správnou třídu. Následné úpravy koeficientů se dopočítávají rekurzivně pomocí algoritmu

gradientního sestupu (SGD) [25]

$$\theta = \theta - \eta \cdot \bigtriangledown_\theta C(\theta) \ , \tag{2.21}$$

kde θ představuje optimalizovaný parametr, η koeficient rychlosti učení a $\bigtriangledown_\theta \mathcal{J}_{CE}(\theta)$ je gradient účelové funkce $\mathcal{J}_{CE}$ podle daného hledaného parametru. Dále existuje kromě modifikovaných SGD řada dalších, jako je například Broyden–Fletcher–Goldfarb–Shanno (BFGS) algoritmus nebo algoritmus adaptivního odhadu momentů [23]. Při trénování je nutné brát ohledy na paměťové nároky. Pro velké trénovací množiny není vhodné provádět trénování na množině celé a proto se trénuje po várkách. Dále se trénování dělí do epoch, kdy jedna epocha zahrnuje cyklus, při kterém se neuronová síť přetrénuje na všech dostupných datech právě jednou.

Pokud se bavíme o DNN-HMM systémech s učitelem, systémech, jejichž výstupem jsou hodnoty pravděpodobností stavů *n-gramů* přiřazené jednotlivým rámcům vstupního dat, je k trénování zapotřebí mít dopředu označená data, v případě akustického modelu to znamená nutnost znalosti sekvence fonémů. Získání těchto dat je v základu možné pomocí natrénování a dekódování trénovací množiny pomocí GMM-HMM dekodéru a následnému použití získaných zarovnaných referencí pro trénování DNN. Existují však i další metody, mezi něž patří částečné učení s učitelem, generativní trénování, nebo CTC [26].

Jelikož je trénování parametrů sítě rekurzivní, je nutné nastavit počáteční hodnoty parametrů tak, aby se zabránilo divergenci a nestabilitě modelu [20]. Podmínkou pro rychlé a efektivní učení je, aby se každý uzel pohyboval v *lineární oblasti* své charakteristiky a tím maximalizovat hodnotu gradientu v nadcházející iteraci. Dalším základním předpokladem je, aby hodnoty vah pro inicializaci byly náhodné, taženy z vhodně zvoleného pravděpodobnostního rozdělení. Pokud by tomu tak nebylo, váhy by se během trénování měnily v dané vrstvě stejně a síť by nebyla schopna rozlišit jednotlivé vzorce v datech, jelikož jednotlivé uzly mají v tu chvíli souhlasné vstupy a váhy.

Jelikož úloha je inherentně o dekódování sekvence dat, bylo by vhodné používat na trénování účelové funkce, které tento fakt odrážejí. Metoda maximální vzájemné informace rozšiřuje kontext křížové entropie z rámce na sekvenci pozorovaných rámců a hledá tak sekvenci slov, resp. fonémů, která maximalizuje účelovou funkci [27]

$$\mathcal{J}_{MMI} = \sum_u \log \frac{p(\mathbf{c}_u|\mathbf{s}_u)^\kappa p(\mathcal{S}_u)}{\sum_\mathcal{S} p(\mathbf{c}_u|\mathbf{s})^\kappa p(\mathcal{S})} \ , \tag{2.22}$$

kde $\mathbf{c}_u = \{c_{u1}, c_{u2}, \ldots, c_{uN}\}$ je pozorování příznaků a $\mathcal{S}_u$ je sekvence slov příslušící sekvenci stavů HMM $\mathbf{s}_u$. I tato metoda vyžaduje v učení vstupní data zarovnaná k příslušným stavům na úrovni slov. Omezení která klade tento požadavek ruší její následná

modifikace navržená v [28]. **Lattice-free Maximum Mutual Information** (LF-MMI) je metoda trénování vah neuronové sítě založená na algoritmu maximalizace vzájemné informace (MMI) s výrazými modifikacemi, které jí umožňují trénování na datech zarovnaných na úrovni jednotlivých fonémů. Dále je dosaženo zrychlení trénování tím, že se hodnota účelová funkce vypočítává při snížené frekvenci vstupních dat. Z toho důvodu je nutná úprava stavového HMM modelu, který nyní umožňuje projití celého modelu během jediného rámce, což v základní třístavové implementaci zabere minimálně tři rámce. Jelikož je kontext v sousedních rámcích nezanedbatelný, v kombinaci se snížením vzorkovací frekvence vstupních příznaků vede k méně výpočetně náročnému ale přesto efektivnímu způsobu učení.

2.2.7 Předtrénování

Kromě inicializace náhodnými hodnotami existují sofistikovanější metody k získání vhodných výchozích hodnot parametrů sítě. Pokud bychom váhy **W** nastavili tak, aby skryté vrstvy již prováděli částečnou diskriminaci, mohli bychom takto urychlit a zpřesnit trénování. Nejjednodušší je, pakliže máme k dispozici natrénovaný model, vzít váhy onoho modelu jako inicializační a vzniklou síť následně **přetrénovat** na nových trénovacích datech. Pokud tyto váhy nemáme, můžeme využít generativních metod trénování. Jejich princip spočívá v iterativním trénování od pouhé jedné vrstvy s přidáním další v každé iteraci trénování. *Deep Belief Network* je generativní síť, získaná vrstvením *Restricted Boltzman Machine* vrstev [29], [30], kdy jednotlivé vrstvy jsou sekvenčně od nulté vstupní vrstvy separovaně trénovány pomocí hladového algoritmu na základě výstupu vrstvy předchozí tak, aby se kumulativní distribuční funkce pro všechny kombinace výstupních proměnných dané vrstvy rovnaly jedné za použití definice pro složenou pravděpodobnostní funkci pomocí energie [31]

$$P(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{Z} , \tag{2.23}$$

kde $\mathbf{v}$ jsou hodnoty na výstupu dané vrstvy a Z je funkce zajišťující rovnost $\sum_{\mathbf{v}} P(\mathbf{v}) = 1$.

2.2.8 Vstupní data DNN

Dalším způsobem, jak zlehčit podmínky trénování je normalizace vstupních dat. Pokud by totiž některý ze vstupních příznaků měl výrazně větší rozptyl, síť by se naučila klasifikovat především z tohoto jednoho příznaku.

Často zmiňovaným problémem je *overfitting*. Ve zkratce jde o to, že trénovací data nemůžou z podstaty věci obsáhnout všechny charakteristiky daného jazyka. Jelikož se snažíme minimalizovat ztrátovou funkci, viz. rovnice (2.21), optimalizujeme latentní parametry pro trénovací data. Pokud bychom nechali síť trénovat příliš dlouho, model by

dokázal přesně predikovat trénovací data. Formálně je tedy *overfitting* definován jako rozdíl mezi chybami vyhodnocení na trénovací a testovací množině [31]. Prvním možným řešením je získat co nejvíce trénovacích dat, ideálně obsahově pestrých. To však není vždy možné. Dalším řešením zabraňujícím *overfittingu* je správné nastavení dimenzí DNN. Jelikož každá z vrstev může být chápána jako extraktor vlastností dat, musí být dimenze vrstev dostatečné pro kompletní popis těchto vlastností, na druhou stranu příliš široká síť bude mít větší sklon k *overfittingu* [20]. Zbylé nástroje přichází na řadu během trénování samotného.

2.2.9 Regularizace

Formálně může být regularizace definována jako „modifikace trénovací fáze s cílem snížení obecné chybovosti ale nikoliv chybovosti v trénovací fázi“ [31] a funguje na principu snížení variance parametrů za cenu zvýšení odchylky. Regularizace Ω se realizuje přidáním penalizace přes normu parametrů k účelové funkci $\mathcal{J}$

$$\tilde{\mathcal{J}}(\theta; \mathbf{c}; \mathbf{y}) = \mathcal{J}(\theta; \mathbf{c}; \mathbf{y}) + \alpha\Omega(\theta) \ , \tag{2.24}$$

kde $\mathbf{y}$ představuje označení trénovacích dat, α je hyperparametr, tedy parametr, který se nastavuje ručně před začátkem trénování a v průběhu trénování se nemění, který určuje míru penalizace. V praxi se penalizují především a pouze váhy, regularizace odchylek nenese pozitivní přínos a také se pro různé vrstvy volí různé hodnoty α.

Mezi používané normy regularizace Ω patří L^2 norma, kde

$$\Omega(\mathbf{w}) = ||\mathbf{w}||_2 = \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} \,, \tag{2.25}$$

nazývaná *weight decay* omezující aktualizace komponent matice vah, které nepřispívají významně k minimalizaci účelové funkce [31]. Její přísnější variantou je pak L^1 norma, kde

$$\Omega(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_i |w_i| \,, \tag{2.26}$$

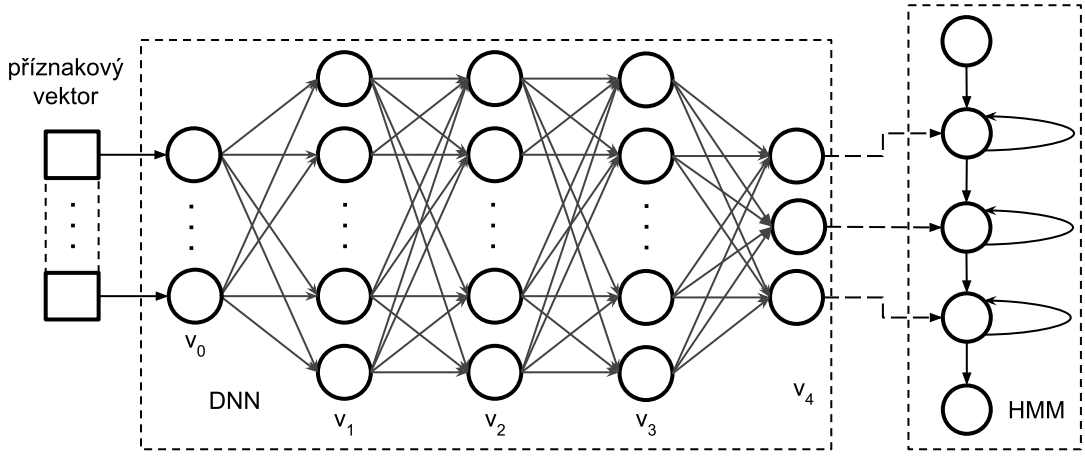
která výrazněji přispívá k řídkosti matic vah tím, že některé jejich členy pokladá rovny nule a provádí tzv. *výběr příznaků* [31]. *Dropout* metoda je přístup, kdy se *overfittingu* předchází tím, že se při trénování na základě jistých pravidel během trénování maskují výstupy vybraných uzlů nulou, a tím je pro danou iteraci efektivně vyřazuje z trénování. Tato metoda zabraňuje koadaptaci neuronů a nutí je vytvořit nezávisle na sobě robustní reprezentaci dat [31].

2.2.10 Adaptace DNN modelů

Z důvodu množství parametrů neuronových sítí a neznalosti vnitřní struktury, známý pojem „černé skříňky“, u nich nelze aplikovat principiálně stejné adaptační mechanismy jako u GMM-HMM systémů. Avšak existují i jiné metody augmentace než je transformace vnitřních parametrů. Ač takové pokusy byly, výsledky přinesly spíše přístupy transformující pouze odchylky vah nebo transformující vstupní data. Nejlépe nasaditelné a interpretovalené jsou však metody rozšiřující vektor vstupních dat o příznaky reprezentující mluvčího nebo prostředí [32]. Mezi takové metody patří tzv. *i-vektory*, *identifikačních vektory*, a *x-vektory*, revidovaná implementace *i-vektorů*, které představují vektorové reprezentace mluvčího a prostředí s konstatní délkou a získají se diskriminativním trénováním. Jejich předností je jejich naprostá nezávislost na implementaci akustického modelu. Princip *i-vektorů*, spočívá ve vytvoření univerzálního modelu řečníka (UBM) nezávislého na mluvčím a modelů mluvčích pomocí GMM, kdy *i-vektory* vyjadřují linární vztah mezi UBM a modely jednotlivých mluvčích, resp. jejich středními hodnotami.

2.3 DNN-HMM hybridní systém

Princip DNN-HMM systému spočívá v realizaci akustického modelu neuronovou sítí, která má na výstupních uzlech posteriorní pravděpodobnosti jednotlivých fonémů, zatímco dynamika řeči se modeluje pomocí HMM modelu. Grafické znázornění architektury takového systému je na obrázku 2.5. Na možnosti trénování se vztahují metody zmíněné



Obrázek 2.5: Architektura DNN-HMM modelu

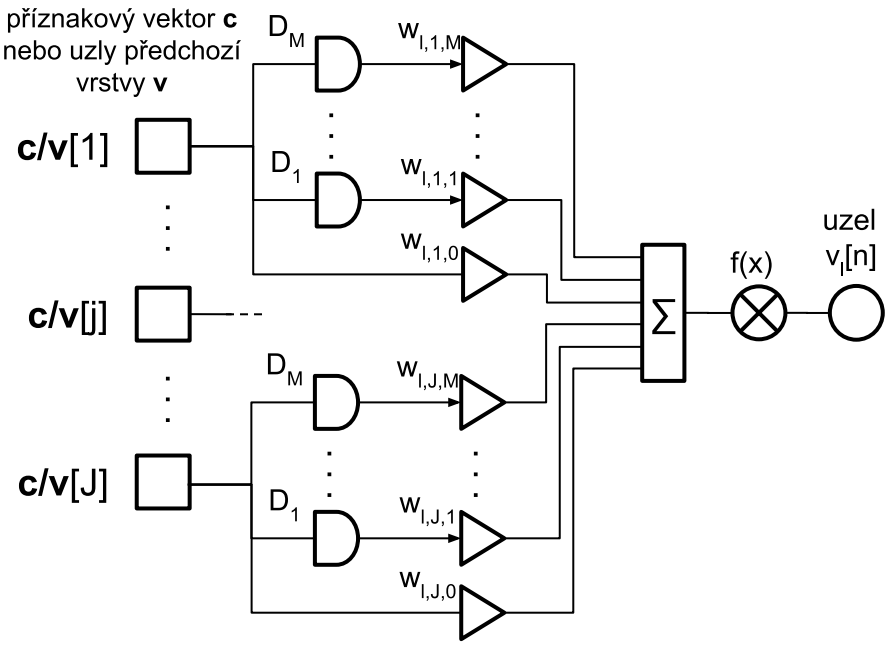
v sekci 2.2.6, avšak nejčastěji se lze setkat s získáním trénovacích referencí pomocí zarovnaných dat z natrénovaného GMM-HMM systému. Existují však i metody, které trénování pomocí GMM-HMM vypouští úplně a vychází z rovnoměrně zarovnaných dat, jejichž zarovnání se iterativně zpřesňuje pomocí DNN-HMM systému [33] .

