FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

**Diplomová práce**

# Segmentace řečových trénovacích dat pro učení systému ASR

*Studijní program:* N0613 – Informační technologie
*Studijní obor:* N0613A140028 – Inteligentní systémy
*Autor práce:* **Bc. Ondřej Vacek**
*Vedoucí práce:* prof. Ing. Jan Nouza, CSc.

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

## Zadání diplomové práce

# Segmentace řečových trénovacích dat pro učení systémů ASR

| | |
|---|---|
| *Jméno a příjmení:* | **Bc. Ondřej Vacek** |
| *Osobní číslo:* | M21000164 |
| *Studijní program:* | N0613A140028 Informační technologie |
| *Specializace:* | Inteligentní systémy |
| *Zadávající katedra:* | Ústav informačních technologií a elektroniky |
| *Akademický rok:* | 2022/2023 |

### Zásady pro vypracování:

1. Seznamte se s principy fungování a návrhu systémů automatického rozpoznávání řeči (ASR) a zejména se způsoby a požadavky na jejich trénování.
2. Navrhněte a implementujte nástroje, které budou schopné vytvářet řečová trénovací data z vhodných, veřejně dostupných zdrojů – zejména ze záznamů TVR pořadů, jednání institucí (např. parlamentu), videí na platformách jako je YouTube, audioknih, apod. Protože výše uvedené zdroje obsahují většinou dlouhé záznamy, vytvořené nástroje je musí v první řadě převést a rozdělit do zvukových souborů splňujících řadu požadavků (zejména ohledně délky, pozice řezu, odstranění delších neřečových zvuků, apod.). U dat, kde existují textové přepisy (např. titulky, stenografické záznamy, či např. e-knihy), musí být ke každému trénovacímu zvukovému souboru přiřazen i co nejlépe odpovídající text. Pro rozčlenění a přiřazení textu lze využít vlastní či existující nástroje založené na ASR.
3. Funkčnost vytvořených nástrojů ověřte a vyhodnoťte nejprve na českých datech a následně na dostupných datech dalších evropských jazyků.

| | |
|---|---|
| *Rozsah grafických prací:* | podle potřeby dokumentace |
| *Rozsah pracovní zprávy:* | 40-50 stran |
| *Forma zpracování práce:* | tištěná/elektronická |
| *Jazyk práce:* | Čeština |

**Seznam odborné literatury:**


[1] Dong Yu, Li Deng. Automatic Speech Recognition: A Deep Learning Approach. Springer. 2015. ISBN 978-1447157786

[2] Guoguo Chen, et al. GigaSpeech: An Evolving, Multi-domain ASR Corpus with 10,000 Hours of Transcribed Audio. arXiv e-prints, arXiv: 2106.06909, 2021.

[3] Patrick K. O'Neill, et al. Spgispeech: 5,000 hours of transcribed financial audio for fully formatted end-to-end speech recognition. *arXiv preprint arXiv:2104.02014*, 2021.

[4] Odborné články z konferencí Interspeech a ICASSP


| | |
|---|---|
| *Vedoucí práce:* | prof. Ing. Jan Nouza, CSc.<br>Ústav informačních technologií a elektroniky |
| *Datum zadání práce:* | 24. října 2022 |
| *Předpokládaný termín odevzdání:* | 22. května 2023 |

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 24. října 2022

# Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

21. 5. 2023 Bc. Ondřej Vacek

# Segmentace řečových trénovacích dat pro učení systému ASR

## Abstrakt

Tato diplomová práce se zabývá vývojem poměrně rozsáhlého systému pro vytěžování a přípravu trénovacích dat pro účely trénování systémů rozpoznávání řeči. Začíná přehledem historického vývoje automatických systémů rozpoznávání řeči od tradičních po E2E systémy, přičemž zdůrazňuje klíčovou roli dat v jejich tvorbě. Na základě potřeby velkého množství dat je navržen systém pro jejich automatickou těžbu, který se skládá ze tří částí: zpracování audia, zpracování textu a přiřazení textů k audio segmentům. Zpracování audia zahrnuje předzpracování, detekci řečové aktivity a následné rozdělení audia na segmenty obsahující řeč. Zpracování textu se věnuje úpravě textů podle požadavků uživatele a poskytuje k tomu potřebné nástroje. Přiřazení textu k audio segmentům zahrnuje rozpoznávání audio segmentů a přiřazení vhodných textů na základě jejich podobnosti. Navržený systém je experimentálně ověřen na jednoduchých českých a komplexních dánských datech, přičemž se podařilo vytěžit téměř 90 % jednoduchých českých dat a téměř 48 % komplexních dánských dat. Data vytěžená v rámci dánského experimentu byla následně použita při trénování nového modelu. Nakonec se tento nový model použil pro opětovné vytěžení dat, kde se ukázalo, že byl schopen vytěžit téměř o 6 % více dat než jeho předchůdce, a tudíž vytěžená data napomohla ke zlepšení modelu.

**Klíčová slova:** rozpoznávání řeči, těžba dat, zpracování audia, zpracování textu, trénovací data

# Abstract

This thesis deals with the development of a rather large-scale system for mining and preparing training data for the purpose of training speech recognition systems. It starts with an overview of the historical development of automatic speech recognition systems, from traditional to E2E systems, highlighting the key role of data in their design. Based on the need for large amounts of data, a system for automatic data mining is proposed, which consists of three parts: audio processing, text processing, and text matching to audio segments. Audio processing involves pre-processing, speech activity detection and subsequent segmentation of audio into segments containing speech. Text processing deals with the editing of texts according to the user's requirements and provides the necessary tools to do so. Text to audio segment matching involves recognizing audio segments and matching appropriate texts based on their similarity. The proposed system is experimentally validated on simple Czech and complex Danish data, and it has been able to extract almost 90 % of the simple Czech data and almost 48 % of the complex Danish data. The data extracted in the Danish experiment were subsequently used to train the new model. Finally, this new model was used to re-extract the data, where it was shown to be able to extract almost 6 % more data than its predecessor and thus the extracted data helped to improve the model.

**Keywords:** speech recognition, data mining, audio processing, text processing, training data

# Poděkování

Rád bych poděkoval svému vedoucímu práce prof. Ing. Janu Nouzovi, CSc. za poskytování věcných rad po celou dobu tvorby práce. Dále bych rád poděkoval celému týmu SpeechLab Liberec, který mi poskytl přístup k jejich systému pro rozpoznávání řeči, bez kterého bych nebyl schopný tuto práci dokončit. Speciální poděkování patří mé přítelkyni za konstantní podporu v čemkoliv, co si zamanu.

# Obsah

# Seznam zkratek

| | |
|---|---|
| AED | Attention Based Encoder-Decoder |
| ANN | Artificial neural network |
| ASR | Automatic speech recognition |
| CNN | Convolution neural network |
| CTC | Connectionist Temporal Classification |
| DCT | Discrete cosine transform |
| DTW | Dynamic time warping |
| DNN | Deep neural network |
| E2E | End to end |
| GMM | Gaussian Mixture Model |
| HMM | Hidden Markov model |
| LSTM | Long-short term memory |
| MHA | Multi-head attention |
| MED | Minimal edit distance |
| MFCC | Mel Frequency Cepstral Coefficients |
| NN | Neural network |
| PCM | Pulse-Code Modulation |
| RNN | Recurrent neural network |
| STFT | Short time Fourier transform |
| TUL | Technická univerzita v Liberci |
| VAD | Voice activity detection |
| WAcc | Word accuracy |

# Seznam obrázků



# Seznam tabulek

# Seznam kódů

# Úvod

Tato práce se zabývá automatickým získáváním trénovacích dat pro učení systémů automatického rozpoznávání řeči. Je zaměřena zejména na systémy typu end-to-end, které ke svému vývoji vyžadují stovky i tisíce hodin anotovaných řečových dat.

Její text je rozčleněn následovně:

V úvodu práce je seznámení s historickým vývojem těchto systémů od tradičních až po moderní E2E systémy. Následně zde vysvětluji, jak a proč jsou pro tyto systémy nesmírně důležitá data, a to především z pohledu množství a kvality.

V návaznosti na významnost dat přecházím k vlastnímu cíli této práce, jímž je systém pro automatické vytěžování dat. Zde nejprve představuji iterativní proces vytěžování a následného trénování modelů, čímž vysvětluji podstatu navrhovaného systému. Následně navrhuji, jak by takový systém automatického vytěžování mohl fungovat. Navržený systém rozdělím do 3 samostatných celků, a to zpracování audia, zpracování textu a přiřazení textu k audio segmentům.

V navazujících částech práce se již věnuji jednotlivým částem navrženého systému. Jako první detailně projdu část zpracování audia, kde se postupně věnuji předzpracování, detekci řečové aktivity a na konec i segmentaci audio záznamů. V rámci detekce řečové aktivity představuji několik zdarma dostupných detektorů, kde pomocí testování volím nejlepší z nich. Při řešení segmentace audia se věnuji návrhu a popisu algoritmu pro optimální segmentaci na základě stanovených kritérií a jeho optimalizaci.

Následně přejdu k části zabývající se zpracováním textu, kde představuji 2 základní typy úprav a rozdíl mezi nimi, a to modifikaci a korekci textu. Ke každé z nich zavádím pravidla, která uživatelům umožní jednoduše zpracovat text do požadované podoby.

Jako poslední část systému představuji přiřazení textů k audio segmentům. Zde se nejprve věnuji rozpoznávání audio segmentů pomocí již existujících ASR modelů vytvořených týmem SpeechLab TUL. Následně se zaměřuji na návrh a popis algoritmu pro zarovnání textů na základě jejich podobnosti. V závěru této části popisuji průchod celým procesem přiřazení včetně uvedení různých technik, které vedou k lepší výtěžnosti systému.

Po představení systému přejdu k experimentům, kde chci ověřit, jak si navržený systém vede, a to jak z pohledu výtěžnosti dat, tak i z pohledu časové náročnosti. V této části jsou provedeny dva experimenty. Jeden na českých datech, která reprezentují jednoduchý typ dat. Druhý na dánských datech, která představují náročnější

typ dat.

Data vytěžená v rámci dánského experimentu jsou použita týmem SpeechLab TUL pro natrénování nového modelu. Tento nově vzniklý model je následně použit v další iteraci vytěžování dat. Tímto ověřím, jestli iterativním přístupem dokáži vytěžit nějaká nová data.

# 1 Systémy automatického rozpoznávání řeči

Automatické rozpoznávání řeči (ASR) je technologie, která umožňuje počítačům rozumět lidské řeči a převádět ji do textového formátu. Základem rozpoznávání řeči je převod zvukového signálu na sekvenci příznakových vektorů reprezentujících velmi krátké úseky řeči (nazývané framy, trvající obvykle cca 25 ms) a následné přiřazení těchto framů k elementárním řečovým jednotkám (nejčastěji fonémům, u nových systémů znakům), z nichž pak modul zvaný dekodér sestavuje nejpravděpodobnější slova a věty.

Během let se ASR technologie výrazně vyvíjela. Proto se na začátku této práce seznámíme s tradičními systémy, které byly populární především v minulosti (přibližně do roku 2000), kdy ještě neuronové sítě nebyly schopny konkurovat zaběhlým pravděpodobnostním modelům. Následně se podíváme na to, jak se do těchto systémů postupem času začaly přidávat neuronové sítě, takzvané hybridní systémy. Nakonec přejdeme k současným systémům, které plně využívají potenciál neuronových sítí. Těmto systémům se říká End-to-End (E2E).

## 1.1 Tradiční systémy

Tradiční ASR systémy potřebují pro rozpoznávání řeči tři hlavní modely:

- `Akustický model` – jedná se o pravděpodobnostní model, který popisuje vztah mezi základními jednotkami řeči (zejména slovy a fonémy) a příznakovým prostorem. U klasických systémů je většinou založen na metodě skrytých Markovových modelů (HMM) a směsích gaussovských rozložení (GMM).
- `Jazykový model` – matematicky postihuje vztahy mezi slovy v daném jazyce, např. pomocí pravděpodobností slov vyskytujících se za sebou (tzv. N-gramy).
- `Výslovnostní model` – jedná se o slovník obsahující nejčastěji se vyskytující slova a slovní tvary v daném jazyce, k nimž jsou připojeny výslovnosti vyjádřené sekvencí fonémů.

Jedním z klíčových prvků tradičních systémů ASR jsou Markovské modely, konkrétně skryté Markovovy modely. Tyto modely byly široce používány pro modelování sekvencí pozorování, jako jsou jednotlivé zvuky řeči (fonémy). HMM se používaly

ke klasifikaci a sekvenčnímu rozpoznání zvuků a umožňovaly kombinovat pravděpodobnosti jednotlivých částí řečového signálu.

Gaussovy směsi jsou často používány v tradičních systémech ASR pro modelování akustických vlastností fonémů. GMM představují pravděpodobnostní model, který umožňuje kompaktně popsat rozdělení pravděpodobnosti vícerozměrných dat.

V rámci vývoje tradičního ASR systému musíme nejprve vytvořit slovník, který obvykle obsahuje stovky tisíc nejčastěji se vyskytujících slov a slovních tvarů. Ke každému musí být vygenerována nebo manuálně vytvořena výslovnost (v případě některých slov i různé výslovnostní varianty). Dále je třeba shromáždit co největší množství (řádově gigabajty) textů v daném jazyce a pomocí nich vypočítat parametry jazykového modelu. Nejpracnější částí ale bývá vytvoření akustického modelu, k jehož trénování potřebujeme desítky hodin záznamů řeči pocházející od co největšího počtu osob. Tyto trénovací nahrávky musí být doprovázeny velmi přesným fonetickým přepisem, jehož příprava představuje jedno z nekritičtějších míst vývoje. K vlastnímu trénování se pak používají již hotové nástroje, jako je např. HTK [1] či Kaldi [2].

Avšak i pro natrénování relativně malého systému je zapotřebí mnoho práce. Potřebujeme mít trénovací data (a znalosti) pro všechny tři modely, které se vyvíjejí a ladí separátně. Má to své výhody, ale nelze zaručit, že individuálně optimalizované modely povedou k celkově optimálnímu systém [3].

## 1.2 Hybridní systémy

S postupem času se vývoj v oblasti ASR systémů začal posouvat směrem k hybridním řešením, které kombinují prvky tradičních ASR systémů s modernějšími technikami založenými na neuronových sítích, což vede k značně lepším výsledkům v rozpoznávání řeči.

Hybridní ASR systémy stále používají akustický, jazykový a výslovnostní model, ale s významným rozdílem ve způsobu, jakým jsou tyto modely zpracovávány a propojovány. Umělé neuronové sítě jsou v těchto systémech použity k modelování akustických a jazykových vlastností řeči, což umožňuje lépe zachytit komplexní vzory a závislosti mezi jednotlivými prvky.

V hybridních systémech se GMM často nahrazují hlubokými neuronovými sítěmi (DNN), které se ukázaly jako účinnější v modelování akustických vlastností řeči. DNN se dokážou naučit složité nelineární závislosti mezi různými částmi řečového signálu, což vede k přesnějšímu rozpoznávání. Jazykové modely založené na neuronových sítích, například rekurentních neuronových sítích (RNN), pak umožňují zachytit delší kontext a strukturu vět, čímž zlepšují predikci slov.

I přes to, že modernější E2E systémy dosahují lepších výsledků z pohledu přesnosti, tak jsou hybridní systémy stále velmi rozšířené v komerční sféře. Důvodem jsou dekády optimalizace různých faktorů, jako je adaptabilita, latence a integrace, které jsou nezbytné pro produkční nasazení [4].

## 1.3 End-to-End systémy

Nejnovějším trendem v oblasti rozpoznávání řeči jsou E2E systémy, které se od tradičních či hybridních systémů liší tím, že nevyužívají akustický, jazykový a výslovnostní model. Místo toho přímo mapují akustický vstup (tj. sekvenci příznakových vektorů) na sekvenci výstupních znaků, což teoreticky umožňuje rozpoznávání i neznámých slov. Neboli tři dřívější modely byly nahrazeny jedním modelem, optimalizovaným s cílem generování očekávané sekvence znaků na základě audio dat.

E2E systémy využívají různé typy neuronových sítí, jako rekurentní neuronové sítě (RNN), konvoluční neuronové sítě (CNN) nebo transformery, které se učí zachytit komplexní vzory a závislosti mezi jednotlivými prvky vstupního signálu. Sekvenční tréninkové metody, jako CTC (Connectionist Temporal Classification) nebo AED (Attention Based Encoder-Decoder), se často používají pro učení správného zarovnání mezi vstupními a výstupními sekvencemi. Při dekódování využívají E2E systémy algoritmy jako Beam Search pro nalezení nejpravděpodobnějšího výstupního textu.

Jednou z hlavních výzev při implementaci E2E systémů je zajištění dostatečného množství kvalitních trénovacích dat. Tyto systémy potřebují řádově více dat než starší přístupy, aby dosáhly stejné úrovně rozpoznávání. Nicméně získávání trénovacích dat je značně zjednodušeno, jelikož E2E systémy vyžadují pouze dvojice audio a text. Tento proces lze do jisté míry velmi dobře automatizovat, což je hlavním předmětem této práce [4].

# 2 Trénovací data pro E2E systémy

Jak již bylo zmíněno v předchozí kapitole, pro trénování E2E systému jsou potřeba audiozáznamy řeči a jim odpovídající textové přepisy (anotace). Je důležité zdůraznit, že textová část dat obsahuje pouze text bez časových značek pro jednotlivé části, které by systému mohly usnadnit trénování. Díky tomu je formát dat, který potřebujeme pro trénink velmi prostý, což zjednodušuje přípravu těchto dat.

Jelikož trénovací data neobsahují žádné pomocné informace, musíme trénovacímu procesu pomoci jinak. Za prvé dodáním řádově vyššího množství dat než u hybridních/klasických systémů, které byly představeny výše. Za druhé dělením trénovacích dat do relativně krátkých úseků, což usnadňuje učení souvislostí mezi částí vstupního signálu a odpovídajícím výstupním znakem. Intuitivně si lze představit, že trénování na dlouhém nepřetržitém kusu audia s přiřazeným textem bez časových značek by bylo značně obtížnější.

Další významnou výhodou dělení trénovacích dat na kratší úseky, která je minimálně stejně důležitá, je možnost paralelního zpracování. Tato schopnost je zcela zásadní pro práci s tak obrovským množstvím dat, jaké vyžadují E2E systémy.

Je důležité zmínit také opačný případ, kterým je rozdělení trénovacích dat na příliš krátké úseky, což by vedlo ke ztrátě zásadní informace, kterou je kontext, ve kterém se data nachází. Ten je opět nesmírně důležitý, protože E2E systémy nemají separátní jazykový model, který by určoval pravděpodobnosti výskytu jednotlivých slov ve vztahu k již určeným slovům. Naopak tyto komplexní vztahy se systém učí sám během tréninkového procesu.

Trénovací data musíme tedy dělit tak, aby byla dostatečně dobře paralelizovatelná, ale zároveň jsme zachovali co největší množství kontextu, který modelu pomáhá s vyprodukováním správné sekvence znaků.

Nyní, když jsme si ukázali formát dat, který potřebujeme pro trénování E2E ASR systémů, přichází otázka, kde získat tato data s co nejmenším úsilím a s ohledem na konkrétní aplikaci. Jinými slovy, nejde nám pouze o množství dat, ale také o jejich obsah a akustické vlastnosti.

Pokud budeme stavět systém pro automatický přepis jednání parlamentu, bude výhodné začít sběrem dat právě z konkrétního parlamentu. Tento přístup bude mít za následek, že i s menším množstvím dat bude náš systém lépe rozpoznávat řeč, jelikož je velká šance, že právě ti lidé, na kterých se systém učil, budou ti stejní, které bude muset rozpoznávat. Stejně tak akustické prostředí, ve kterém bude systém trénován, bude shodné s tím, ve kterém bude muset fungovat.

