VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MULTI-MODÁLNÍ PŘEPIS TEXTU

MULTI-MODAL TEXT RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MICHAL KABÁČ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN KIŠŠ**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Kabáč Michal, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Strojové učení
Název: **Multi-modální přepis textu**
Multi-Modal Text Recognition
Kategorie: Zpracování obrazu
Zadání:

1. Prostudujte základy neuronových sítí.
2. Vytvořte si přehled o současných metodách rozpoznávání řeči a textu pomocí neuronových sítí.
3. Vyberte nebo navrhněte metodu aplikovatelnou pro korekci výstupů rozpoznávače textu pomocí rozpoznávání řeči.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- SHI, Baoguang; BAI, Xiang; YAO, Cong. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2016, 39.11: 2298-2304.
- KANG, Lei, J. Ignacio TOLEDO, Pau RIBA, Mauricio VILLEGAS, Alicia FORNÉS a Marçal RUSIOL. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In: *Pattern Recognition*. Stuttgart, Germany: Springer International Publishing, 2019, s. 459-472. ISBN 978-3-030-12938-5.
- SHENG, Fenfen; CHEN, Zhineng; XU, Bo. NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019. p. 781-786.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kišš Martin, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 2. listopadu 2021

Abstrakt

Cieľom tejto práce je popísať a vytvoriť metódu pre korekciu výstupov rozpoznávača textu pomocou rozpoznávača reči. Práca popisuje prehľad súčasných metód pre rozpoznávanie textu a reči pomocou neurónových sietí. Popisuje tiež existujúce metódy prepájania výstupov dvoch modalít. V rámci práce je navrhnutých a implementovaných niekoľko prístupov pre korekciu rozpoznávačov, ktoré sú založené na algoritmoch, alebo neurónových sieťach. Ako najlepší prístup sa ukázal algoritmus založený na princípe prehľadávania výstupov rozpoznávačov zarovnaných pomocou levenshtainového zarovnania. Algoritmus prehľadáva výstupy v prípade že neistota znaku rozpoznávača textu je menšia ako predom zvolená hranica. V rámci práce bol ku textovým prepisom vytvorený anotačný server, pomocou ktorého sa robil zber nahrávok pre vyhodnotenie experimentov.

Abstract

The aim of this thesis is to describe and create a method for correcting text recognizer outputs using speech recognition. The thesis presents an overview of current methods for text and speech recognition using neural networks. It also presents a few existing methods of connecting the outputs of two modalities. Within the thesis, several approaches for the correction of recognizers, which are based on algorithms or neural networks, are designed and implemented. An algorithm based on the principle of searching the outputs of recognizers using levenshtain alignment was proven to be the best approach. It scans the outputs, if the uncertainty of the text recognizer character is less than the pre-selected limit. As part of the work, an annotation server was created for the text transcripts, which was used to collect recordings for the evaluation of experiments.

Kľúčové slová

rozpoznávanie reči, rozpoznávanie textu, multimodálny prepis, neurónové siete, anotačný server, prepis textu, prepájanie písaného textu a reči, korekcia výstupu rozpoznávačov, multimodálny systém

Keywords

automatic speech recognition, automatic text recognition, multimodal transcription, neural network, annotation server, text recognition, connection between handwriting text and speech, correction output of recognizers, multimodal system

Citácia

KABÁČ, Michal. *Multi-modální přepis textu.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Kišš

Multi-modální přepis textu

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Kišša. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Kabáč
18.mája 2022

Poďakovanie

Chcel by som podakovať pánovi Ing. Martinovi Kiššovi za odbornú pomoc a rady pri vedení práce, ktoré mi pomohli túto prácu dokončiť.

Obsah

Kapitola 1

Úvod

V posledných rokoch vývoj neurónových sietí značne napreduje. Vďaka tomu neurónové siete našli uplatnenie pri riešení rôznych problémov ako napríklad: rozpoznávanie obrázkov, odhaľovanie dezinformácií, ale aj rozpoznávanie textu a reči, na ktoré je táto práca zameraná. Rozpoznávanie textu je proces, pri ktorom sa menia znaky do podoby, ktoré sú zrozumiteľné pre počítač [26].

Pomocou rozpoznania textu vieme ušetriť čas a financie v oblastiach ako napríklad: rozpoznávanie evidenčných čísel vozidiel [30], rozpoznávanie textu v dotazníkoch [11], rozpoznávanie historických dokumentov [23], rozpoznávanie reči pre komentovanie lekárom [27].

Rozpoznávanie evidenčných čísel vozidiel. Úloha rozpoznávania evidenčných čísel vozidiel je využívaná najmä v policajných zložkách [30]. Tento systém funguje na princípe kamery, ktorá nasníma danú značku vozidla. Následne sa pomocou systému značka rozpozná. Tieto systémy sú tiež používané aj na diaľniciach, kde pomáhajú pri zisťovaní platnosti diaľničnej známky.

Rozpoznávanie textu v dotazníkoch. Rozpoznávanie textu môže pomôcť pri vyhodnocovaní rukou písaných formulárov [11]. Rozpoznávanie v prípade dotazníkov môže byť v niektorých prípadoch náročnejšie, ako rozpoznávanie dokumentov a dopisov z dôvodu rôznych predtlačených a ohraničujúcich políčok (angl. bounding box). Ukázalo sa že dotazníky v porovnaní s dopismi obsahujú viac šumu z dôvodu väčšieho množstva prepisovania a opravovania odpovedí [11].

Rozpoznávanie historických dokumentov. Rozpoznávanie textu môže byť užitočné aj pri prepise textu z fotiek historických dokumentov do digitálnej podoby [23]. Rozpoznávanie textu v historických dokumentoch je náročná úloha z dôvodu rôznych štýlov, hrúbky a tvaru písma, ako aj rôznych typov pozadia. Digitalizácia historického textu má významný vplyv pre zachovanie kultúrneho dedičstva.

Rozpoznávanie reči pre komentovanie lekárom. Prepis hovorenej reči do textovej podoby môže výrazne skrátiť dobu spracovania hlásenia lekárom [27]. Nevýhoda v tejto oblasti spočíva v chybách, ktoré môže systém urobiť. Takýto text musí byť následne skontrolovaný a opravený, nakoľko chyba pri odbornom medicínskom prepise môže mať závažné následky. Systémy na prepis reči do textu boli testované vo viacerých amerických nemoc-

niciach [28, 27].

Rozpoznávanie a prepis textu je náročná úloha [31]. Charakteristické črty typu písma a výslovnosti sú pre každého jedinca iné. Rovnako aj podmienky a spôsob vzniku rôznych dokumentov je odlišný. Na jeho kvalitu môžu vplývať rôzne faktory. Jedným z najpoužívanejších prístupov pre rozpoznávanie textu sú rekurentné siete [17].

Táto práca je zameraná na korekciu výstupu neurónových sietí rozpoznávača textu, pomocou rozpoznávača reči. V niektorých prípadoch môže byť pre používateľa jednoduchšie nahratie hlasovej nahrávky ako manuálna korekcia výstupu. Tento prístup môže byť vhodný aj v prípade veľkého počtu chýb pri výstupe rozpoznávača textu, alebo fyzickej neschopnosti používateľa manuálne opraviť nahrávky napríklad z dôvodu úrazu.
Pri rozpoznávaní textu je potrebné poznať základné architektúry neurónových sietí. Tieto architektúry sú popísané v kapitole 2.1. Takto kapitola tiež popisuje CTC funkciu, ktorá sa používa pri trénovaní modelov a popisuje existujúce metódy prepojenia výstupov rozpoznávačov. Kapitola 3 popisuje navrhnuté metódy, ako je možné túto problematiku riešiť. V kapitole 4 sú opísané niektoré implementačné detaily ktoré sú súčasťou implementácie. Ďalšia kapitola 5 popisuje vyhodnocovanie a výsledky jednotlivých experimentov ako aj celkové zhodnotenie a porovnanie výsledkov.

Kapitola 2

Rozpoznávanie textu a reči pomocou neurónových sietí

Táto kapitola predstavuje základné prístupy a architektúry pre rozpoznávanie reči. Analyzujú sa v nej aj metódy prepojenia rozpoznávačov. Nasledujúca časť predstavuje najznámejšie neurónové siete a princípy, slúžiace ako stavebné prvky architektúr pre rozpoznávanie reči a textu.

Skryté markovové modely (angl. Hidden markov model - HMM). Skryté markovove modely boli jednou z najčastejšie používaných parametrických štatistických metód začiatkom roku 2000 [24]. Skryté markovové modely modelujú údaje, ktoré predpokladajú skryté stavy v markovovom modeli, kde pravdepodobnosť jedného stavu závisí od jeho predchádzajúceho stavu. Skrytý markov model je definovaný ako stochastický proces generovaný dvoma vzájomne súvisiacimi mechanizmami - Markovov reťazec s konečným počtom stavov a množinou náhodných funkcií, kde je každá prepojená so stavom [4].

V diskrétnych časových okamihoch sa predpokladá, že proces je v nejakom stave a pozorovanie je generované náhodnou funkciou zodpovedajúcou aktuálnemu stavu [4]. Markovov reťazec potom mení stavy na základe svojich pravdepodobností prechodu. Tuto nastáva úloha zostavenia modelu, ktorý vysvetľuje a charakterizuje výskyt pozorovaných symbolov [6]. Výstup zodpovedajúci jednému symbolu môžno reprezentovať ako diskrétny, alebo spojitý. Diskrétne jednotky môžu byť znaky z konečnej abecedy, alebo kvantované výstupy z číselníka kódov zatiaľ čo spojité hodnoty nadobúdajú hodnoty zo spojitého priebehu. Pri generovaní slova alebo znaku systém prechádza z jedného stavu do ďalšieho, pričom každý stav vydáva výstup podľa určitej pravdepodobnosti až kým nie je celé slovo alebo znak vygenerovaný.

Existujú 2 základné prístupy na rozpoznávanie slov používajúce HMM [4]: 1. HMM diskriminujúci model. Pre každú triedu je vo fáze trénovania vytvorený model. Stavy v skrytom markovovom modeli predstavujú zhlukové centrá pre priestorové črty. Cieľom klasifikácie je následne rozhodnutie modelu, ktorý vytvoril neznámu sekvenciu pozorovaní. Každý stĺpec obrázku slova, je reprezentovaný ako vektor čŕt, ktorý je označením pozície každého pixlu. Pre každý znak je natrénovaný osobitný model na obrázkoch slov, pričom identifikuje hranice medzi začiatkom a koncom každého znaku. Následne nad každým slovom v súbore údajov sú použité jednotlivé modely na základe procesu spájania slov. Výhodou tohoto prístupu je nerozdeľovanie slov na znaky pre proces porovnávania [5].