2.3.1 TDNN architektura

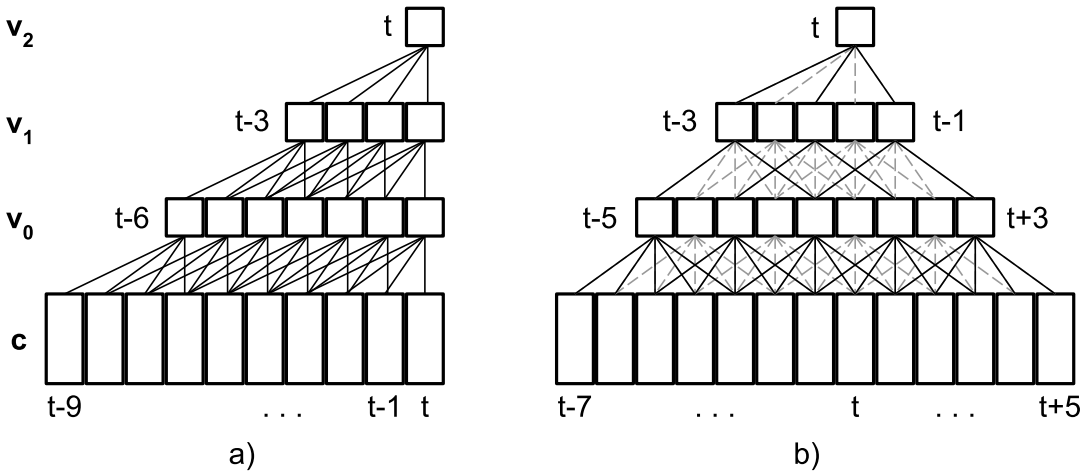
Specifickým požadavkem, kladeným na akustický model v případě rozpoznání řeči je bezpochyby temporální kontext. V předchozích kapitole 2.1.2 byla popsána metodu, která kontextu dosahovala přidáním dynamických příznaků. Dopředná architektura *Time Delay Neural Network*, neuronových sítí se zpožďovacími linkami, dosahuje porušení předpokladu nezávislosti jednotlivých pozorování bez dynamických příznaků. Oproti běžné struktuře představené, v sekci 2.2.5 jsou všechny excitační vektory rozšířené o jejich zpožděné reprezentace [34], což dává síti možnost kontextu v čase. V závislosti na aplikaci, tedy jde-li o rozpoznání v reálném čase nebo nikoliv, je možné použití dopředných vazeb a tudíž oboustranného kontextu, viz. obrázek 2.7. Jelikož lze očekávat vysokou korelaci mezi sousedními rámci, efektivní úpravou této architektury je podvzorkování, kdy se některé časové kroky vypustí a tím je dosaženo výrazného snížení výpočetní náročnosti bez ztráty získané informace temporálního kontextu. Na trénování těchto sítí lze aplikovat postupy uvedené v sekci 2.2.6 [35].

2.3.2 Faktorizovaná TDNN síť

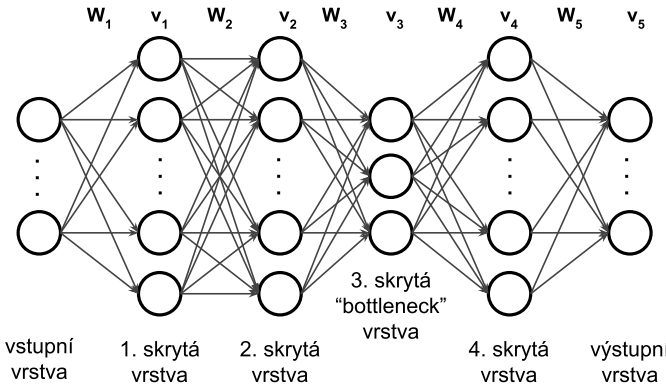
Faktorizací se v kontextu neuronových sítí myslí postup, kterým se dosáhne snížení počtu parametrů sítě a zároveň se matice parametrů reprezentují pomocí dvou faktorů, matic, s nižší dimenzionalitou. Tento postup stojí na teorii rozkladu na singulární hod-



Obrázek 2.6: Blok vstupního uzlu TDNN sítě



Obrázek 2.7: Architektura TDNN sítě: *a)* TDNN se zpětným kontextem *b)* TDNN s oboustranným kontextem - s podvzorkováním (plné) a bez podvzorkování (plné a čárkované)



Obrázek 2.8: Ukázka struktury neuronové sítě se 4 skrytými vrstvami, z nichž třetí je *bottleneck* vrstva

noty (SVD) a pomocí dosažení semi-ortogonality jedné z matic lze dojít k modelu, jehož výsledky přesahují výsledky jiných sítí s dvojnásobným počtem parametrů [36]. Při implementaci je tato faktorizace provedena pomocí vložení takzvané *bottleneck* vrstvy mezi vrstvy, na jejichž matici vah se faktorizace provádí. Hlavní charakteristikou *bottleneck* vrstvy je výrazně menší počet uzlů oproti ostatním vrstvám.

2.4 Jazykový model

Jazykové modely jsou základním stavebním blokem veškerých systémů na zpracování přirozeného jazyka, dále jen NLP. Jedná se o stochastické modely mnohdy využívající strojového učení s cílem určit pravděpodobnost slov či posloupností slov na základě slova, posloupnosti slov či jiných dat předložených modelu. Základní vztah [37], ze kterého vychází nejjednodušší modely vyjadřuje pravděpodobnost sekvence slov a větních elementů $w_1, w_2, \ldots, w_n$ o délce n jako

$$p(w_1, w_2, \ldots, w_n) = p(w_1) \times p(w_2|w_1) \times \ldots \times p(w_n|w_1, w_2, \ldots, w_{n-1}) \; . \tag{2.27}$$

Pravděpodobnosti jednotlivých sekvencí získáme iterativně pomocí poměru četnosti výskytu dané sekvence k četnosti výskytu sekvence o jedno slovo či element kratší, tedy

$$p(w_n|w_1, w_2, \ldots, w_{n-1}) = \frac{C(w_1, w_2, \ldots, w_n)}{C(w_1, w_2, \ldots, w_{n-1})} , \tag{2.28}$$

kde $C(.)$ představuje absolutní četnost. Při maximálním zobecnění bychom mohli tyto modely rozdělit do dvou hlavních kategorií, konvenční jazykové modely, dále jen CLM a předtrénované jazykové modely, dále jen PLM [38]. Druhá zmiňovaná kategorie je charakterizována nutností velkého množství trénovacích dat a využití strojového učení. Ač se tento přístup v dnešní době těší hojné popularitě a u známých projektů (GPT od OpenAI, Claude od Anthropic nebo LLaMa od Meta) je použit ve valné většině, není v praktické části této práce věnující se jazykovým modelům zohledněn a dále se jí tedy nezabýváme.

Druhá kategorie, tedy CLM, zahrnuje tradiční modely, které předcházely dnešním LLM a jež podléhaly základnímu omezení z dob svých počátků, nízkému výpočetnímu výkonu, paměťovým omezením ale především pak nedostupnosti tak velkého množství trénovacích dat.

Základním předpokladem rozumného chování takového modelu jsou vhodná trénovací data. Ta by měla obsahově reprezentovat sledovaný účel. Tyto modely využívají statistických metod a modelování pomocí n-gramů. Nejzákladnější z těchto jsou dopředné n-gramové modely. Jejich princip je založen na stanovení pravděpodobnosti posloupnosti na základě relativní četnosti v předloženém textovém korpusu. Například trigramový model bere jako kontext pouze dvě předchozí slova a rovnice (2.28) má pak tvar

$$p(w_n|w_{n-2}, w_{n-1}) = \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})} . \tag{2.29}$$

2.4.1 N-gramové modely

Základní n-gramové modely jsou reprezentovány pomocí HMM a vycházejí principiálně z výrazu (2.27). Díky Markovově předpokladu (2.11) zobecněného na více stavů omezují podmíněnou pravděpodobnost výskytu stavu na $N - 1$ stavů předchozích. Základním algoritmem pro výpočet této podmíněné pravděpodobnosti je metoda maximální věrohodnosti, MLE. Jelikož se pro větší korpusy a širší n-gramy pravděpodobnosti pro méně časté sekvence slov blíží nule a hrozí tak při výpočtech s takovými hodnotami podtečení, v praxi se počítá s logaritmickými hodnotami pravděpodobností, což mimo jiné přináší i výhodu nahrazení násobení pravděpodobností součtem.

2.4.2 Hodnocení jazykových modelů

Vyhodnocování reprezentace trénovacích dat jazykovým modelem může probíhat dvěma způsoby, vnějším vyhodnocením a vnitřním. Ke vnějšímu hodnocení je zapotřebí vyhodnotit danou aplikaci, ve které se porovnávané modely používají a v obou scénářích měnit pouze parametry jazykového modelu. Ač je tento způsob jednodušší na interpretaci, postrádá přímé kvantifikovatelné porovnání daných jazykových modelů. Proto se při vyhodnocení jazykových modelů častěji přistupuje k vnitřnímu vyhodnocení. Základním měřítkem přesnosti modelu je **perplexita**. Obecně je perplexita definována jako míra nejistoty při predikci. Mějme natrénovaný model a testovací sekvenci $W = w_1, w_2, \ldots, w_N$ kde w_i jsou jednotlivé slovní elementy tažené z testovací množiny. Perplexita modelu je pak dána výrazem

$$PP(W) = \sqrt[N]{\frac{1}{P(W)}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}\,, \tag{2.30}$$

jinými slovy to je inverzní hodnota pravděpodobnosti. Jelikož je perplexita silně svázána se slovníkem trénovacích dat, je možné na základě této veličiny srovnávat pouze modely trénované na datech se stejným slovníkem [11].

2.4.3 Tokenizace

Doposud jsme jako aproximovaný stav brali v potaz jen slova. Často však vyvstává problém v případě, že narazíme sekvenci fonémů, která ve slovníku žádnému slovu neodpovídá. Mohli bychom vždy takové slovo do modelu přidat, avšak narazíme na problém, že nám bude chybět kontext a zároveň vyvstane otázka, jakou pravděpodobnost takovému slovu přiřadit. Zaveďme si proto termín tokenizace, což je proces, kdy se sekvenci po sobě jdoucích elementům, v našem případě příznaků, přiřadí tzv. token. V aplikaci samotné se pro reprezentaci tokenů a práci s nimi volí jednoduše celá čísla která identifikují v dříve zmiňovaném případě slova. Jazykové modely však mohou pro tokeny místo slov použít i jiné elementy. Nejmenším takovým elementem jsou pak psaná písmena, *grafémy*, což přináší výhodu značně menšího slovníku, ale na druhou stranu pro efektivní aplikaci vyžadují mnohem hlubší kontext, než mají trigramové modely [39].

Dobrou alternativu za cenu složitějšího modelování nabízí subslovní modelování. Tento přístup využívá kromě běžného slovníku se slovy druhý slovník, který obsahuje nejčastěji se vyskytující subslovní elementy získané z slovníku se slovy. Existují dva přístupy generování subslovních elementů, statistický a lingvistický. První jmenovaný nejdříve rozdělí všechna slova ve slovníku na jednotlivé grafémy a následně je pomocí jistého algoritmu skládá iterativně dohromady tak dlouho, dokud se nedosáhne předem stanovené velikosti slovníku subslovních elementů [40]. Druhý pak získává subslovní elementy pomocí pravi-

del daného jazyka [41], což je zvlášť vhodné pro jazyky, které uplatňují skloňování. Dále také existují tokenizační přístupy na úrovni frází, tedy běžných sekvencí slov.

Pokud se bavíme o konvenčním jazykových modelech, jedná se většinou o dopředné n-gramové modelování (FLM), tedy že pravděpodobnost výskytu tokenu predikujeme pouze na základě pravěpodobnosti výskytu předcházejících tokenů. Existují však i další postupy, zejména bidirekcionální či obousměrné modely, strukturální modely a permutační modely [38]. Obousměrné modely, také nazývané maskované, hledají pravděpodobnost výskytu daného tokenu ze znalosti oboustranného kontextu. Strukturální modely rozdělují slova na základě znalosti slovních druhů a heuristickým algoritmem staví hierarchii vět podle pravidel větné stavby daného jazyka. Pravděpodobnost hledaného slova je pak dána jeho předky v hierarchickém stromu. Permutační pracuje podobně jako konvenční dopředný model, s tím rozdílem, že hodnotí pravděpodobnost na základě množiny všech permutací sekvence přechozích slov.