Rovněž nás zajímá i obsahová stránka dat, která zajistí dostatečnou „slovní zásobu“ pro danou aplikaci. Pokud si opět vezmeme příklad parlamentu, bude lepší, když systém bude trénován na datech, která tématicky zapadají do dění ve veřejném prostoru, než na romantických e-knihách.

Pro efektivní trénování ASR systému je potřeba, aby trénovací data byla relevantní a reprezentativní pro danou doménu, což nám umožní dosáhnout lepších výsledků i při nižším množství dat.

Existuje několik možností, jak získat data pro trénování ASR systémů. Některé z nich jsou komerční, zatímco jiné jsou open-source nebo založené na veřejně dostupných zdrojích. Níže jsou uvedeny některé z těchto možností spolu s jejich výhodami a nevýhodami:

- `Vlastní tvorba dat` – Samozřejmě si můžeme data vytvořit sami a to například tím, že napíšeme krátké texty a poté nahrajeme jejich přečtení. Hlavní výhodou takto vytvořených dat je, že máme absolutní kontrolu nad jejich kvalitou a jsou také okamžitě připravena k trénování. Na druhou stranu nevýhodou může být časová náročnost a nutnost sehnat dostatečné množství lidí, kteří pro nás přečtou připravené texty.

- `Open-source zdroje` – Open-source databáze, jako například CommonVoice nebo LibriSpeech jsou skvělým zdrojem zdarma dostupných dat. Každopádně kvalita dat může být různorodá a někdy méně přesná než u komerčních poskytovatelů. Stejně tak dostupnost dat pro některé jazyky a domény může být omezená.

- `Zakoupení dat` – Data lze zakoupit od komerčních subjektů, které se zaměřují na sběr a distribuci dat. Jedním z takových poskytovatelů je například čínská firma SpeechOcean. Výhodou zakoupení dat je jejich rychlá dostupnost, obvykle vysoká kvalita a výběr dat pro konkretní doménu. Nevýhodou je finanční náročnost a malá nabídka pro jazyky s malou poptávkou. Pro rozšířené jazyky jako je angličtina, španělština nebo čínština, je těchto dat obrovské množství. Naopak pro jazyky, které nejsou tak rozšířené jako je například dánština, je těchto dat velmi málo.

- `Veřejné orgány/média s povinností zveřejňovat přepis` – Ve vyspělých zemích bývá zákonnou povinností zveřejňovat určitý typ dat z veřejného prostoru. Typickým příkladem těchto dat jsou záznamy jednaní parlamentu. Ty se musí zveřejňovat v podobě video/audio záznamu spolu se zápisem jednání. Výhodou těchto zdrojů je jejich neustálá tvorba a nulová cena. Nevýhodou je, že se data musí konvertovat do tvaru, který požaduje E2E systémem a shoda mezi audiem a přidaným textem není 100%.

- `Audioknihy` – Audioknihy, ke kterým existuje i e-kniha jsou dalším velmi užitečným zdrojem trénovacích dat. Některé jsou na internetu dostupné zdarma, jiné jsou za mírný poplatek. Jejich výhodou je velká shoda mezi textem a audiem spolu s čistým audio záznamem. Nevýhodou je opět potřeba převedení dat

do formátu, který vyžaduje E2E systém a také to, že jedna kniha nepokrývá velkou škálu témat a má velmi málo mluvčích, typicky pouze jednoho.

- `Ostatní veřejně dostupné zdroje` – Ostatní veřejné zdroje, jako například YouTube, seriály, filmy a jejich titulky, poskytují téměř nekonečnou zásobou lehce dostupných dat, které jsou tématicky rozmanité a diverzita mluvčích je obrovská. Získání dostatečného množství kvalitních trénovacích dat z těchto zdrojů je však velmi obtížné, jelikož jsou často nekvalitní jak z pohledu audia, tak i přesnosti odpovídajícího textu. Navíc zde nesmíme opomíjet autorská práva.

- `Využití jiného ASR systému` – Při získávání dat pro trénování vlastního ASR systémů můžeme využít již existující rozpoznávač, který může v podstatě k libovolnému audiu vygenerovat jeho přepis. Tato možnost se na první pohled může zdát geniální, jelikož můžeme vzít nejlepší dostupný rozpoznávač a vygenerovat si obrovské množství trénovacích dat, čímž bychom v podstatě „svůj“ nový systém dostali za velmi krátkou dobu na úroveň konkurence. Do určité míry je to pravda, ale musíme počítat s tím, že takto vygenerovaná tréninková data budou obsahovat nedostatky rozpoznávače, který se snažíme využít. A jelikož pro natrénování kvalitního systému potřebujeme obrovské množství dat, typicky přes 1000 hodin, tak následné nalezení a eliminace chyb v trénovacích datech může být téměř nemožná. Nicméně použití již existujících ASR jakožto pomocníků pro získaní nových dat je velmi běžné a nezbytné pro automatické vytěžování trénovacích dat.

Při trénování ASR systému jsou data základním stavebním kamenem celého procesu. Vzhledem k tomu, že potřebujeme opravdu velké množství trénovacích dat, je důležité minimalizovat čas strávený jejich získáváním a přípravou. Při trénování nového ASR systému se tedy snažíme, co nejjednodušší cestou získat, co největší množství dat a postupně, malými kroky, náš systém iterativně zlepšovat. S každým vylepšením můžeme náš systém opětovně použít pro získání více trénovacích dat, než jsme získali s využitím předchozí verze.

Tento přístup může být na první pohled matoucí, neboť jsme uvedli, že kvalitní ASR systémy bývají trénovány na 1000 hodinách nahrávek, zatímco my bychom chtěli použít náš systém k podpoře vytěžování dat již při 10 hodinách. Důvod, proč je toto možné, spočívá v tom, že vztah mezi množstvím dat a úspěšností rozpoznání u těchto systémů je logaritmický. To znamená, že každých 10 hodin dat na začátku přispěje k velkému zlepšení výkonu, zatímco přidání 10 hodin do systému, který je již trénován na 1000 hodinách, přinese velmi malý posun.

Závěrem lze říci, že klíčem k úspěchu ASR systémů je iterativní proces trénování a vytěžování dat, který umožňuje postupné zlepšování systému prostřednictvím zpracování nových trénovacích dat. Po několika iteracích trénování a extrahování dat by měl být ASR systém natolik pokročilý, že bude schopen rozpoznávat řeč s vysokou přesností a minimální potřebou manuálních korekcí. Důležitým aspektem je pečlivé zvolení zdrojů trénovacích dat a zajištění jejich kvality, aby byl ASR systém schopen správně rozpoznávat různorodé řečové situace a přizpůsobit se různým dialektům, akcentům a stylům mluvy.

# 3 Návrh systému pro automatické vytěžování dat

Nyní, když máme základní představu o potřebných datech, můžeme se podívat na iterativní proces trénování E2E ASR systému a jeho využití pro automatické vytěžování trénovacích dat.

Na úplném začátku je vhodné využít volně dostupné datasety, případně i nějaké zakoupit. Tato data jsou obvykle již připravená ve formátu, který potřebujeme, nebo jejich konverze bude velmi jednoduchá. S těmito daty, kterých mohou být i pouze nižší desítky hodin, natrénujeme model, který i přes malé množství dat již bude mít určitou úspěšnost rozpoznávání, například 30 %. Jakmile máme alespoň základní model, můžeme začít využívat další zdroje. Mezi tyto zdroje patří například audioknihy nebo záznamy z parlamentů, které ale vyžadují extrakci užitečné informace, k čemuž využijeme právě natrénovaný model.

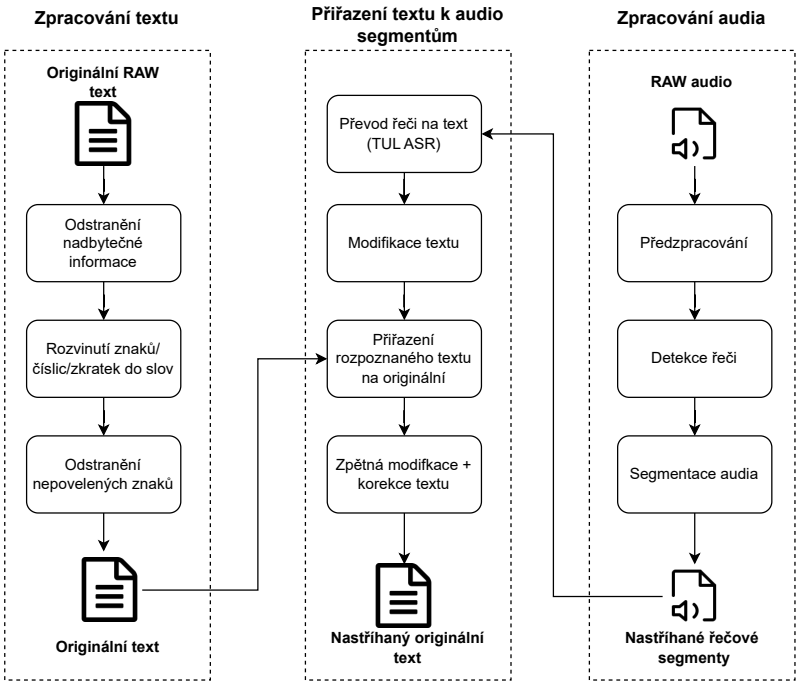
Na počátku, kdy je náš model ještě nedokonalý, budeme vytěžovat malé množství trénovacích dat a bude nutná řada manuálních korekcí, které zajistí jejich vysokou kvalitu. Vyextrahovaná data přidáme do trénovacích a model znovu natrénujeme, čímž se zlepší jeho rozpoznatelnost. Tento proces opakujeme.

S rostoucí kvalitou rozpoznávače se snižuje doba potřebná pro korekci dat. Přesto budeme potřebovat korekce i nadále, abychom modelu pomohli s oblastmi, kde bude slabší. Takovými oblastmi mohou být například složité číslovky, cizí slova a jména, slang nebo prostě jen slova, která nebyla v trénovacích datech dostatečně zastoupena.

V uvedeném iteračním procesu, mluvíme o vytěžení užitečné informace z dat. Jak ale takový proces extrakce vypadá? Na obrázku níže (3.1) je ukázán vývojový diagram navrženého procesu automatického vytěžování dat. Hlavním cílem tohoto procesu je z audia a k němu poskytnutému textu získat krátké řečové úseky a k nim přiřadit odpovídající text. Neboli automaticky vytěžit nová data ve formátu odpovídajícímu požadavkům na trénování. Navržený proces se skládá z 3 hlavních částí, které jsou popsaný níže.

## 3.1 Zpracování audia

Cílem této sekce je rozdělit vstupní audio signál na sadu krátkých audio segmentů obsahujících řeč, tak aby řeč zaujímala co největší podíl jednotlivých segmentů.




Obrázek 3.1: Vývojový diagram systému

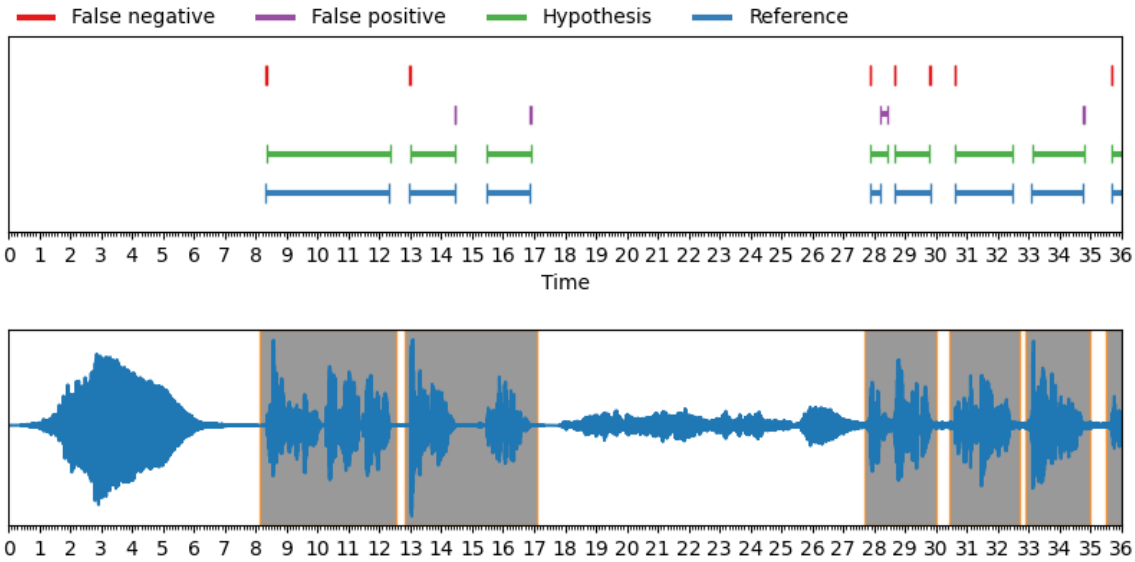
Nejprve je třeba předzpracovat audio do formátu, který nám ulehčí další zpracování a zároveň bude vyhovovat vstupním požadavkům E2E systému. Tento krok zahrnuje převzorkování signálu, konverzi do požadovaného audio formátu a případně i odstranění příliš dlouhých tichých částí, které jsou typicky způsobeny vypnutím mikrofonu při zapnutém nahrávání.

Takto předzpracovaný signál se předá do dalšího bloku, který má za úkol detekovat řečovou aktivitu. Pro zajištění co nejlepší detekce řeči se zde používá neuronová síť (NN), která je natrénovaná na rozpoznávání dvou typů signálu „řeč“ a „ne-řeč“. Řeč by měla být vyhodnocena pouze v případě, že se skutečně jedná o řeč a ne o jiný zvuk, kterým může být například hluk stroje, zvířat a nebo hudba. Podrobný popis detekce řečového signálu naleznete v kapitole 4.2.

Jako poslední krok se na základě detekované řeči určí místa řezu pro vytvoření řečových segmentů. Tyto segmenty musí splňovat určité požadavky jako je minimální a maximální délka segmentu, minimální doba ne-řeči na hranici střihu a nebo maximální doba neřečového signálu mezi dvěma řečovými úseky. Kromě těchto základních podmínek se také určuje cílová délka, ke které by se měly jednotlivé segmenty co nejvíce přiblížit. Detailní popis procesu audio segmentace lze nalézt v kapitole 4.3.

Na níže uvedeném obrázku (3.2) je ukázka výsledku zpracování audia. V horní části obrázku je srovnání výstupu detektoru řeči (Hypothesis) s ručně anotovanými řečovými částmi (Reference). Součástí této části je také znázornění v jakých částech detektor špatně detekoval řeč (False positive) a v jakých částech špatně detekoval ne-řeč (False negative). Zde můžeme pozorovat, že detekovaná řeč téměř dokonale

koresponduje s referencí. Ve spodní části obrázku pak vidíme samotný audio záznam s vyznačenými řečovými segmenty, které představují výstup tohoto procesu. Zde si můžeme všimnout, že některé části audia, kde je zřetelná aktivita, nebyly určeny jako řeč, což je správně, jelikož na začátku audia jsou slyšet bubny a činely, zatímco v prostřední části se objevuje hudba se zpěvem.




Obrázek 3.2: Ukázka detekce řeči a následného určení řečových segmentů

## 3.2 Zpracování textu

Cílem této sekce je připravit vstupní textová data do formátu, který je vhodný pro pozdější trénování. Nejprve se z textu odstraňují nechtěné části, tedy ty, u kterých víme, že nebyly v audiu řečeny. Typicky se jedná o poznámky pod čarou a strukturní data. Následně se text začne upravovat, do formátu, který je shodný s daty na kterých je ASR systém trénován. V tomto kroku se například rozepíší zkratky, číslovky a speciální znaky do jejich mluvené podoby. Nakonec se z textu odstraní nepovolené znaky, jako je interpunkce nebo jiné speciální symboly. V tabulce níže (3.1) je názorná ukázka zpracování textu. Podrobný popis této části je v kapitole 5.

Tabulka 3.1: Ukázka zpracování textu

| **Původní tvar** | **Převedený tvar** |
|---|---|
| Jel jsi pouze na 50 %. (Důrazně) | jel jsi pouze na padesát procent |
| Byl tam Tomáš, Ondra atd. | byl tam tomáš ondra a tak dále |
| Dnes je 20.1.2023. | dnes je dvacátého ledna dva tisíce dvacet tři |

## 3.3 Přiřazení textu k audio segmentům

Cílem této sekce je přiřadit originální text k audio segmentům. Nejprve pomocí rozpoznávače zjistíme jaký je obsah jednotlivých audio segmentů. Následně pomocí přepisu a algoritmu pro přiřazení, na základě minimální editační vzdálenosti, nalezneme co nejlepší shodu s originálním textem. Po aplikování tohoto procesu získáme data v požadovaném formátu, tedy krátké audio soubory a jejich obsah. Veškerý určený obsah přesně neodpovídá tomu, co se v audiu skutečně říká. Ve skutečnosti získáme, v závislosti na obtížnosti těženého zdroje a pokročilosti rozpoznávače, pouze určitou část 100% shodných dat. Úroveň shody se počítá jako podobnost referenčního textu s textem z rozpoznávače. Pro začlenění do trénování použijeme data, jejichž shoda bude větší než určená hranice, přičemž data, která nemají 100% shodu, je nutné manuálně upravit.

V níže uvedené tabulce (3.2) je zobrazen rozdílů mezi tím, co určil rozpoznávač, co bylo přiřazeno z originálního textu a co bylo skutečně řečeno v audio souboru. Tato ukázka slouží k demonstraci chybovosti rozpoznávače a nesrovnalostí mezi referenčním textem a skutečným obsahem audia.

Je důležité zmínit, že úplná shoda mezi referenčním textem a rozpoznaným textem nezaručuje absolutní správnost vzhledem k audiu. Pracujeme ale s předpokladem, že pravděpodobnost, že rozpoznávač udělá naprosto stejné chyby jako jsou v originálním textu, je velmi malá.

Dále je také podstatné uvést, že ASR systémy od SpeechLab Tul, které se využívají v této práci, jsou již značně pokročilé a tudíž budou mít obrovský vliv na množství vytěžených dat. Podrobný popis této části naleznete v kapitole 6.

Tabulka 3.2: Ukázka přiřazení textového obsahu k audio záznamu

| | | | | |
|---|---|---|---|---|
| **Skutečný obsah** | Toto | jest | testovací | text |
| **Dodaný přepis** | Tady | je | testovací | text |
| **Rozpoznaný obsah** | Toto | je | testovací | text |

# 4 Zpracování audia

## 4.1 Předzpracování

Předzpracování audio signálu je nezbytné pro kompatibilitu se vstupními požadavky dalších použitých systémů, zejména ASR systém a detektor řečové aktivity (VAD). Kromě kompatibility je tento krok také užitečný pro ulehčení dalšího zpracování z pohledu výpočetní náročnosti a udržení jednotného formátu trénovacích dat.

Vstupní audio soubory tedy nejprve převedeme do následujícího formátu.

- `Vzorkovací frekvence 16 kHz` – Vzorkovací frekvence 16kHz je naprosto dostatečná pro práci s lidskou řečí. Důvodem je, že člověk běžně sděluje informace pomocí řeči s frekvencí do 8 kHz a tudíž kvůli dodržení vzorkovacího teorému je minimální dostatečná vzorkovací frekvence právě zmíněných 16 kHz [5]. Zároveň s touto vzorkovací frekvenci pracuje ASR systém a VAD, který budeme později používat.
- `Monofonní zvuk` – Použitý ASR systém i VAD využívají pouze jednokanálovou informaci, tudíž není důvod pro nadbytečné držení dat, které nijak nevyužijeme.
- `WAV formát` – WAV je audio formát, který se používá pro ukládání nekomprimovaného jedno nebo vícekanálového zvuku kódovaného pomocí pulzně kódové modulace (PCM) [6]. Důvodem pro tento formát je zachování originálního nezkresleného signálu.