2. HMM diskriminujúci cestu [5]. V tomto prípade je skonštruovaný jediný skrytý markovov model pre celý jazyk alebo kontext. Stav môže byť reprezentovaný znakom, častou znaku, alebo zreťazenými znakmi. Rozpoznanie potom vychádza z najpravdepodobnejšej cesty pre každú triedu použitím Viterbiho algoritmu.

Konvolučné neurónové siete sú špecializovaným druhom neurónových sietí, slúžiacich na spracovanie dát, ktoré majú štruktúru podobnú mriežke [13]. Svoj názov dostali podľa operácie *konvolúcie*, ktorá sa v nich vykonáva.

Konvolúcia je matematická operácia na funkciách [13]. Z pohľadu strojového učenia je prvá funkcia braná ako *vstup* a druhá funkcia je braná ako *filter*. Pod pojmom vstup rozumieme viacrozmerné pole údajov. Pojem funkcia definujeme ako viacrozmerné pole parametrov, ktoré sú prispôsobené algoritmom učenia. Viacrozmerné polia môžeme pomenovať ako tensory. Operácia konvolúcie je definovaná vzorcom:

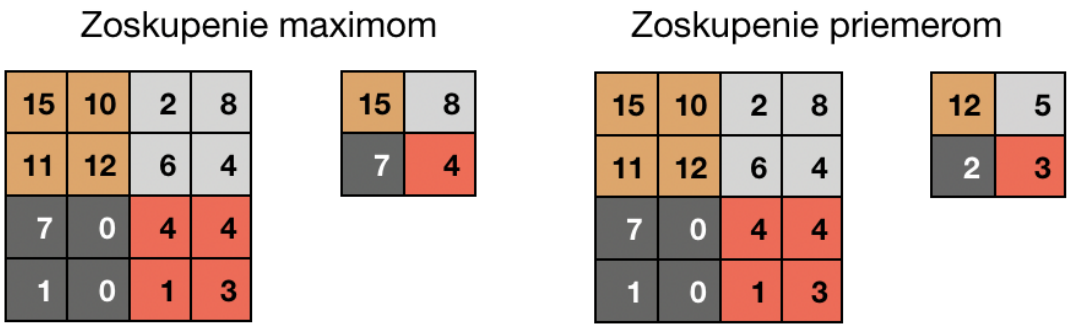
$$K(i,j) = (F * V)(i,j) = \sum_m \sum_n V(i-m, j-n)F(m,n) \tag{2.1}$$

kde K reprezentuje počítanú hodnotu na pozícii i a j, F je filter, V je vstup, m je šírka vstupnej matice a n je výška vstupnej matice. Veľkosť filtra býva menšia ako veľkosť matice. Pri násobení vstupu filtrom definujeme krok, ktorý nám reprezentuje o koľko sa posunieme na vstupných dátach pri výpočte.

Funkcia konvolúcie sa pri strojovom učení nepoužíva sama, ale kombinuje sa s inými funkciami [13]. Výstupy operácie konvolúcie, prejdú nelineárnou aktivačnou funkciou. Príkladom takejto nelineárnej aktivačnej funkcie je $ReLU(x) = max(0, x)$, ktorá vráti vypočítanú hodnotu konvolúcie v prípade že je kladná, inak vráti nulu. Po aktivačnej vrstve použijeme funkciu zoskupenia (angl. pooling). Úlohou tejto funkcie je redukcia rozmerov vstupnej matice. Základné funkcie ktoré používame pri zoskupovaní sú maximum, kde zo združovacej oblasti vyberáme maximálnu hodnotu a priemer pri ktorom je pre združovaciu oblasť počítaná priemerná hodnota. Rozdiel pri použití zoskupenia maximom a priemerom môžeme vidieť na obrázku 2.1

Plne prepojená vrstva je ďalšia vrstva konvolučných neurónových sieti [2]. Jej architektúra je podobná tradičným neurónovým sieťam. Pre každý neurón platí, že je prepojený s každým neurónom v predchádzajúcej vrstve a s každým neurónom v nasledujúcej vrstve. Veľký počet zapamätateľných parametrov, ktoré potrebujú komplexné výpočty pri trénovaní je jedna z nevýhod plne prepojenej vrstvy.

Rekurentné neurónové siete (angl. Reccurent neural network) sú neurónové siete určené najmä na spracovávanie sekvencií [29]. Rovnako ako konvolučné neurónové siete, aj rekurentné neurónové siete sa špecializujú na dáta so štruktúrou mriežky [13]. Ich výhoda spočíva v zdieľaní informácií, vďaka čomu majú schopnosť lepšej generalizácie. Konvolučné neurónové siete majú v porovnaní s rekurentnými neurónovými sieťami veľmi nízku schopnosť zdieľania informácií. Nakoľko pri konvolučných neurónových sieťach je výstup funkcia malého počtu okolitých vstupov. Táto informácia zdieľania informácií pri konvolučných neurónových sieťach má zakaždým rovnaký tvar a množstvo zdieľaných informácií záleží na veľkosti filtra. Pri rekurentných neurónových sieťach je každý člen výstupu funkciou predchádzajúcich výstupov. Každý výstup je vytvorený použitím rovnakého aktualizačného pravidla použitého na predošlý vstup. Výsledkom tejto opakujúcej sa operácie je zdieľanie



Obr. 2.1: Ukážka zoskupenia pri nastavení maximálnej a priemernej hodnoty.

parametrov prostredníctvom hlbokého výpočtového grafu. Pomocou rekurentných sieti je možné modelovať rôzne závislosti v texte.

Základným stavebným prvkom rekurentných neurónových sietí sú rekurentné bunky [18]. Tieto bunky môžu mať rôzne verzie a modifikácie. Najčastejšie sa používa nelineárna funkcia mapujúca priamy súčet (alebo zreťazenie) prichádzajúceho skrytého stavu h_{t-1} a vstupného vektora X_{t-1} na skrytý výstupný vektor h_t s dimenziou d_h daného vzorcom:

$$h_t = f(W[h_{t-1}; X_{t-1}] + b), \tag{2.2}$$

kde h_t je skrytý vstup vektora, f je nelineárna aktivačná funkcia, parametre rekurentnej neurónovej siete sú dané maticou váh $W \in \mathbb{R}^{d_h \times (d_h + d_v)}$, kde d_v je vstupná dimenzia reprezentujúca počet možných hodnôt ktoré môže vstup X_t nadobúdať a b je bias $b \in \mathbb{R}^{d_h}$. Výpočet plnej pravdepodobnosti $P(X)$ sa vykonáva sekvenčným výpočtom začínajúcim na $P(X_1)$, nakoľko hodnota $P(X_0)$ sa nastavuje ako konštanta. Výpočet plnej pravdepodobnosti dostávame výpočtom podmienenej pravdepodobnosti

$$P(X_t|X_{t-1}, ..., X_1) = Y_t \cdot X_t, \tag{2.3}$$

kde pravá strana reprezentuje skalárny súčin medzi vektormi a Y_t je dané vzťahom:

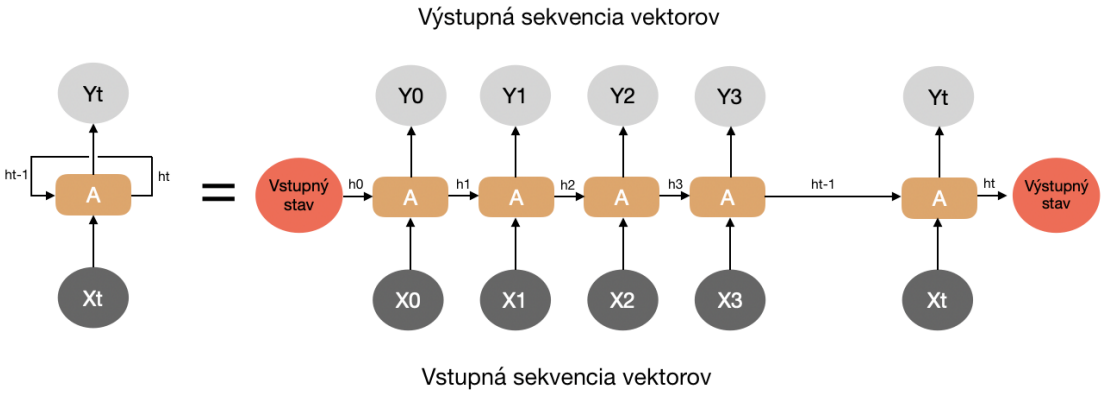
$$Y_t \equiv S(Uh_t + c), \tag{2.4}$$

kde $U \in \mathbb{R}^{d_h \times d_v}$ a $c \in \mathbb{R}^{d_v}$ sú váhy a biasy prislúchajúce k vrstve *Softmax* a jej aktivačná funkcia S je daná nasledovne

$$S(v_t) = \frac{exp(v_t)}{\sum_i exp(v_i))} \tag{2.5}$$

a $Y_t = (Y_t^1, ..., Y_t^{d_v})$ je d_v rozmerný vektor kladných reálnych čísel sumujúcich sa do 1. Zo vzorcov h_t a Y_t vyplýva, že skrytý vektor h_t kóduje informáciu o predchádzajúcich konfiguráciach vstupu $X_{<t}$. Vďaka zdieľaniu skrytých stavov je rekurentná neurónová sieť schopná modelovať silne korelované rozdelenia. Architektúra rekurentnej jednovrstvovej siete je znázornená na obrázku 2.2.

Nevýhodou rekurentných neurónových sietí je slabé šírenie informácií pri dlhých sekvenciách [21]. Pri trénovaní rekurentných sietí môže nastať problém miznúceho gradientu (angl. vanishing gradient problem). Problém nastáva v prípade dosiahnutia malých hodnôt gradientov dôvodom čoho nenastane zmena váhy neurónu. Tento problém môže spôsobiť úplne



Obr. 2.2: Ukážka architektúry rekurentnej neurónovej siete, kde oranžová farba reprezentuje bunku rekurentnej neurónovej siete. Jej vstup tvorí vstupný vektor X_t spolu so skrytým stavom h_t. Bunka A vygeneruje skrytý stav h_{t+1}. Tento stav je vstupom ďalšej rekurentnej bunky a zároveň je vstupom pre Softmax, ktorý vypočíta podmienené pravdepodobnosti vrátené vo vektore Y_t.

zastavenie trénovania siete.