2.4.4 Optimalizace

Základním problémem jsou slova mimo slovník jazykového modelu. Modely využívající tokenizace subslovních elementů mohou při dostatečné velikosti slovníku tento problém eliminovat a stávají se tak systémem s *uzavřeným slovníkem*, jinými slovy jejich slovník neobsahuje token pro neznámé slovo [11]. Opačným přístupem je slovník otevřený, který obsahuje symbol, token, pro neznámá slova.

Pokud chceme, aby uměl náš model spolehlivě pracovat se slovy, která existují exkluzivně pro danou tématickou oblast, ale v porovnání s běžně používanými slovy mají tak malou četnost, že jejich *n-gramy* se vyskytují v trénovacích datech minimálně či dokonce vůbec, je potřeba provést tzv. vyhlazení či diskontování. V principu se jedná o úpravu pravděpodobností výskytu tak, aby se zvýšila pravděpodobnost velmi řídkým a neviděným *n-gramům* na úkor těch pozorovaných často ale zároveň musí platit, že součet všech pravděpodobností se rovná jedné. Základní metodou je *add-k* vyhlazování, kdy se absolutní četnost všech možných kombinací zvedne on $1/k$ kde k je kladné celé číslo. Pokud je k rovno jedné, jde o tzv. *Laplaceovo* vyhlazování.

Tento způsob vyhlazování se prakticky nepoužívá jelikož přistupuje ke všem neporozorovaným *n-gramům* stejně. *Katz backoff* je rekurzivní metoda diskontující pravděpodobnost *n-gramu* pokud byl pozorován a přiřazující váhovanou pravděpodobnost o řád nižšího *n-gramu*, pokud pozorován není. Parametru pro diskontování a váhování se získá maximalizací účelové funkce nad tzv. *held-out* korpusem, tedy částí trénovacího korpusu, který je vyhraněn pro tento účel. Interpolace přiřazuje nepozorovanému *n-gramu* kombinaci

pravděpodobností všech *n-gramů* nižších řádů [11].

$$\hat{p}(w_n|w_{n-2}w_{n-1}) = \lambda_1 p(w_n) + \lambda_2 p(w_n|w_n{-}1) + \lambda_3 p(w_n|w_{n-2}w_{n-1}) \ , \tag{2.31}$$

kde

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \ . \tag{2.32}$$

Absolutní diskontování je způsob snižování pravděpodobnosti pro pozorované *n-gramy* ponížením jejich absolutní četnosti v přepočtu pravděpodobnosti o konstatní hodnotu d s výjimkou velmi málo pozorovaných

$$p_{abs}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda_{w_{i-1}} p(w_i) \qquad 0 \leq d \leq 1 \ , \tag{2.33}$$

kde λ je interpolační váha. Tento způsob byl navržen v [42] z pozorování konstantního rozdílu v počtu *n-gramů* se specifickou absolutní četností v trénovacích a validačních datech.

Nejpopulárnější metodou je *Kneser-Ney* vyhlazování. Ta zavádí parametr pravděpodobnosti pokračování pro *n-gramy* pro slovo w jako počet nově pozorovaných *n-gramů* vzniklých přidáním slova w normalizovaným celkovým počtem *n-gramů*

$$p_{pokr}(w) = \frac{|\{v; C(vw) > 0\}|}{|\{(v', w'); C(v'w') > 0\}|} \ , \tag{2.34}$$

kde $C(.)$ představuje četnost, v je *n-gram* o jeden řád nižší, v' je jakýkoliv *n-gram* o jeden řád nižší a w' je jakékoliv slovo. Pro bigramy pak dostáváme pravděpodobnost

$$p_{KN}(w_i|w_{i-1}) = \frac{\max(C(w_{i-1}w_i - d), 0)}{C(w_{i-1})} + \lambda_{w_{i-1}} p_{pokr}(w_i) \ , \tag{2.35}$$

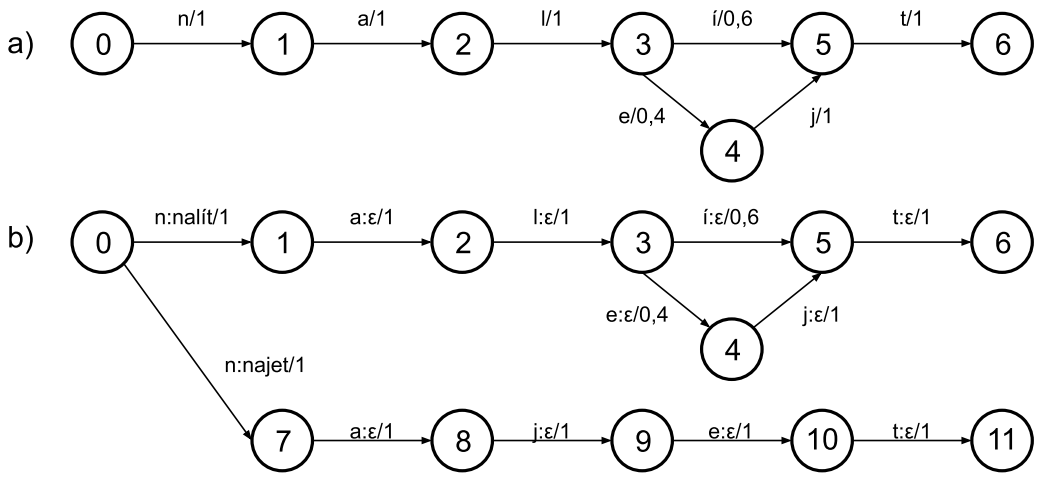
kde $\lambda_{w_{i-1}}$ je normalizační konstanta použitá pro diskontování

$$\lambda_{w_{i-1}} = \frac{d}{\sum_v C(w_{i-1}v} |\{w; C(w_{i-1}w) > 0\}| \ , \tag{2.36}$$

kde první člen je normalizovaná konstanta diskontování d a druhý je počet *bigramů*, na které bylo normalizované diskontování vztaženo.

2.5 Dekódování

Poté, co získáme všechny potřebné parametry akustického a jazykového modelu, je nutné je dát do souvislosti. K mapování od skrytých Markovových modelů po sekvenci neprav-



Obrázek 2.9: Příklady akceptoru (a) a transduktoru (b) pro slova *nalít* a *najet*

děpodobnějších slov se používá realizace pomocí váhovaných konečných transduktorů (WFST) [43]. Transduktory vychází principem z akceptorů. Akceptory jsou automaty, sekvence stavů, u nichž přechody mezi nimi představují vstupní znaky, např. stavy HMM, fonémy nebo tokeny či slova a jim příslušné pravděpodobnosti přechodu. Pravděpodobnost sekvence dané průchodem transduktorem je dána součinen pravděpodobností daných přechodů. Transduktory jsou kromě kromě vstupních znaků charakterizovány i výstupními znaky (viz. porovnání obr. 2.9), které mapují vstupní znaky do dalšího tranduktoru pro vyšší úroveň reprezentace.

Za povšimnutí stojí, že výstupní znak, v případě uvedeného příkladu na obr. 2.9 slovo, emituje pouze první přechod, jelikož z logiky věci by měl transduktor daný token emitovat jen jednou. Ostatní přechody emitují znak ϵ, který v podstatě znamená žádnou akci. Teorie konečných automatů nad těmito strukturami definuje množství operací, které s nimi tyto a další manipulace umožňují, zejména *kompozici*, *determinizaci* a *minimalizaci*, pomocí nichž vytvoří z tranduktorů pro HMM reprezentaci H, mapování z kontextově závislých (*trifonů*) na kontextově nezávislé (*monofóny*) C, výslovnostního slovníku L a jazykového model G finální integrovaný graf

$$HCLG = H \circ C \circ L \circ G \,, \tag{2.37}$$

kde symbol $\circ$ značí kompozici.

Determinizace je operace nad automatem, jež zajistí, že každý stav má pro jakýkoliv daný vstupní znak nejvýše jeden přechod do jiného stavu a neexistují pro něj vstupní znaky ϵ. Znak ϵ představuje přechod, který nemá vstupní znak, resp. negeneruje znak výstupní. Minimalizace je pak operací prováděnou na determinizovanými automaty, dále snižuje

počet stavů a přechodů pomocí přesouvání vah, resp. pravděpodobností přechodů [43]. V praxi se po každé kompozici provádí determinizace a nakonec se celý integrovaný graf minimalizuje.

2.5.1 Zarovnání modelu

Výstupem dekódování GMM-HMM modelu je nepravděpodobnější sekvence fonémů a po aplikaci jazykového modelu (kapitola 2.4) sekvence slov. Avšak pro účely dalšího zpracování je vhodné mít časové značky jednotlivých fonémů korespondující se zvukovou stopou. Za tímto účelem se používají modifikace Viterbiho algoritmu, ač je jeho výkon oproti Baum-Welchově algoritmu suboptimální, pro jeho nižší výpočetní náročnost [44]. Cílem algoritmu je najít průchod transduktorem, představujícím sekvenci stavů stanovenou přepisem záznamu, maximalizujícím účelovou funkci algoritmu, v našem případě maximalixovat věrohodnost účelové funkce pokud známe sekvenci stavů. Výstupem zarovnání je pak finální reprezentace akustického modelu, obsahující natrénované pravděpodobnosti přechodů mezi stavy.

Kapitola 3

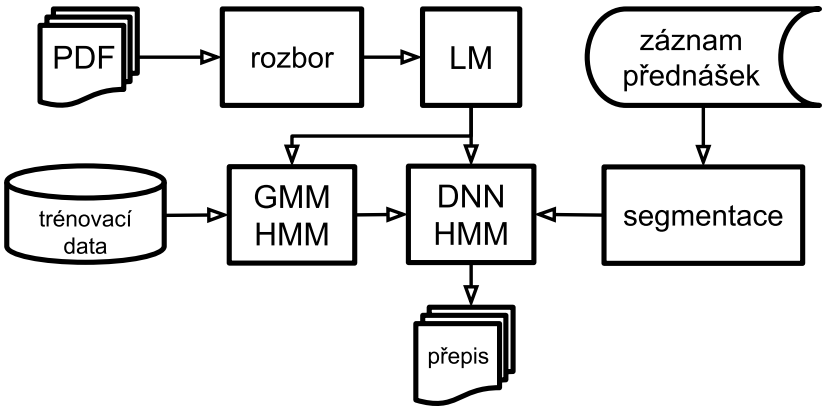
ASR pro přepis spontánní řeči

Hlavním cílem práce je na základě nástrojové sady Kaldi vybrat vhodnou architekturu DNN-HMM pro přepis spontánní řeči, získat postup vytvoření vhodného jazykového modelu a na předložených datech demonstrovat aplikaci zvolené architektury. Jako výchozí bod pro určení vhodného DNN modelu byly zvoleny výsledky práce [45], z několika důvodů. Prvním z důvodů je fakt, že jednou z množin testovacích dat byla databáze segmentů nahrávek spontánní řeči, což odpovídá typově datům, která jsou zpracována i v případě této práce v rámci trénovací množiny. Práce porovnává několik dostupných typů neuronových sítí, z nichž nejlépe vycházejí sítě založené na architektuře TDNN. Tyto důvody byly hlavním přispěvatelem k rozhodnutí implementovat právě tuto strukturu, a to v obou jejích variantách.

Před implementací rozpoznávání samotného je však nejprve nutné vyřešit otázku zpracování přiložených dat, zejména pak audio a videozáznamů přednášek a učebního materiálu ve formátu PDF. Pro možnost prezentace získaných přepisů je nakonec nutné vytvořit skript pro vygenerování příslušných výstupních souborů. Celkový pohled na jednotlivé stavební bloky celého mechanismu popisuje obr. 3.1.

3.1 Struktura a uživatelské rozhraní

Jednou z myšlenek, ač to není přímo zadáním práce, je vytvořit skript na spouštění jednotlivých komponent, oddělit kód samotný od konfigurace komponent a zvláště pak oddělit umístění kódu implementace od vygenerovaných dat. Je zapotřebí do značné hloubky projít a upravit mnohé skripty, které jsou postavné na tzv. „Kaldi receptech“ [13]. Zároveň však není žádoucí jejich strukturu úplně změnit z důvodu zachování alespoň částečné zpětné kompatibility a možnosti snadné integrace částí nově vzniklých receptů Kaldi. Tyto „recepty“ jsou skripty ASR postavenými nad jistými databázemi dat a slouží jako ukázky experimentů, které jsou jednoduše replikovatelné. Jelikož však nebyly tvořeny za jiným



Obrázek 3.1: Grafické znázornění toku dat a závislosti jednotlivých komponent zvolené implementace

účelem než experimentálním, jejich struktura kombinuje kód, parametry a data do stejné struktury složek a souborů. To by však pro naši implementaci a potažmo i její vývoj bylo značnou překážkou.

Jádro frameworku *Kaldi* je napsáno v jazyce `C++` a práce s jeho funkcemi a vyššími logickými jednotkami probíhá v této implementaci pouze skrze skriptování v jazyce *Bash*. Pokud by k vývoji zadané aplikace nad nástrojovou sadou *Kaldi* bylo přistoupeno s širšími možnostmi, rozhodně by bylo vhodné zvolit si jiný jazyk pro implementaci. Nabízí se například jazyk *Python*, pro který je dostupný obaleč [46]. Jelikož se tato možnost v rámci časových dispozic nenabízí a stávající architektura navržená v receptech není vhodná, je zvoleno kompromisní řešení. Téměř všechna práce se odehrává v příkazové řádce a často se hodí mít více terminálů. Z tím účelem bylo vytvořeno prostředí pomocí přizpůsobitelného terminálového multiplexeru *tmux*.