K provedení těchto kroků můžeme využít FFmpeg, což je zdarma dostupný nástroj pro zpracování multimediálních souborů. FFmpeg umožňuje převzorkování, konverzi formátů a další operace s audiem či videem. Následující ukázka (4.1) demonstruje, jak s použitím FFmpeg převést MP3 soubor na WAV soubor s 32 bitovým rozlišením se vzorkovací frekvencí 16kHz a monofonním zvukem [7]:

```
ffmpeg -i input.mp3 -acodec pcm_s32le -ac 1 -ar 16000 output.wav
```

Kód 4.1: Použití nástroje FFmpeg pro předzpracování signálu

Kromě výše zmíněného předzpracování má pro naše účely smysl i odstranění dlouhých tichých úseků, u kterých si můžeme být jisti, že neobsahují žádnou zajímavou informaci. K detekci těchto částí se používá energeticky založený detektor aktivity, který je násobně rychlejší než detektor založený na neuronových sítích. A tudíž nám ušetři čas v následujícím bloku detekce řeči.

Je důležité mít detekční hranici nastavenou opravdu nízko a dobu, po kterou musí být signál pod ní, rozumně dlouhou, aby se předešlo riziku odstranění řečových částí ze signálu. Tato metoda se typicky používá pro odstranění částí, kde byl mikrofon ztlumen nebo úplně vypnut, ale nahrávání zůstalo zapnuté. Toto se často vyskytuje například v parlamentu během dlouhých jednání, kde se na hodinovou pauzu nechává zapnuté nahrávání.

## 4.2 Detekce řečové aktivity

Detekce řeči, známá také jako Voice Activity Detection (VAD), je proces rozpoznávání a rozlišování řečových a neřečových zvuků v daném zvukovém signálu.

### 4.2.1 Metody hodnocení detekce řeči

Než se podíváme na konkrétní detektory řeči a jejich funkce, tak je vhodné si uvést, podle čeho budeme rozhodovat, jak dobrý je konkrétní VAD a jaké typy parametrů budeme sledovat.

Jako první si uvedeme čtyři základní typy klasifikace, které může detektor vyprodukovat a co znamenají.

- `True positive (TP)` – Správně rozpoznaný řečový úsek.
- `True negative (TN)` – Správně rozpoznaný neřečový úsek
- `False positive (FP)` – Nesprávně rozpoznaný jako řečový úsek, který je ve skutečnosti neřečový
- `False negative (FN)` – Nesprávně rozpoznaný jako neřečový úsek, který je ve skutečnosti řečový.

Při hodnocení VAD pracujeme s časem neboli úseky, které spadají do jedné z výše zmíněných kategorií. Jelikož signál, se kterým pracujeme, má 16 tisíc vzorků v jedné sekundě, tak pro nás nemá smysl rozlišovat rozdíl mezi hranicí reference a rozpoznaní v rámci pár stovek vzorků. Z toho důvodu se u hodnocení VAD často zavádí toleranční oblast, která určuje, jak moc se může lišit rozpoznání od reference z pohledu času. V našem případě budeme používat toleranci 100 ms, neboli 50 ms na levé straně a 50 ms na pravé straně [8].

**Přesnost (Accuracy)** představuje celkovou úspěšnost algoritmu ve smyslu správného rozpoznání řečových a neřečových úseků. Tento parametr ukazuje, jak často je detekce správně rozpoznána, tedy poměr správně klasifikovaných úseků (TP a TN) ku celkovému počtu úseků [8].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \cdot 100[\%] \tag{4.1}$$

**Preciznost (Precision)** ukazuje, jak dobře algoritmus rozlišuje řečové úseky od neřečových úseků, když řečový úsek detekuje. Tento parametr vyjadřuje, jak často je řečový úsek správně rozpoznán jako řečový, tedy poměr TP ku součtu TP a FP [8].

$$Precision = \frac{TP}{TP + FP} \cdot 100[\%] \tag{4.2}$$

**Úplnost (Recall)** zobrazuje, jak dobře algoritmus detekuje řečové úseky jako řeč, aniž by označoval neřečové úseky jako řeč. Tento parametr vyjadřuje, jaký podíl skutečných řečových úseků byl správně rozpoznán, tedy poměr TP ku součtu TP a FN [8].

$$Recall = \frac{TP}{TP + FN} \cdot 100[\%] \tag{4.3}$$

**F1 skóre** poskytuje hodnocení, které zohledňuje jak preciznost, tak úplnost. Tento parametr je harmonickým průměrem preciznosti a úplnosti, což je číslo, které vyjadřuje úspěšnost detekce řeči. F1 skóre představuje dobrý kompromis mezi precizností a úplností, zejména v případě, že jejich hodnoty jsou velmi rozdílné [9].

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{4.4}$$

Při hodnocení algoritmu detekce řeči je důležité brát v úvahu všechny tyto metriky, protože žádná z nich sama o sobě neposkytuje úplný obraz o kvalitě detekce. Například vysoká preciznost může být dosažena na úkor úplnosti, což znamená, že algoritmus má tendenci rozpoznávat řečové úseky, ale může přehlédnout některé skutečné řečové úseky. Naopak vysoká úplnost může být dosažena na úkor přesnosti, což znamená, že algoritmus detekuje většinu řečových úseků, ale může také nesprávně zahrnovat některé neřečové úseky jako řečové. Neboli když o všem bude tvrdit, že je řeč, tak jeho úplnost bude 100 %. F1 skóre zohledňuje obě tyto metriky a poskytuje lepší představu o celkové úspěšnosti algoritmu detekce řeči.

Všechny výše zmíněné metriky patří mezi obecné a lze je aplikovat na libovolnou úlohu binární klasifikace. Nyní si uvedeme metriky, které jsou specifické pro detekci řeči. Tyto metriky nám později pomohou správně nastavit parametry pro VAD.

**Front end clipping (FEC)** hodnotí, jak dobře algoritmus detekuje začátek řečového úseku. FEC nastává, když VAD nesprávně detekuje začátek řečového úseku později, než ve skutečnosti začíná. Vyšší hodnota znamená horší detekci začátku řečových úseků. FEC se počítá jako poměr mezi dobou chybné klasifikace ne-řeči při přechodu z neřečové aktivity do řečové a celkovou dobou trvání řeči v referenčních datech [10].

$$FEC = \frac{T_{FEC}}{T_{speech}} \cdot 100[\%] \tag{4.5}$$

**Mid speech clipping (MSC)** hodnotí, jak dobře algoritmus detekuje řeč během řečového úseku. MSC nastává, když VAD nesprávně přeruší řečový úsek a označí část řeči jako ne-řeč. Vyšší hodnota znamená horší detekci řeči během řečových úseků.

MSC se počítá jako poměr mezi dobou chybné klasifikace ne-řeči uprostřed řečového úseku a celkovou dobou trvání řeči v referenčních datech [10].

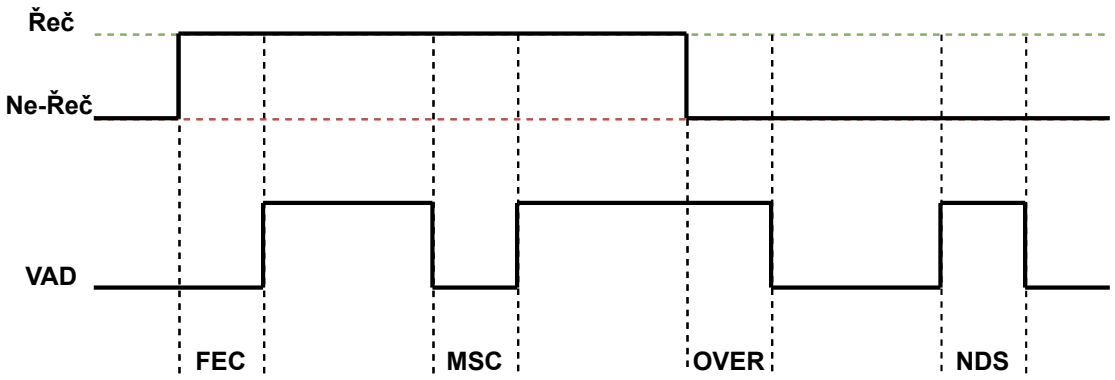
$$MSC = \frac{T_{MSC}}{T_{speech}} \cdot 100[\%] \tag{4.6}$$

**Over hang (OVER)** hodnotí, jak dobře algoritmus detekuje konec řečového úseku. OVER nastává, když VAD nesprávně detekuje konec řečového úseku později, než ve skutečnosti končí. Vyšší hodnota znamená horší detekci konce řečových úseků. OVER se počítá jako poměr mezi dobou chybné klasifikace řeči při přechodu z řečové aktivity do neřečové a celkovou dobou trvání ne-řeči v referenčních datech [10].

$$OVER = \frac{T_{OVER}}{T_{nonspeech}} \cdot 100[\%] \tag{4.7}$$

**Noise detected as speech (NDS)** hodnotí, jak často je ne-řeč nesprávně rozpoznán jako řečový úsek. Vyšší hodnota této metriky znamená horší detekci neřečových úseků a větší pravděpodobnost FP. NDS se počítá jako poměr mezi dobou chybné klasifikace řeči v době neřečového úseku a celkovou dobrou trvání ne-řeči v referenčních datech [10].

$$NDS = \frac{T_{NDS}}{T_{nonspeech}} \cdot 100[\%] \tag{4.8}$$




Obrázek 4.1: Ukázka metrik specifických pro VAD

K výše uvedeným metrikám můžeme také vytvořit čtveřici inverzních metrik. Tedy například v případě FEC můžeme vytvořit metriku Early hang (EARLY), která bude měřit dobu nesprávné klasifikace řeči, kdy by měla být ne-řeč v době přechodu z ne-řečové do řečové aktivity atd.

### 4.2.2 Detekce řečové aktivity pomocí měření energie signálu

V této části se zaměříme na fungování jednoduchého detektoru řeči založeného na měření energie signálu. Pro takovýto detektor máme využití v již zmíněném

předzpracování audia, kde ho kvůli úspoře času používáme pro odstranění dlouhých tichých segmentů. Další využití tohoto detektoru nalezneme později, kdy ho budeme chtít použít pro zjemnění výstupu z detektoru řeči založeném na neuronových sítích.

Pro proměnlivý signál, jako je řeč, je nutné energii signálu počítat po krátkých časově omezených úsecích zvaných framy (rámce). Krátkodobou energii signálu $E_n$ vypočítáme pomocí následujícího vztahu:

$$E_n = \log \sum_{m=n-N+1}^{n} x(n-m)^2 \quad n = 0, 1T, 2T, ... \tag{4.9}$$

Symbol N značí délku framu, T je časový posun mezi jednotlivými framy a x je vektor obsahující časově proměnný signál, v našem případě řeč [11].

Můžeme si všimnout, že součástí vzorce je také logaritmus. Důvodem pro jeho použití je komprimace rozsahu výstupních hodnot, což nám pomůže při dalším zpracování. Jelikož logaritmus není definován pro nulu, je vhodné před logaritmizací přičíst malou hodnotu, aby byl algoritmus prakticky použitelný.

Pro efektivní využití vypočítané energie signálu při detekci řečové aktivity je třeba energii normalizovat do rozsahu 0 až 1. To uživatelům usnadní nastavování rozhodovacího prahu. Pro normalizaci signálu použijeme následující vzorec:

$$\widetilde{E}_n = \frac{E_n - \bar{E}}{2\sigma(E)} + 0,5 \tag{4.10}$$

Symbol $\bar{E}$ představuje průměr a $\sigma(E)$ značí směrodatnou odchylku vypočtené energie E.

Detekce řeči probíhá na základě nastavení horního a spodního prahu. Když energie signálu překročí horní práh, následující framy jsou označeny jako řeč. Tento proces pokračuje, dokud energie signálu neklesne pod spodní práh. Poté jsou následující framy označeny jako ne-řeč a to až do momentu, kdy energie signálu opětovně překročí horní práh. Důvodem dvou prahů je zavedení hystereze, která omezuje rapidní změny mezi detekovanými stavy.

### 4.2.3 Detekce řečové aktivity pomocí neuronových sítí

V této části se zaměříme na obecné fungování detektoru řeči založeném na neuronových sítí. Především si zde představíme typy sítí, které se k této úloze používají a jaké reprezentace dat můžeme použít.

V kontextu detekce řečové aktivity se neuronové sítě učí rozpoznávat vzory, které jsou charakteristické pro řečové a neřečové segmenty. Toho lze dosáhnout různými typy neuronových sítí, jako jsou hluboké neuronové sítě (DNN), konvoluční neuronové sítě (CNN) a rekurentní neuronové sítě (RNN).

Hluboké neuronové sítě jsou založeny na vícevrstvých perceptronech, které mají vysoký počet skrytých vrstev. Tyto sítě jsou schopné efektivně modelovat komplexní nelineární vztahy, což jim umožňuje efektivně rozpoznávat závislosti rozlišující řečový a neřečový signál.

Konvoluční neuronové sítě jsou vhodné pro detekci řeči, neboť efektivně rozpoznávají lokální vzory v datech, jako jsou například spektrogramy. V případě zpracování zvukových signálů se CNN zaměřují na extrakci různých frekvenčních a časových vlastností, které jsou charakteristické pro řečové a neřečové segmenty.

Rekurentní neuronové sítě mají schopnost zpracovávat sekvenční data, což je pro detekci řeči výhodné, jelikož řeč je časově závislý signál. RNN disponují paměťovými buňkami, které umožňují uchovávat informace z předchozích časových kroků. Tato vlastnost je klíčová pro modelování kontextu a závislostí v řečových signálech.

Kombinace různých typů neuronových sítí do jedné architektury může vést ke zlepšení výkonu detekce řečové aktivity. Například můžeme použít konvoluční vrstvy pro extrakci lokálních vlastností ze spektrogramů, následované rekurentními vrstvami pro zpracování časových závislostí a zakončené hlubokou neuronovou sítí pro finální klasifikaci mezi řeči a ne-řečí.

Při použití neuronových sítí pro detekci řečové aktivity je klíčovým faktorem zvolení vhodné reprezentace vstupních dat. Obvykle se používají spektrogramy a mel-frekvenční kepstrální koeficienty (MFCC). Tyto reprezentace zachycují různé aspekty řečového signálu, které jsou užitečné pro detekci řečové aktivity.

Spektrogram je vizuální reprezentace frekvenčního spektra zvuku, který ukazuje, jak se intenzita různých frekvencí mění v průběhu času. Vytvoří se pomocí krátkodobé Fourierovy transformace (STFT), která rozděluje signál na malé časové úseky a analyzuje frekvenční složky každého z nich. Výsledný spektrogram má konstantní rozlišení frekvencí, což neodpovídá způsobu, jakým lidé zpracovávají zvuk. Z toho důvodu byla zavedena mel-frekvenční škála, která na rozdíl od lineárního rozložení frekvencí lépe odpovídá lidskému vnímání. Mel-spektrogram se tak zaměřuje na frekvence, které jsou pro lidské ucho důležitější, což může být užitečné při detekci řečové aktivity [12].

Mel-frekvenční kepstrální koeficienty (MFCC) jsou kompaktní reprezentace zvukových signálů založené na mel-frekvenční škále. MFCC se získávají pomocí několika kroků, včetně aplikace mel-frekvenčního filtru na spektrogram, logaritmovaní energie jednotlivých frekvencí a diskrétní kosinové transformace (DCT). Tato reprezentace vytváří malý soubor koeficientů, které účinně zachycují hlavní charakteristiky řečového signálu, což je důležité pro detekci řečové aktivity [3].

Každá z těchto reprezentací zachycuje různé aspekty řečového signálu, které mohou být užitečné pro detekci řečové aktivity. Spektrogramy a mel-spektrogramy poskytují zobrazení frekvenčních složek zvuku v čase, zatímco MFCC nabízí kompaktnější a efektivnější reprezentaci. Pro zlepšení výkonu neuronových sítí je vhodné kombinovat tyto a nebo i jiné reprezentace.

Tvorba detektoru řeči založeném na NN pak probíhá tak, že se navrhne architektura sítě, která používá výše zmíněné typy sítí. Výstupem takové sítě bude jedna

hodnota reprezentující pravděpodobnost výskytu řeči v daném framu. Součástí návrhu architektury bude také formát vstupních dat, tedy například využijeme mel-spektrum a MFCC jako vstupní příznaky.

Následně se začne síť učit na trénovacích datech, ke kterým musíme mít anotační data, která určují jaký úsek odpovídá řeči a jaký ne-řeči. Tato data se následně zpracují do formátu, který požaduje navržená architektura. Při samotném tréninku se pak upravují váhy jednotlivých částí NN s cílem minimalizovat chybu mezi skutečnými a predikovanými hodnotami řečové aktivity. Optimalizace se provádí na základě metrik probíraných v sekci 4.2.1.

Po úspěšném natrénování sítě je její použití velmi jednoduché. Stačí upravit vstupní audio do požadovaného formátu a na výstupu sítě získáme pravděpodobnost, která určuje jestli se v daném framu vyskytuje řeč.

### 4.2.4 Představení dostupných systémů detekce řeči

V této sekci si představíme veřejně dostupné modely pro detekci řečové aktivity. Konkrétně si zde popíšeme pouze dva modely ze čtyř vyzkoušených a to Speechbrain VAD a Pyannote VAD. Zbylé dva modely Silero VAD a MarbleNet VAD zde z důvodu zkrácení práce zmíníme pouze okrajově

**SpeechBrain VAD** je předtrénovaný model pro detekci řečové aktivity vyvinut týmem stojícím za balíčkem nástrojů SpeechBrain. Tento výzkumný tým se specializuje na vývoj open-source nástrojů pro práci s řečí založených na PyTorch [13].

Tento model využívá CRDNN (Convolutional, Recurrent, Deep Neural Network) architekturu. Konkrétně se jedná o kompaktní model, který začíná dvěma konvolučními vrstvami. První konvoluční vrstva má 16 kanálů, zatímco druhá má 32 kanálů. Tyto konvoluční vrstvy jsou následovány dvěma vrstvami obousměrné GRU rekurentní neuronové sítě (RNN), každá s 32 neurony. Po RNN vrstvách následuje malá DNN, s hloubkou 2 a 16 neurony v každé vrstvě. Její výstupní vrstva pak má pouze jeden neuron, který signalizuje pravděpodobnost výskytu řeči pro jednotlivé framy.

Vstupní data modelu jsou reprezentována pomocí mel-frekvenčního spektra, což je technika popsána v předchozí kapitole. Pro tento model je nezbytné, aby vstupní zvuková data měla vzorkovací frekvenci 16 kHz.

Při výpočtu mel-frekvenčního spektra jsou data rozdělena do rámců. Každý rámec má šířku 25 ms a mezi jednotlivými rámci je posun o 10 ms. Příznaky získané z jednotlivých rámců jsou reprezentovány pomocí 40 mel-frekvenčních bank. Tyto banky fungují jako filtry, které transformují spektrum každého rámce do sady příznaků, které model používá pro učení a predikci [14].

**Pyannote VAD** je předtrénovaný model pro detekci řečové aktivity, který byl vyvinut výzkumným týmem stojícím za sadou nástrojů Pyannote. Tento výzkumný tým se specializuje na vývoj open-source nástrojů pro audio diarizaci [15].

Tento model využívá architekturu sítě zvanou PyanNet [15], která je založena na obecné CRDNN architektuře. Mimořádně zajímavý je přístup, jakým PyanNet zachází s vstupními daty. Na rozdíl od běžných modelů, které používají předzpraco-

vaná data ve formátu jako například MFCC, PyanNet pracuje přímo s nezpracovaným zvukem. V případě modelu pro detekci řeči je délka vstupního audio segmentu nastavena na 5 sekund a předpokládaná vzorkovací frekvence audia je 16 kHz.