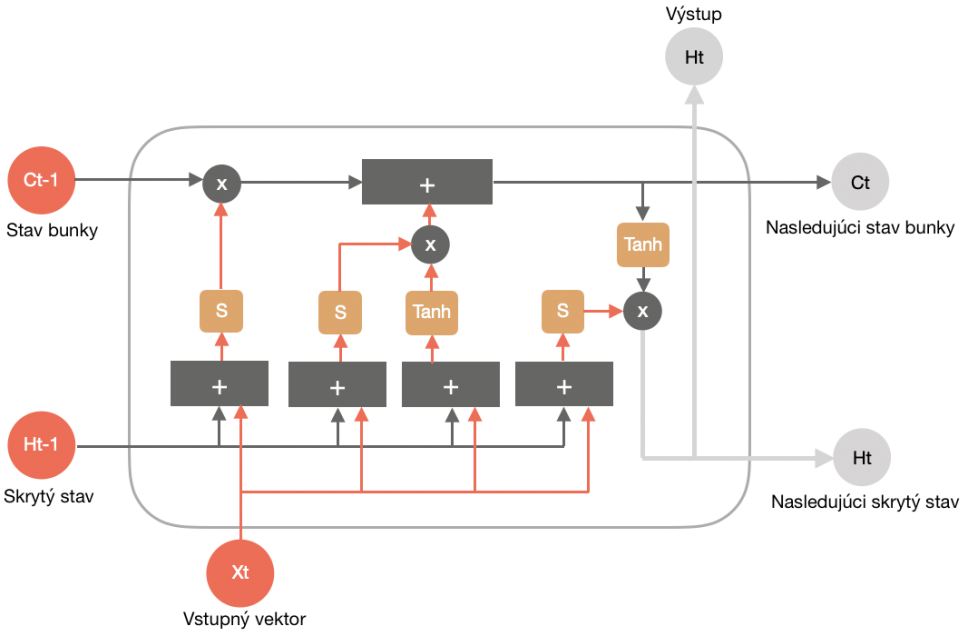
Long-short term memory (LSTM) patria medzi rekurentné neurónové siete, špecializujúce sa na riešenie sekvenčných problémov [17]. LSTM sú siete, ktoré dokážu riešiť problém miznúceho gradientu. Základnou myšlienkou LSTM je pamäťová bunka udržujúca si stav v priebehu času, ktorá obsahuje hradlové jednotky (angl. gates) - *input gate*, *output gate*, *reset gate*. Hradlové jednotky regulujú množstvo informácií vstupujúcich do bunky a množstvo výstupných informácií. Na obrázku 2.3 môžeme vidieť zobrazenie pamäťovej bunky LSTM.

Vstupom do pamäťovej bunky sú stavy, ktoré prichádzajú z predchádzajúcej bunky - stav bunky a skrytý stav [22]. Stav bunky umožňuje tok dát v nezmenenej forme. Môžu však nastať lineárne transformácie ktoré pridajú alebo odoberú dáta prostredníctvom sigmoidálnych hradiel. V týchto hradlách sa vykonávajú maticové operácie, ktoré obsahujú rôzne váhy. Dôležitosť informácií a fakt či sa majú informácie vynechať alebo nie rozhoduje sigmoidálna funkcia. Vstupom tejto sigmoidálnej funkcie je výstup skrytého stavu predchádzajúcej LSTM bunky H_{t-1} a aktuálny vstupný vektor X_t. Sigmoidálna funkcia môže tiež vybrať úsek údajov, ktoré majú byť zabudnuté. Zabudnutie údajov umožnuje forget gate. Vzorec pre forget gate je daný vzťahom:

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f), \tag{2.6}$$

kde F_t symbolizuje forget gate ktoré nadobúda hodnoty z rozsahu 0 až 1, σ je sigmoidálna funkcia, W_f je váhová matica a b_f je matica biasov.

Ďalší krok uloží údaje z nového vstupu X_t do stavu bunky a aktualizuje stav bunky [22]. Táto časť sa skladá z 2 funkcií - sigmoidálne a tangesovej. Najskôr sigmoidálna funkcia rozhodne o uchovaní alebo ignorácií vstupných údajov. Potom tangesová funkcia priradí týmto údajom váhu, ktorá hovorí o ich dôležitosti. Táto váha je z rozmedzia -1 až 1. Následne sa sigmoidálna a tangesová funkcia vynásobia. Výsledná hodnota sa uloží ako nový stav bunky. Nový stav bunky sa pridá ku starému stavu bunky, čím vzniká stav C_t V poslednom kroku sigmoidálna funkcia rozhodne, ktorá časť stavu bunky sa dostane na výstup [22]. Potom je sigmoidálna funkcia násobená spolu s tangesovou funkciou. Výsledok tejto operácie tvorí



Obr. 2.3: Zobrazenie pamäťovej bunky LSTM. Čierna farba reprezentuje maticové a vektorové operácie. Oranžová farba reprezentuje sigmoidálne a tangesové funkcie. Červená farba reprezentuje vstupné hodnoty pre bunku LSTM a sivá farba reprezentuje výstupné hodnoty z bunky.

výstup bunky a zároveň slúži ako vstupný stav ďalšej pamäťovej bunky LSTM.

Gated recurrent unit (GRU) patri medzi rekurentné neurónové siete a rovnako ako LSTM dokáže riešiť problém miznúceho gradientu [19]. Od LSTM sa líši v počte hradlových jednotiek [12]. GRU obsahuje dve hradlové jednotky, ktoré sa nazývajú *update gate* a *reset gate*. Reset gate rozhoduje o kombinácii predchádzajúceho vnútorného stavu s aktuálnym vstupom. Výstupná hodnota r_t pre reset gate je daná vzťahom

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2.7}$$

a výstupná hodnota z_t pre update gate je daná vzťahom

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z),$$

kde W_r,W_z sú matice váh pre príslušný vstupný vektor a U_r,U_z predstavujú matice predchádzajúceho časového kroku. Výsledný výstup h_t je potom vypočítaný ako

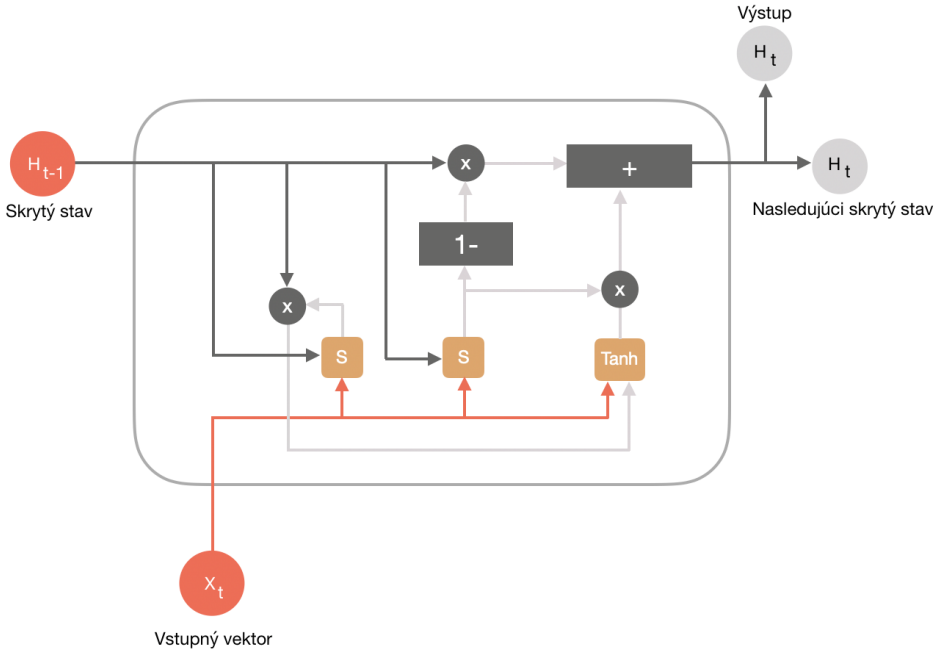
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.8}$$

$$\tilde{h}_t = g(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h). \tag{2.9}$$

Bunka GRU je znázornená na obrázku 2.4.

2.1 Rozpoznávanie rukou písaného textu

V dnešnej dobe je rukou písaný text všade prítomný a preto má jeho rozpoznávanie prakticky neoceniteľnú hodnotu [24]. Rozpoznávanie písaného textu umožňuje preklad rôznych



Obr. 2.4: Zobrazenie všeobecnej schémy bunky GRU.

druhov dokumentov na analyzovateľné a upravovateľné dáta, v ktorých je možné rýchle vyhľadávanie. Tieto dokumenty sú najčastejšie pomocou umelej inteligencie prekladané do elektronického formátu. Pre rozpoznávanie rukou písaného textu existujú rôzne architektúry neurónových sieti.

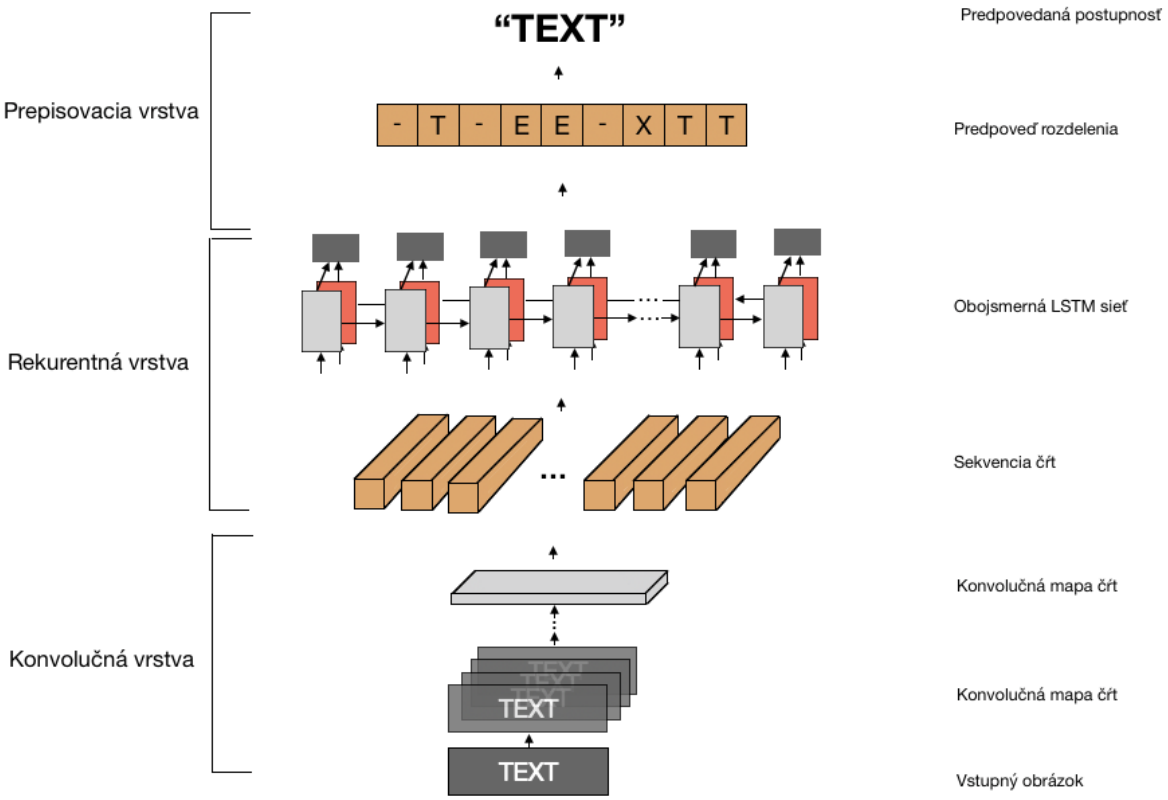
Architektúry pre rozpoznávanie rukou písaného textu

Základ pre nasledujúce architektúry tvoria konvolučné neurónové siete, pomocou ktorých sa spracujú črty. Črty získané konvolučnými neurónovými sieťami tvoria vstupy pre ďalšie architektúry.