Jednotlivé bloky na obr. 3.1 představují kroky, které jsou spouštěny centrálně pomocí skriptu `main_run.sh`. Volaný krok musí souhlasit s aktuálním profilem nastaveným ve skriptu `config.sh`. Tento skript slouží k definování profilů a volání příslušných konfiguračních skriptů. Profil je skupina konfiguračních skriptů jednotlivých kroků. Konfigurační skripty jsou ve složce `config`, která obsahuje také složku `blueprints` se vzory jednotlivých skriptů, ve kterých je detailně popsáno, které proměnné musí být pro daný konfigurační soubor definovány. Vývoj probíhal na platformě *Metacentrum* [47] a implementace tak obsahuje jak možnost spuštění na lokálním stroji, tak přes plánovač *OpenPBS*. Skript `switch_links.sh` slouží k nastavení cest k binárním souborům a nástrojům v závislosti na stroji, na kterém mají být kroky provedeny.

3.2 Zpracování zvukových dat

Jelikož nástrojová sada Kaldi neobsahuje nástroje pro dekódování v reálném čase, její nástroje pracují se záznamy v jejich originální délce. Pro delší nahrávky výrazně narůstá výpočetní náročnost a při špatném nastavení může zároveň klesat přesnost. Sami autoři se ve fórech opakovaně zmiňují, že zpracování nahrávek delších než 15 sekund vede na snížení přesnosti kvůli knihovně *OpenFST*. V našem případě jsou vstupními daty kompletní záznamy přednášek ve formátu `.wav`, které je nutné nasegmentovat. Jelikož standartním rozhraním přístupu nástrojů Kaldi k audio souborům je program `sox`, stačí pomocí možností tohoto nástroje poukázat na jednotlivé úseky nahrávek a není nutné je přímo segmentovat, je však nutné zajistit, aby střih nahrávky probíhal, pokud možno, v úseku, který neobsahuje řečový úsek.

Jádrem skriptu na segmentaci je nástroj `compute_vad_decision.sh`, který nejprve získá MFCC příznaky celé nahrávky a ze získaných příznaků jednotlivých rámců určí na základě logaritmické hodnoty energie v melovských pásmech [48], zda obsahují mluvené slovo či nikoliv. Z tohoto binárního výstupu se následně za pomoci primitivních pravidel najdou místa vhodná pro střih. Kromě střihu skript také provádí nastavení vzorkovací frekvence, hloubky kvantizace a ohraničení nově vzniklých segmentů krátkými úseky šumu pro správnou inicializaci prvního stavu HMM, kdy je možné si vybrat z několika typů šumů generovaných pomocí programu `sox` nebo nulových vzorků.

Slova u trigramových modelů začínají zpravidla modelem s počátečním stavem reprezentujícím ticho. Pokud jsou první rámce segmentu detekovány jako fonémy, může dojít ke snížení přesnosti detekce, jelikož veškerá trénovací data obsahují ticho na začátku. Kaldi generuje upozornění, pokud se na počátku či konci testovacích dat neobjevuje krátké ticho. Na závěr se vygeneruje nezbytné minimum souborů potřebných pro zařazení dat do testovací množiny. Jedním z takových souborů je soubor `text`, který má obsahovat přepisy nahrávek, které jsou použity na konci pro zjištění přesnosti automatického přepisu.

3.3 Zpracování textových dat

Pro možnost tvorby jazykového modelu je nezbytné mít obsáhlý jazykový korpus. Předložený dokument ve formátu PDF bylo nejprve nutné převést do textové podoby. V prvním kroku se pomocí nástroje na extrakci textu ze strukturovaných PDF souborů, modulu [49] v jazyce *Python*, získá hrubý text jednotlivých stránek a druhém kroku se pomocí běžných nástrojů na editaci souborů na platformě *Linux* získají dva soubory. První soubor obsahuje na každém řádku jednu větu, která vyhovuje všem podmínkám stanoveným v konfiguračních souborech pro daný krok a druhý soubor obsahuje výčet jednotlivých

slov obsažených v souboru prvním. Také je vygenerován soubor, který obsahuje veškeré zkratky, resp. ty, které byly v originálním dokumentu zapsány velkými písmeny.

Pro získání parametrů pro extrakci textu z PDF souboru je vhodné si soubor otevřít pomocí vizualizačního nástroje který je součástí modulu [49]. Ostatní parametry jsou povětšinou regulární výrazy, pro něž existuje množství dokumentace. Pro tvoření výrazů doporučuji použít stránku `https://regexr.com/`.

3.4 Příprava jazykového modelu

V prostředí *Kaldi* se pro práci s jazykovými modely používá sada nástrojů SRILM [50]. Jelikož pro tvorbu úplně nového jazykového modelu je zapotřebí velkého množství dat a dostatečně rozsáhlé jazykové modely jsou k dispozici již zkompilované [51], byl tento krok koncipován jako rozšíření obecného modelu. První fází je vytvoření malého jazykového modelu specifického pro danou tématickou oblast. Tento model vytvořit pomocí příkazu `ngram-count`. Následně se příkazem `ngram` provede interpolace obecného modelu s modelem specifickým a v poslední fázi se rozšíří výslovnostní slovník obecného modelu o unikátní slova z modelu specifického. Získané soubory jsou pak soubor s logaritmickou pravděpodobností jednotlivých n-gramů, slovník a výslovnostní slovník.

Jelikož je výslovnostní slovník unikátních slov ve specifickém modelu vytvářen automaticky a zároveň neobsahuje veškerá slova, která se mohou vyskytovat, byla přidána možnost přidat slova s vlastnoručně vypracovanými výslovnostmi. Sice se přidají do jazykového modelu pouze jako unigramy, jelikož u nich schází kontext, ten se však přidá díky algoritmům vyhlazování. U tohoto souboru byla implementována zároveň i kontrola obsahu.

3.5 Databáze, příznaky a graf pro trénovací fázi

Tato část implementace vychází principiálně z receptů *Kaldi* pro *Wall Street Journal* (WSJ) databázi pouze s obměnami realizujícími myšlenky uvedené v úvodu kapitoly 3. V prvním kroku se pro zvolené databáze z dostupných souborů s přepisy vytvoří pomocí jim příslušných skriptů složky se soubory nutnými pro další zpracování v prostředí *Kaldi*. Dále se připraví složka se soubory specifikujícími parametry pro akustický model. Ta obsahuje informace o definovaných fonémech, mapování fonémů k jejich tokenům, rozdělení na symboly fonémů a symboly reprezentující neřečové události jako nádechy, nesrozumitelnou řeč a ticho. Nakonec je pomocí nástroje `fstcompile` knihovny *OpenFst* vytvořeno mapování fonémů `L.fst`. Tento a následně i transduktor G se však v jedné věci liší od principu uvedeném v kapitole 2.5. Transduktor L představující výslovnostní slovník má

v tomto případě jako výstupní znaky fonémy kromě znaku pro ticho (<*sil*>). G je pak lineární akceptor pro fonémy odpovídající přepisům pro trénování [48].

Pro zvolené trénovací a testovací databáze se na základě parametrů nastavených v konfiguračních souborech získají MFCC koeficienty popsané v sekci 2.1. Následně se provede CMVN normalizace získaných příznaků podle principů popsaných v sekci 2.1.2.

3.6 Trénování GMM-HMM modelu

Trénování GMM-HMM modelů probíhá iterativně v následujících krocích. Postupně se trénují akustické modely monofónů a trifónů přes různou kombinaci normalizovaných MFCC koeficientů nebo některou z variant adaptovaných MFCC koeficientů. Po každém natrénování se provádí zarovnání přepisů získaných dekódováním transduktoru akustického modelu pomocí algoritmů zmíněných v sekci 2.5.1.

Po natrénování monofónů se trénují trifóny s *delta* a *delta-delta* koeficienty na 35 iterací. Trifóny s aplikací LDA se trénují ze zarovnaných dat z trifónů s *delta* a *delta-delta* koeficienty opět na 35 iterací a nakonec jsou ze zarovnaných dat předchozího kroku natrénovány GMM-HMM modely pro trifóny s LDA, fMLLR a SAT adaptacemi na 35 iteracích.

SAT je zkratka pro *Speaker Adapted Training* a provádí trénování modelů GMM-HMM na příznacích, které jsou transformovány na základě znalosti mluvčího pomocí fMLLR, což je jen jiný název pro MLLR aplikovanou přímo na příznakové vektory, které obsahují v tomto případě jak MFCC, tak *delta*, *delta-delta* a LDA příznaky.

U monofónů se zarovnávání přepočítává celkově 21 krát, u posledních dvou trénovacích kroků se zarovnání přepočítává při 10, 20 a 30 iteraci a transformace MLLT a fMLLR jsou přepočítány každou 2,4,6 a 12 iteraci.

Před každým trénováním je nejdříve potřeba dokončit *HCLG* integrovaný graf pro daný model n-gramu, jak je popsáno v kapitole 2.5. Toho lze docílit pomocí příkazu `compile-train-graphs`. Následně se inicializuje trénování rovnoměrným zarovnáním dat podle přepisů trénovacích dat, provede se první reestimace akustického modelu G pomocí programu `gmm-est`. Tento program realizuje aplikaci Baum-Welchovova algoritmu pro standartní MFCC koeficienty a jeho varianty pro adaptované příznakové vektory.

3.7 Dekódování pomocí GMM-HMM

V tomto kroku se pomocí nástroje `fstcompile` vytvoří jazykový model, transduktor G, který je ve skutečnosti akceptor, který má vstupní a výstupní znaky identické a mapují vztahy mezi slovy získané v kroku 3.4. Před kompilací jazykového modelu se ještě provedou

modifikace transduktoru představujícímu mapování z monofónů na slova L, ten se totiž oproti transduktoru L z trénovací fáze liší. Oproti němu je nutné přidat výstupní znaky ϵ, aby bylo možné komponovat L a G, kde G nyní představuje mapování ze slov do sekvencí slov. Další nutnou úpravou je přidání symbolů bránících nejednoznačnosti, vzniklé faktem, že některé dvojice různých slov mají stejnou výslovnost, jinak nazývané spodoba znělosti. Jako příklad lze uvést slova *plot* a *plod*. To se následně promítne i do úpravy transduktoru C tak, aby respektoval symboly bránící nejednoznačnosti. U transduktoru pro trénování se sice prakticky používají, ale pouze pro notaci prázdných slov. Jako akustický model se použije natrénovaný model z poslední, obecně nejpřesnější, varianty trifónů, ale je možné použít jakýkoliv jiný, který se opět adaptuje na nově zahrnuté symboly bránící nejednoznačnosti.

Dekódování následně probíhá pomocí získaného integrovaného grafu $HCLG$, natrénovaného akustického modelu a sekvence příznakových vektorů. Dekodér `gmm-latgen-faster` vytvoří ze zmíněných vstupů pomocí algoritmu dynamického programování založeném na *Viterbiho* algoritmu takzvanou mřížku s několika nejpravděpodobnějšími hypotézami na možnou sekvenci slov [52]. Mřížka je vlastní reprezentace *OpenFST* implementace transduktoru, který obsahuje několik sekvencí slov, každou právě jednou (dekodér provádí determinizaci). Každá sekvence slov je dále dekomponovaná na sekvenci fonémů zarovnaných na odpovídající sekvenci příznakových vektorů. Váhované pravděpodobnosti přechodů jsou v ní reprezentovány dvěma číselnými hodnotami s pohyblivou řádovou čárkou. První reprezentuje pravděpodnosti přechodů mezi fonémy, akustický model, a druhá pravděpodobnosti přechodů mezi slovy, jazykový model. Na tyto mřížky je následně aplikováno několik úprav, včetně přehodnocení pravděpodobností na základě váhy jazykového modelu LMW a nakonec je vybrána sekvence s nejvyšší pravděpodobností.

3.8 Rozpoznání pomocí DNN-HMM

Akustický model DNN-HMM se trénuje na stejných datech jako v případě GMM-HMM a reference pro SGD algoritmus jsou získané zarovnáním dat pomocí akustického modelu na bázi GMM-HMM zvoleným HMM modelem.

V implementaci je na výběr mezi dvěma strukturami DNN-HMM akustického modelu. Podstata standardního TDNN modelu, v *Kaldi* je označovaný *nnet3* podle toho, že jde o třetí generaci, je popsána v kapitole 2.3.1. Nejprve se podle postupu navrženém v [53] provede augmentace původních trénovacích dat, která uměle zvětší celkové množství trénovacích dat. Aplikuje se změna časového měřítka s koeficienty $0,9$, $1,0$ a $1,1$. Tento postup se osvědčil jako obrana proti *overfittingu*. Pro data se změnou časového měřítka je nejprve nutné získat standartní koeficienty a pomocí nich nová zárovnání, jelikož při

změně časového měřítka se změní i počet rámců. Následně se aplikuje změna intenzity hlasitosti, která pomáhá při zpracování dat s různou hlasitostí. Jedním z aspektů neuronových sítí, který je odlišuje od GMM-HMM přístupu, je ten, že dokáží těžit z více MFCC koeficientů. 40 MFCC koeficientů se získá z augmentovaných dat a následně se na ně aplikuje CMVN.

V druhém kroku jsou získána data pro adaptaci na základě principu *i-vektorů* popsaného v 2.2.10. Na získání univerzálního modelu UBM se vyhradí zhruba čtvrtina trénovacích dat na které se aplikuje PCA. Následně je natrénován *extraktor*, který z dat získává *i-vektory* o délce 100. Tyto vektory jsou v posledním kroku získány pro všechny mluvčí jak v trénovací, tak v testovací množině dat.