První vrstvou v síti je speciální konvoluční neuronová síť zvaná SincNet. Tato síť je navržena tak, aby se přímo učila filtry schopné zachytit základní frekvenční charakteristiky audio signálů. SincNet je speciálně trénován na parametrizaci sinc funkcí, které implementují pásmové propusti. Na rozdíl od běžné CNN se tato síť snaží určit pouze vhodné mezní frekvence pro jednotlivé filtry. Tyto filtry se navíc učí přímo na trénovacích datech odpovídajících doméně využití výsledného modelu, takže se přizpůsobí dané aplikaci [16].

Po konvoluční síti SincNet následuje RNN složená ze čtyř obousměrných LSTM vrstev, kde každá obsahuje 128 neuronů. Síť je zakončena dvouvrstvovou DNN s 128 neurony v každé vrstvě a jedním výstupním neuronem, který určuje pravděpodobnost výskytu řeči [17].

**MarbleNet VAD** je předtrénovaný model pro detekci řečové aktivity od společnosti Nvidia. Model je založený na architektuře s názvem MarbleNet, kterou společnost Nvidia vyvinula speciálně pro účely detekce řečové aktivity.

Hlavní charakteristikou této sítě je opakované využití 1D konvoluce, což ve výsledku vede k menšímu počtu parametrů sítě. Důležité je také poznamenat, že na vstupu očekává 64 MFCC příznaků. Bližší specifikace a popis fungování dané architektury jsou uvedeny v příslušném článku [18].

**Silero VAD** je předtrénovaný model pro detekci řeči od společnosti Silero AI, která se zabývá vývojem produktů v oblasti zpracování řeči a počítačového vidění.

Jejich cílem při vývoji daného modelu bylo vytvoření produktu připraveného pro streamování, který bude zachovávat slušnou přesnost. Navržený model je využívá neuronovou síť založenou na multi-head attention (MHA) mechanismu a výstup krátkodobé Fourierovy transofmrace (STFT) jako vstupní parametry signálu.

Zajímavý je také jejich přístup k trénovacím datům, kde místo časových úseků rozlišujících řeč a ne-řeč využívají sadu krátkých nahrávek o průměrné délce 7 sekund, kde celé nahrávky jsou označeny buď jako řeč a nebo jako ne-řeč, podle toho jestli nahrávka obsahuje alespoň nějakou řeč. Bližší specifikace je opět možné dohledat v příslušné literatuře [19] [20].

### 4.2.5 Výběr detektoru

Po představení uvažovaných VAD modelů musíme zjistit, který se nejlépe hodí pro naší aplikaci. Z tohoto důvodu si připravíme testovací data, která budou typově odpovídat datům, která budeme těžit. Pro anotaci dat využijeme open-source nástroj LabelStudio, který nám v uživatelsky příjemném prostředí umožní k daným časovým úsekům testovacích dat přiřadit odpovídající značku. Tato data si následně vyexportujeme do formátu JSON a použijeme jako referenci při vyhodnocení jednotlivých modelů [21].

Pro testování si pomocí uvedeného programu připravíme přibližně 20 minut záznamu v češtině a 20 minut záznamu v dánštině. Česká data se skládají z audioknihy, reportáže a online pořadu. Dánska data obsahují úryvky jednání z dánského parlamentu za rok 2020-2021.

V rámci testování byla pro každý detektor určena aktivační a deaktivační hranice na základě manuální optimalizace na datech z dánského parlamentu. Pro správné zvolení těchto parametrů jsme využili metriky specifické pro VAD, jako jsou FEC, MSC, OVER a NDS (viz. kapitola 4.2.1). Například, velká hodnota FEC pravděpodobně znamená, že aktivační hranice je nastavena příliš vysoko a měli bychom ji trochu snížit. Na druhou stranu velká hodnota OVER znamená, že deaktivační hranice je pravděpodobně nastavena příliš nízko a tudíž bychom ji měli trochu zvýšit.

Zároveň je důležité zmínit, že vzhledem k naší aplikaci preferujeme chybu typu označení ne-řeči za řeč než opačně. To by totiž vedlo k nevyžádanému poškození řečového signálu, čímž bychom narušili celý proces vytěžení dat, zejména pak části rozpoznávání audia a následnému přiřazení vůči originálnímu textu.

Tabulka 4.1: Výsledky testování detektorů řeči

| **Parlament (DK)** | **Přesnost** [%] | **Preciznost** [%] | **Úplnost** [%] | **F1** [-] |
|---|---|---|---|---|
| BrainSpeech VAD | **93,95** | **92,86** | 98,39 | **95,54** |
| Pyannote VAD | 88,52 | 86,47 | 97,89 | 91,83 |
| MarbleNet | 89,74 | 87,27 | 98,95 | 92,70 |
| Silero VAD | 88,64 | 85,71 | **99,32** | 92,02 |
| **Audiokniha (CZ)** | | | | |
| BrainSpeech VAD | 87,17 | **98,93** | 96,12 | **97,51** |
| Pyannote VAD | 85,64 | 82,17 | 95,92 | 88,51 |
| MarbleNet | **93,14** | 90,14 | 98,52 | 94,31 |
| Silero VAD | 92,10 | 88,85 | **98,68** | 93,51 |
| **Reportáž (CZ)** | | | | |
| BrainSpeech VAD | 89,22 | **89,78** | 97,00 | 93,25 |
| Pyannote VAD | 86,71 | 85,66 | 99,30 | 91,98 |
| MarbleNet | 86,72 | 85,64 | **99,35** | 91,99 |
| Silero VAD | **90,51** | 89,68 | 99,05 | **94,13** |
| **Pořad (CZ)** | | | | |
| BrainSpeech VAD | 87,17 | **98,93** | 96,12 | **97,51** |
| Pyannote VAD | 85,64 | 82,17 | 95,92 | 88,51 |
| MarbleNet | **93,14** | 90,14 | 98,52 | 94,31 |
| Silero VAD | 92,10 | 88,85 | **98,68** | 93,51 |

Tabulka 4.1 obsahuje výsledky testování pro zmíněné detektory řeči. Výsledky jsou rozděleny do 4 sekcí, kde každá sekce zobrazuje výsledky pro jinou skupinu

testovacích dat. Důvodem pro rozdělení testování do těchto celků je možnost porovnání jednotlivých detektorů v různých akustických prostředích. Audiokniha a pořad mají akusticky nejlepší prostředí, kde se téměř nevyskytují nechtěné hluky. Naopak parlament a reportáž jsou pořizovaný v běžném prostředí, kde se v pozadí nahrávek vyskytují nechtěné zvuky. Například reportáž je točena v automobilu za jízdy, kde je v pozadí puštěná hudba.

Při volbě nejlepšího detektoru se budeme řídit především F1 skórem, které jsme si představili v kapitole 4.2.1. Tato metrika zahrnuje jak preciznost, tak i úplnost a tudíž je ideálním ukazatelem pro objektivní hodnocení. Z tabulky vidíme, že BrainSpeech VAD měl nejlepší výsledek F1 skóre ve 3 ze 4 sekcí. To pro nás znamená, že je jasným kandidátem na zvolený detektor. Nicméně předtím než ho jím prohlásíme, musíme zhodnotit ještě jednu metriku - dobu zpracování.

Tabulka 4.2: Porovnání rychlosti zpracování dat jednotlivých detektorů řeči

| | **CPU** [s] | **CPU**× [-] | **GPU** [s] | **GPU**× [-] |
|---|---|---|---|---|
| BrainSpeech VAD | **25,75** | **92,16** | 40,77 | 58,20 |
| Pyannote VAD | 213,78 | 11,10 | **11,03** | **215,14** |
| MarbleNet | 184,00 | 12,90 | 318,31 | 7,45 |
| Silero VAD | 78,90 | 30,08 | 20,83 | 113,92 |

V tabulce 4.2 je srovnána výkonost jednotlivých detektorů řeči. Test byl prováděn na počítači s procesorem AMD Ryzen 9 5900X a grafickou kartou NVIDIA GeForce RTX 3060. Sloupce CPU a GPU obsahují konkrétní dobu zpracování všech testovacích dat pomocí daného detektoru. Sloupce CPU× a GPU× určují poměr mezi skutečnou délkou trénovacích dat a dobou jejich zpracování pomocí daného detektoru. Jinými slovy, nám určují kolikanásobně rychleji dokážeme zpracovat audio vzhledem k jeho délce.

Z tabulky vidíme, že v případě zpracování pomocí CPU je jednoznačně nejrychlejší BrainSpeech VAD. Na druhou stranu při použití GPU je jednoznačně nejrychlejší Pyannote VAD. Zajímavé je také to, že výkonost BrainSpeech A MarbleNet značně klesla při použiti GPU.

Z výkonnostního porovnání jsme chtěli zjistit především to, jestli daný detektor není příliš pomalý pro použití na velkém množství dat. Jak je vidět z tabulky výše, zvolený BrainSpeech VAD si se svým zrychlením 100x vede velmi dobře. Z těchto důvodů použijeme BrainSpeech VAD pro finální vytěžení dat popsané v kapitole 7. Důležité je zmínit, že daného zrychlení jsme schopni dosáhnout na každém jádře procesoru, takže ve finále budeme moci zpracovávat data ještě rychleji.

### 4.2.6 Konkrétní implementace

Tato sekce se věnuje podrobnému popisu procesu detekce řeči. Pro detekci řeči jsme si zvolili model od SpeechBrain. Níže popsaný postup platí, až na drobné odchylky, i při použití ostatních modelů. Pro uživatele znamená přechod na jiný detektor

z výše představených pouze změnu adaptéru při definování VAD. Dále v popisu předpokládáme, že audio již prošlo předzpracováním a má požadovanou vzorkovací frekvenci, která je pro všechny modely 16 kHz.

Proces detekce řeči začíná bezprostřední aplikací modelu na vstupní audio data. Výstupem je pravděpodobnost výskytu řeči pro jednotlivé rámce. Tuto pravděpodobnost, která se pohybuje v rozmezí 0 až 1, kvantizujeme do jedné ze dvou kategorií: řeč (1) a ne-řeč (0). Pro určení kategorie využíváme dvou prahů stanovených uživatelem, a to aktivačního a deaktivačního prahu. Určení kategorie probíhá následovně: Rámce jsou zařazovány do kategorie 0 až do momentu, kdy je překročen aktivační práh. Poté jsou řazeny do kategorie 1. Zpět do kategorie 0 jsou zařazovány v případě, že pravděpodobnost výskytu řeči klesne pod deaktivační práh. V kategorii 0 zůstávají až do dalšího překročení aktivačního prahu.

Dalším krokem je aplikace energeticky založeného detektoru řeči na úseky, kde detekujeme řeč. Důvodem pro jeho použití je rozmělnění detekované řeči tak, abychom minimalizovali výskyt dlouhých řečových úseků, které nejsou vhodné pro další zpracování. Proces funguje tedy tak, že se na úsecích s detekovanou řečí aplikuje energeticky založený detektor (viz. kapitola 4.2.2), kde se na základě energie signálu opět rozděluje do jedné ze dvou skupin. Uživatel si zde může nastavit jiné hodnoty aktivačního a deaktivačního prahu než v předešlé detekci. Je důležité poznamenat, že tento krok je nepovinný a je na uživateli, zda se rozhodne pro jeho použití.

Nakonec sloučíme řečové segmenty, které jsou příliš blízko u sebe, a odstraníme ty, které jsou příliš krátké. Je vhodné nastavit tyto hodnoty s ohledem na další použití. V našem případě budeme u segmentace řeči požadovat, aby segment začínal a končil 200 ms ticha, a tedy můžeme spojit segmenty, které mezi sebou mají mezeru kratší než tato hodnota.

Obvykle by tímto celý proces detekce řeči končil, avšak jak již bylo zmíněno, detekce řeči je pouze částí celkového procesu zpracování audia. To znamená, že by do detekce řeči měl být zahrnut další parametr, konkrétně maximální povolená délka segmentu. Tento parametr je odvozen z požadavků na segmentaci, které budou prezentovány později v kapitole 4.3.1. V kontextu detekce řeči to znamená, že bychom ideálně chtěli, aby všechny řečové úseky byly kratší než je maximální povolená délka segmentu.

Pro dosažení tohoto cíle zavedeme rekurzivní volání detekce řeči na úsecích, které tuto podmínku nesplňuji. Rekurzivní volání postupně zvyšují aktivační i deaktivační práh detektoru, čímž zajišťuji neustále zpřísňovaní detektoru a to až do hodnot prahu 0.99. Pokud by detektor stále tvrdil, že se jedná o řeč, která je delší než je povoleno, tak tuto situaci prostě akceptujeme. Při segmentaci audia se však tato část nebude moci použít.

Na závěr vygenerujeme podrobnou zprávu o detekci řeči (viz. 4.2). Tato zpráva obsahuje informace o nastavení detektoru, použitém audiu a samotném procesu detekce. Součástí zprávy je také základní statistika detekce řeči, která zahrnuje například počet detekovaných úseků, jejich minimální, maximální a průměrnou délku. Nezbytnou součástí je také seznam všech detekovaných řečových segmentů s infor-

macemi o jejich začátku, konci a délce.

```
"vad_type": "BrainSpeech",
"execution_time": 0.99,
"configuration": {
  "activation_th": 0.5,
  "deactivation_th": 0.4,
  "min_duration_on": 0.2,
  "min_duration_off": 0.2,
  "max_segment_duration": 24.0,
  "apply_energy_vad": true,
  "en_activation_th": 0.4,
  "en_deactivation_th": 0,
},
"audio": {
  "file": "D:\\andele_a_demoni_01.wav",
  "duration": 98.59,
  "uem": { "start": 0.0, "end": 98.59 }
},
"speech": {
  "count": 31.0,
  "durations": {
    "total": 55.13,
    "min": 0.51,
    "avg": 1.78,
    "max": 4.24,
    "std": 0.99
  },
  "segments": [
    { "segment": { "start": 8.35, "end": 12.34 }, "duration": 3.99 },
```

Kód 4.2: Ukázka generované zprávy o výsledku detekce řeči

## 4.3 Segmentace audio dat

Nyní když pomocí detekce řečové aktivity víme v jakých částech audia je řeč, tak můžeme přejít k poslední částí zpracování audia a tím je jeho segmentace. Cílem segmentace je nastříhat audio soubory tak, aby co nejlépe odpovídali vstupním požadavků, které si specifikujeme níže.

### 4.3.1 Požadavky na segmentaci

Segmentace podléhá vstupním požadavkům, které jsou určeny uživatelem. Obecně jsou požadavky stanoveny tak, že délka segmentu musí být v rozsahu 2 s až 25 s,

střihy musí být mimo řeč, na začátku a konci segmentu by měly být neřečové části a segment by měl obsahovat co nejméně neřečového signálu. Tyto požadavky jsou níže shrnuty do parametrů, kterými budeme nastavovat segmentační algoritmus. U každého parametru je i jeho značení, které budeme používat při vysvětlování algoritmu.

- `Cílová délka segmentu` ($t_d$) – Určuje délku segmentu, ke kterému by si měl střižený segment co nejvíce blížit. Vůči tomuto parametru se počítá skóre při hledání nejlepšího rozložení segmentů.
- `Minimální délka segmentu` ($min_d$) – Žádný segment, nemůže být kratší než tato délka. V našem případě je stanovený pevný limit 2 s, který je určený TUL ASR systémem.
- `Maximální délka segmentu` ($max_d$) – Žádný segment, nemůže být delší než tato délka. V našem případě je stanovený pevný limit 25 s, který je určený TUL ASR systémem.
- `Maximální délka ne-řeči` ($max_{ns}$) – Určuje maximální délku mezi jednotlivými řečovými úseky v jednom segmentu. Tento parametr zajišťuje to, že pokud budou existovat dva krátké řečové signály, které budou mít mezi sebou příliš ne-řeči, tak se vytvoří dva krátké segmenty obsahující převážně řeč než jeden segment, který by z větší části obsahoval ne-řeč a to i na úkor skóre segmentace.
- `Minimální délka přechodové ne-řeči` ($min_{tr}$) – Určuje minimální délku neřečového signálu na hranicích segmentu. Tento parametr pomáhá kompenzovat případné chyby detektoru řeči, které by mohly způsobit odříznutí malých částí řeči na začátku a konci segmentu.

### 4.3.2 Dynamické programování

Ještě než se pustíme na představení segmentačního algoritmu, tak si musíme udělat rychlou odbočku k dynamickému programování, jelikož navržený segmentační algoritmus je na něm založený. Kromě segmentace se nám dynamické programování bude hodit i později při přiřazení textu k audio segmentům.

Dynamické programování je optimalizační technika používaná v informatice a matematice, která se zaměřuje na řešení složitých problémů rozdělením do jednodušších, přehlednějších podproblémů, jejichž optimální řešení lze použít při řešení původní úlohy (ne všechny úlohy toto splňují). Tyto podproblémy se řeší pouze jednou a jejich řešení se ukládá (tzv. memoizace), aby se předešlo opakovanému výpočtu, při řešení složitějších podproblémů. Tato technika vede ke značným úsporám z pohledu výpočetních zdrojů [22].

### 4.3.3 Algoritmus pro nalezení optimálních míst řezu za využití dynamického programování

Po krátkém představení konceptu dynamického programování přišel čas na představení algoritmu pro nalezení optimálních míst řezu. Pro zjednodušení si uvedeme zobecněný popis úlohy, kterou potřebujeme řešit.

Mějme pole $X$ s $N$ kladnými čísly. Toto pole chceme rozdělit do sady polí, které mohou obsahovat pouze souvislé prvky z pole $X$ a to tak, aby suma prvků každého pole byla v rozmezí 2-25 s. Zároveň chceme takové rozdělení, kde rozdíl mezi cílovou a skutečnou sumou všech polí je minimální.

V takovém případě nalezení optimálního řešení pomocí hrubé síly bude mít složitost $O(n!)$. To je náročnost, která už při stovkách prvcích je i pro moderní počítače téměř neřešitelná. Z tohoto důvodu se buď musíme smířit se sub-optimálním řešení založeném například na jednocestném průchodu s výbornou časovou složitostí $O(n)$ a nebo přijít se sofistikovanějším řešením.

Jelikož by byla škoda kvůli tomuto kroku přijít i třeba o 0,5% konečné výtěžnosti, tak se uchýlíme k variantě sofistikovanějšího řešení. To založíme na dynamickém programování, konkrétně se inspirujeme algoritmem [23], který řeší rozložení pole na M částí s vyváženým součtem. Níže je detailně popsaný postup navrženého algoritmu.

Vstupem do segmentačního algoritmu je seřazené pole řečových úseků $X$ o délce $N$, které se vyznačují počátečním a koncovým časem. Na základě tohoto pole si vytvoříme 2D matici $D$ o rozměrech $N \times N$. Tato matice bude obsahovat skóre pro každou podmnožinu obsahující $i$ až $j$tý řečový segment z $X$. Jelikož musí platit $j \geq i$, tak se omezíme pouze na počítání nad diagonálou. Skóre vypočítáme dle následující rovnice (4.11), která říká, že pokud vybraná podmnožina nesplňuje kritérium na minimální délku $min_d$, maximální délku $max_d$ a nebo na maximální délku ne-řeči $max_{ns}$, tak její skóre bude $\infty$ a tedy daná podmnožina nemůže tvořit segment. V opačném případě, kdy podmnožina splňuje kritéria, tak se její skóre vypočítá jako kvadrát rozdílu její délky $d$ a cílové délky $t_d$.

$$D[i,j] = \begin{cases} \infty & if \quad d < min_d \\ \infty & if \quad d > max_d \\ \infty & if \quad ns > max_{ns} \\ (d - t_d)^2 \end{cases} \tag{4.11}$$

kde $d$ označuje délku podmnožiny $X[i-j]$, která je zvětšena o $2 \cdot min_{tr}$. Dále $ns$ označuje dobu neřečového úseku mezi jednotlivými řečovými úseky v dané podmnožině $X[i-j]$.