Shi a spol. sa vo svojej práci [29] zamerali na rozpoznávanie sekvenčných objektov z obrázkov. Ich navrhnutý model tvorila konvolučná rekurentná sieť, ktorá je kombináciou konvolučných neurónových sietí a rekurentných neurónových sietí. Medzi výhody konvolučných rekurentných neurónových sietí, ktoré opisujú v ich práci patria 1) dokážu sa učiť priamo zo sekvencií a nevyžadujú podrobné označkovanie (napríklad hranice znakov), 2) dokážu sa učiť priamo z obrázkov, nepotrebujú žiadne predom vytvorené črty, 3) dokážu spracovať dlhé sekvencie za použitia výškovej normalizácie.

Predstavili tiež architektúru konvolučných rekurentných neurónových sietí, ktorá sa skladala z 3 komponentov - konvolučné vrstvy, rekurentné vrstvy a prepisovacia vrstva [29]. Ukážku takejto architektúry môžeme vidieť na obrázku 2.5. Konvolučné vrstvy získajú sekvenciu čŕt z každého obrázka. Výstup konvolučnej neurónovej siete tvorí vstup rekurentnej neurónovej siete, ktorá dáva predpoveď pre každý rámec sekvencie čŕt. Prepisovacia vrstva prekladá každý predpovedaný rámec z rekurentnej vrstvy do označkovanej postupnosti.

Pri extrakcii čŕt používali zoskupujúce vrstvy konvolučnej neurónovej siete s nastavením maximálneho zoskupenia [29]. Pri takejto architektúre je potrebná normalizácia výšky pre



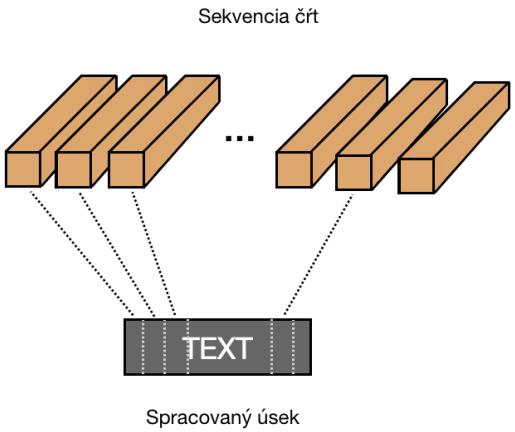
Obr. 2.5: Architektúra konvolučno rekurentnej neurónovej siete.

každý obrázok. Spracovaním obrázka sa dostanú konvolučné mapy čŕt z konvolučnej vrstvy, ktorá tvorí vstup rekurentnej vrstvy. Ako je zobrazené na obrázku 2.6, každý vektor zo sekvencie čŕt reprezentuje spracovaný úsek z pôvodného obrázka.

V prepisovacej vrstve sa pre každú predpoveď zo sekvencie čŕt, ktorá je výstupom rekurentnej vrstvy, priraďuje značka [29]. Z matematického pohľadu ide o výber najpravdepodobnejšej značky danej sekvencie. Takúto architektúru je možné trénovať pomocou *Connectionist Temporal Classification* (CTC), ktorá je bližšie špecifikovaná v kapitole 2.3.

Sequence to sequence model (Seq2seq) funguje na princípe enkódera a dekódera. Ako jeden z možných prístupov pre seq2seq model založený na rozpoznávaní textu bol predstavený v práci *Kang a spol* [20]. Ako enkóder bola použitá konvolučná a rekurentnú neurónovú sieť. Konvolučná neurónovú sieť extrahuje črty z rukou písaných obrázkov. Tieto črty slúžia ako vstup rekurentnej neurónovej sieti. Dekóder obsahoval rekurentnú neurónovú sieť, ktorá v každom kroku rozpozná 1 znak zo vstupnej postupnosti. Ako architektúra konvolučnej neurónovej siete bola použitá VGG-19-BN, ktorá má vopred predtrénované váhy na ImageNet. Za konvolučnou neurónovou sieťou nasledovala viacvrstvová obojsmerná rekurentná sieť (BGRU), ktorá zahrňuje vzájomné informácie a polohové informácie pre každý stĺpec a enkóduje sekvenciu rukou písaného textu.

Za enkóderom bol použitý *attention* mechanizmus [20]. *Kang a spol.* opísali dva hlavné attention mechanizmy pri rozpoznávaní textu - attention založená na obsahu a attention založená na polohe. Attention založená na obsahu je zameraná na podobnosť medzi aktuál-

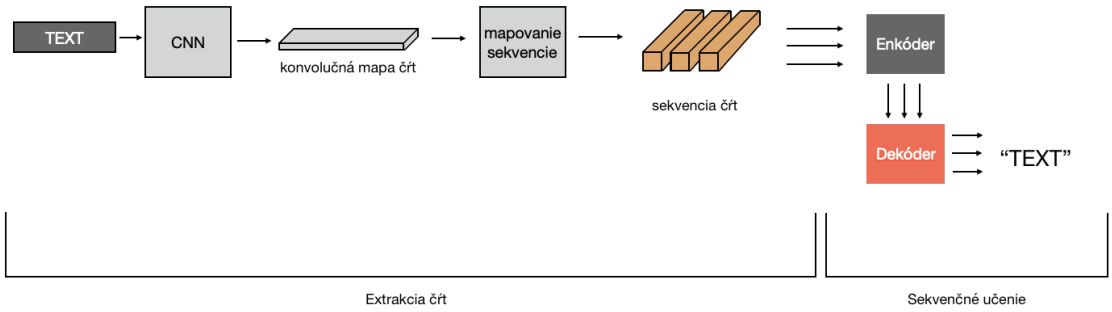


Obr. 2.6: Zobrazenie sekvencie čŕt a spracovaného úseku.

nym skrytým stavom a mapou čŕt z enkódera. Taktiež očakáva informácie o polohe zahrnuté vo výstupe enkódera. Attention založená na polohe explicitne berie do úvahy informácie o polohe. Výstupom attention je kontextový vektor, ktorý dekóderu pomáha pri predikovaní aktuálneho stavu.

Dekóder môže mať architektúru jednosmernej viacvrstvová GRU [20]. Vstupom GRU je zreťazenie *embedding vecktora* z predchádzajúceho vstupu a kontextového vektora. Embedding vektor pre každý znak zo slovnej zásoby pochádza z matice vyhľadávacej tabuľky (angl. lookup table matrix). Matica je na začiatku náhodne inicializovaná a v priebehu trénovania je upravovaná. Začiatok činnosti dekódera vždy začína štartovacím signálom ⟨GO⟩, ktorý sa nachádza ako prvý znak na vstupe. Dekóder končí dekódovací proces po obdržaní koncového signálu ⟨EOS⟩, alebo po dosiahnutí maximálneho počtu krokov T. Pomocou pamäťovej jednotky GRU dekóder využíva informáciu o predchádzajúcom znaku, čo pomáha pri predikovaní aktuálneho znaku.

Pri rozpoznávaní textu môže byť použitá detekcia na detekovanie jednotlivých riadkov, na ktorých sa následne vykoná rozpoznávanie textu [32]. *Wigington a spol.* v ich
práci [32] navrhli model založený na 3 podmodeloch - SOL na vyhľadávanie začiatku riadka, LF na výber celého riadka a HWR na rozpoznávanie textu. Na nájdenie začiatku riadku bola použitá sieť založená na návrhu regiónov (angl. Region Proposal Network). Vstupom LF modelu sú súradnice predikované SOL modelom. LF model postupuje pozdĺž textu, kde nasleduje zakrivenia a vytvára normalizovaný textový obrázok. Pre každý detekovaný riadok je spustený model HWR, ktorý rozpoznáva text. Modele SOL používa architektúru siete VGG-11. Z tejto siete boli odstránená plne prepojené a koncové zoskupovacie vrstvy. Výstupom SOL siete sú predpovedané súradnice x, y začiatku riadku. Po predikcii súradníc bol na týchto súradniciach spustený model LF. Tento model v malých inkrementačných krokoch nasleduje text a vytvára obraz pre model HRW. Výhodou modelu LF je schopnosť pracovať aj na zakrivenom texte. LF má architektúru rekurentnej neurónovej siete. Vstupom tejto siete sú aktuálna pozícia x_i, y_i a uhol rotácie textu. Pomocou malého zobrazovacieho okna, ktoré má pevnú veľkosť sa spracuje daná pozícia. Výstup zobrazovacieho okna ide do konvolučnej neurónovej siete. Konvolučná neurónová sieť vracia novú pozíciu zobrazovacieho okna posunutú na ďalšiu oblasť, pokiaľ nenastane okraj obrázka. Pomocou HWR určili skutočný koniec riadku v texte.Výstup z podmodelu LF je vstupom pre model



Obr. 2.7: Zobrazenie architektúry rekurentnej enkóder-dekóder siete.

HWR. Model HWR používa CNN-LSTM neurónovú sieť. CNN vytvorí horizontálnu 1D sekvenciu ktorá je vstupom pre obojsmerný LSTM model. Na trénovanie HWR modelu použili CTC.

Chowdhury a spol. v ich práci [8] navrhli architektúru rekurentnej enkóder-dekóder siete. Náčrt tejto architektúry sa nachádza na obrázku 2.7. Na extrakciu čŕt použili konvolučnú neurónovú sieť. Vstupný obrázok transformovali do konvolučnej mapy čŕt. Konvolučná mapa čŕt je prevedená do sekvencie čŕt rovnako ako na obrázku 2.6.