Struktura TDNN odpovídá principem nákresu sítě s podvzorkováním na obr. 2.7. Vstupní vektor kromě *i-vektoru* obsahuje pět po sobě jdoucích rámců MFCC koeficientů. Následující vrstva je afinní vrstva provádějící LDA, jejíž výstup vstupuje do neuronové sítě jako takové. Následují čtyři skryté vrstvy s $ReLU$ aktivační funkcí. Délka výstupního vektoru odpovídá počtu jedinečných stavů HMM, v tomto případě trigramů, znaků pro ticho a znaků pro odstranění nejednoznačnosti. Přehled struktury je v tabulce 3.1. Účelovou funkcí této architektury je křížová entropie, viz. sekce 2.2.6, a implementace zpětné propagace je upravený algoritmus SGD.

vrstva	typ	poč. dimenzí vstupního vektoru	šířka vrstvy (výstup)	časový kontext
vstup	-	-	300	-
-	LDA	300	650	-
1	ReLU	650	650	{0}
2	ReLU	1950	650	{-1,0,1}
3	ReLU	1950	650	{-1,0,1}
4	ReLU	1950	650	{-3,0,3}
5	ReLU	1950	650	{-6,-3,0}
výstup	log-softmax	650	3560	-

Tabulka 3.1: Hyperparametry standardní TDNN sítě

Druhou dostupnou architekturou TDNN je tzv. *řetězový model*. Tato implementace používá za účelem snížení výpočetní náročnosti třetinovou vzorkovací frekvenci rámců. To s sebou nese nutnost specifické topologie HMM modelu, která umožňuje projití modelu pouze jedním stavem a tudíž je nutné vytvořit integrovaný graf specifický pro tuto implementaci. Další změnou oproti standardní TDNN je použití účelové funkce založené na MMI. Do sítě vstupuje pět po sobě jdoucích rámců MFCC o 40 koeficientech a z nich se pomocí inverzní diskrétní kosinové transformace získají energie v pásmech melovské banky filtrů, viz. kapitola 2.1.1, 20 pro každý rámec. Střední hodnota a variance výsledného vektoru se normalizuje na 1. Následuje jedna skrytá vrstva pro standardní TDNN

s *ReLU* aktivační funkcí a 12 faktorizovaných TDNN vrstev s normalizací, viz. sekce 2.3.2. Strukturu shrnuje tabulka 3.2.

Tato architektura také využívá regularizave jak pomocí L2, tak pomocí L1 normy (*"dropout"* metoda) na většině vrstev. Za 14 vrstvou se síť větví na dvě větve, tzv. *chain* a *xent*. První zmíněná je zde popsána, jelikož její výstup je použit při vyhodnocení. Druhá se od první liší použitou aktivační funkcí ve výstupní vrstvě, *softmax* funkcí, a také použitou účelovou funkcí, kterou je křížová entropie stejně jako v případě standardní TDNN architektury. Algoritmus zpětné propagace je opět upravený algoritmus SGD.

vrstva	typ	poč. dimenzí vstupního vektoru	šířka vrstvy (výstup)	šířka *bottleneck* vrstvy	časový kontext
vstup	-	-	300	-	-
-	IDCT	200 (pouze MFCC)	120	-	-
-	normalizace	120	120	-	-
1	afinní ReLU normalizace	220 (MFCC, *i-vektor*)	1024	-	{0}
2-4	afinní ReLU normalizace	1024	1024	128	{-1,0,1}
5	afinní ReLU normalizace	1024	1024	128	{0}
6-13	afinní ReLU normalizace	1024	1024	128	{-3,0,3}
14	lin. ort. omezení ReLU normalizace	1024	192	-	-
15	afinní ReLU normalizace	192	1024	-	-
16	lineární normalizace	1024	192	-	-
výstup	afinní	192	2256	-	-

Tabulka 3.2: Hyperparametry řetězové struktury TDNN sítě

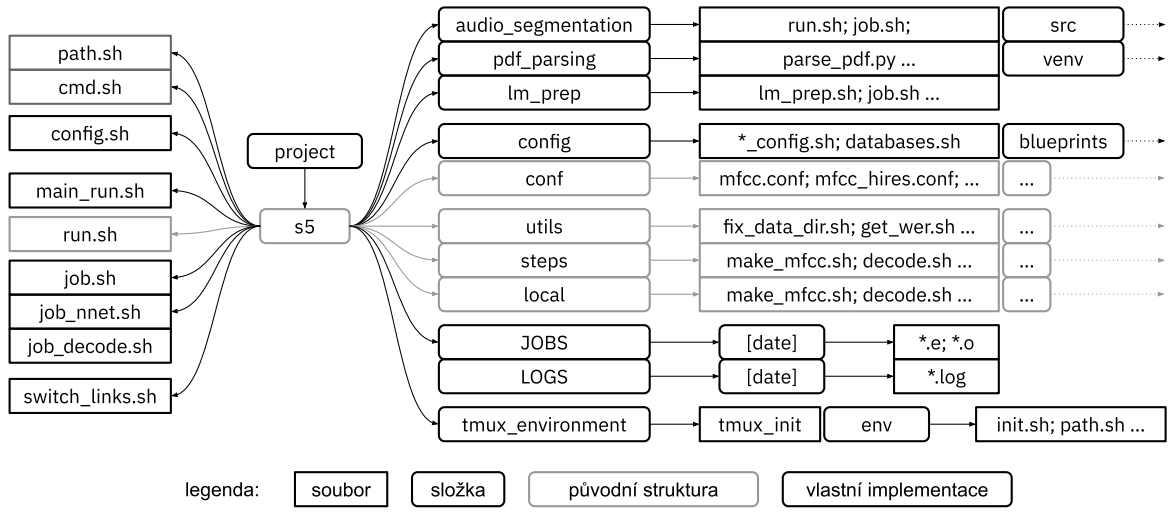
3.9 Dekódování

Dekódování probíhá v podobně jako v případě GMM-HMM s tím rozdílem, že pro generování mřížky je použit program `nnet3-latgen-faster`. Ten je upravený pro integrované grafy vytvořené pro DNN síť a natrénovaný akustický model neuronové sítě. Pro standardní TDNN architekturu se použije stejný integrovaný graf jako pro GMM-HMM a pro řetězovou architekturu jeho vlastní.

Kapitola 4

Implementace

V této kapitole je podán stručný popis struktury implementace a postupu při generování přepisů. Pro detailnější popis obsahují všechny relevantní složky soubor `README` s popisem fungování a rozhraní daného modulu. Většina skriptů vrací po zadání argumentu `-h` popis funkcionalit. V práci je použito množství skriptů vzniklých v rámci laboratoře zpracování řeči při Katedře teorie obvodů na Fakultě Elektrotechnické. Schéma struktury adresáře je na obrázku 4.1.



Obrázek 4.1: Grafické znázornění hierarchie adresáře

4.1 Uživatelské rozhraní

Prostředí v příkazové řádce lze evokovat zavoláním skriptu `tmux_init`. Ten pro svou funkci vyžaduje složku `env` včetně přiložených souborů v domácí složce uživatele. Pokud je uživatel zvyklý na pohybování se v editoru `Vim`, obsahuje složka `env` také přemapování tlačítek na orientaci `tmux.conf`. Návod na použití a nastavení prostředí lze zobrazit pomocí pří-

kazu `tmux_init -h`. Prostředí obsahuje kromě několika dalších oken tři okna v záhlaví, která zobrazují status posledního běhu, což značně urychluje hledání a opravování chyb.

Implementace vychází z práce [45] Ing. Martina Šuberta, která vychází z implementace pro Wall Street Journal korpus, která je standardně součástí instalace *Kaldi*. Jelikož většina parametrů a cest k souborům byla obsažena v kódu, nebyla pro naše účely daná struktura vhodná, jelikož byl mimo jiné stanoven i cíl, aby výsledná implementace byla uživatelsky přístupnější a lehce rozšiřitelná. Za tímto účelem bylo stanoveno několik předpokladů pro rozumnou strukturu, a to:

- všechny **kroky** se spouští pomocí jednoho skriptu,
- všechna generovaná data jsou ukládána mimo složku s implementací,
- všechny podstatné proměnné jsou uchovány mimo skripty,
- nastavení je možné shlukovat do profilů.

Jako kroky jsou myšleny segmentace, zpracování PDF a podobně. Těmito principy by se mělo dosáhnout obecnosti kódu a možnosti migrace kódu. Kvůli množství a různorodosti kódu však nebylo možné těchto cílů dosáhnout v úplnosti, ale částečně požadovaného efektu dosaženo bylo. Jednotlivé kroky se spouštějí pomocí příkazu `main_run.sh`, jehož funkcionalitu lze zjistit pomocí argumentu `main_run.sh -h`. V souladu se spouštěným krokem musí být vyplněno nastavení v souboru `config.sh`, který slouží k nastavení profilu a agregaci jednotlivých souborů s nastavením ve složce `config`, patřícímu k danému profilu.

Před spouštěním jakýchkoliv kroků je nutné si připravit nový profil a alespoň hlavní konfigurační soubor. Ve skriptu `config.sh` je vhodné nový profil nejprve deklarovat v záhlaví jako pole, nastavit proměnnou `profile` podle názvu aktuálního profilu a následně pole profilu definovat jako pole konfiguračních souborů pro jednotlivé kroky. Vzorové konfigurační soubory jsou dostupné ve složce `config/blueprints/`. Vzorové soubory obsahují i vysvětlivky k jednotlivým proměnným. Jakmile je ke každému kroku připraven odpovídající konfigurační soubor, je možné daný krok spustit pomocí `main_run.sh` s parametrem `-r [krok]`. Jednotlivé kroky popisuje tabulka 4.1. Aby bylo možné krok spustit, musí být správně nastaveny cesty ke zdrojovým souborům nástrojových sad *Kaldi*, *OpenFST*, *IRSTLM* a *SRILM* podle toho, zda se má daný krok spouštět lokálně nebo v kontejneru virtualizačního softwaru *Singularity*. K tomu slouží skript `switch_links.sh`, kterým lze cesty změnit a způsob spuštění je nutné zadat do `main_run.sh` pomocí paramteru `-m`. Nakonec je nutné zadat fázi daného kroku. Všechny soubory následně generované se ukládají na adresu zadanou v konfiguraci a nikoliv do složky s implementací. Pokud budou dále uvedeny adresy souborů, vždy jsou myšleny relativně k adrese daného profilu.

Každé jednotlivé spuštění kroku generuje kromě vlastních dat tři soubory. Záznam o průběhu jednotlivých skriptů a spustitelných souborů se ukládá do složky `LOGS`, soubory se standartním a standartním chybovým výstupem se ukládají do složky `JOBS`.

parametr	krok
`-r audio_segmentation`	segmentace zvukových záznamů
`-r pdf_parsing`	zpracování PDF souborů
`-r lm_preparation`	zpracování PDF souborů
`-r gmm`	trénování a dekódování GMM-HMM systému
`-r dnn`	trénování a dekódování DMM-HMM systému
`-r trans`	generování přepisů
`-r dnn`	samostatné dekódování pomocí DNN-HMM systému

Tabulka 4.1: Kroky dostupné v implementaci

4.2 Příprava zvukových záznamů

Samotné segmentování prochází iterativně signál a po každém střihu určí okno několik sekund za střihem, ve kterém hledá nejdelší úsek bez řečové aktivity, jehož spodní hranice trvání nesmí překročit stanovený práh. Pokud se takový úsek nenalezne, okno prodlouží a hledání se opakuje. Tento cyklus se několikrát opakuje, dokud není nalezen vhodný úsek. Pakliže se nenajde, pokusí se skript nalézt jakýkoliv nejdelší úsek bez spodní hranice trvání neřečového úseku. Pokud selže i tento způsob, provede se střih na konci okna a pokračuje se na další segment. Výstupem segmentace je soubor `result/run_result` v příslušné složce, který pro všechny nahrávky a pro dané nastavení parametrů segmentace spočítá průměrnou délku nalezeného bezřečového segmentu, průměrné prodloužení segmentu v sekundách na hodinu záznamu, počet překročení spodní hranice délky bezřečového úseku na hodinu záznamu a počet střihů mimo bezřečový úsek na hodinu. Pomocí těchto hodnot lze empiricky nastavit parametry segmentace tak, aby se co možná nejvíce zamezilo střihům v rámci řeči. Výsledkem je několik souborů, které stačí překopírovat do složky `data/test/` profilu pro rozpoznání.

4.3 Zpracování textových dat

Tento krok provádí rozbor PDF souboru a vytažení relevantních textových dat. V první fázi vyhledá modul `py-pdf-parser` ve struktuře PDF souboru bloky obsahující text. Následně se text filtruje podle rozměru bloku, fontu a velikosti textu a obsahu textu odmítáním celých bloků či části textu na základě regulárních výrazů. Výsledkem je textový soubor v kódování `UTF-8` pro každou požadovanou stránku. Pomocí nástroje zmiňovaného v kapitole 3.3 je vhodné najít font, jímž je prezentován běžný obsah dokumentu,

jelikož ostatní textový obsah, jakým jsou seznamy, popisky obrázků, tabulky neodpovídá hlavnímu odbornému obsahu, který je podstatný pro účely tvorby kontextu tematicky specifického jazykového modelu. Dále je doporučeno odstranit pomocí regulárních výrazů zkratky (tzn., apod., ...) a speciální znaky. Jejich psaná forma neodpovídá mluvené a navíc by zhoršily díky interpunkci na konci rozdělení do vět v následujícím kroku. V druhé fázi se pomocí běžných nástrojů programu *Bash* na zpracování textu upraví získané textové soubory tak, aby vyhovovaly tvorbě jazykového modelu. Tato fáze již nebere žádné parametry od uživatele, tj. text jednotlivých stránek je spojen do souvislého textu, odstraněno dělení slov na koncích řádků, rozdělen na věty podle interpunkce a je na něm provedena finální filtrace pomocí regulárních výrazů. Výstupem je soubor se všemi slovy, soubor s větami bez interpunkce na konci a soubor se seznamem zkratek. Zkratky jsou získány jednoduše jako všechna slova zapsaná velkými písmeny. Soubor se zkratkami nelze dále použít bez ručního zásahu uživatele. Jelikož jsou zkratky hláskovány, vyslovovány s výslovností originálního jazyka nebo mají zažitý specifický způsob výslovnosti, je nutné výslovnost a její možné varianty doplnit ručně. Přitom je nutné dodržet formátování výslovnostních slovníků používaných v nástrojové sadě *Kaldi* a pravidla používaného zápisu výslovnosti, které lze nalézt v knize [54].