Nyní když máme vypočítané skóre pro každou možnou souvislou kombinaci řečových úseků, tak můžeme nalézt jejich optimální rozdělení. To získáme z matice $Q$ s rozměry $N \times N$, kterou spočítáme dle vztahu 4.12. Tato matice obsahuje minimální skóre posledních $i$ prvků z $X$ rozdělených do $j$ segmentů. V této matici nikdy nemůže být $i > j$ a tudíž pracujeme pouze s částí pod diagonálou. Spočtením hodnot matice $Q$ získáme řešení naší úlohy, které se nachází na posledním řádku.

Ten obsahuje skóre rozdělení všech řečových úseků do 1 až N segmentů. Sloupec, ve kterém se nachází nejmenší hodnota, určuje na kolik segmentů musíme rozdělit řečové úseky pro dosažení optimálního rozdělení. Jak konkrétně se mají vstupní data rozdělit, zjistíme zpětným průchodem matice $Q$.

$$Q[i,j] = \begin{cases} D[N+1-i,N] & if \quad j=1 \\ min(Q[i-1,j-1] + D[N+1-i, N+1-i] \\ \quad Q[i-2,j-1] + D[N+1-i, N+2-i] \\ \quad \vdots \\ \quad Q[j-1,j-1] + D[N+1-i, N+1-j]) \end{cases} \tag{4.12}$$

Z tohoto vzorce je patrné, že když rozdělujeme $i$ posledních prvků do jedné skupiny ($j = 1$), tak nejlepší skóre které můžeme získat odpovídá přímo vypočítanému skóre $D[N+1-i, N]$. Jinými slovy poslední sloupec matice D odpovídá prvnímu sloupci matice Q, ale v opačném pořadí. Dále vidíme, že pro nalezení nejlepšího rozdělení posledních $i$ prvků do $j$ skupin používáme výsledky z jednodušších úloh, kde rozdělujeme $i-1$ až $j-1$ prvků do $j-1$ skupin. K těmto výsledkům přičítáme skóre skupiny vytvořené spojením zbývajících prvků od $N+1-i$ do $N+1-j$. Nejlepší rozdělení v daném bodě je potom to s nejmenším výsledkem.

Pro nalezení přesného rozdělení řečových úseků je třeba určit kroky, které vedly k optimálnímu výsledku - tzv. zpětný průchod. Toho dosáhneme tím, že při výpočtu matice $Q$ vytvoříme pomocnou matici $I$ o stejných rozměrech, která bude uchovávat informaci o tom, který z možných členů byl vybrán jako nejmenší - jinými slovy argument funkce $min(...)$ z výše zmíněného vzorce. Tato hodnota nám říká, kolik řečových úseků z $X$ bylo použito na vytvoření j-tého rozdělení.

Konkretně to tedy funguje tak, že hledání započneme na řádku $i = N$ a sloupci $j = argmin(Q[N, :])$, který odpovídá optimálnímu rozdělení. Této kombinaci řádku a sloupce bude odpovídat nějaké číslo $y$ z $I$, které značí, že prvních $y$ prvků z $X$ tvoří první segmentu. Prvky použité na sestavení segmentu odebereme z $X$ a současně se o tento počet prvků posuneme v řádcích, tedy $i = i - y$. Zároveň se za vytvořený segment posuneme o jeden sloupec, tedy $j = j - 1$. Tímto způsobem se dostaneme na novou pozici a tento proces opakujeme, dokud $j > 0$. Jakmile dosáhneme tohoto bodu, tak vytvoříme poslední segment obsahující všechny zbývající prvky z $X$.

Časová náročnost výše popsaném algoritmu je $O(N^2)$ pro výpočet matice $D$ a $O(N^3)$ pro výpočet matice $Q$. To je podstatně lepší než náročnost pomocí metody hrubé síly $O(N!)$, avšak stále se jedná o velmi náročný algoritmus. Časovou náročnost je dále možné zmírnit správnou implementací, kde využijeme vlastnost matice $Q$, která říká, že každý její prvek závisí na prvcích z předchozího sloupce. Neboli hodnoty ve sloupcích matice Q můžeme zpracovávat současně.

### 4.3.4 Konkrétní implementace

V této části si představíme implementační nadstavby, které usnadňují praktické použití výše zmíněného algoritmu. Především z pohledu předzpracování vstupních

dat a výpočetní náročnosti.

Před tím než aplikujeme algoritmus segmentace na řečové úseky, tak musíme dostat vstupní data do formátu, kdy je zaručeno, že algoritmus bude schopný data rozřezat tak, aby splnit všechny uživatelské požadavky. Toho docílíme pomocí následujících 4 kroků:

1. Spojíme řečové úseky, mezi kterými nepůjde provést řez tj. úseky mez kterými je mezera menší než minimální délka přechodové ne-řeči $min_{tr}$.

2. Odstraníme příliš dlouhé úseků tj. úseky, které mají délku větší než maximální délka segmentu $max_d$.

3. Odstraníme příliš krátké úseků, které nebudeme moct přiřadit do segmentu. Neboli ty, které vedle sebe nemají dostatečně ne-řeči, abychom je mohli rozšířit na alespoň na minimální délku segmentu $mid_d$ a zároveň jsou obklopeny úseky, kde by při spojení došlo k porušení buď maximální délky segmentu $max_d$ a nebo maximální délky ne-řeči $max_{ns}$.

4. Rozšíříme příliš krátké úseky, které nelze přiřadit do většího celku, ale mají vedle sebe dostatek ne-řeči, aby jejich výsledná délka včetně přechodů byla alespoň $min_d$.

Ještě předtím než aplikujeme segmentační algoritmus, tak si představíme další úpravu, která nám pomůže výrazně urychlit časovou náročnost. Uváděli jsem si, že časová náročnost algoritmu je $O(n^3)$, což znamená, že s počtem řečových úseků nám exponenciálně stoupá náročnost a pokud budeme mít za úkol segmentovat víc jak 100000 řečových úseků, tak to bude i na moderním hardwaru chvíli trvat. Pokud by se nám ale například podařilo rozdělit, těchto 100000 úseků na 100 částí o 1000 úsecích, tak by časová náročnost dané úlohy klesla 10000 krát. Což je přesně to, co můžeme udělat.

Vstupní data rozdělíme na části tak, abychom měli zaručeno, že neporušíme nalezení optimálního řešení. Data můžeme tedy rozdělit pouze v případě, že sousední úseky nemohou tvořit segment tj. mezera mezi nimi je větší než maximální délka ne-řeči $max_{ns}$. Uživatel tedy může správným nastavením ovlivnit rychlost algoritmu.

Na takto rozdělené části již aplikujeme výše popsaný algoritmus a výsledky jednotlivých částí následně spojíme dohromady abychom získali optimální rozdělení pro celá vstupní data. V samotném algoritmu je ještě poslední vylepšení a to již zmíněné zpracování všech prvků v sloupci matice $Q$ zároveň, které provedeme vektorizací algoritmu viz. ukázka kódu (4.3) pro výpočet matice $Q$ .

```
Q = np.matrix(np.ones((N, N)) * np.inf)
I = np.zeros((N, N))

# Q(i,0) = D(N-i,N-1)
for i in range(N):
```

```
        Q[i, 0] = D[N - 1 - i, N - 1]

    # Q(i,j) = MIN{Q(i-1,j-1)+D(N-i,N-i);
    #              Q(i-2,j-1)+D(N-i,N-i+1);
    #              ...;
    #              Q(j-1,j-1)+D(N-i,N-j)}
    for j in range(1, N):
        for i in range(j, N):
            lookup_count = i - j + 1
            lookup_i = np.arange(0, lookup_count)[:, np.newaxis]
            lookup = Q[i - 1 - lookup_i, j - 1] + D[N - 1 - i, N - 1 - i +
            ↪  lookup_i]

            min_arg = np.argmin(lookup)
            I[i, j] = lookup_i[min_arg] + 1
            Q[i, j] = lookup[min_arg]
```

Kód 4.3: Ukázka vektorizovaného výpočtu matice Q

Na závěr vygenerujeme podrobnou zprávu o segmentaci zvukového záznamu. Tato zpráva obsahuje informace o nastavení segmentace, o audiu, na kterém byla segmentace provedena, a o detekci řeči, která byla použita jako vstupní data. Zpráva poskytuje detailní údaje o segmentaci, jako je skóre segmentace, počet vytvořených segmentů a základní statistiku délky segmentů, včetně minimální, maximální a průměrné délky.

Dále zpráva zahrnuje seznam všech segmentů s jejich počátečním a koncovým časem, stejně jako dobu trvání jednotlivých segmentů. Navíc report obsahuje seznam vyloučených řečových úseků, které byly vyřazeny kvůli porušení pravidel. Tyto informace nám umožní podrobně analyzovat segmentaci zvukového záznamu a případně upravit nastavení pro dosažení lepších výsledků.


```
    "audio_segmenter_type": "Optimal",
    "execution_time": 114.43,
    "configuration": {
      "target_duration": 10,
      "min_duration": 2,
      "max_duration": 25,
      "max_noise_duration": 5,
      "min_transition_silence": 0.199,
    },
    "audio": {
      "file": "D:\\01-06-2021_13-00_2020-21.wav",
      "duration": 34470.99
    },
    "vad": "D:\\01-06-2021_13-00_2020-21_vad_BrainSpeech.json",
```

```
"cut_segments": {
  "score": 3694.76,
  "count": 3246.0,
  "over_max_count": 0,
  "under_min_count": 0,
  "durations": {
    "total": 32546.97,
    "min": 4.36,
    "avg": 10.03,
    "max": 23.12,
    "std": 1.07
  },
  "segments": [
    { "segment": { "start": 26.491, "end": 36.659 }, "duration": 10.17 },
```

Kód 4.4: Ukázka generované zprávy o výsledku segmentace

# 5 Zpracování textu

Další částí navrženého systému je sekce zpracování textu, která má za úkol upravovat text tak, abychom vytěžili co nejvíce dat, neboli dosáhli co největší shody mezi originálním textem a rozpoznaným textem. Z praktických důvodů ji dělíme na část modifikace textu a část korekce textu. Význam a rozdíl v použití těchto částí je vysvětlen u každé z nich.

## 5.1 Modifikace textu

Modifikací textu jsou myšleny operace, které na základě předem definovaných pravidel upravují vstupní text. V našem případě se nám modifikace textu hodí ve dvou situacích. Za prvé pomocí modifikačních pravidel upravíme originální text do podoby, která co nejlépe odpovídá mluvenému projevu v přiřazeném audio souboru. Za druhé dočasně aplikujeme modifikaci na rozpoznaný text, abychom zvýšili šanci na korektní přiřazení k originálnímu textu. V tomto druhém případě nás zejména budou zajímat zkratky slov, které se vyslovují jak v zkráceném tvaru, tak i v rozvinutém.

Modifikací textu budeme povětšinou chtít řešit jednu z těchto oblastí:

- `Odstranění neřečových částí textu` – Tím je myšleno odstranění strukturních informací, jako je jméno řečníka a čas nebo nadbytečné informace, jako například věnování, poznámky pod čarou atd.
- `Převod psané formy na mluvenou` – Tím je myšleno nahrazení všech znaků, zkratek a číslic do tvarů, ve kterých se vyslovují, ale pouze v případě, že se vyslovují. Například tedy chceme převést 100% na sto procentní, nebo 2.1.2023 na druhého ledna dva tisíce dvacet tři. Nechceme však například převádět ”.“ na konci vět na slovo tečka.
- `Odstranění nepovolených znaků` – Tím je myšleno, že například u českých textů budeme povolovat pouze Českou abecedu a tudíž odstraníme všechny nechtěné znaky, jako jsou například již zmíněné tečky na konci vět.

### 5.1.1 Modifikační pravidla

Abychom dosáhli výše zmíněné modifikace, musíme vytvořit ideálně univerzální systém pravidel, pomocí kterého budeme moci docílit téměř libovolné modifikace. Naštěstí tady takový systém již existuje a nazývá se regex, neboli regulární výrazy.

Pomocí regexu jsme schopni pružně a efektivně pracovat s texty ve smyslu vyhledávání, nahrazování a rozdělování.

Pro naše účely vytvoříme nadstavbu nad tímto systémem, která umožní uživatelům jednoduše definovat soubor pravidel. Modifikační pravidla budou mít následující strukturu:

- `Popis (volitelný)` – Slouží výhradně uživatelům proto, aby věděli, co je smyslem daného pravidla.
- `Cíl` – Regulární výraz charakterizující cíl modifikace.
- `Nahrazení` – Obyčejný text, kterým se má nahradit nalezený cíl.
- `Kontext před (volitelný)` – Regulární výraz charakterizující kontext před cílem modifikace.
- `Kontext po (volitelný)` – Regulární výraz charakterizující kontext za cílem modifikace.
- `Počet (volitelný)` – Určuje maximální počet aplikací tohoto pravidla na vstupní text.
- `Vratný (volitelný)` – Určuje, jestli je dané pravidlo vratné. Lze použít pouze na jednoduché výrazy.
- `Testy (volitelný)` – Sada uživatelských testů, které mají uživatelům pomoci ověřit, že se dané pravidlo chová dle očekávání.

Uživatel si může takto definovat pravidla v souborech typu JSON. Je důležité zmínit, že pravidla se aplikují v pořadí, v jakém byla vytvořena. Nedodržení pořadí pravidel v definici může vést k nefunkčnosti dané modifikace.

Pro lepší pochopení schopností tohoto systému následuje ukázka několika pravidel, kde každé z nich je zaměřené na jednu z výše zmíněných oblastí:

```
"description": "Replace abbrevation osv. or o.s.v. with og så videre",
"target": "o[.]?s[.]?v[.]?",
"replacement": " og så videre ",
"context_before": "(^| |\n)",
"context_after": "($| |,|\.|\n)",
"reversible": true
```

Kód 5.1: Ukázka pravidla pro rozepsání zkratky

Pravidlo 5.1 v textu nalezne všechny zkratky „osv“ a nahradí je celým tvarem „og så videre“. Podmínkou pro nahrazení je, že před zkratkou je začátek textu, mezera nebo konec řádku a zároveň za zkratkou je konec textu, mezera, čárka, tečka nebo konec řádku. Kromě toho je pravidlo reverzibilní, což znamená, že pokud bude aplikováno na rozpoznaný text, tak po přiřazení k originálu bude zpětně modifikováváno do formátu zkratky a to jak v rozpoznaném textu tak i přiraženém textu.

```
    "description": "Keep only allowed characters",
    "target": "[^\p{L}'' \n\r]+",
    "replacement": " ",
    "tests": [
      {
        "input": "?!:;.><»«"\"(){}[]/§%&$£*+|~^@#fortsættelseü af
↪  forespørgsel''s",
        "output": " fortsættelseü af forespørgsel''s"
      }
    ]
```

Kód 5.2: Ukázka pravidla pro ponechání pouze povolených znaků

Pravidlo 5.2 v textu nahradí všechny nepovolené znaky mezerou. Nepovolené je v tomto případě cokoliv, co není uvozovka, mezera, konec řádku a nebo nespadá do obecné kategorie písma. K definici pravidla je přidaný i test, který má ověřit, že toto pravidlo opravdu ponechá pouze chtěné znaky.

```
    "description": "Remove lines announcing time, role and speaker",
    "target": "kl\\.\\s+[0-9]{1,2}:[0-9]{1,2}[\\S\\s]{0,60}?\\(.*\\):\\s+",
    "replacement": "\n",
    "tests": [
      {
        "input": "kl. 10:00\nMeddelelser fra formanden\nFørste næstformand (Karen
↪  Ellemann):\nMødet er åbnet.",
        "output": "\nMødet er åbnet."
      }
    ]
```

Kód 5.3: Ukázka pravidla pro odstranění nadbytečného textu

Jako poslední ukázku zde máme pravidlo 5.3, které z textu odstraňuje nadbytečné informace, které uvádějí čas, postavení a jméno řečníka. Toto pravidlo je velmi užitečné při zpracování záznamů jednání z dánských parlamentů. Jak můžeme vidět v přidaném testu, tak touto modifikací můžeme odstranit významnou část přebytečného textu.

## 5.2 Korekce textu

Korekce textu se aplikuje po přiřazení textu k audio souborům. Využívá korekční pravidla ke sjednocení tvarů slov.

### 5.2.1 Korekční pravidla

Korekční pravidla jsou stejně jako modifikační pravidla založena na regexu, ale jejich smysl a struktura je rozdílná.

- `Popis (volitelný)` – Slouží výhradně uživatelům proto, aby věděli, co je smyslem daného pravidla.
- `Nahradit v` – Určuje, jestli se má korekce aplikovat v originálním textu a nebo rozpoznaném.
- `Pravidlo pro originál` – Pravidlo, pomocí kterého se bude hledat shoda v originálním textu. Toto pravidlo se skládá z cíle, kontextu před a kontextu po, stejně tak jako tomu bylo u modifikačního pravidla.
- `Pravidlo pro rozpoznání` – Ekvivalent pravidlu pro originální text s tím rozdílem, že je aplikováno na rozpoznaný text.

Korekční pravidla se aplikují pouze v případě, že daná část byla nalezena jak v originálním textu, tak i v rozpoznaném textu. Aplikace pravidel funguje následovně: pokud je cílem nahrazení „originální text“, cíl z pravidla pro originální text se nahradí cílem z pravidla pro rozpoznaný text a naopak.

```
"description": "Replace words ending from pt to punkt",
"replace_in": "estimation",
"original_rule": {
    "target": "punkt",
    "context_before": ".*"
},
"estimation_rule": {
    "target": "pt",
    "context_before": ".*"
}
```

Kód 5.4: Ukázka pravidla pro korekci slov

Pravidlo 5.4 slouží pro rozvinutí koncovky *pt* do tvaru *punkt* u rozpoznaných slov. V dánštině totiž *pt* a *punkt* znamenají totéž a to je *bod*. Například tedy *tidpt* i *tidpunkt* označují to samé a tím je bod v čase. Díky této korekci, se odstraní preference tvaru od tvůrce textu a tím se zvýší šance na správné přiřazení.

# 6 Přiřazení textů k audio segmentům

Konečně se dostavme k poslední části navrženého systému, která přiřadí k jednotlivým audio segmentům, získaným ze sekce zpracování audia (4) část textu, získaného ze sekce zpracování textu (5).

## 6.1 Rozpoznání audia

Proto, abychom mohli k jednotlivým audio segmentům přiřadit vhodnou část textu z dodaného zdroje, musíme nejprve zjistit, co se v daných audio segmentech říká. K tomu nám poslouží systém automatického rozpoznávání řeči. Neboli toto je přesně ta část, kde využíváme již existují model k tomu abychom získali data pro vylepšení toho stejného modelu.

V našem případě využijeme systém vyvinutý v týmu SpeechLab TUL, k jehož dalšímu rozvoji směřuje i tato diplomová práce. Komunikace se Speechlab systémem probíhá pomocí konzolové aplikace s názvem ntx20 [24], která umožňuje zadat úlohy na cluster s vybraným typem rozpoznávače. Níže je ukázka použití tohoto nastroje, která pro daný rozpoznávač, definovaný jménem a verzí spustí na clusteru rozpoznávání souboru *source.wav*, jehož výsledek uloží do souboru *results.json*

```
ntx20 run name:version@cluster -i source.wav -o results.json
```

Kód 6.1: Použití nástroje ntx20 pro zadání úlohy na rozpoznání textu

Výstupní JSON obsahuje informaci o tom, co bylo rozpoznáno jako obsah audia, včetně časových značek jednotlivých slov. Důvodem, proč se nepoužijí časové značky z rozpoznávače pro segmentaci audia, je fakt, že tento systém není trénován na datech, jejichž součástí je časová značka. Tudíž je tato schopnost určení času naučena jako vedlejší efekt a tím pádem není zaručena její přesnost. Obzvláště u delších nahrávek se může vyskytnout chyba i v řádu jednotek sekund, což znamená, že použití tohoto parametru pro nalezení míst střihu by nemělo tak dobré výsledky jako metoda navržená v sekci zpracování audia (4.3).