V enkóderi bola z dôvodu miznúceho gradientu použitá obojsmernú LSTM sieť [8]. Detailné zobrazenie enkódera a dekódera sa nachádza na obrázku 2.8. Táto sieť produkuje v každom kroku výstupný kontextový vektor C_{Ts}, ktorý obsahuje výstup z attention. Ts reprezentuje dĺžku sekvencie. Tento výstupný vektor C_{Ts} tvorí vstup pre ďalšiu rekurentnú sieť nachádzajúcu sa v dekóderi.

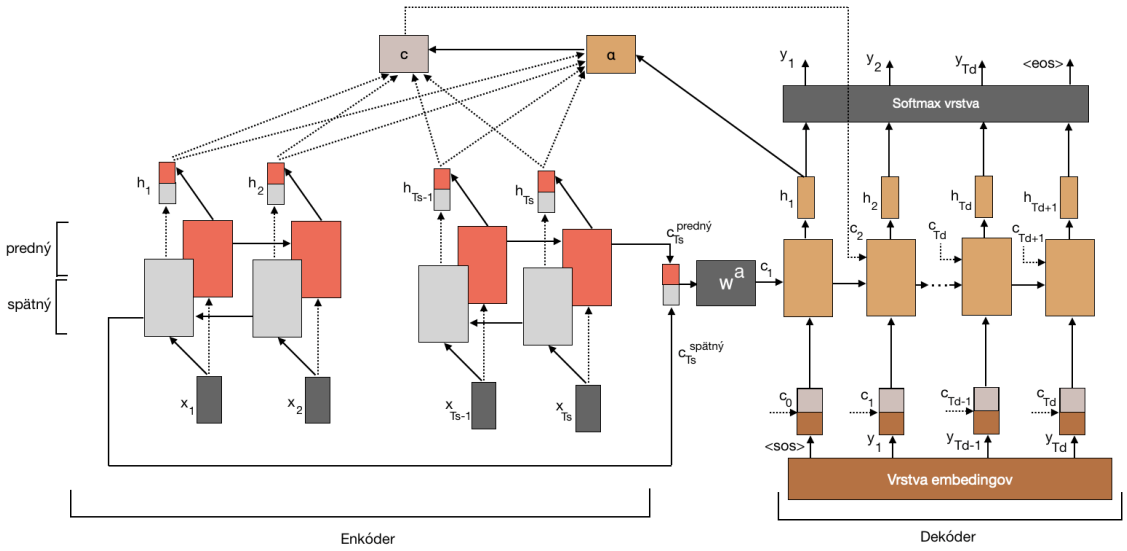
Dekóder je navrhnutý s jednosmernou LSTM sieťou z dôvodu generovania v každom kroku t token y_t výstupnej sekvencie [8]. Token y_t je podmienený c_1 a jeho vlastnými predchádzajúcimi predikciami $\{y_1, ..., y_{t-1}\}$. Jedná sa teda o podmienené rozdelenie pravdepodobnosti $p(y) = \prod_{t=1}^{T_d} p(y_t|\{y_1, ..., y_{t-1}\}, c_1)$ pre výslednú sekvenciu $\{y_1, ..., y_{T_d}\}$ kde T_d reprezentuje dĺžku sekvencie. Použitím rekurentných sietí je každá podmienka modelovaná ako $p(y_t|\{y_1, ..., y_{T_d}\}, c_1) = Softmax(g(y_t, h_{t-1}, c_1))$, kde g je nelineárna funkcia a h_{t-1} je skrytý stav rekurentnej siete. Kontextový vektor tvoril prepojenie medzi enkóderom a dekóderom. Z dôvodu nedostatočného šírenia informácií zo začiatku sekvencie pri dlhých sekvenciách bol upravený kontexotvý vektor. Kontextový vektor bol založený na podobnosti skrytých predchádzajúcich stavov dekódera, so sekvenciou $\{h_1^{encoder}, ..., h_{Ts}^{encoder}\}$, vygenerovanou enkóderom pre vstupnú sekvenciu. Kontextový vektor c_i pre i-ty krok je daný vzťahom

$$c_i = \sum_{j=1}^{Ts} \alpha_{ij} h_j^{encoder}, \tag{2.10}$$

kde váha α_{ij} pre každý $h_j^{encoder}$ bola daná ako

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{Ts} \exp(e_{ik})}, \tag{2.11}$$

kde $e_{ij} = a(h_{i-1}^{decoder}, h_j^{encoder})$, a je dopredná sieť. Kontextový vektor je teda upravený ako vážený súčet vstupných anotácií, kde váhy reprezentujú podobnosť výstupu na pozícii i s



Obr. 2.8: Detailenjšie zobrazenie enkódera a dekódera.

výstupom na pozícii j. Táto modifikácia sa ukázala ako vhodná pre dlhšie sekvencie. Dekóder pracuje dokým neobdrží token $< eos >$. Pre koncové dekódovanie použili Beam Search, ktorého úlohou je maximalizovať spoločnú pravdepodobnosť. Pri greedy prístupe, keď vyberáme najpravdepodobnejšiu značku pre daný krok t, nemusí byť táto značka správna. Algoritmus Beam Search vyberá K najlepších značiek a maximalizuje pravdepodobnosť až do príchodu $< eos >$ tokena. Potom na výstup posiela značky s najväčšou spoločnou pravdepodobnosťou.

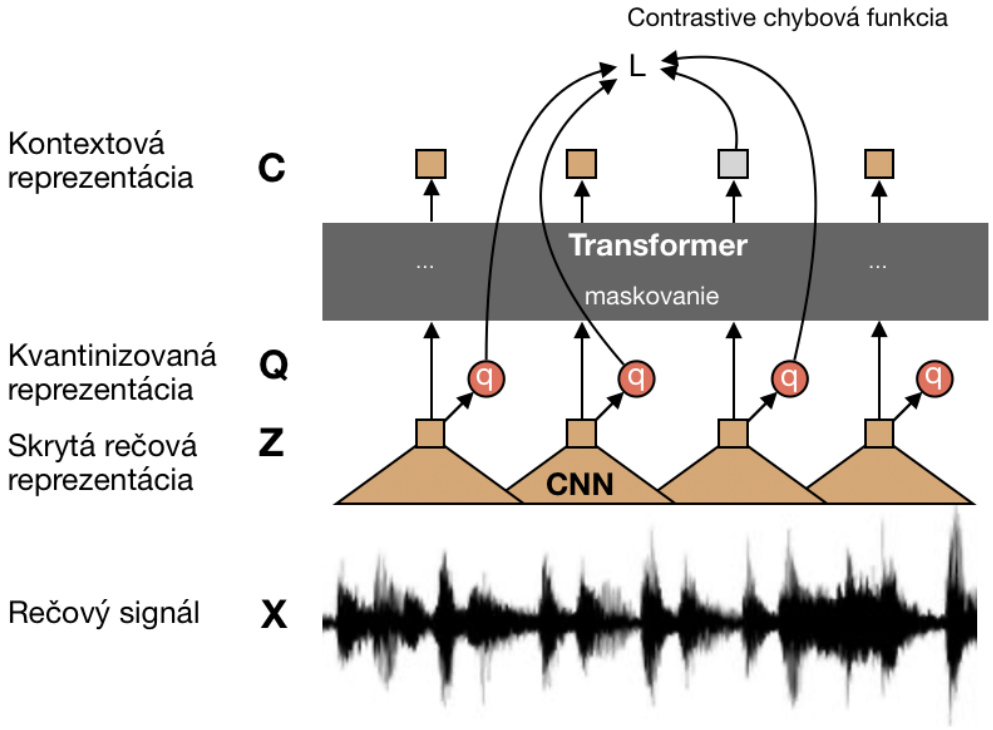
2.2 Rozpoznávanie hovorenej reči

Rozpoznávanie hovorenej reči slúži na prepis hovorenej ľudskej reči do textovej podoby [1]. Jedná sa o náročnú úlohu nakoľko hlasové prejavy rečníka môžu byť rôzne. Medzi faktory, ktoré môžu ovplyvniť kvalitu zvukového prejavu patria: nálada rečníka, tempo rozprávania, okolitý hluk, výška tónu rečníka a mnoho ďalších.

Architektúry pre rozpoznávanie reči

V minulosti sa na rozpoznávanie reči používali najmä skryté markovové modely a gausovské zmiešané modely [1]. Tieto prístupy sa používajú ešte aj dnes, no z časti boli nahradené prístupmi založenými na hlbokom učení [25]. Vo väčšine prípadov sú architektúry podobné architektúram opísaným v kapitole 2.1 s úpravami vstupu pre rozpoznávanie reči.

HMM sa pri rozpoznávaní reči používajú hlavne kvôli jednoduchosti a praktickosti [25]. Každý HMM používa pri reprezentácii zvukovej vlny gaussovské modely. Ako črty sa používaju MFCC príznaky (angl. Mel-Frequency Cepstral Coefficients). HMM dosahujú pri súvislých vetách malú úspešnosť a sú skôr vhodné na rozpoznávanie krátkych zvukových signálov, najčastejšie slov. Na rozdiel od rekurentných neurónových sietí nedokážu pri zvukovej nahrávke modelovať závislosti.



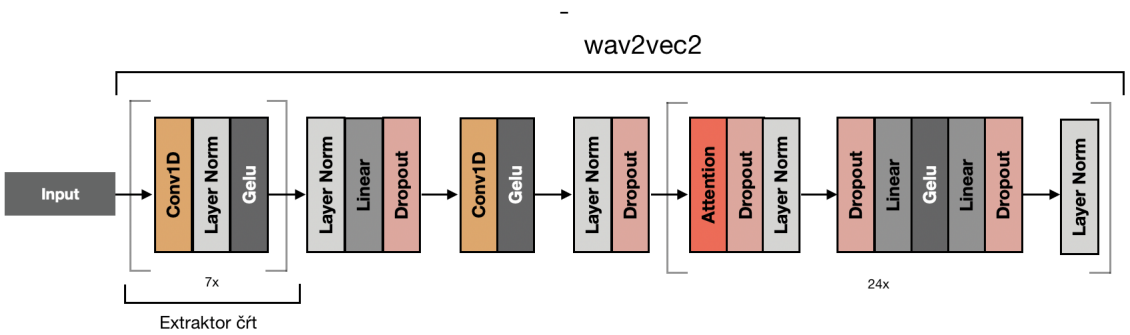
Obr. 2.9: Architektúra modelu wav2vec2.