4.4 Příprava jazykového modelu

Výskyt některých vybraných slovních spojení specifických pro naši aplikaci může být v dodaném dokumentu velmi řídký. Navíc například spojení *„kepstrální analýzou“* a *„kepstrální analýzy“* jsou obsahově prakticky stejná, ale díky tvarosloví *flektivních* jazyků, jakým čeština je, je každé spojení reprezentováno jedinečným bigramem. U *izolačních* jazyků, jakými jsou angličtina nebo francouzština, tento problém nenastává. Z tohoto důvodu je při tvorbě specifického jazykového modelu nutné dbát na vyhlazování modelu, viz. sekce 2.4.4 a omezit prořezávání modelu. Prořezávání modelu je postup, při kterém se u velkých jazykových modelů vyřazují n-gramy, které mají limitně nízkou četnost.

V první fázi se vytváří model pomocí nástroje `ngram-count`. Nástrojová sada SRILM nabízí pro tento kompilátor několik algoritmů pro vyhlazování. Jelikož se aplikují rekurzivně pro jednotlivé n-gramy, je možné je kombinovat. Výchozí je *Good-Turing* algoritmus, který navíc nabízí možnost omezit diskontování pro n-gramy s nízkou absolutní četností. Diskontuje pak pouze ty, které přesahují zadaný práh nastavený příkazem `-gt[n]max [t]`, kde za `n` se dosadí šířka kontextu n-gramu a za `t` práh. Výrazného zlepšení se dá dosáhnout také získáním relativních četností n-gramů o jeden řád vyšších, než je řád modelu který tvoříme, jelikož četnosti n-gramů jsou zpětně přepočítávány z vyšších [50].

U lineární interpolace jazykových modelů programem `ngram` je nejpodstatnějším para-

metrem váha jednotlivých modelů. Při interpolaci dvou se zadává váhá jednoho, v tomto případě obecného, v konfiguračních souborech proměnná `general_lambda`. V obou fázích lze kromě implicitně zadaných parametrů přidat jakékoliv další, které jednotlivé programy nabízí, pomocí proměnných `ngram_count_extra_options` a `interp_extra_options` v příslušných konfiguračních souborech.

Nakonec se v poslední fázi vytvoří výslovnostní slovník nových slov. K tomuto účelu je použit nástroj `transc` [55] který vytváří výslovnosti automaticky. Pokud byl vytvořen soubor s výslovnostmi zkratek a cesta k jeho umístění je specifikována proměnnou `abbrevs_path` v příslušném konfiguračním souboru, budou tyto zkratky součástí vytvořeného jazykového modelu. Před jejich integrací je soubor se zkratkami zkontrolován pro správnost formátu a použitého značení fonémů. Je nutné podotknout, že pokud není soubor se zkratkami dodán, neznamená to, že ve slovníku jazykového modelu žádné zkratky nejsou. Soubor se zkratkami pouze přidává ručně korigovanou výslovnost všech zkratek nalezených před hlavní filtrací vět.

4.5 Databáze a získání příznaků

Trénovací data jsou získána ze tří databází. První databáze, v kódu značená jako *temic*, vznikla na zadání od *TEMIC Speech Dialog Systems GmbH* [56]. Nahrávky byly pořízeny v prostředí automobilu více mikrofony a za různých podmínek. Pro účely práce však bylo použito pouze 12724 nahrávek od 301 mluvčích z „head-set“ mikrofonu *Sennheiser* při vypnutém motoru. Ostatní nahrávky neodpovídaly aplikaci. Druhá databáze, v kódu označovaná jako *speecon* [57], vznikla pro *Castel GmbH* a obsahuje nahrávky z různých prostředí. Použity byly pouze nahrávky z „head-set“ mikrofonů o celkovém počtu 60877 od 225 mluvčích. Poslední použitá databáze jsou nahrávky přednášek předmětu *Digitální zpracování signálu* [58] v kódu označované *CzLecDsp* o 9452 nahrávkách od 41 mluvčích nahrané na „head-set“ mikrofon. Důležitou charakteristikou této databáze je, že se jedná o spontánní řeč. Celková doba záznamu činí 84,5 hodiny. Všechny audiozáznamy jsou zpracovány program *Sox* při vzorkovací frekvenci 16 kHz a bitové hloubce 16 bitů. Všechna trénovací data i testovací data, resp. složky se soubory odkazujícími na segmenty záznamů přednášek, jsou agregována do odpovídajících složek `data/test` a `data/train`. Parametry testovacích dat se nastavují již v kroku popsaném v sekci 4.2 a je nutné, aby odpovídaly parametrům trénovacích dat.

Nastavení extrakce příznaků je v souboru `conf/mfcc.conf` a počet melfrekvenčních koeficientů je nastaven pro všechny vyhodnocené profily na 20 a mezní frekvence pro melovskou banku filtrů je 7,8 kHz, tzn. -200 Hz od horní hranice kmitočtového pásma.

4.6 Příprava zarovnaných dat pomocí GMM-MM

V první fázi jsou zkombinována veškerá testovací data zadaná v konfiguračním souboru. Následně se připraví akustický model na základě seznamu fonémů z nástrojové sady HTK [54]. Bližší popis je v sekci 3.5. Trénování a dekódování probíhá tak jak je popsáno v kapitolách 3.6 a 3.7. Pro další postup není dekódování nutné, ale slouží jako způsob kontroly, zda získaná zarovnání dostatečně odpovídají realitě.

Zarovnání dat s HMM modelem se běžně ukládají v binárním formátu, ale pomocí nástroje `show-alignments` lze výsledky zarovnání vizualizovat. Zjednodušený výstup můžeme vidět ve výpisu 4.2. V první řádce jsou identifikátory přechodů HMM modelu, které odpovídají jedinečným pravděpodobnostem přechodů v HMM modelu a lze pomocí nich mapovat další parametry modelu, jako jsou fonémy [48]. V druhém řádku jsou již fonémy, které daná část řetězce představuje. Druhým potřebným výstupem zarovnání je HMM model s natrénovanými pravděpodobnostmi přechodů označovaný `final.mdl`

```
me39bc115015  [ 2 1 4 3 3 3 6 5 ] [ 254 253 258 ] [ 74 73 75 78 ] ...
me39bc115015  sil                 v                 e                ...
```

Tabulka 4.2: Zobrazení souboru zarovnání dat

4.7 Rozpoznání pomocí DNN-HMM

Skripty pro spouštění rozpoznání na principu hybridního systému DNN-HMM zůstaly prakticky nezměněny, pouze bylo nastavení podstatných proměnných přesunuto do konfiguračních souborů. Pomocí natrénovaného HMM modelu `final.mdl` jsou získána zarovnání identifikátorů přechodů pro data se změněným časovým měřítkem. Dál pokračuje rozpoznání podle popisu v sekci 3.8 a 3.9. Jako zdroj zarovnání pro trénování DNN sítě je zvolen model pro trifóny s LDA, fMLLR s SAT adaptacemi (`tri3`).

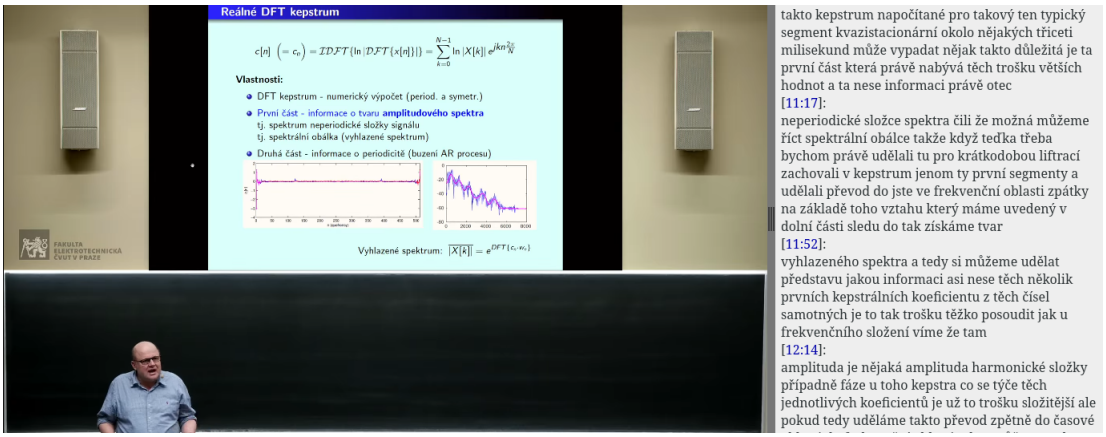
4.8 Vytvoření přepisů v požadovaných formátech

Pro možnost prezentace výsledků a možnosti další práce s přepisy je implementován skript, který zajišťuje automatickou agregaci přepisů ze standartního výstupu programu `lattice-best-path`. Implementovány jsou 4 formáty. První je textový soubor, který odpovídá formátu přepisů používaných u trénovacích dat a umožňuje výpočet metriky WER. Tato metrika je implementována pomocí *Levenshteinovy vzdálenosti* nad slovy, tedy jako poměr nejmenšího počtu nutných úprav, vymazání, editace a vložení, k počtu slov v referenční větě, kterým lze dosáhnout identity dané věty s její referencí. Implementovány

jsou i další metriky, provádějící stejný úkon nad písmeny a fonémy, avšak tato metrika je nejvíce diskriminativní, jelikož dvě slova se mohou psát či vyslovovat podobně, ale významem mohou být vzdálená.

Další tři formáty jsou pro práci a zobrazení v různých multimediálních programech. Soubory s příponou `.srt` jsou titulky běžného formátu, které lze snadno zobrazit pomocí volně dostupného přehrávače *VLC Media Player*. Soubory formátu `.json` jsou specifické soubory, imitující strukturu souboru s titulky získanými pomocí Google speech-to-text API. Poslední formát `.trs` je používaný v programu *Transcriber* pro editaci titulků.

Přepisy všech přednášek předmětu Zpracování řeči ve formátu `json` jsou použity v rámci vizualizace na stránách předmětu vytvořené v rámci práce [59]. Ukázka je na obrázku 4.2. Vizualizace umožňuje vyhledávat jednotlivá slova a spouštět záznam z odpovídajících časových značek.



Obrázek 4.2: Ukázka vizualizace na stránkách předmětu

4.9 Izolovaný krok dekódování

Pro zjednodušení dekódování nových nahrávek pomocí vytvořeného modelu byl vytvořen krátký skript, který pro vybraná nasegmentovaná data specifikovaná v konfiguračním souboru proměnnou `extra_decode` vytvoří přepisy pomocí `chain` modelu vybraného profilu spuštěním kroku `decode` a uloží je na adresu specifikovanou proměnnou `extra_trans_path`.

4.10 Vlastní kód

Vlastní implementace je v archivu přiloženém k práci. Všechny významné složky včetně hlavního adresáře obsahují `README` soubor se stručným popisem.

Kapitola 5

Experimentální část

V rámci této kapitoly jsou popsány výsledky experimentů, jejichž výsledky slouží jako nastavení jednotlivých dílčích modulů celého systému, a to od základní konfigurace rozpoznávače DNN-HMM, přes kvalitu jazykových modelů, až po přesnost finálních přepisů. Nejprve byly získány takové parametry segmentace zvukových nahrávek, aby se docílilo minimálního střihu v úsecích obsahujících promluvu. Po odladění zpracování PDF souborů bylo experimentováno s parametry jazykových modelů. Nastavení GMM-HMM systému je na základě teoretických poznatků a parametry neuronových sítí zůstaly nezměněny, odpovídají tedy parametrům použitým pro *Kaldi* „recept“ WSJ.

Finální parametry vedoucí k výsledkům prezentovaným v této sekci jsou v konfiguračních souborech uvedených v tabulce 5.1. Výpis všech logovacích souborů pro daný profil je možný příkazem `head LOGS/*/*.log | grep -B1 [profil]`.

krok	profil
audio segmentace	`zre_ls2324_v3_segmentation`
zpracování PDF	`zre_book_parsing_v2`
příprava LM	`zre_lm_preparation_v3_1`
ASR	`zre_ls2324_v3_abbrevs`

Tabulka 5.1: Použité nastavení parametrů

5.1 Testování DNN-HMM ASR

Před samotnou tvorbou přepisů přednášek je nutné se na základě dříve dosažených výsledků přesvědčit o správném vyhodnocování GMM-HMM systému. Jako reference byly použity výsledky rozpoznání pro český jazyk z práce [45]. Pro trénování byly použity databáze *temic* a *speecon* a vyhodnocení proběhlo na části databáze *speecon*. Pro rozpoznání na struktuře TDNN sítě *nnet3* bylo dosaženo hodnoty 20,55 WER a bylo tudíž usouzeno, že je akustický model správně nastaven. Nastavení pro toto vyhodnocení jsou uložena

pod profilem `accuracy_test`. Pro příznakový vektor bylo použito 20 melfrekvenčních kepstrálních koeficientů.