## 6.2 Levenstheinova vzdálenost

Předtím než se podíváme na algoritmus pro zarovnávání textů na základě podobnosti se seznámíme s Levenshteinovou vzdáleností. Ta je pro nás důležitá, protože navr-

hovaný algoritmus pro zarovnávání textů je modifikací Wagner-Fisherova algoritmu pro výpočet Levenshteinovy vzdálenosti.

Levenshteinova vzdálenost je metrika, která se používá pro porovnávání dvou řetězců. Konkrétně nám udává minimální editační vzdálenost, což je nejmenší počet editačních operací nutných k převedení jednoho řetězce na druhý. V tomto kontextu máme k dispozici tři editační operace: vložení (inzerce), nahrazení (substituce) a odstranění (delece) [25].

Ukážeme si výpočet Levenshteinovy vzdálenosti mezi dvěma řetězci A a B pomocí Wagner-Fisherova algoritmu, který je založen na dynamickém programování. Pro výpočet si definujeme matici D o rozměrech $|A|+1$ a $|B|+1$, kde $|A|$ a $|B|$ označují délky řetězců. Pro každý prvek matice $D[i,j]$ vypočítáme minimální počet editačních operací potřebných k převedení prvních $i$ znaků řetězce $A$ na prvních $j$ znaků řetězce $B$. Výpočet jednotlivých prvků matice $D$ je vyjádřen vzorcem 6.1 [26].

$$D[i,j] = \begin{cases} i & if \quad j = 0 & & \\ j & if \quad i = 0 & & \\ min(D[i-1,j]+1 & & & delece \\ \quad D[i,j-1]+1 & & & inzerce \\ \quad D[i-1,j-1]+1 & if A[i] \neq B[j] & & substituce \\ \quad D[i-1,j-1]) & if A[i] = B[j] & & shoda \end{cases} \tag{6.1}$$

Výsledné skóre v právem spodním rohu matice, tedy $i = |A|$ a $j = |B|$, odpovídá počtu operací, které je potřeba provést abychom transformovali řetězec $A$ na řetězec $B$. Pokud chceme zjistit, jaké operace a na kterém znaku máme použít k dosažení výsledku, použijeme techniku zpětného průchodu, při které vždy vybereme nejmenší hodnotu z možností, které vedly k cíli.

| | | **k** | **r** | **o** | **n** | **i** | **k** | **a** |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **m** | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **o** | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 |
| **n** | 3 | 3 | 3 | 3 | 2 | 3 | 4 | 5 |
| **i** | 4 | 4 | 4 | 4 | 3 | 2 | 3 | 4 |
| **k** | 5 | 4 | 5 | 5 | 4 | 3 | 2 | 3 |
| **a** | 6 | 5 | 5 | 6 | 5 | 4 | 3 | 2 |

Obrázek 6.1: Ukázka aplikace Wagner-Fischerova algoritmu

Pro lepší pochopení je na obrázku 6.1 vizualizovaná matice D pro přechod ze slova *monika* na slovo *kronika*. Z matice vidíme, že tato ukázková kombinace má editační vzdálenost 2, což znamená, že je nutné provést 2 editační operace k převedení slova *monika* na slovo *kronika*. Konkrétně se jedná o přidání písmenka *k* a nahrazení písmenka *m* za *r*. Kromě toho je v tabulce vyznačená cesta, v tom to případě jedna z cest, která vedla k dosažení daného výsledku.

## 6.3 Algoritmus pro zarovnání textů na základě podobnosti

Nyní, když jsme si představili, co to je Leventainova vzdálenost a jak ji spočítat, můžeme přejít k vysvětlení navrženého algoritmu, který vychází z výše zmíněného postupu a jehož cílem je zarovnání textů na základě podobnosti.

První rozdíl je v tom, že naší základní jednotkou jsou jednotlivá slova a nikoli znaky. Jinak je základní postup stejný a to takový, že se snažíme zjistit minimální počet editačních operací, které musíme provést k tomu, abychom převedli jednu sekvenci slov na druhou. To opět zjistíme z matice $D$, jejíž definice je v tomto algoritmu definovaná vzorcem 6.2.

$$D[i,j] = \begin{cases} 5 \cdot i & if \quad j = 0 \\ 5 \cdot j & if \quad i = 0 \\ min(D[i-1,j] + 5 & delece \\ \quad D[i,j-1] + 5 & inzerce \\ \quad D[i-1,j-1] + LevDiff(A[i],B[j]) \cdot 20 & substituce/shoda \end{cases} \tag{6.2}$$

Rozdíl oproti původnímu výpočtu matice $D$ je v přidělování penalizací za jednotlivé operace. Cena inzerce a delece se změnila z 1 na 5. Zároveň se sloučil výpočet pro substituci a shodu, který reflektuje rozdílnost porovnávaných slov. Konkrétně se tato penalizace počítá jako 20 násobek rozdílnosti dvou slov, kde rozdílnost je vyjádřená pomocí funkce $LevDiff(...)$, která je definovaná vztahem 6.3. Důvodem pro zavedení tohoto způsobu porovnání je zvýšeni šance na přiřazení dvou podobných slov na stejnou pozici při zpětném průchodu. Neboli chceme zvýšíme šanci na správné zarovnání slov, která se liší pouze tvarem nebo obsahují překlep.

$$LevDiff(a,b) = \frac{LevEditDistance(a,b)}{LevEditOperations(a,b)} \tag{6.3}$$

Vzorec pro výpočet $LevDiff(...)$ obsahuje $LevEditDistance(a,b)$, což je Levensteinova vzdálenost a $LevEditOperations(a,b)$, což určuje počet editačních operací potřebných k dosažení této vzdálenosti. Pokud bychom tedy využili ukázkový příklad výpočtu vzdálenosti mezi slovy *monika* a *kronika*, tak bychom dostali $LevEditDistance(...) = 2$ (červená číslice na obrázku 6.1), $LevEditOperations(...) = 8$ (počet šedých políček na obrázku 6.1) a $LevDiff(...) = 0,25$.

Po vypočítání matice D pro vstupní sekvence A a B získáme vše potřebné pro jejich zarovnání. Zarovnání se provádí zpětným průchodem matice D. V každém kroku určíme, jaká operace vedla k dosažení daného místa, a na základě toho vkládáme příslušná slova do výstupních sekvencí. Konkrétní implementace zpětného průchodu je popsána algoritmem 6.2. Výstupem tohoto algoritmu jsou dvě stejně dlouhé sekvence, které mají patřičně zarovnaná slova na základě jejich podobnosti.

```
def back_trace(D: np.ndarray, A: [str], B [str]) -> ([Word], [Word]):
    aligned_A, aligned_B = [], []

    i, j = len(A) - 1, len(B) - 1
    while i > 0 and j > 0:
        action = np.argmin([D[i - 1, j - 1], D[i - 1, j], D[i, j - 1]])
        if action == 0: #Substituce/Shoda
            operation = Operation.Equal if D[i, j] == D[i - 1, j - 1] else
            ↪  Operation.Replace

            aligned_A.append(Word(A[i], operation))
            aligned_B.append(Word(B[j], operation))
            i -= 1
            j -= 1
        elif action == 1: #Inzerce
            aligned_A.append(Word(A[i], Operation.Insert))
            aligned_B.append(Word())
            i -= 1
        elif action == 2: #Delece
            aligned_A.append(Word())
            aligned_B.append(Word(B[j], Operation.Delete))
            j -= 1

    return aligned_A[::-1], aligned_B[::-1]
```

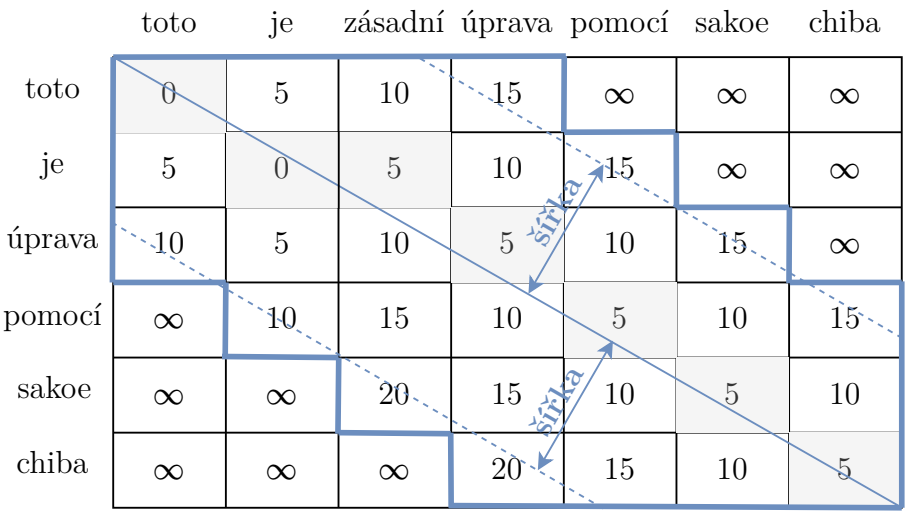
Kód 6.2: Algoritmus pro zarovnání dvou sekvencí pomocí zpětného průchodu

### 6.3.1 Praktická optimalizace

Výše popsaný algoritmus má výkonnostní problém, který se projeví, při pokusu o zarovnání dvou velmi dlouhých textů. Jeho náročnost je $O(nm)$ pro čas i paměť, kde $n$ je počet slov ze sekvence $A$ a $m$ označuje počet slov ze sekvence $B$.

Nicméně v praxi můžeme tento algoritmus výrazně urychlit. Místo počítání celé matice D budeme počítat pouze její část. Tuto část definujeme přímkou mezi počátkem $(0, 0)$ a koncem $(n, m)$ matice $D$. Uživatel pak definuje šířku okolí, která v tomto kontextu určuje maximální počet slov, které můžeme při hledání shody přeskočit. Jinými slovy označuje rozsah slov, které pro dané slovo považujeme za možné kandidáty na přiřazení. Díky tomuto můžeme v závislosti na šířce pásma značně urychlit algoritmus, avšak za cenu ztráty optimálního řešení, pokud se nachází mimo dané pásmo.

Tato metoda se běžně používá v algoritmu dynamického bortění času (DTW), který se používá k nalezení optimální shody mezi dvěma časovými řadami, které mohou mít rozdílnou délku. V případě DTW se této optimalizaci říká zavedení omezujícího pásma podle *Sakoe-Chiba* [27].




Obrázek 6.2: Ukázka matice D s omezením podle Sakoe-Chiba

Obrázek 6.2 zobrazuje matici D po uplatnění omezení dle Sakoe-Chiba. Modré čárkované linie vyznačují šířku pásma, které vymezuje užitnou část matice D, jež je také modře ohraničena. Vidíme, že v matici je optimální řešení (políčka se šedým pozadím) umístěno uvnitř definované oblasti.

Optimalizací popsanou výše jsme algoritmus zlepšili z hlediska času, avšak z hlediska paměti stále inicializujeme celou matici $D$ o rozměrech $n \times m$. Toto je pro nás nevýhodné, protože často potřebujeme pouze poměrně úzké pásmo a většina matice $D$ tak obsahuje základní hodnotu ($\infty$). Abychom ušetřili paměť, změníme datovou strukturu z 2D pole na hašovací tabulku, jejímž klíčem bude dvojice indexů, stejně jako by to bylo při práci s polem. Tímto způsobem budeme v paměti uchovávat pouze hodnoty, které se nacházejí uvnitř omezujícího pásma.

## 6.4 Konkrétní implementace

Nyní, když rozumíme principu algoritmu pro zarovnání dvou sekvencí slov, se můžeme věnovat přiřazování částí textu z originálního zdroje k jednotlivým audio segmentům. V této části si postupně popíšeme kroky, které vedou na tížený výsledek.

Celý proces začíná načtením dat. Na jedné straně máme upravený originální text, jehož části chceme přiřadit k audio segmentům. Na druhé straně máme výsledky rozpoznávání řeči pro jednotlivé audio segmenty. Ty načteme do hašovací tabulky, kde klíčem je název souboru a hodnotou jsou rozpoznaná slova po aplikaci modifikačních pravidel. Je důležité, aby z názvu souborů bylo možné určit pořadí segmentů, které obsahují.

S takto načtenými daty můžeme začít s přiřazováním. Nejjednodušší by bylo aplikovat algoritmus pro zarovnání načtených textů a následně zpětně přiřadit odpovídající části k jednotlivým souborům. To však nebude možné, jelikož často pracujeme s texty, které jsou extrémně dlouhé a tudíž jejich zpracování je jak časově, tak i pa-

měťově náročné a to i přes optimalizace popsané v 6.3.1.

Z toho vyplývá nutnost zavést další úpravu, která umožní zpracování celého textu v rozumném časovém rámci. Tato úprava spočívá v přiřazování po skupinách souborů, u kterých budeme hledat odpovídající části pouze v omezené části originálního textu. Neboli místo jednoho obrovského přiřazení, uděláme spoustu malých, jejichž velikost je stanovena uživatelem. Tento parametr nazveme *velikost skupiny* a pro potřeby vysvětlení předpokládejme, že je nastaven na 100.

Nyní tedy budeme pracovat s prvními 100 záznamy rozpoznání. K těmto 100 záznamům určíme část původního textu, ve které budeme hledat odpovídající úseky pro přiřazení. Důvodem, proč můžeme hledat pouze v omezené části textu, je skutečnost, že audio a tedy i původní text respektují časovou posloupnost. Můžeme tedy předpokládat, že odpovídající obsah pro první segment bude někde na počátku dostupného textu. Velikost oblasti z originálního textu určíme jako velikost odpovídající velikosti 100 rozpoznaných záznamů, ke které přičteme toleranci. Velikost tolerance určuje uživatel a pro nejlepší výsledek by ji neměl nastavovat větší než je šířka pásma u algoritmu pro zarovnání.

Takto vybrané skupiny nyní zarovnáme pomocí dříve popsaného algoritmu pro zarovnání sekvencí. Získáme tím dvě stejně dlouhé sekvence, které na odpovídajících pozicích obsahují přiřazená slova. Následně namapujeme odpovídající části zpět na rozpoznané záznamy, čímž automaticky získáme i přiřazenou část z původního textu.

Abychom zvýšili možnou shodu mezi rozpoznanou a přiřazenou částí, aplikujeme tři následující úpravy:

1. `Hledání chybějících slov na začátku/konci sekvence u sousedů` – Pokud jsou na začátku nebo na konci přiřazené sekvence prázdné slova (tj. algoritmus zarovnání k prvním/posledním slovům rozpoznání nenašel vhodný protějšek), podíváme se na konec/začátek přiřazené části v předchozím/následujícím rozpoznání. Pokud obsahuje správnou sekvenci slov, použijeme ji jako náhradu za prázdnou část na začátku/konci. Toto pravidlo je potřeba, jelikož segmentační algoritmus má občas tendenci upřednostnit podobná slova před stejnými, zejména pokud audio obsahuje mnohem více slov než je dostupné v originálním textu.

2. `Prohození pořadí slov v přiřazeném textu` – Pokud jsou sousedící slova nebo slova, která jsou o jedno od sebe v přiřazené sekvenci prohozena na odpovídajících místech v rozpoznané sekvenci, tak prohodíme jejich pořadí. Toto pravidlo je potřeba, jelikož se v originálních textech často stává, že někdo zapisuje slova v pořadí, na které je zvyklý a ne v pořadí, ve kterém byla slova řečena.

3. `Odstranění nepřiřazených slov z originálního textu` – Z přiřazené sekvence odstraníme všechna slova, ke kterým není přiřazeno slovo v rozpoznané sekvenci. Toto pravidlo slouží především pro odstranění doplňkových informací, které nebylo možné, kvůli jejich náhodnému charakteru, odstranit

pomocí modifikačních pravidel. Zároveň u některých sekvencí může toto pravidlo vést na horší skóre přiřazení, to ale není závažný problém, jelikož toto může nastat pouze u sekvencí, kde by stejně nedošlo k 100% shodě.

Po provedení těchto úprav vytvoříme záznam pro každé přiřazení, který obsahuje název souboru, rozpoznaný text, přiřazený text a jejich procentuální podobnost, která je vyjádřena vztahem $(1 - LevDiff(A, B)) \cdot 100$, kde $LevDiff(A, B)$ je definován pomocí vzorce 6.3.

Po dokončení přiřazení pro danou skupinu tyto výsledky uložíme. Současně se musíme rozhodnout, jestli se pro jednotlivá přiřazení pokusíme najít lepší shodu či nikoli. Důvodem pro tuto úvahu je skutečnost, že pracujeme pouze s částí originálního textu, který považujeme za nejpravděpodobnější místo nalezení shody. Toto však můžeme s jistotou tvrdit pouze pro první položku ve skupině nikoli pro všechny, natož pro poslední. Může se stát, že obsah prvních 60 souborů bude zahrnut v dané části, ale obsah zbývajících 40 najdeme až hlouběji v originálním textu.

Proto musíme určit hranici, za kterou již nebudeme přiřazení důvěřovat a dané soubory rozpoznání použijeme znovu v další iteraci, kde budeme pracovat s jinou částí originálního textu. Tuto hranici určíme následujícím pravidlem: Kandidáti na hraniční rozdělení jsou ti, jejichž délka je alespoň 3 slova a podobnost je alespoň 75 %. Z těchto kandidátů se pak vezme poslední, jehož sousedé na obou stranách jsou taktéž kandidáti. Ponecháváme jednoho kandidáta (souseda na pravé straně) do další iterace, což donutí zarovnávací algoritmus méně vynechávat slova a případně doplnit správná slova, pokud by tento kandidát byl poslední.

Pro rozpoznání, které bylo před hranicí rozdělení, již nebudeme dále hledat lepší přiřazení a tudíž můžeme odstranit k nim přiřazený text z originálního textu. To nám zajistí, že v další iteraci budeme mít opět největší pravděpodobnost na nalezení vhodného přiřazení na začátku originálního textu.

Je důležité poznamenat, že jako finální přiřazení pro každé rozpoznání bude vybráno to s největší podobností. Algoritmus také obsahuje pravidla, která zajišťují, že se nezasekne. Tato pravidla nejsou pro jednoduchost v tomto vysvětlení uvedena. Zaseknutí může například nastat v případě, že nenajdeme vhodného kandidáta pro rozdělení.

Po ukončení procesu získáme pro každý audio segment přiřazení, které má nejlepší podobnost vůči rozpoznanému obsahu. Před tím, než uložíme tyto výsledky, provedeme několik drobných úprav. Za prvé revertujeme modifikační pravidla (5.1.1, která jsou označeny jako *reversible*. Typicky chceme zpětně modifikovat rozepsané zkratky, které byli rozpoznány ve své zkrácené podobě a my je pro potřeby přiřazeni rozepsali do plné podoby. Za druhé provedeme korekci textu pomocí korekčních pravidel (5.2.1), která nám slouží pro vylepšení podobnosti u slov, které mohou mít více tvarů výslovnosti, ale my chceme akceptovat pouze jeden z nich.

Na závěr vygenerujeme podrobnou zprávu o přiřazení textů k audio segmentům. Tato zpráva obsahuje obsahuje informace o nastavení přiřazení, zdroji originálního textu a použitých modifikačních a korekčních pravidlech. Součástí zprávy je také základní statistika týkající se podobnosti textů. Ta zahrnuje například průměrnou

hodnotu podobnosti nebo jejich distribuci. Tyto údaje nám umožní určit, jaká část přiřazených souborů je vhodné pro tréninkové účely.