RNN. Pre rozpoznávanie reči môže byť použitá architektúra rekurentných neurónových sietí [3]. Medzi najčastejšie prístupy patrí využitie LSTM sietí. Architektúry LSTM sieti boli vysvetlené v kapitole 2.1.

wav2vec2 je framework pre self-supervised učenie zameraného na reč [7]. Viacvrstvový konvolučný enkóder siete wav2vec2 pre extrakciu čŕt $f : X \mapsto Z$. má na vstupe nespracovanú reč X a jeho výstupom je skrytá reprezentácia reči $z_1, ..., z_T$ pre T časových krokov. Skrytá reprezentácia reči tvorí vstup transformera $g : Z \mapsto C$ pre vytvorenie kontextovej reprezentácie $c_1, ..., c_T$ slúžiacej pre zachytenie informácie celej sekvencie. Výstup enkódera je diskretizovaný do kvantinizovanej jednotky q_t pomocou kvantovacieho modulu $Z \mapsto Q$, ktorý má reprezentovať ciele pre self-supervised učenie. Model vytvára kontextové reprezentácie súvislej reči a pomocou pozornosti (angl. attention) zachytáva závislosť v celej sekvencii skrytých reprezentácií od začiatku až po koniec.

Enkóder pre extrakciu čŕt pozostáva z niekoľkých blokov obsahujúcich časovú konvolúciu po ktorej nasleduje normalizačná vrstva a aktivačná funkcia GELU [7]. Vstupný nespracovaný signál pre enkóder je normalizovaný na nulový priemer a jednotkovú varianciu. Celkový počet krokov (angl. stride) enkódera je určený počtom časových krokov T, ktoré sú na vstupe transformera.

Celý model bol predtrénovaný pomocou contrastive learningu L, pomocou maskovania. Maskovala sa časť výstupov enkódera, alebo časové kroky pred ich odovzaním do kontextovej siete. Cieľom bolo správne identifikovať správnu skrytú zvukovú reprezentáciu zo skutočnej a umelo vytvorených skrytých zvukových reprezentácií. Pri contrastive learningu bola ako chybová funkcia použitá Contrastive loss. Vstupy do kvantovacieho modulu sa



Obr. 2.10: Jednotlivé vrstvy modelu wav2vec2.

nemaskovali. Celý model bol dotrénovaný na označkovaných dátach tak, že nad kontextovú vrstvu bola pridaná náhodne inicializovaná lineárna vrstva s C triedami. Kde každá trieda reprezentovala písmeno z abecedy. Model bol následne optimalizovaný minimalizáciou CTC chybovej funkcie. Architektúru modelu je možné vidieť na obrázku 2.9. Jednotlivé vrstvy nachádzajúce sa v tejto architektúre môžeme vidieť na obrázku 2.10.

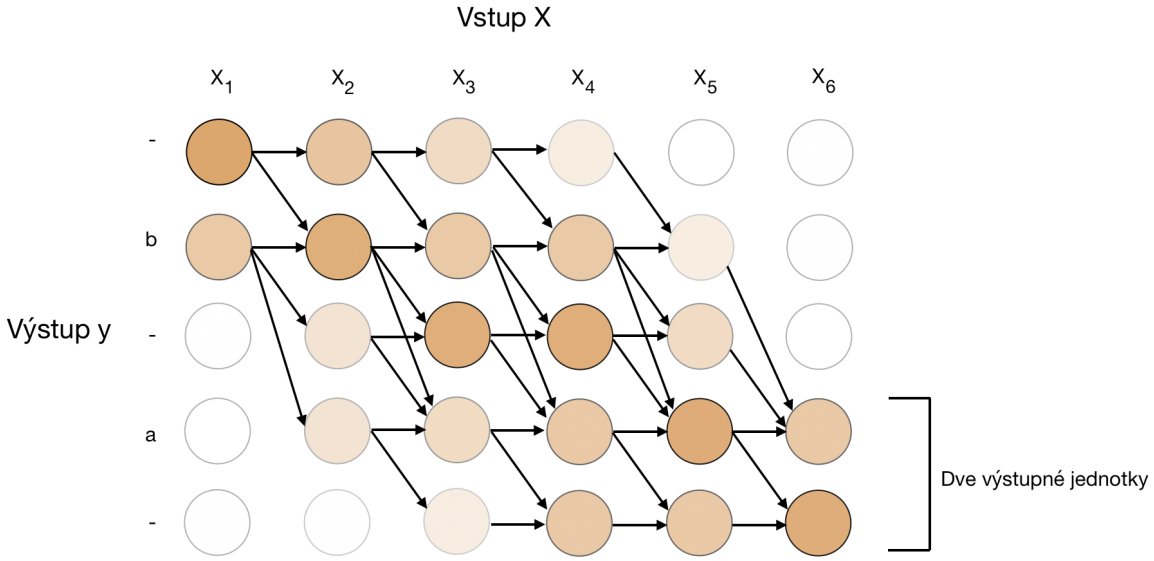
2.3 CTC funkcia

Pri úlohách rozpoznávania textu je použitie CTC funkcie časté [31, 29, 26, 16].Pre jej použitie sú potrebné označkované reťazce, ktoré sú označkované s diskrétnymi hodnotami ako napríklad slová alebo písmená [16]. Rozhodujúcim krokom pri CTC funkcii je transformácia výstupov siete, ktoré tvoria vstupy CTC funkcie, na podmienené rozdelenie pravdepodobností pre značkovanie sekvencie. Sieť potom predikuje najpravdepodobnejšie značky pre danú vstupnú sekvenciu a ako výstupnú vrstvu má $Softmax$ ktorá obsahuje o 1 jednotku viac ako je počet vstupných značiek abecedy L. Prvých L značiek abecedy je reprezentovaných ako pravdepodobnosť že v danom kroku t uvidím konkrétnu značku na vstupe. L+1 znak je reprezentovaný ako pravdepodobnosť prázdneho, alebo žiadneho znaku (angl. blank). Tieto výstupy definujú všetky možné pravdepodobnosti namapovania značiek na vstupnú sekvenciu. Výsledná pravdepodobnosť značkovanej sekvencie sa zistí ako súčet pravdepodobností jej rôznych zarovnaní.

Formálne sa pre vstupnú sekvenciu x dĺžky T definuje rekurentná neurónová sieť s m vstupmi, n výstupmi a váhovým vektorom w ako spojité namapovanie $\mathcal{N}_w : (\mathbb{R}^m)^T \mapsto (\mathbb{R}^n)^T$ [16]. Nech $y = \mathcal{N}_w(x)$ je sekvencia výstupov siete označená ako y_k^t aktivácia k-tej výstupnej jednotky v kroku t. Potom y_k^t je reprezentované ako pravdepodobnosť obdržania značky k v kroku t, ktorá definuje rozloženie cez množinu L'^T dĺžky T sekvencií v abecede $L' = L \cup \{blank\}$: $p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T$. Prvky L'^T sa označujú aj ako cesty π.

Sekvencia sa mapujeme spôsobom veľa do jedného (angl. many to one) [16]. Tento spôsob si označme ako $\mathcal{B}$. Pod týmto pojmom rozumieme že veľkosť pôvodnej sekvencie T bude mať dĺžku menšiu alebo rovnú T nad pôvodnou abecedou L. Toto dosiahneme odstránením opakujúcich sa znakov z cesty. Napríklad $\mathcal{B}(b-ba-) = \mathcal{B}(-bb---baa) = bba$. Z príkladu môžeme vidieť, že keď po sebe nasledujú 2 rovnaké znaky a sú oddelené symbolom prázdneho znaku -, tieto symboly budú oba ponechané.

Na dekódovanie sekvencie slúži metóda najlepšie dekódovanej cesty [16]. Táto metóda je založená na predpoklade, že najpravdepodobnejšia cesta bude zodpovedať najpravdepo-



Obr. 2.11: Zobrazenie výpočtov CTC použitím dynamického programovania. Čím je farba silnejšia, tým je väčšia pravdepodobnosť danej jednotky.

dobnejšiemu označkovaniu $h(x) \approx \mathcal{B}(\pi^*)$, kde

$$\pi^* = \underset{\pi \in N^t}{\operatorname{argmax}}\, p(\pi|x), \tag{2.12}$$

kde π^* je zreťazenie najpravdepodobnejších značiek v danom kroku t.

CTC dopredno spätný algoritmus (angl. CTC forward-backward algorithm) je algoritmus založený na dynamickom programovaní [16]. Túto myšlienku môžeme vidieť aj na obrázku 2.11. Na obrázku môžeme vidieť že každé možné zarovnanie má svoju cestu. Intenzita farby symbolizuje pravdepodobnosť značky t v kroku x_t. Môžeme vidieť že dekódované slovo by obsahovalo znaky "*ba*". Hlavnou myšlienkou algoritmu je že suma cez cesty zodpovedajúce označkovaniu môže byť rozdelená na iteratívny súčet prefixov tohoto označkovania. Iterácia môže byť potom efektívne vypočítaná pomocou rekurzívnych dopredných a spätných premenných.

2.4 Multimodálny systém

Pod pojmom multimodálny systém rozumieme systém, ktorý sa skladá aspoň z 2 modalít a tieto modality sa snažíme medzi sebou prepojiť [14]. Keďže rozpoznávanie písaného textu a rozpoznávanie reči majú podobný proces rozpoznávania, môžeme oba tieto systémy medzi sebou kombinovať [15]. Dekódovanie takéhoto systému, v našom prípade reč a rukou písaný text, je náročný proces kvôli časovej asynchronizácii medzi týmito signálmi. Príkladom môže byť rôzna dĺžka každej sekvencie a náročnosť nájdenia časového bodu, kde sú rovnaké slová synchronizované.

V [15] rozdelili rozpoznávanie pre multimodálny systém na 3 základné časti: 1) rozpoznávanie v čase extrakcie, 2) rozpoznávanie vo vyhľadávacom procese, 3) rozpoznávanie pri dekódovaní.

Rozpoznávanie v čase extrakcie: kombinácia čŕt sa vytvorí zreťazením rôznych čŕt na úrovni vektora, aby sa vytvorila nová sekvencia vektora čŕt ktorá sa použije pri rozpoznávaní [15].

Rozpoznávanie vo vyhľadávacom procese: Pri tejto časti sa využíva kombinovanie pravdepodobnosti tried rozpoznania pred procesom vyhľadávania [15]. Kombinácia pravdepodobnosti môže byť synchrónna - kombinovaním pravdepodobnosti optických/akustických modelov rámec po rámci, alebo asynchrónne kombináciou pravdepodobnosti na vyššej úrovni ako napríklad fonémy alebo znaky. Potrebný je však paralelný tok dát.

Rozpoznávanie pri dekódovaní: V tejto časti sa deje rozpoznávanie na výstupe dekóderov z oboch modalít [15].

Zvýšiť pravdepodobnosť dekódovania multimodálneho systému môžme aj pomocou rozpoznania jednej z modalít vo forme mriežky, alebo slovného grafu (angl. word graph) [14].