5.2 Nastavení parametrů segmentace

Parametry segmentace závisí na kadenci promluvy mluvčího a je nutné ji tomu přizpůsobit. Zároveň je nutné dbát doporučení autorů nástrojové sady *Kaldi* ohledně délky segmentů. Parametry byly postupně nastavovány za vyhodnocování na části trénovací množiny nahrávek, dokud nebylo dosaženo nulového počtu střihů v časech nahrávky obsahujících promluvu. S ohledem na časté prodlužování délky segmentu byly nastaveny hranice délky na 8, resp. 13 sekund a průměrná délka segmentu se pak pohybuje okolo **7** sekund. Pro ohraničení segmentů byl zvolen bílý šum s hladinou zesílení -55 dB. V tabulce 5.2 jsou vypsány jak parametry segmentace, tak metriky segmentace z testovacích dat.

parametr	hodnota
minimální délka úseku	3 s
maximální délka úseku	8 s
minimální délka neřečového úseku	0,1 s
délka prodloužení	1 s
maximální počet prodloužení	5
mezera mezi střihem a hranicí	0,05 s
trvání ohraničení	0,2 s
typ šumu ohraničení	bílý
zesílení šumu	-55 dB
metrika	**hodnota**
průměrná délka volených neřečových úseků	0,299
průměrná délka segmentu	7,502 s
průměrný počet střihů mimo neřečové segmenty na hodinu záznamu	0
průměrný počet porušení minimální délky neřečového segmentu na hodinu záznamu	128
průměrný počet prodloužení na hodinu záznamu	933

Tabulka 5.2: Parametry a výsledky segmentace

5.3 Testování přínosu jazykových modelů

Rozšíření jazykového modelu bylo provedeno na jazykovém modelu vytvořeném v rámci práce [51]. Ten byl získán z Českého národní korpusu *SYN2006PUB* [60] s maximálním řádem n-gramů 5. Tato databáze má v originální podobě 340 milionů slovních elementů. Z ní byl po filtraci a omezení popsaném v [51] vytvořen model s parametry popsanými v tabulce 5.3, kde jsou pro porovnání prezentovány zároveň i parametry ostatních modelů.

Jak bylo uvedeno v sekci 2.4.2, jazykové modely lze porovnávat dvěma způsoby, vnějším a vnitřním. Vnější vyhodnocení, tedy výsledky rozpoznání, jednoznačně dokazuje jak přínos rozšíření jazykového modelu tak přidání výslovností zkratek. Tento fakt lze dokázat částečně pomocí počtu OOV, slov neobsažených ve slovníku modelu. Při vyhodnocení na 100 korigovaných přepisech měl obecný model těchto slov **113** a rozšířený včetně zkratek **71**, což je zvláště pro tak malý vzorek výrazný rozdíl.

-	obecný	specifický	rozšířený obecný
unigramy	340 002	9 355	341 719
bigramy	44 602 819	37 317	44 622 566
trigram	20 443 214	2 176	20 444 996

Tabulka 5.3: Parametry jednotlivých jazykových modelů

Oba kroky, interpolace obecného jazykového modelu se specifickým modelem a přidání zkratek, rozšiřují slovník modelu a tudíž nelze přímo srovnávat obecný s rozšířenými modely pomocí perplexity. Co však můžeme, je pokusit se nastavit parametry rozšířeného jazykového modelu tak, abychom minimalizovali perplexitu a získat tak vhodné parametry pro tvorbu a interpolaci jazykových modelů. Pro rozšířený jazykový model se zkratkami bylo realizováno několik kombinací parametrů tvorby specifického a váhy obecného modelu, viz. tabulka 5.4. Ve všech případech byl výstupní řád modelu roven 3. GT, WB, KN a UKN jsou zkratky pro algoritmy diskontování, a to v pořadí *Good-Turing*, *Witten-Bell*, *Kneser-Ney* a *Unmodified Kneser-Ney*. Pro výpočet perplexity byly použity níže popsané ručně korigované přepisy 100 vybraných segmentů. Z hodnot perplexity jednotlivých variant je vidět výrazný přínos generování modelu s řádem kontextu o jeden vyšším, než je řád výstupní. Optimálních hodnot perplexity je dosaženo použitím *Good-Turingova* algoritmu diskontování společně s váhou obecného modelu při interpolaci okolo hodnoty 0,55.

generovaný řád	3	**4**	5	4	4	**4**	4	4	4
algoritmus diskontování	GT	GT	GT	GT	GT	**GT**	WB	KN	UKN
váha obecného modelu	0,5	0,5	0,5	0,3	0,45	**0,55**	0,55	0,55	0,55
perplexita	1395	**1022**	1022	1105	1030	**1020**	1086	1036	1049

Tabulka 5.4: Perplexity pro různá nastavení jazykového modelu

5.4 Testování kvality finálních přepisů

Pro účely vyhodnocení přesnosti rozpoznání pomocí metriky WER bylo náhodně vybráno 100 segmentů, jejichž automatické přepisy měly dostatečný počet slov, konkrétně více než

20. Testovací množina má pak celkově 2302 slov, 896 unikátních. Celkový čas testovací množiny je 20 minut s průměrnou délkou segmentu 11,8 sekund. Přepisy těchto segmentů byly ručně zkorigovány v programu *Transcriber* tak, aby odpovídaly obsahu, který byl v daných segmentech vysloven a zároveň splňovaly parametry referencí pro tento systém.

5.4.1 Dosažené výsledky WER

Výsledky úspěšnosti rozpoznání potvrdily výsledky dosažené v [45] a to konkrétně tvrzení, že řetězová struktura TDNN sítí je pro aplikaci rozpoznání spontánní řeči v českém jazyce vhodnější než standardní struktura. Nejlepších výsledků, **15,62** %, dosáhl specifický trigramový model bez zkratek vyhodnocený řetězovou architekturou. Prakticky stejného výsledku dosáhl rozšířený trigramový model se zkratkami, konkrétně **16,12** %. Rozdíl mezi použitím bigramového a trigramového jazykového modelu jsou 2 promile, což odráží nízký počet trigramů v jazykovém modelu. Po interpolaci specifického modelu s obecným je počet trigramů poloviční oproti počtu bigramů. U samostatného specifického modelu byl tento poměr dokonce 1:17. Aby se přínos trigramů projevil, bylo by zapotřebí výrazně více textových dat pro trénování.

Kromě zmíněných měly všechny jazykové modely parametry trénování stejné, konrétně váhu obecného modelu $0,5$ a diskontování pomocí *Good-Turingova* algoritmu. Pomocí metriky WER bylo na testovací množině vyhodnoceno několik kombinací jazykového modelu a struktury TDNN. Přehled výsledků je v tabulce 5.5. Jelikož je testovací množina segmentů záznamů značně omezená, je nutné brát výsledky spíše jako orientační.

jaz. model:	obecný	rozšířený tématicky specifický			long
verze:	trigram	bigram	trigram	trigram se zkratkami	latest
nnet3	-	-	43,14	34,15	-
chain	19,53	15,80	15,62	16,12	-
Google	-	-	-	-	11,58

Tabulka 5.5: Přehled získaných WER pro různé kombinace jednotlivých částí systému

Pro srovnání jsou v tabulce 5.6 uvedeny hodnoty WER dosažené v práci Ing. Šuberta [45]. Toto vyhodnocení proběhlo se stejnými parametry extrakce příznakových vektorů a akustického modelu. Zásadním rozdílem je trénovací množina, která neobsahuje spontánní promluvy. Testovací množina je výběr ze záznamu přednášek předmětu pokročilé metody DSP, jedná se tedy o záznam spontánní řeči srovnatelné se záznamy přednášek zpracování řeči. Z výsledků lze pozorovat inkrementální zlepšení po přidání databáze se spontánní řečí a po upravení jazykového modelu.

-	Šubert	obecný model	rozšířený model
nnet3	31,60	-	34,15
chain	25,98	19,53	16,12

Tabulka 5.6: Porovnání s výsledky práce Ing. Šuberta

5.4.2 Vliv výslovnostních variant zkratek

Přidání zkratek snížilo úspěšnost detekce. Mohlo by to být dáno malým vzorkem použitým k vyhodnocení, ale jako příklad mějme překlad věty z následující tabulky.

reference	`i ten wav jako obsahuje nejenom standardní pcm ale může tam být třeba ad /zaváhání/ adpcm`
bez zkratek	`i ten vás jako obsahuje nejenom standardní péčéčkem ale může tam být třeba v adelaide péčéčkem`
se zkratkami	`i ten wav jako obsahuje nejenom standardní pcm ale můžete být třeba vad a dpcm`

Tabulka 5.7: Porovnání rozpoznání se zkratkami a bez nich

Je zřejmé, že rozpoznání zkratek funguje, ale jejich přidání, zdá se, obecně snížilo schopnost rozpoznání. To může být dáno mimo jiné tím, že podobně znějící slova mohou být nyní zaměněna za tyto zkratky jelikož ty mají více variant výslovnosti, zejména českou a anglickou. Navíc, jak bylo uvedeno v kapitole 4.4, jazykový model bez zkratek obsahuje některé zkratky s jejich automaticky generovanou výslovností. Řešením by pak mohlo teoreticky být uložení zkratek do vlastního malého jazykového modelu a ten pak interpolovat s obecným jazykovým modelem s větší vahou obecného modelu.

5.4.3 Srovnání s Google Chirp

Pro možnost srovnání s dostupnými nástroji byla testovací množina vyhodnocena i pomocí nástroje speech-to-text API modelu Chirp [7], viz tabulka 5.5. Skript pro vyhodnocení pomocí tohoto modelu a výsledky jsou ve složce `google_chirp`. Pro český jazyk je na výběr několik modelů rozlišených podle typu a délky záznamu. Pro naši aplikaci se nejlépe hodil model označovaný jako *latest_long*. Tento model očekávaně dosáhl lepších výsledků než naše aplikace, konkrétně **11,58**% WER. Model vracel čísla ve formě číslovek, a nikoliv ve formě slov, a tím se vyhodnocení WER zhoršilo a ve skutečnosti se pohybuje okolo 11 %. API rozhraní modelu bohužel možnost takové změny neumožňuje. Tento výsledek byl bez rozšíření o neznámá slova, který daný model poskytuje. Následuje výběr vět na kterých lze snadno pozorovat některé zmíněné aspekty.

Google:	plus slov a v rámci těch 10 slov už ten fonetický obsah bude dostatečně bohatý a to ke pstruhům řeči už se k té nule začne
Kaldi	plus slov a v rámci těch deseti slov už ten fonetický obsah bude dostatečně bohatý a to kepstrum řeči useknuté nule začne
Google:	intenzity signálu čili že můžeme tedy počítat energii respektive výkon senzorku velmi jednoduchý odhad výkon je střední kvadratická hodnota
Kaldi:	intenzity signálu čili že můžeme tedy počítat energie respektive výkon z n vzorků velmi jednoduchý odhad výkon je střední kvadratická hodnotách
Google:	my vlastně pomocí toho darwinova algoritmu počítáme vždycky ten nový koeficientka a pomocí té levinsonovy rekurze přepočítáme ten transverzální model na ten vyšší
Kaldi:	vlastně pomocí toho darwinova algoritmů počítáme vždycky ten nový koeficient k a pomocí televize nově rekurzí přepočítáme ten transverzální model na ten vyšších

Tabulka 5.8: Porovnání výsledků rozpoznání pomocí Kaldi a Google Chirp

Za povšimnutí stojí schopnost implementace v *Kaldi* správně rozpoznat v tématice zpracování signálu běžné termíny jakými jsou *„z n vzorků“* a *„koeficient k“*. Na druhou stranu má v některých případech značnou potíž se spontánností řeči. Jediná reprezentace neřečové události je symbol *<sil>*, který dokáže stěží pokrýt všechny aspekty neřečových událostí. Jako příklad důsledku mějme rozpoznání slova *„hodnota“* jako *„hodnotách“*. Z poslechu nahrávky vyšlo najevo, že po slově *„hodnota“* následovalo výrazné nadechnutí které bylo interpretováno jako hláska *„ch“*.

Kapitola 6

Závěr

Předložená práce popisuje implementaci funkčního rozpoznávače dlouhých nahrávek spontánní řeči. Po popisu základní teorie z oboru rozpoznávání řeči je v práci prezentováno základní řešení pomocí existujících nástrojů *Kaldi* pro realizaci DNN-HMM rozpoznávače. Kompletní řetězec úkonů zahrnuje přípravu textových dat pro modifikaci jazykového modelu, nasegmentování zvukových stop, vytvoření jazykového modelu a jeho interpolaci, trénování a zarovnání GMM-HMM modelu, trénování hybridního DNN-HMM a nakonec dekódování a agregaci přepisů. V rámci práce vznikly přepisy pro přednášky předmětu Zpracování řeči za poslední semestr a jsou dostupné studentům.

Výsledky rozpoznání za použití rozšířeného jazykového modelu získané vyhodnocením ručně korigovaných segmentů vykazují jednoznačný přínos. Zlepšení o **20**% přineslo rozšíření z jednoho dokumentu s použitím 250 stránek z nichž bylo použito 5200 vět. Ač je získaný specifický model oproti obecnému nesrovnatelně menší, výsledný interpolovaný model přináší lepší výsledky, mimo jiné díky rozšíření slovníku. Pokus s přidáním ručně korigovaných zkratek do modelu a výslovnostního slovníku sice vedl na lepší rozpoznání zkratek samotných, ale zároveň měl negativní vedlejší účinek na přesnost obecně.

Aktuální implementace systému pro přepis přednášek zahrnuje všechny potřebné výše zmíněné kroky, poskytuje přepisy v akceptovatelné přesnosti pro účely zaindexování a rychlejšího vyhledávání v záznamu přednášky s přepisem, zároveň také odhaluje místa, kde je prostor pro vylepšení daného přístupu.