Zpráva dále obsahuje seznam všech přiřazení. Každý záznam v tomto seznamu obsahuje cestu k souboru s výsledkem rozpoznání daného audio segmentu, obsah rozpoznání, přiřazený text a míru jejich podobnosti. Tato data mohou být následně extrahována do samostatných souborů, které jsou pak přiřazeny k patřičným audio segmentům, čímž získáme trénovací data.

```
"source": "D:\\01-06-2021_13-00_2020-21_clean.txt",
"execution_time": 172.08,
"configuration": { "group_size": 100, "max_ignore_count": 400},
"text_loader": {
  "type": "JSON",
  "text_modifier": {
    "modification_rules_files": [ "C:\\Rules\DK_modifications_rules.json"]
    },
"text_corrector": {
   "correction_rules_files":["C:\\Rules\DK_corrections_rules.json"]
 },
"count": 8305,
"similarity": {
  "min": 0.0, "avg": 94.77, "max": 100.0, "std": 11.15,
  "ranges": {
    "(-0.001, 50.0]": { "count": 103, "percent": 1.24 },
    "(50.0, 60.0]": { "count": 38, "percent": 0.46 },
    "(60.0, 70.0]": { "count": 119, "percent": 1.43 },
    "(70.0, 80.0]": { "count": 274, "percent": 3.3 },
    "(90.0, 99.0]": { "count": 2911, "percent": 35.05 },
    "(99.0, 99.99]": { "count": 146, "percent": 1.76 },
    "(99.99, 100.0]": { "count": 3968, "percent": 47.78 }
  }
},
"matches": [
  {
    "source": "D:\\CUT_2_BrainSpeech\01-06-2021_13-00_2020-21_00000.json",
    "similarity": 98.18,
    "estimated_text": "ro i salen vi starter med mine ord",
    "original_text": "ro i salen vi starter med mindeord"
  },
```

Kód 6.3: Ukázka generované zprávy o výsledku přiřazení textu k audio segmentům

# 7 Vyhodnocení funkčnosti

V této části provedeme experimenty, které mají za úkol zjistit, jak si navržený systém vede. Zejména nás bude zajímat, jakou část vstupních dat jsme schopni plně automaticky vytěžit.

Experimenty byly prováděny na počítači s procesorem AMD Ryzen 9 5900X. Žádné výpočty nebyly převáděny na grafickou kartu.

Tabulka 7.1: Konfigurace experimentů

| **Detekce řeči** | | **Segmentace** | | **Přiřazení** | |
|---|---|---|---|---|---|
| Aktivační hranice | 50 % | Cílová délka | 2 s | Velikost skupiny | 100 |
| Deaktivační hranice | 40 % | Minimální délka | 2 s | Tolerance výběru | 400 |
| En. aktivační hranice | 40 % | Maximální délka | 25 s | Šířka pásma | 1000 |
| En. deaktivační hranice | 0 % | Maximální neřeč | 5 s | | |
| Minimální délka řeči | 0,2 s | Přechodá neřeč | 0,2 s | | |
| Maximální délka řeči | 24 s | | | | |
| Minimální délka ticha | 0,2 s | | | | |

Tabulka 7.1 obsahuje nastavení, se kterým byly experimenty prováděny. K danému nastavení detekce řeči jsme došli optimalizací na testovací části dat z dánského parlamentu, kterou jsme provedli v rámci výběru detektoru řeči viz. 4.2.5.

U nastavení segmentace jsme určovali pouze cílovou délku a maximální délku ne-řeči. Maximální délka ne-řeči mezi segmenty je určena tak, abychom měli co nejméně ne-řeči v segmentech, ale zároveň dosáhli vysoké rychlosti algoritmu, která je tímto parametrem zásadně ovlivněna viz. konkrétní implementace segmentačního algoritmu (4.3.4). Cílová délka byla vybrána tak, abychom co nejlépe ovlivnili následné přiřazení k daným segmentům, kde vycházíme z předpokladu, že čím delší segment, tím větší šance na chybu. Následně si toto ověříme i číselně v rámci experimentu na dánských datech (7.2.1). Ostatní parametry byly pevně určeny viz. požadavky na segmentaci (4.3.1).

Parametry pro přiřazení byly zvoleny na základě 210 experimentů na testovacích dánských datech, kde byl pro každý parametr určen list možností, které chceme vyzkoušet a vybrána byla kombinace s nejlepší výtěžností s přihlédnutím k času potřebnému na zpracování.

## 7.1 Experiment na jednoduchých českých datech

První experiment provedeme na české audio knize, která bude představovat jednoduchý typ dat. To znamená, že daná data mají vhodné akustické prostředí bez nadbytečného hluku v pozadí a mluvčí velmi dobře artikuluje. Stejně tak i dodaná textová data mají velmi velkou shodu s tím, co se skutečně říká v audio záznamu.

Tabulka 7.2: Výsledek detekce řečové aktivity na českých datech

| **Obecné** | | **Řečové úseky** | |
|---|---|---|---|
| Délka vstupního audia | 390,48 min | Počet | 5889 |
| Délka detekované řeči | 301,33 min | Minimální délka | 0,20 s |
| Zastoupení řeči v audiu | 77,17 % | Průměrná délka | 3,07 s |
| Doba zpracování | 4,09 min | Maximální délka | 17,74 s |
| Zrychlení vůči vstupnímu audiu | 95,59× | Směrodatná odchylka | 2,18 s |

Tabulka 7.2 zobrazuje výsledky detekce řečové aktivity na české audio knize. Z tabulky vidíme, že jsme z celé audio knihy získali 301,33 minut čisté řeči, což odpovídá 77,17 % z celé její délky. Zároveň vidíme, že průměrná délka řečového úseku je 3,07 s, z čehož můžeme předpokládat, že při segmentaci budeme značně mimo náš cíl 2 s. Dalším zajímavým parametrem je doba zpracování, ze které vidíme, že jsme celých 390,48 min byli schopni zpracovat za 4 minuty, což odpovídá 95,59 násobnému zrychlení.

Tabulka 7.3: Výsledek segmentace na českých datech

| **Obecné** | | **Vytvořené segmenty** | |
|---|---|---|---|
| Délka vstupního audia | 390,48 min | Počet | 4681 |
| Délka detekované řeči | 301,33 min | Minimální délka | 2,01 s |
| Délka vytvořených segmentů | 345,05 min | Průměrná délka | 4,42 s |
| Zastoupení řeči v segmentech | 87,33 % | Maximální délka | 18,14 s |
| Doba zpracování | 0,10 min | Směrodatná odchylka | 2,04 s |
| Zrychlení vůči vstupnímu audiu | 3815,76× | | |

Po detekci řeči přišla na řadu segmentace, jejíž výsledky jsou zobrazeny v tabulce 7.3. Z dat vidíme, že na základě detekované řeči jsme vytvořili 4681 segmentů s celkovou délkou 345 minut. To znamená, že obsah těchto segmentů tvoří z 87 % řeč, a zbytek je ne-řeč přidána zejména povinnými přechody na okrajích segmentů. Zároveň vidíme, že průměrná délka segmentů, je značně vzdálená od cílových 2 s, což je způsobeno tím, že 2 s jsou zároveň spodní hranicí segmentace a vysokým průměrem řečových úseků. Nakonec v tabulce vidíme i dobu zpracování, která je pouze 0.1 minuty, což odpovídá 3815 násobnému zrychlení oproti délce vstupního audia.

Následně rozpoznáme obsah jednotlivých segmentů, abychom k nim následně mohli přiřadit příslušný text. K tomu využijeme ASR model s označením atran-cz-

-openlex:23.0224.1 od SpeechLab TUL. Data byla zpracována pomocí 15 paralelních úloh, kterým rozpoznání všech segmentů zabralo necelých 7 minut.

Ještě před přiřazením je potřeba upravit dostupné texty (rozpoznané a originální), k čemuž použijeme modifikační pravidla. Cílem tohoto jednoduchého experimentu bylo především ověřit funkčnost systému a tudíž nebyla věnována pozornost maximalizaci vytěžených dat. Z toho důvodu jsme zde použili pouze pár jednoduchých pravidel, která například zajišťovala, že texty nebudou obsahovat nepovolené znaky. Žádná pokročilejší pravidla, která by například rozepisovala datum, nebo zkraty nebyla použita.

Tabulka 7.4: Výsledek přiřazení textů k audio segmentům na českých datech

| **Obecné** | | **Rozdělení podobnosti** | |
|---|---|---|---|
| Délka vytvořených segmentů | 345,05 min | (0%, 50%] | 1,13 % |
| Délka vytěžených dat | 308,79 min | (50%, 60%] | 0,04 % |
| Procento vytěžených dat | 89,49 % | (60%, 70%] | 0,00 % |
| Doba zpracování | 1,85 min | (70%, 80%] | 0,06 % |
| Zrychlení vůči vstupnímu audiu | 186,11× | (80%, 90%] | 0,21 % |
| | | (90%, 99%] | 7,95 % |
| | | (99%, 100%) | 1,11 % |
| | | 100% | 89,49 % |

Po rozpoznání segmentů a přípravě textů se můžeme pokusit přiřadit texty k patřičným segmentům. Výsledky tohoto kroku jsou uveden v tabulce 7.4. V pravé polovině tabulky vidíme procentuální zastoupení segmentů podle podobnosti přiřazených textů. Nejpodstatnější je pro nás hodnota v posledním řádku a tedy jak velká část byla přiřazena se 100% shodou. Zde vidíme, že je to skvělých 89,49 %. Z toho plyne, že jsme byli schopni z dodaných segmentů vytěžit necelých 309 minut dat připravených pro trénovaní. Zajímavá je také rychlost zpracování, která je 1,85 minuty, což odpovídá 186 násobnému zrychlení oproti délce vstupních segmentů.

Tabulka 7.5: Shrnutí experimentu na českých datech

| **Výtěžnost** | | **Doba zpracování** | |
|---|---|---|---|
| Původní délka audia | 390,48 min | Předzpracování audia | 0,02 min |
| Délka po předzpracování | 390,48 min | Detekce řeči | 4,09 min |
| Délka detekované řeči | 301,33 min | Segmentace audia | 0,10 min |
| Délka po segmentaci | 345,05 min | Rozpoznávání řeči | 6,62 min |
| Délka vytěžených dat | 308,79 min | Zpracování textů | 0,00 min |
| Výtěžnost vůči původnímu | 79,08% | Přiřazení textů k segmentům | 1,85 min |
| Výtěžnost vůči segmentům | 89,49% | Celková doba zpracování | 12,68 min |
| | | Zrychlení vůči původnímu audiu | 30,80× |

Na závěr máme v tabulce 7.5 shrnutí experimentu. V levé části je pro nás zásadní skvělá výtěžnost 89.49 % oproti získaným segmentům a 79,08 % oproti původnímu

audiu. V pravé části vidíme, že celý proces zabral necelých 13 minut, což odpovídá zrychlení 30,8 oproti původní délce dat. Zároveň také vidíme, že časově nejnáročnější je rozpoznání řeči v segmentech, což je způsobenou komplexitou ASR modelů.

## 7.2 Experiment na komplexních dánských datech

Druhý a mnohem rozsáhlejší experiment provedeme na záznamech jednání dánského parlamentu za rok 2021-2022. Důvodem pro zpracování dánštiny je projekt NordTrans, který se zabývá vývojem ASR systémů pro severské jazyky a na jehož řešení pracuje tým SpeechLab Tul [28]. Data získaná v tomto experimentu pomohou s tréninkem ASR modelu pro dánštinu.

Parlamentní záznamy svým charakterem představují komplexní typ dat. To znamená, že data jsou nahrávána v prostředí, kde na nahrávce jsou nechtěné zvuky, více mluvčích může mluvit zároveň a objevují se problémy typu přeřeky, přízvuk, koktání, atd. Stejně tak dodaná textová data nemají již tak přímou strukturu a obsahují spoustu textu, který není skutečně řeč a naopak.

Data použitá v tomto experimentu jsou mnohem rozsáhlejší než u českého experimentu, a proto zde místo minut budeme pracovat s hodinami a zavedeme paralelní zpracovaní, které nebylo u předchozí úlohy potřeba.

V rámci předzpracování audia byla provedena redukce, která z původních 730 hodin udělala 707.47 hodin. Vymazané úseky odpovídají pauzám, kde nebyl vypnut mikrofon.

Tabulka 7.6: Výsledek detekce řečové aktivity na dánských datech

| **Obecné** | | **Řečové úseky** | |
|---|---|---|---|
| Délka vstupního audia | 707,47 h | Počet | 1000352 |
| Délka detekovaná řeč | 521,35 h | Minimální délka | 0,20 s |
| Poměr řeči v audiu | 73,69 % | Průměrná délka | 1,88 s |
| Výpočetní doba zpracování | 7,01 h | Maximální délka | 23,68 s |
| Paralelismus | 7 | Směrodatná odchylka | 1,50 s |
| Skutečné zpracování | 1,19 h | | |
| Zrychlení vůči vstupnímu audiu | 595,54× | | |

Tabulka 7.6 zobrazuje výsledky detekce řečové aktivity na dánských datech. Z tabulky vidíme, že jsme ve všech jednání detekovali 521,35 hodin jako řeč, což odpovídá 73,69 % z celkové délky dat po předzpracování. Z pohledu rychlosti byla výpočetní doba zpracování 7,01 hodin - výpočetní doba počítače. To při rozdělení práce mezi 7 úloh znamenalo, že skutečná doba zpracování klesla na 1,19 hodiny. Ve výsledku jsme tedy dosáhli 595,54 násobného zrychlení oproti délce vstupního audia. To je značný skok v porovnání s českým experimentem, kde nebyl využíván paralelismus.

Po detekci řeči přejdeme k segmentaci audia, jejíž výsledky jsou zobrazeny v tabulce 7.7. Z tabulky vidíme, že jsme vytvořili 629240 segmentů (souborů), o celkové

Tabulka 7.7: Výsledek segmentace na dánských datech

| **Obecné** | | **Vytvořené segmenty** | |
|---|---|---|---|
| Délka vstupního audia | 707,47 h | Počet | 629240 |
| Délka detekované řeči | 521,35 h | Minimální délka | 2,01 s |
| Délka vytvořených segmentů | 644,66 h | Průměrná délka | 3,69 s |
| Poměr řeči v segmentech | 80,87 % | Maximální délka | 24,08 s |
| Výpočetní doba zpracování | 1,38 h | Směrodatná odchylka | 1,32 s |
| Paralelismus | 7 | | |
| Skutečné doba zpracování | 0,23 h | | |
| Zrychlení vůči vstupnímu audiu | 3129,06× | | |

délce 644,66 hodin. Vytvořené segmenty obsahují z 80,87 % řeč. Zbytek opět odpovídá ne-řeči, která je tvořena povinnými přechody a spojováním jednotlivých úseků. I zde vidíme, že průměrná délka segmentu se značně liší od cílové, což je opět způsobeno tím, že spodní hranice segmentace je shodná s cílovou a tudíž spojováním řečových úseků, jejichž průměrná délka je 1,88 s často docházelo k značnému překročení cílové hranice. Dále v dané tabulce vidíme, že výpočetní doba zpracování je 1,38 hodiny. Skutečná doba zpracovaní při rozdělení na 7 úloh, pak byla pouze 0,23 hodiny. Z čehož plyne, že jsme dosáhli 3129 násobného zrychlení oproti vstupnímu audiu. Zde si můžeme všimnout že i při použiti paralelismu je zrychlení menší, než tomu bylo u českých dat. To je způsobeno tím, že u českých dat byla kniha rozdělena do 56 částí, a tudíž jsme v každé části pracovali s velmi malým množstvím řečových úseků, což se značně projevilo na rychlosti segmentace.

Jako další krok opět necháme rozpoznat všechny segmenty, tentokrát ale s dánským modelem. Konkrétně využijeme model 22.1224.000000-no-e60-a10-atran-dk-e2e od SpeechLab TUL, který byl natrénován na konci roku 2022. Všechny segmenty byly rozpoznány za necelých 14 hodin při použití 15 paralelních úloh.

Následně je potřeba upravit texty na základě modifikačních pravidel. V tomto případě již pouze nechceme otestovat systém, ale chceme vytěžit co největší množství dat, a proto definování těchto pravidel je mimořádně důležité. Pro zpracování dánských dat se ve spolupráci s profesorem Nouzou vytvořilo 1909 modifikačních pravidel, která pozitivně přispěla k finální výtěžnosti systému. Vytvořená pravidla obsahují obecné definice, které například umožňují převod číslic, dat a jiných speciálních symbolů do textové podoby. Kromě obecných pravidel byla vytvořena i pravidla řešící konkrétní textový formát a parlamentní doménu. Tato pravidla zajišťují například odstranění přebytečných informací ze záznamů jednání nebo rozepisování zkratek dánských politických stran. Aplikace modifikačních pravidel na originální a rozpoznané texty zabrala 1,04 hodiny.

Jako poslední krok přiřadíme upravené texty k rozpoznaným segmentům. Výsledky této části jsou zobrazeny v tabulce 7.8. V pravé části tabulky vidíme, jak velká část dat byla přiražena s danou podobností. Nejdůležitější je hodnota v posledním řádku, která udává jak velké procento segmentů bylo přiřazeno se 100% podobností. Z toho vychází, že 49,17 % dat, což odpovídá 308,79 hodinám bylo plně automatic-

Tabulka 7.8: Výsledek přiřazení textů k audio segmentům na dánských datech

| **Obecné** | | **Rozdělení podobnosti** | |
|---|---|---|---|
| Délka audio segmentů | 644,66 h | (0%, 50%] | 1,14% |
| Délka vytěžených dat | 316,98 h | (50%, 60%] | 0,46% |
| Výpočetní zpracování | 7,69 h | (60%, 70%] | 1,09% |
| Paralelismus | 7 | (70%, 80%] | 2,96% |
| Skutečné zpracování | 2,14 h | (80%, 90%] | 9,24% |
| Zrychlení vůči audio segmentům | 300,92× | (90%, 99%] | 34,19% |
| | | (99%, 100%) | 1,76% |
| | | 100% | 49,17% |

ky vytěženo a jsou připraveny na trénovaní. Tento výsledek je sice značně horší než u audio knihy ale stále ho můžeme považovat za velmi dobrý vzhledem k obtížnosti těženého zdroje. Dále vidíme, že v rozsahu (90%, 100%) podobnosti je dalších téměř 36 % dat, které vyžadují drobnou manuální korekci, po které budou i ony připraveny na trénování. Z pohledu časové náročnosti vidíme, že výpočetní čas byl 7,69 hodiny. Skutečná doba zpracování byla pouze 2,14 hodiny. Z toho plyne, že jsme dosáhli 300,92 násobného zrychlení oproti délce audio segmentů.

Tabulka 7.9: Shrnutí experimentu na dánských datech

| **Výtěžnost** | | **Doba zpracování** | |
|---|---|---|---|
| Původní délka audia | 730,00 h | Předzpracování audia | 0,07 h |
| Délka po předzpracovaní | 707,47 h | Detekce řeči | 1,19 h |
| Délka detekované řeči | 521,35 h | Segmentace audia | 0,23 h |
| Délka po segmentaci | 644,66 h | Rozpoznávání řeči | 13,58 h |
| Délka vytěžených dat | 316,98 h | Zpracování textů | 1,04 h |
| Výtěžnost vůči původnímu | 43,42% | Přiřazení textů k segmentům | 2,14 h |
| Výtěžnost vůči segmentům | 49,17% | Celková doba zpracování | 18,25 h |
| | | Zrychlení vůči původnímu audiu | 40,01× |

Na závěr máme v tabulce 7.5 shrnutí experimentu. V levé části je pro nás zásadní výtěžnost systému, která je 316,98 hodin, což odpovídá 49,17 % oproti získaným segmentům a 43,42 % oproti původnímu audiu. V pravé části vidíme, že celý proces zabral 18,25 hodin, což odpovídá zrychlení 40,01 oproti původnímu audiu.