Sieť zmätenia (angl. confusion network) bola použitá v práci [14] na riešenie problému prepojenia dvoch modalít. Každá modalita mala vlastnú sieť zmätenia. Následne sa vykonali 2 kroky: vyhľadanie kotviacej podsiete na zarovnanie oboch podsietí sieti zmätenia a vytvorenie novej siete zmätenia.

Pri zarovnávaní podsieti je dôležité nájsť referenčné podsiete slúžiace ako kotvy medzi týmito modalitami [14]. Zobrazenie dvoch podsietí zmätenia a novej siete zmätenia môžeme vidieť na obrázku 5.3. Z obrázka môžeme vidieť že kotviacie podsiete sú $SN^0_{htr} - SN^0_{asr}, SN^2_{htr} - SN^1_{asr}, SN^3_{htr} - SN^2_{asr}$ a $SN^5_{htr} - SN^5_{asr}$.

Referenčné podsiete sú dôležité z dôvodu možných chýb v jednom z týchto štýlov. Algoritmus navrhnutí v práci [14] prehľadáva kotviace podsiete v oboch smeroch (zľava doprava a naopak) súčasne, pričom ako kotviace podsiete vyberá tie kde sa obe vyhľadávania zhodujú. Týmto sa zabráni vzniku nových chýb pri kombinácii z dôsledku vzdialeností medzi podsieťami. Gramatická zhoda medzi slovami oboch štýlov bola hodnotená pomocou kvadratického priemeru chybovosti znakov (CER) a chybovosti foném (PER) medzi týmito slovami. Chybu gramatickej zhody môžeme vypočítať ako:

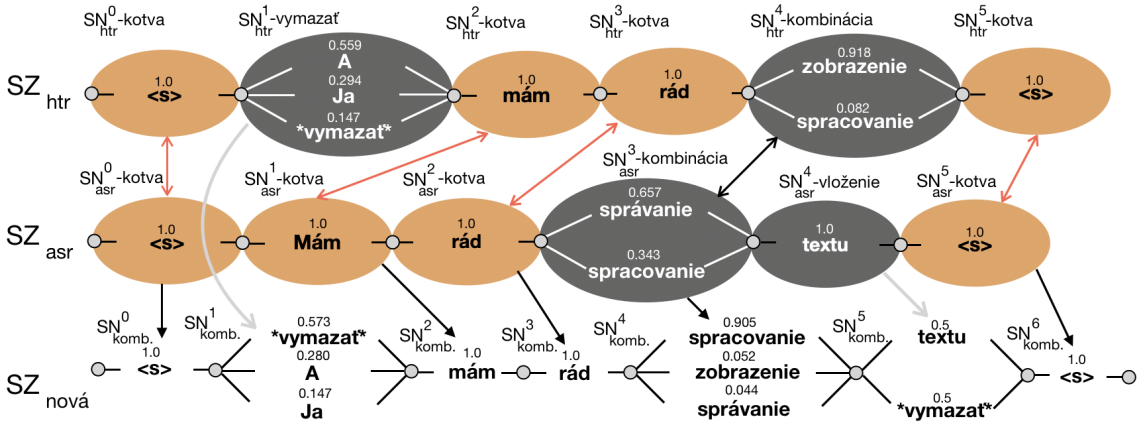
$$E(W_{htr}, W_{asr}) = \sqrt{\frac{CER(W_{htr}, W_{asr})^2 + PER(W_{htr}, W_{asr})^2}{2}}, \tag{2.13}$$

kde W_{htr} reprezentuje slovo z rozpoznávača písaného textu a W_{asr} reprezentuje slovo z rozpoznávača reči. CER a PER je Levensthainová vzdialenosť medzi slovami oboch štýlov na úrovni znakov a fonémov. E predstavuje chybu gramatickej zhody.
Cieľom vytvorenia novej siete zmätenia je redukcia chyby [14]. Na zostavenie novej siete môžeme použiť kombináciu, vloženie a vymazanie podsietí.

Cieľom multimodálnej kombinácie je maximalizovať pravdepodobnosť správneho slova (v prípade že sa nachádza v oboch podsieťach) vzhľadom na podsiete SN_{htr} pre rozpoznávanie písaného textu a SN_{asr} pre rozpoznávanie reči [14]. Na základe bayesovej vety a predpokladu silnej nezávislosti $P(w|SN_{htr}, SN_{asr})$ dostávame: $P(w|SN_{htr}, SN_{asr}) \simeq P(w|SN_{htr})P(w|SN_{asr})$, keďže nevieme, ktorý štýl je presnejší použijeme váhovanú verziu:

$$P(w|SN_{htr}, SN_{asr}) \simeq P(w|SN_{htr})^{\alpha} P(w|SN_{asr})^{1-\alpha}, \tag{2.14}$$

kde α je váhový faktor vyvažujúci relatívnu spoľahlivosť medzi rozpoznávaním písaného textu a reči. Z tejto rovnice by menej časté slová mali nulovú pravdepodobnosť pri kombinácii dvoch podsietí. Preto je potrebné vyhladiť pravdepodobnosti slov pred kombináciou. Na vyhladenie pravdepodobností slov sa používa vzorec



Obr. 2.12: Vytvorenie novej siete zmätenia z dvoch podsietí zmätenia. Pre výslednú vetu <s>Mám rád spracovanie textu<s>

$$P_{vyhladene}(w|SN) = \frac{P(w|SN) + \theta}{i + n\theta}, \tag{2.15}$$

kde θ je predom definovaná granularita a n je celkový počet slov. Na obrázku 5.3 môžeme vidieť SN^4_{htr} a SN^3_{asr} sú vybrané na skombinovanie. Môžeme vidieť že správne slovo "spracovanie", je najmenej pravdepodobné v obidvoch sieťach, avšak po zvolení $\alpha = 0.5$ a $\theta = 10^{-4}$ sa stáva najpravdepodobnejšie, čo môžeme vidieť aj v $SN^4_{komb.}$.

V prípade nesúhlasu medzi modalitami z dôvodu nezhodnutia sa v pozícii slov je potrebné rozhodnúť, či dané slovo v podsieti má byť ponechané, alebo vymazané [14]. Pridanie slova do podsiete sa vykonáva v prípade rozhodnutia druhej modality, že dané slovo na pozícii chýba a presahuje rozhodovací prah γ. Slovo sa z výslednej podsiete vymaže ak druhá modalita rozhodne že dané slovo je v prvej modalite navyše a najpravdepodobnejšie slovo pre vymazanie nedosiahne prah δ.

Pri vytvorení novej podsiete sa najskôr vytvorí multimodálna kombinácia pre podsiete označené ako kotva [14]. Takto sa medzi po sebe idúcimi zakotvenými podsieťami objaví séria zarovnaných fragmentov obidvoch štýlov. Fragmenty môžeme vidieť na obrázku 5.3 ako elipsy. Každý fragment môže obsahovať buď žiadnu, alebo niekoľko podsietí. Pre rozhodnutie čo s každou podsieťou robiť, použijeme veľkosť fragmentu. Pri porovnaní veľkostí dvoch fragmentov môžeme nájsť nasledujúce prípady, keď obe veľkosti fragmentov nie sú nulové: 1) ak sa veľkosti oboch fragmentov rovnajú, postupne sa skúsia skombinovať všetky podsiete, 2) ak je veľkosť fragmentu jednej modality nulová, pre všetky podsiete fragmentu druhej modality musím vybrať, či sa majú vložiť alebo vymazať, 3) ak sú obidva veľkosti fragmentov rozdielne a žiadny z nich nie je nulový, tak musí nájsť všetky nové kotviace podsiete uvoľneným vyhľadávaním (angl. relaxed search) a rozhodnúť či sa má zvyšok podsieti vložiť alebo odstrániť. Ako výsledok tohoto procesu získavame novú sieť zmätenia.

Znižovanie chybovosti hlasovaním pri výstupe rozpoznávača (angl. Recogniser Output Voting Error Reduction) rozpoznanie vykonáva kombináciou hlasovaním (na úrovni slov) medzi rôznymi výstupmi s použitím iba 1 najlepšej hypotézy [15]. Metóda zníženia chybovosti hlasovaním je implementovaná v 2 moduloch: 1) prvý modul zarovná a skombinuje najlepšie výstupy z dekódovania v sieti prechodu slov (podobná sieť ako sieť zmätenia).

2) druhý modul, ktorý je hlasovací, vyhodnotí každú podsieť aby vybrala najlepšie ohodnotené slovo na nový prepis.

Hlasovanie je vykonané nasledovne: pre každú podsieť počet výskytov každého slova w v zodpovedajúcej podsieti i je uložený do poľa N ako $N(w,i)$ a normalizovaný vydelením $N(w,i)$ počtom kombinovaných systémov N_s [15]. Miera istoty slova w v podsieti i záleží na hlasovacej schéme. Miera istoty je spočítaná a normalizovaná v poli $C(w,i)$. Rovnováha medzi použitím frekvencie výskytu slova s mierou istoty môže byť upravená pomocou parametra α:

$$Score(w,i) = \alpha \left(\frac{N(w,i)}{N_s} \right) + (1-\alpha)C(w,i). \tag{2.16}$$

Modul hlasovania môže byť rozdelený na 3 rôzne schémy [15]: 1) frekvencia výskytu - pri hlasovaní o frekvencii výskytu sa všetky miery istoty ignorujú a parameter α sa nastaví na 1, 2) frekvencia výskytu a priemerná spoľahlivosť slova - pri tomto spôsobe hlasovaním miera istoty každého slova w v poli $C(w,i)$ je nastavená na priemernú hodnotu výskytu tohoto slova v podsieti i. Parameter α musí byť predtrénovaný. 3) frekvencia výskytu a maximálna spoľahlivosť slova - v poslednej hlasovacej schéme miera istoty každého slova w v poli $C(w,i)$ je nastavená na maximálnu hodnotu vzhľadom na toto slovo w v podsieti i. Parameter α musí byť predtrénovaný.

N najlepších hlasovačov pri znížení chybovosti hlasovaním, funguje na podobnom princípe ako znižovanie chybovosti hlasovaním pri výstupe rozpoznávača avšak s tým rozdielom, že namiesto jednej najlepšej hypotézy využíva kombináciu viacerých hypotéz [15]. V prvom kroku sa vyberie h najlepších hypotéz z dekódovanej sekvencie x z rôznych systémov S_i, ktoré sú zarovnané rovnako ako v metóde spomenutej vyššie. V druhom kroku sa pre každý systém odhadnú normalizované a logaritmicky lineárne vážené posteriórne slová. V poslednom kroku je kombinovaný posterior slova vypočítaný ako lineárna kombinácia.