Příprava textových dat pro trénování jazykového modelu byla implementována pouze pro strukturované PDF soubory, ty však představují jen část možné zdrojové literatury. Samotné zpracování PDF představuje samostatnou kapitolu vyžadující důsledné úpravy, protože současná implementace vyžaduje značný vstup od uživatele a filtrace je sice robustní, ale nekompromisní a zbytečně velké množství dat sítem neprojde. V pozdní fázi práce byl nalezen potencionálně lepší přístup k segmentaci zvukových nahrávek využívající segmentace generující překryvy jednotlivých segmentů a *time-marked conversation*

(CTM) soubory obsahující časové značky jednotlivých slov. Tento přístup může zajistit plný kontext pro všechna slova ve finálním přepisu a je jedním z doporučených vylepšení navrženého systému. Dalšími úpravami, které by vedly k možnému zlepšení, jsou rozšíření akustických modelů o neřečové události, získání více trénovacích dat obsahujících spontánní řeč a nakonec využití novějšího obecného jazykového modelu s roszáhlejším slovníkem.

Implementace představuje koncept, na jehož principu by bylo možné vytvořit modulární systém. Takový systém, postavený na základě nástrojové sady *Kaldi* s využitím vhodného prostředí nabízejícím výhody objektového programování, jakým je například obaleč *PyKaldi*, by bylo možné opatřit i dalšími schopnostmi, jako je například zpracování v reálném čase.

Bibliografie

[1] *(1961) Shoebox - IBM Archives (78-013)*, [Online; citováno květen 2024]. URL: https://mediacenter.ibm.com/media/(1961)+Shoebox+-+IBM+Archives+(78-013)/0_4m2ynnkk.

[2] F. Jelinek, „Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, roč. 64, č. 4, s. 532–556, 1976. DOI: 10.1109/PROC.1976.10159.

[3] K.-F. Lee, H.-W. Hon, M.-Y. Hwang, S. Mahajan a R. Reddy, „The SPHINX speech recognition system," in *International Conference on Acoustics, Speech, and Signal Processing,*, 1989, 445–448 vol.1. DOI: 10.1109/ICASSP.1989.266459.

[4] C.-H. Lee, E. Giachin, L. Rabiner, R. Pieraccini a A. Rosenberg, „Improved acoustic modeling for large vocabulary continuous speech recognition," *Computer Speech & Language*, roč. 6, č. 2, s. 103–127, 1992, ISSN: 0885-2308. DOI: https://doi.org/10.1016/0885-2308(92)90022-V.

[5] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey et al., „Robust speech recognition via large-scale weak supervision," in *Proceedings of the 40th International Conference on Machine Learning*, , Honolulu, Hawaii, USA, Journal of Machine Learning Research, 2023.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit a L. e. a. Jones, „Attention is All you Need," in *Advances in Neural Information Processing Systems*, sv. 30, Curran Associates, Inc., 2017.

[7] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar a Y. Z. et al., „Conformer: Convolution-augmented Transformer for Speech Recognition," *ArXiv*, roč. abs/2005.08100, 2020.

[8] P. Tremblay, I. Deschamps a V. L. Gracco, „Chapter 59 - Neurobiology of Speech Production: A Motor Control Perspective," in *Neurobiology of Language*, G. Hickok a S. L. Small, ed., San Diego: Academic Press, 2016, s. 741–750, ISBN: 978-0-12-407794-2. DOI: https://doi.org/10.1016/B978-0-12-407794-2.00059-6.

[9] D. O'Shaughnessy, „Trends and developments in automatic speech recognition research," *Computer Speech & Language*, roč. 83, s. 101 538, 2024, ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2023.101538.

[10] M. Gales a S. Young, *Application of Hidden Markov Models in Speech Recognition*. Now Foundations a Trends, 2008. DOI: 10.1561/2000000004.

[11] D. Jurafsky a J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2008, sv. 2.

[12] P. Mermelstein, „Distance measures for speech recognition, psychological and instrumental," *Pattern Recognition and Artificial Intelligence*, s. 374–388, 1976.

[13] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek et al., „The Kaldi speech recognition toolkit," *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, led. 2011.

[14] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh et al., „Sphinx-4: A flexible open source framework for speech recognition," *Sun Microsystems*, pros. 2004.

[15] G. T. Fechner, *Elemente der psychophysik*. Breitkopf u. Härtel, 1860, sv. 2.

[16] I. Mporas, T. Ganchev, M. Siafarikas a N. Fakotakis, „Comparison of speech features on the speech recognition task," *Journal of Computer Science*, roč. 3, č. 8, s. 608–616, 2007.

[17] S. Furui, „Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, roč. 34, č. 1, s. 52–59, 1986. DOI: 10.1109/TASSP.1986.1164788.

[18] K. Johnson, „The Δf method of vocal tract length normalization for vowels," *Laboratory Phonology*, roč. 11, č. 1, 2020, ISSN: 18686346. DOI: 10.5334/LABPHON.196.

[19] L. E. Baum, T. Petrie, G. W. Soules a N. Weiss, „A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics*, roč. 41, s. 164–171, 1970.

[20] L. D. Dong Yu, *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2014.

[21] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed et al., „Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *Signal Processing Magazine, IEEE*, roč. 29, s. 82–97, lis. 2012. DOI: 10.1109/MSP.2012.2205597.

[22] L. Deng, „Computational Models for Speech Production," in *Computational Models of Speech Pattern Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, s. 199–213, ISBN: 978-3-642-60087-6. DOI: 10.1007/978-3-642-60087-6_20.

[23] I. H. Sarker, „Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *Sn Computer Science*, roč. 2, 2021.

[24] X. Zhang, J. Trmal, D. Povey a S. Khudanpur, „Improving deep neural network acoustic models using generalized maxout networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, s. 215–219. DOI: 10.1109/ICASSP.2014.6853589.

[25] S. Ruder, „An overview of gradient descent optimization algorithms," *ArXiv*, roč. abs 1609.04747, 2016.

[26] A. Graves, S. Fernández, F. Gomez a J. Schmidhuber, „Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, s. 369–376, ISBN: 1595933832. DOI: 10.1145/1143844.1143891.

[27] K. Veselý, A. Ghoshal, L. Burget a D. Povey, „Sequence-discriminative training of deep neural networks," in *Proc. Interspeech 2013*, srp. 2013, s. 2345–2349. DOI: 10.21437/Interspeech.2013-548.

[28] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar et al., „Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI," in *Proc. Interspeech 2016*, zář. 2016, s. 2751–2755. DOI: 10.21437/Interspeech.2016-595.

[29] N. Jaitly, P. Nguyen, A. Senior a V. Vanhoucke, „Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech 2012*, 2012, s. 2578–2581. DOI: 10.21437/Interspeech.2012-10.

[30] G. Dahl, M. Ranzato, A.-r. Mohamed a G. E. Hinton, „Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel a A. Culotta, ed., sv. 23, Curran Associates, Inc., 2010.

[31] I. Goodfellow, Y. Bengio a A. Courville, *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[32] G. Saon, H. Soltau, D. Nahamoo a M. Picheny, „Speaker adaptation of neural network acoustic models using i-vectors," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, s. 55–59. DOI: 10.1109/ASRU.2013.6707705.

[33] A. Senior, G. Heigold, M. Bacchiani a H. Liao, „GMM-free DNN acoustic model training," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, s. 5602–5606. DOI: 10.1109/ICASSP.2014.6854675.

[34] A. H. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano a K. J. Lang, „Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech Signal Process.*, roč. 37, s. 328–339, 1989.

[35] V. Peddinti, D. Povey a S. Khudanpur, „A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. Interspeech 2015*, 2015, s. 3214–3218. DOI: 10.21437/Interspeech.2015-647.

[36] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu et al., „Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks," in *Proc. Interspeech 2018*, 2018, s. 3743–3747. DOI: 10.21437/Interspeech.2018-1417.

[37] J. T. Goodman, „A bit of progress in language modeling," *Computer Speech & Language*, roč. 15, č. 4, s. 403–434, 2001, ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.2001.0174.

[38] C. Wei, Y. C. Wang, B. Wang a C.-C. J. Kuo, „An Overview on Language Models: Recent Developments and Outlook," *ArXiv*, roč. abs/2303.05759, 2023.

[39] K. Hwang a W. Sung, „Character-level language modeling with hierarchical recurrent neural networks," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, s. 5720–5724, 2016.

[40] P. Gage, „A new algorithm for data compression," *The C Users Journal archive*, roč. 12, s. 23–38, 1994.

[41] T. Rotovnik, M. S. Maučec a Z. Kačič, „Large vocabulary continuous speech recognition of an inflected language using stems and endings," *Speech Communication*, roč. 49, č. 6, s. 437–452, 2007, ISSN: 0167-6393. DOI: https://doi.org/10.1016/j.specom.2007.02.010.

[42] K. W. Church a W. A. Gale, „A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams," *Computer Speech & Language*, roč. 5, č. 1, s. 19–54, 1991, ISSN: 0885-2308. DOI: https://doi.org/10.1016/0885-2308(91)90016-J.

[43] M. Mohri, F. Pereira a M. Riley, „Speech Recognition with Weighted Finite-State Transducers," in *Springer Handbook of Speech Processing*. Springer Berlin Heidelberg, 2008, s. 559–584.

[44] L. J. Rodriguez-Fuentes a M. Torres, „Comparative Study of the Baum-Welch and Viterbi Training Algorithms Applied to Read and Spontaneous Speech Recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, čvn. 2003, s. 847–857, ISBN: 978-3-540-40217-6. DOI: 10.1007/978-3-540-44871-6_98.

[45] M. Šubert, *Continuous Speech Recognition using Advanced Deep Neural Networks*, Diplomová práce, ČVUT FEL, Praha, 2021.

[46] *PyKaldi*, ver. 0.2.2, [Online; citováno květen 2024], 19. ún. 2010. URL: `https://github.com/pykaldi/pykaldi`.

[47] *MetaCentrum*, [Online; citováno květen 2024]. URL: `https://metavo.metacentrum.cz/`.

[48] D. Povey, *Kaldi*, [Online; citováno květen 2024]. URL: `https://kaldi-asr.org/doc/index.html`.

[49] J. Stockwin, *PDF parser*, [Online; citováno květen 2024]. URL: `https://py-pdf-parser.readthedocs.io/en/latest/overview.html`.

[50] A. Stolcke, „SRILM - an extensible language modeling toolkit,“ in *Proc. 7th International Conference on Spoken Language Processing (ICSLP 2002)*, 2002, s. 901–904. DOI: `10.21437/ICSLP.2002-303`.

[51] V. Prochazka, P. Pollak, J. Zdánský a J. Nouza, „Performance of Czech Speech Recognition with Language Models Created from Public Resources,“ *Radioengineering*, roč. 20, č. 4, s. 1002–1008, pros. 2011, ISSN: 1210-2512.

[52] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal et al., „Generating exact lattices in the WFST framework,“ *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, břez. 2012. DOI: `10.1109/ICASSP.2012.6288848`.

[53] T. Ko, V. Peddinti, D. Povey a S. Khudanpur, „Audio augmentation for speech recognition,“ in *Proc. Interspeech 2015*, 2015, s. 3586–3589. DOI: `10.21437/Interspeech.2015-711`.

[54] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw et al., *The HTK Book*. pros. 2015.

[55] P. Pollák a V. Hanžl, „Tool for Czech Pronunciation Generation Combining Fixed Rules with Pronunciation Lexicon and Lexicon Management Tool,“ in *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain*, European Language Resources Association, srp. 2002.

[56] P. Pollák, *710 Speaker Czech Database from Car*, Final report of the project, ČVUT FEL, 2004.

[57] P. Pollák a J. Černocký, *Czech SPEECON adult database*, vol. 104, 2004.

[58] J. Rajnoha a P. Pollák, „Czech Spontaneous Speech Collection and Annotation: The database of Technical lectures," in *Lecture Notes in Artificial Intelligence*, sv. 5641, 2009, s. 377–385.

[59] A. Jirkovský, *Speech Recognition Based on Available Internet Modules*, Bakalářská práce, ČVUT FEL, Praha, 2021.

[60] *Institut Českého národního korpusu*, [Online; citováno květen 2024]. URL: `http://wiki.korpus.cz/doku.php?id=cnk:syn2006pub&rev=1661789279`.

[61] O. Plátek, *Rozpoznávání řeči pomocí KALDI*, Diplomová práce, Ústav formální a aplikované lingvistiky, Karlova Univerzita, 2014.

[62] A. Stolcke, J. Zheng, W. Wang a V. Abrash, „SRILM at sixteen: update and outlook," *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, led. 2011.

[63] B.-H. Juang a L. R. Rabiner, „Automatic speech recognition–a brief history of the technology development," *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, roč. 1, č. 67, s. 1, 2005.

[64] D. O'shaughnessy, *Speech communications: Human and machine (IEEE)*. Universities press, 2000.

[65] Y. Zhang, W. Han, J. Qin, Y. Wang a A. B. et al., „Google USM: Scaling Automatic Speech Recognition Beyond 100 Languages," *ArXiv*, roč. abs/2303.01037, 2023.

[66] M. Gales, „Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech & Language*, roč. 12, č. 2, s. 75–98, 1998, ISSN: 0885-2308. DOI: `https://doi.org/10.1006/csla.1998.0043`.

[67] M. J. F. Gales, „Semi-tied covariance matrices for hidden Markov models," *IEEE Trans. Speech Audio Process.*, roč. 7, s. 272–281, 1999.

[68] D. Povey, X. Zhang a S. Khudanpur, „Parallel training of Deep Neural Networks with Natural Gradient and Parameter Averaging," in *International Conference on Learning Representations*, říj. 2014.

[69] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey a S. Khudanpur, „X-Vectors: Robust DNN Embeddings for Speaker Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, s. 5329–5333. DOI: `10.1109/ICASSP.2018.8461375`.

[70] R. Kneser a H. Ney, „Improved backing-off for M-gram language modeling," *1995 International Conference on Acoustics, Speech, and Signal Processing*, roč. 1, 181–184 vol.1, 1995.