### 7.2.1 Testování výtěžnosti v závislosti na délce segmentu

Jak již bylo zmíněno, tak cílem toho experimentu bylo vytěžit co největší množství dat. Z toho důvodu byla v rámci experimentu ověřena myšlenka, že čím kratší bude délka segmentů, tím větší máme šanci na bezchybné přiřazení textů. V tabulce 7.10 je porovnání výtěžnosti systému na dánských testovacích datech v závislosti na cílové délce segmentů. Důležitý je pro nás poslední sloupec, který zobrazuje, jak velké procento dat bylo perfektně přiřazeno na základě cílové délky segmentu. Vidíme,

Tabulka 7.10: Porovnání výsledků přiřazení textů k audio segmentům na základě cílové délky segmentu

| | **(0%, 50%]** | **(50%, 90%]** | **(90%, 100%]** | **100%** |
|---|---|---|---|---|
| **Cíl 2 s** | 1,31 % | 13,76 % | 35,66 % | 49,27 % |
| **Cíl 3 s** | 1,77 % | 15,08 % | 37,04 % | 46,11 % |
| **Cíl 4 s** | 0,95 % | 14,74 % | 42,15 % | 42,16 % |
| **Cíl 5 s** | 0,71 % | 14,27 % | 48,48 % | 36,54 % |
| **Cíl 6 s** | 0,58 % | 13,80 % | 54,06 % | 31,56 % |
| **Cíl 7 s** | 0,83 % | 13,26 % | 58,74 % | 27,17 % |
| **Cíl 8 s** | 0,39 % | 12,82 % | 62,92 % | 23,87 % |
| **Cíl 9 s** | 0,35 % | 12,50 % | 65,90 % | 21,25 % |
| **Cíl 10 s** | 0,16 % | 9,13 % | 71,32 % | 19,39 % |

že s klesající délkou segmentu značně stoupá výtěžnost, čímž jsme si potvrdili výše zmíněnou hypotézu. Zvolení minimální cílové délky byla tedy správná volba.

Zde by bylo vhodné namítnout, že sice s klesající délkou segmentu roste výtěžnost, ale pokud budou daná data příliš krátká, tak se model nebude moci učit kontext a mohl by se trénováním na daných datech zhoršit. To je samozřejmě pravda, ale to by snad nemělo nastat, jelikož průměrný počet slov ve vytěžených datech je přibližně 10 se směrodatnou odchylkou 4. Z toho můžeme vyvodit, že vytěžená dat budou sice obsahovat i velmi krátké soubory o pár slovech, ale také v nich bude spousta delších úseků, kde by měl být již dostatečný kontext. Pro zajištění minimálního kontextu byl vedoucím vyžádán seznam souborů, které měly pod 20 znaků. Těchto nevhodných souborů byla 3 % z perfektně přiřazených.

# 8 Praktické přínosy práce

## 8.1 Využití v řešení mezinárodního projektu NordTrans

Experiment na dánských datech byl proveden v tak velkém rozsahu především proto, že vytěžená data měla posloužit k natrénovaní nového modelu, který je součástí řešení mezinárodního výzkumného projektu NordTrans (zaměřeného na rozpoznávání řeči severských jazyků) a který by v ideálním případě měl být lepší než stávající model použitý v experimentu [28].

Vytěžená data byla předána týmu SpeechLab Tul, který je následně využil pro trénování nového modelu. Konkrétně se použilo 317 hodin, které nahradily dříve používaných 700 hodin trénovacích dat z dánského parlamentu. Původních dat bylo sice více než dvojnásobek, ale jejich kvalita z pohledu umístění střihů a přiřazených textů nebyla tak vysoká jako u mnou zpracovaných dat. Proto by i přes snížené množství dat mělo dojít ke zlepšení modelu. Součástí tréninkové sady nového modelu bylo také dalších 400 hodin typu audioknihy, čtené věty a jiné, které tým získal již dříve. Nový model se tedy celkově trénoval na 700 hodinách, kdežto model použitý v dánském experimentu byl trénován na přibližně 1100 hodinách.

Nově natrénovaný model se i přes menší množství dat, ale o větší kvalitě opravdu ukázal být lepší. Při testování na nezávislých datech došlo k zvýšení slovní přesnosti (WAcc) o 0,3 % a to z 90,7 % na 91,0 %. Toto zlepšení může vypadat na první pohled jako malé, ale v kontextu automatického rozpoznávání řeči jsou to právě tato malá procenta, která ve výsledku rozlišují dobré a vynikající ASR systémy.

Z pohledu této práce je také zásadní zjistit, jestli nový model dokáže vytěžit více dat než původní. Tím bychom dokázali, že myšlenka iterativního těženi a trénování dat je správná. Z toho důvodu byl nově natrénovaný model použit na rozpoznání segmentů, které nebyly v rámci dánského experimentu určeny se 100% shodou. Zejména nás zajímají data, která byla v první iteraci přiřazena s podobností 90 - 100 %, u kterých je největší pravděpodobnost, že dojde k zlepšení.

Tabulka 8.1: Porovnání výtěžnosti při využití původního a nového modelu

| | **Původní model** | **Nový model** | **Rozdíl** |
|---|---|---|---|
| **Výtěžnost vůči segmentům** | 49,17 % | 55,12 % | **5,95 %** |
| **Délka vytěžených dat** | 316,98 h | 355,34 h | **38,36 h** |

V tabulce 8.1 je porovnaná výtěžnost při použití původního a nového modelu. Z tabulky vidíme, že došlo k zlepšeni a to celkem nezanedbatelnému. S novým modelem jsme dokázali se 100% shodou přiřadit téměř o 6 % více dat než u původního modelu. Jinými slovy jsme automaticky získali dalších více než 38 hodin trénovacích dat. Toto potvrzuje, že nový model je opravdu lepší a iterativní přístup k těžení a trénování je správný.

## 8.2 Přenositelnost vytvořeného řešení pro jiné jazyky

Tým SpeechLab Tul se specializuje na vývoj systémů pro automatické rozpoznávání řeči (ASR) pro více než 20 jazyků, především těch evropských. Právě proto klade důraz na nástroje, které lze snadno přenášet mezi různými jazyky.

Vytvořený nástroj pro automatické vytěžování trénovacích dat byl v rámci experimentování vyzkoušen na dvou jazycích - češtině a dánštině. Pokud by bylo třeba tento nástroj použít v jiném jazyce, stačilo by přidat specifická pravidla pro modifikaci a korekci daného jazyka. Toto jednoduché přizpůsobení by mělo být dostatečné pro správné fungování systému v jakémkoli jazyce.

Jediným potenciálním problémem může být detekce řeči. V případě, že by detektor v cílovém jazyce fungoval podstatně slaběji, bylo by nutné detekci řeči speciálně dotrénovat pro daný jazyk. Nicméně všechny dosud testované detektory řeči byly trénovány na multijazyčných datech, takže by se tento problém neměl vyskytnout.

# 9 Závěr

Diplomová práce byla zaměřena na návrh, implementaci a testování nástrojů pro automatické generování řečových trénovacích dat z vhodných veřejně dostupných zdrojů. Tento cíl byl úspěšně splněn a byl vytvořen systém, který se skládá ze tří částí - zpracování audia, zpracování textu a přiřazení textu k audio segmentům.

Část zpracování audia jsem rozdělil na předzpracování, detekci řečové aktivity a segmentaci audia. K předzpracovaní audia jsem využil nástroj FFmpeg, který mi umožnil transformovat audio do požadovaného formátu. Pro detekci řečové aktivity jsem otestoval několik přístupů, a na základě experimentálního vyhodnocení jsem vybral natrénovaný CRDNN model od týmu SpeechBrain. Pro segmentaci audia jsem navrhl algoritmus, který na základě uživatelských požadavků vhodně vystřihá řečové úseky z audio záznamu.

V části zpracování textu jsem navrhl systémy pravidel využívající regex, které uživateli umožňují definovat velkou škálu textových úprav. Uživatel je tak například schopen odstranit přebytečné části, přepsat zkratky do plné podoby a nebo rozvinout čísla do jejich mluvené podoby.

Poslední část systému má za úkol přiřadit k vytvořeným audio segmentům odpovídající text. K tomu, abych mohl přiřadit odpovídající text je nutné znát obsah audio segmentů, k čemuž využívám natrénované ASR modely od SpeechLab TUL. Pro samotné přiřazení textů k audio segmentům jsem navrhl algoritmus, který na základě minimální editační vzdálenosti nalezne nejvhodnější shodu mezi rozpoznaným a dodaným textem.

Vytvořený systém jsem otestoval nejprve na jednoduchých českých datech a následně na komplexních dánských datech. Experiment na českých datech jsem provedl na audioknize, která měla velkou shodu mezi audiem a dodaným textem, a tudíž se dá považovat za jednoduchý typ dat. V tomto experimentu se mi podařilo vytěžit skvělých 89,49 % dat, což odpovídá necelým 309 minutám. Průchod celým systémem včetně externího rozpoznání audio segmentů zabral necelých 13 minut.

Experiment na dánských datech jsem provedl na záznamech jednání dánského parlamentu za rok 2020-2021. Tato data představovala již komplexnější typ, jelikož kvalita audia a stejně tak i dodaného přepisu nebyla ani zdaleka tak dobrá jako u audioknihy. Navíc oficiální přepisy parlamentních promluv se vždy více či méně liší od toho, co bylo skutečně řečeno, což je vedeno snahou o maximální srozumitelnost a gramatickou správnost textu. V tomto experimentu se mi podařilo vytěžit 49,17 % dat, což odpovídá téměř 317 hodinám. Kompletní zpracování v tomto případě zabralo 18,25 hodin.

Vytěžená data z dánského parlamentu byla použita SpeechLab TUL na natrénování nového ASR modelu. Tento nový model dosáhl při testování o 0,3 % lepší WAcc než předchozí. Stejně tak i při následném použití nového modelu pro vytěžení dat došlo k nárůstu výtěžnosti o necelých 6 %, což znamená, že v druhé iteraci bylo vytěženo o 38,36 hodin více dat.

Všechny body zadaní byly splněny a i samotné výsledky navrženého systému jsou velmi dobré. Je však důležité podotknout, že veškeré testování bylo provedeno s pokročilými ASR modely. V rámci práce tedy nebylo otestováno chovaní systému při použití v počátečních fázích trénování.

Jako možné vylepšení systému bych navrhl přidání možnosti ladění VAD modelu na uživatelem dodaných datech. Další možností je zdokonalení zpracování textu, které trvá zbytečně dlouho z důvodu separátního zpracování jednotlivých segmentů rozpoznaného textu.

# Použitá literatura


[1] STEVE YOUNG, et. al. *HTK Speech Recognition Toolkit* [http://htk.eng.cam.ac.uk]. Cambridge University Engineering Department, 2006.

[2] DANIEL POVEY, et. al. *Kaldi Speech Recognition Toolkit* [http://kaldi-asr.org]. 2011.

[3] CUTAJAR, Michelle et al. Comparative study of automatic speech recognition techniques. *IET Signal Processing*. 2013, roč. 7, č. 1, s. 25–46. Dostupné z DOI: https://doi.org/10.1049/iet-spr.2012.0151.

[4] LI, Jinyu. *Recent Advances in End-to-End Automatic Speech Recognition*. 2022. Dostupné z arXiv: 2111.01690 [eess.AS].

[5] PULKKI, V.; KARJALAINEN, M. *Communication Acoustics: An Introduction to Speech, Audio and Psychoacoustics*. Wiley, 2015. ISBN 9781118866559. Dostupné také z: https://books.google.cz/books?id=r%5C_TqCAAAQBAJ.

[6] WIKIPEDIE. *WAV*. 2021. Dostupné také z: https://cs.wikipedia.org/w/index.php?title=WAV&oldid=20355868. [3. 04. 2023].

[7] FFMPEG. *FFmpeg* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://ffmpeg.org/ffmpeg.html.

[8] PYANNOTE. *Pyannote Evaluation metrics* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://pyannote.github.io/pyannote-metrics/reference.html#detection.

[9] WIKIPEDIE. *F-score*. 2023. Dostupné také z: https://en.wikipedia.org/w/index.php?title=F-score&oldid=1139808388. [3. 04. 2023].

[10] SESHASHYAMA SAMEERAJ MEDURI, Rufus Ananth. *A Survey and Evaluation of Voice Activity Detection Algorithms*. 2012. Dostupné z diva: 2:831566.

[11] JALIL, Madiha; BUTT, Faran Awais; MALIK, Ahmed. Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced segments of speech signals. In: *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*. 2013, s. 208–212. Dostupné z DOI: 10.1109/TAEECE.2013.6557272.

[12] ROBERTS, Leland. *Understanding the Mel Spectrogram* [online]. 2020. [cit. 2023-04-16]. Dostupné z: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.

[13] RAVANELLI, Mirco; PARCOLLET, Titouan et al. *SpeechBrain: A General-Purpose Speech Toolkit*. 2021. Dostupné z arXiv: 2106.04624 [eess.AS]. arXiv:2106.04624.

[14] SPEECHBRAIN. *Vad model*. Hugging Face, 2022. Dostupné také z: https://huggingface.co/speechbrain/vad-crdnn-libriparty. [30. 04. 2023].

[15] AL, Hervé Bredin et. *pyannote.audio: neural building blocks for speaker diarization*. 2019. Dostupné z arXiv: 1911.01255 [eess.AS].

[16] RAVANELLI, Mirco; BENGIO, Yoshua. *Speaker Recognition from Raw Waveform with SincNet*. 2019. Dostupné z arXiv: 1808.00158 [eess.AS].

[17] PYANNOTE. *VAD model*. Hugging Face, 2021. Dostupné také z: https://huggingface.co/pyannote/segmentation. [30. 04. 2023].

[18] JIA, Fei; MAJUMDAR, Somshubra; GINSBURG, Boris. *MarbleNet: Deep 1D Time-Channel Separable Convolutional Neural Network for Voice Activity Detection*. 2021. Dostupné z arXiv: 2010.13886 [eess.AS].

[19] TEAM, Silero. *Silero VAD: pre-trained enterprise-grade Voice Activity Detector (VAD), Number Detector and Language Classifier* [https://github.com/snakers4/silero-vad]. GitHub, 2021.

[20] VEYSOV, Alexander; VORONIN, Dimitrii. One Voice Detector to Rule Them All. *The Gradient* [https://thegradient.pub/one-voice-detector-to-rule-them-all/]. 2022.

[21] MAXIM TKACHENKO Mikhail Malyuk, Andrey Holmanyuk; LIUBIMOV, Nikolai. *Label Studio: Data labeling software*. 2022. Dostupné také z: https://github.com/heartexlabs/label-studio.

[22] WIKIPEDIE. *Dynamické programování*. 2021. Dostupné také z: https://cs.wikipedia.org/w/index.php?title=Dynamick%C3%A9_programov%C3%A1n%C3%AD&oldid=20192593. [16. 04. 2023].

[23] GAO, Harold. *Algorithm to split an array into P subarrays of balanced sum* [online]. 2019. [cit. 2023-04-17]. Dostupné z: https://stackoverflow.com/a/55025876.

[24] ŽĎÁNSKÝ, Jindřich. *ntx20*. GitLab, 2023. Dostupné také z: https://gitlab.com/nanotrix-public/ntx20. [28. 04. 2023].

[25] WIKIPEDIE. *Levenshtein distance*. 2023. Dostupné také z: https://en.wikipedia.org/w/index.php?title=Levenshtein_distance&oldid=1150303438. [24. 04. 2023].

[26] WIKIPEDIE. *Wagner-Fischer algorithm*. 2023. Dostupné také z: https://en.wikipedia.org/w/index.php?title=Wagner%E2%80%93Fischer_algorithm&oldid=1117965265. [24. 04. 2023].

[27] PYTS. *Sakoe-Chiba band*. 2023. Dostupné také z: https://pyts.readthedocs.io/en/stable/auto_examples/metrics/plot_sakoe_chiba.html. [29. 04. 2023].

[28] NEWTON TECHNOLOGIES A.S. Technická univerzita v Liberci, Norges teknisk-naturvitenskapelige universitet. *NordTrans* [https://www.tacr.cz/en/nordtrans/]. 2021.

# Přílohy

# A Ukázka automatických modifikací referenčního textu (dánština) - rozepisování číslovek a zkratek

**Referenční text:**
I dag er der følgende anmeldelser: Mette Hjermind Dencker (DF) m.fl.: Beslutningsforslag nr. B 136 (Forslag til folketingsbeslutning om ændring af forbrugeraftaleloven).

**Rozpoznaný text:**
i dag er der følgende anmeldelser mette henning dencker dansk folkeparti med flere beslutningsforslag nummer hundrede og seksogtredive om ændring af forbrugeraftaleloven

**Přiřazený text:**
i dag er der følgende anmeldelser mette hjermind dencker dansk folkeparti med flere beslutningsforslag nummer hundred og seksogtredive om ændring af forbrugeraftaleloven

**Referenční text:**
Leif Lahn Jensen (S), Peter Skaarup (DF), Andreas Steenberg (RV), Karsten Hønge (SF), Peder Hvelplund (EL)

**Rozpoznaný text:**
leif lahn jensen socialdemokratiet peter skaarup dansk folkeparti andreas steenberg radikale venstre karsten hønge socialistisk folkeparti peder hvelplund enhedslisten

**Přiřazený text:**
leif lahn jensen socialdemokratiet peter skaarup dansk folkeparti andreas steenberg radikale venstre karsten hønge socialistisk folkeparti peder hvelplund enhedslisten

# B Ukázka automatických modifikací referenčního textu provedených na základě rozpoznané řeči (dánština) - přehození pořadí slov

**Referenční text:**
Mødet er åbnet. Jeg skal lige sige, for der er temmelig mange i salen, at der ikke bliver afstemning før i næste omgang. Det er bare, så folk er klar over det.

**Rozpoznaný text:**
mødet er åbnet jeg skal lige sige fordi der er temmelig mange i salen at der bliver ikke afstemning så i næste omgang det er bare så folk er klar over det

**Přiřazený text:**
mødet er åbnet jeg skal lige sige for der er temmelig mange i salen at der bliver ikke afstemning før i næste omgang det er bare så folk er klar over det

# C Ukázka automatických korektur textu (dánština) - přizpůsobení tvaru slov

**Referenční text:**
så vi kan godt støtte det. Vi er ikke enige i de kritikpunkter, der kommer, og jeg synes faktisk, at regeringen har landet det et godt sted her. Så det skal være ordene

**Rozpoznaný text:**
så vi kan godt støtte det vi er ikke enige i de kritikpter der kommer jeg synes faktisk ikke at regeringen har landet et godt sted her så det skal være ordene

**Přiřazený text:**
så vi kan godt støtte det vi er ikke enige i de kritikpunkter der kommer jeg synes faktisk at regeringen har landet det godt sted her så det skal være ordene

**Rozpoznaný text po korekci:**
så vi kan godt støtte det vi er ikke enige i de kritikpunkter der kommer jeg synes faktisk ikke at regeringen har landet et godt sted her så det skal være ordene

**Referenční text:**
Men hvad nu, hvis der kommer 1,6 mio. t eller mindre ind om året?

**Rozpoznaný text:**
men hvad nu hvis der kommer en komma seks millioner ton eller mindre ind om året

**Přiřazený text:**
men hvad nu hvis der kommer en komma seks million ton eller mindre ind om året

**Přiřazený text po korekci:**
men hvad nu hvis der kommer en komma seks millioner ton eller mindre ind om året