Posterior slova pre každé slovo w a systém i je vypočítaný pre každú podsieť j logaritmicky lineárnym váženým skóre, po ktorom nasleduje normalizácia všetkých hypotéz [15].

$$P_i(w|x) = \frac{\sum_{h:w\in h} \exp\left(\sum_j \lambda_{ij} s_{ij}(h|x)\right)}{\sum_{\forall h} \exp\left(\sum_j \lambda_{ij} s_{ij}(h|x)\right)}, \tag{2.17}$$

kde $s_{ij}(h|x)$ je logaritmické skóre a λ_{ij} sú kombinované váhy logaritmického skóre v podsieti j, hypotézy h, systému i. Potom kombinovaný posterior je vypočítaný ako lineárna kombinácia: $P_i(w|x) = \sum_i \mu_i P_i(w|x)$, kde μ_i je reprezentované ako váhy systému. Kombinovaná hypotéza je vytvorená zreťazením najpravdepodobnejších slovných hypotéz zarovnaných na každej pozícii. Výsledok kombinácie je vytvorený iba z 1 hypotézy, čo je rovnaké obmedzenie ako u znižovania chybovosti hlasovaním.

Hodnotenie mriežok (angl. Lattices Rescoring) [15]. Kombinácia viacerých mriežok do novej mriežky nemusí iba zlepšiť najpravdepodobnejšiu hypotézu, ale nová mriežka môže obsahovať lepšie hypotézy ako bola najpravdepodobnejšia hypotéza. Metóda N najlepších mriežok optimalizuje skóre rozpoznávania na úrovni slov a vytvára mriežku slov zo všetkých informácií v mriežkach, ktoré sa majú kombinovať. Algoritmus má 2 časti: v prvej časti je skóre hypotézy uložené v mriežkach ktoré sa majú kombinovať. Skóre je váhované pomocou parametra, všetky hypotézy sa zarovnajú a spoja do jedného N najlepšieho zoznamu. V druhej časti sa optimalizácia rozpoznávania skóre na úrovni slov vykonáva pomocou substi-

túcie normalizačného výrazu $P(x)$ konečnou sumou nad množinou W zo všetkých hypotéz v spoločnom N najlepšom zozname:

$$P(x) = \sum_{w \in W} P(w|x). \tag{2.18}$$

Nakoniec sa z N najlepších nanovo ohodnotených hypotéz vytvorí nová kombinovaná mriežka.

Kapitola 3

Návrh multimodálneho systému

Ako vyplynulo z analýzy, metódy pre rozpoznávanie reči a rozpoznávanie textu sú založené na podobných princípoch a je možné ich kombinovať. Nami navrhnutá metóda má slúžiť pre korekciu výstupov rozpoznávača textu. V prípade že rozpoznávač urobí chybu, táto chyba by mala byť odstránená pomocou rozpoznávača reči.

Vstupom pre rozpoznávač textu by mal byť obrázok textu. Tento obrázok by mal obsahovať iba rukou písaný text. Nemal by v sebe obsahovať ďalšie obrázky, prečiarknutý text a podobné rušivé elementy, ktoré by mohli zhoršiť rozpoznanie daného textu. Text na obrázku by mal byť v českom, alebo slovenskom jazyku.

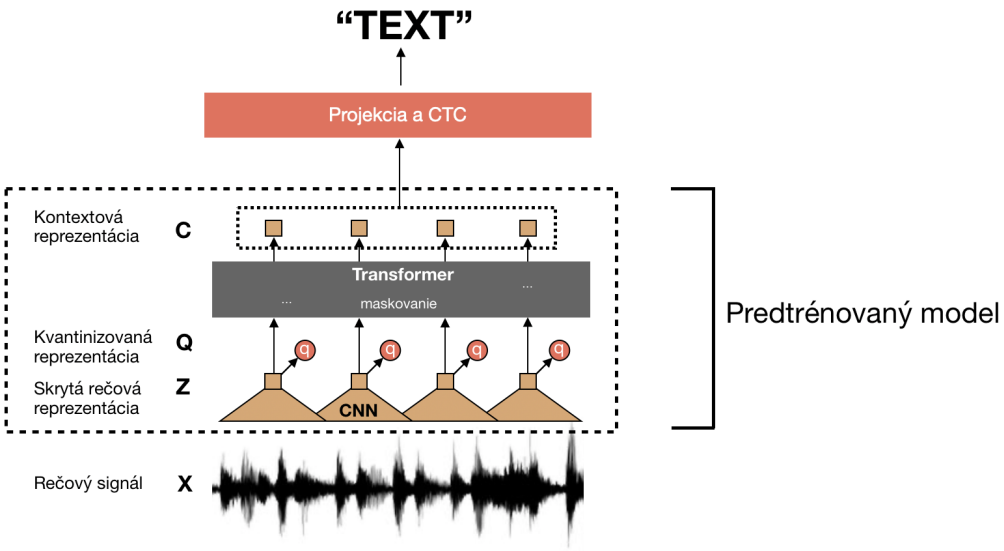
Vstupom pre rozpoznávač reči by mala byť nahrávka textu. Táto nahrávka by mala byť v českom, alebo slovenskom jazyku a nemala by obsahovať komunikačný šum. Slová v nahrávke by mali byť zreteľné a zrozumiteľné.

Požiadavky pre súbor údajov, na ktorých by sa mali systémy trénovať sú: 1) súbor údajov by mal byť dostatočne veľký, 2) súbor údajov by mal obsahovať štruktúru a formát ktorý je opísaný vyššie a mali by byť k nemu príslušné prepisy.

Dátová sada môže byť stiahnutá z nejakého zdroja, alebo vytvorená manuálne pomocou zberu údajov. Pre zber údajov by mohol byť vytvorený anotačný server. Na anotačnom serveri by sa v strede obrazovky zobrazila veta, ktorú by mal používateľ za úlohu prečítať a nahrať. Po nahratí vety by sa záznam uložil spolu s jej prepisom. Prípadne by časť dátovej sady mohla byť vygenerovaná pomocou generátora.

3.1 Možnosti implementačného riešenia pre rozpoznávanie textu

Rozpoznávanie písaného textu. Pri rozpoznávaní textu by na vstupe mohla byť konvolučná neurónová sieť, ktorá by z obrázka extrahovala črty a vytvorila by mapu čŕt. Konvolučná mapa by sa následne rozdelila na sekvencie čŕt, ktoré by tvorili vstup rekurentnej siete. Ako často používaná sa ukázala obojsmerná LSTM sieť. Výstup rekurentnej siete by mohol byť napojený na lineárnu vrstvu. Celý model by mohol byť trénovaný objektívnou CTC funkciou. Táto architektúra by bola rovnaká ako architektúra v práci [29]. Výstupom CTC funkcie by boli logity pre jednotlivé písmena, ktoré by určovali ako si je sieť istá daným znakom. Do úvahy pripadá aj možnosť prevzatia nejakej existujúcej implementácie, ktorá bude upravená do nami požadovaného tvaru tak, aby jej výstupom boli logity.



Obr. 3.1: Navrhnutý model pre rozpoznávanie reči. Základ tvorí predtrénovaný wav2vev2 enkóder. Na enkóder je pridaná lineárna vrstva.

Rozpoznávanie reči. Ako popisuje vo svojej práci *Granell a spol.* [15] rozpoznávače pre rozpoznávanie reči a rozpoznávanie textu môžu mať rovnakú, alebo podobnú architektúru s rôznymi vstupmi. Pri rozpoznávači reči by vstupom rozpoznávača bola hlasová nahrávka. Požadovaná forma výstupu z rozpoznávača reči je vo forme logitov, ktoré určujú mieru istoty pre každý znak.

Jednou z možností by bolo použiť konvolučnú neurónovú sieť pre extrakciu čŕt z nahrávky. Konvolučná neurónová sieť by obsahovala konvolučné a pooling vrstvy. Za konvolučnou neurónovou sieťou by mohla ísť rekurentná neurónová sieť, ktorá je vhodná na spracovanie sekvenčných dát. Iba rekurentná neurónová sieť by nebola vhodná z dôvodu miznúceho gradientu, preto by sme použili LSTM, alebo GRU. Celý model by bol trénovaný pomocou CTC objektívnej funkcie. Ako abecedu pre model by sme mohli použiť abecedu extrahovanú z trénovacej a validačnej sady, prípadne si vytvoriť vlastnú abecedu, ktorá by obsahovala znaky českého jazyka.

Ďalšou alternatívou je použitie už predtrénovaného modelu. Na extrakciu čŕt by sa mohol použiť predtrénovaný wav2vec2 model na výstup ktorého by bola vytvorená projekcia a pridaná lineárna vrstva [33]. Táto vrstva by mala počet výstupných tried rovný počtu znakov v našej abecede. Celý tento model by mohol byť dotrénovaný pomocou CTC objektívnej funkcie. Navrhnutý model je možné vidieť na obrázku 3.1.

3.2 Možnosti prepojanie rozpoznávača textu s rozpoznávačom reči

Prepojenie pomocou siete zmätenia. Sieť zmätenia vyplynula z analýzy a článku [15] ako jeden z najlepších spôsobov prepojenia pre multimodálny systém. Z výstupov dekódera pre text a reč vo forme logitov, vytvoríme sieť zmätenia pre každú modalitu. Siete zmätenia

zarovnáme pomocou minimalizácie Levenshtainovej vzdialenosti, ktorú vypočítame ako

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{ak} \quad min(i,j) = 0 \\ \min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & inak \end{cases}, \tag{3.1}$$

a,b reprezentujú znaky jednotlivých modalít. Zarovnané siete zmätenia preiterujeme a vytvoríme kotvy. Princíp kotvenia preberieme z kapitoly 2.4, ktorý bol opísaný v článku [14], kde zafixujeme rovnaké znaky, ktoré budú tvoriť kotvy podsieti. Po vytvorení kotiev plánujeme prehľadávať neisté znaky v sieti zmätenia. Pre tieto znaky sa budeme snažiť maximalizovať spoločnú pravdepodobnosť znaku oboch modalít. Výsledný prepis vety budú spoločné najpravdepodobnejšie znaky pre obe modality.

Prehľadávanie s použitím logitov. Druhý navrhnutý prístup bude založený na prehľadávaní s použitím logitov. Z logitov vyberieme pozície najpravdepodobnejších znakov pre každú modalitu. Pozície kde je najpravdepodobnejšia hodnota BLANK vynecháme. Po výbere pozícií začneme nimi iterovať. Ak sa znaky budú zhodovať vytvoríme kotvu. V prípade že sa znaky zhodovať nebudú, začneme prehľadávanie logitov od pozície posledného zakotveného znaku po aktuálny znak. V prípade že nájdeme spoločný najpravdepodobnejší znak, vytvoríme kotvu. V opačnom prípade tento znak odstránime.

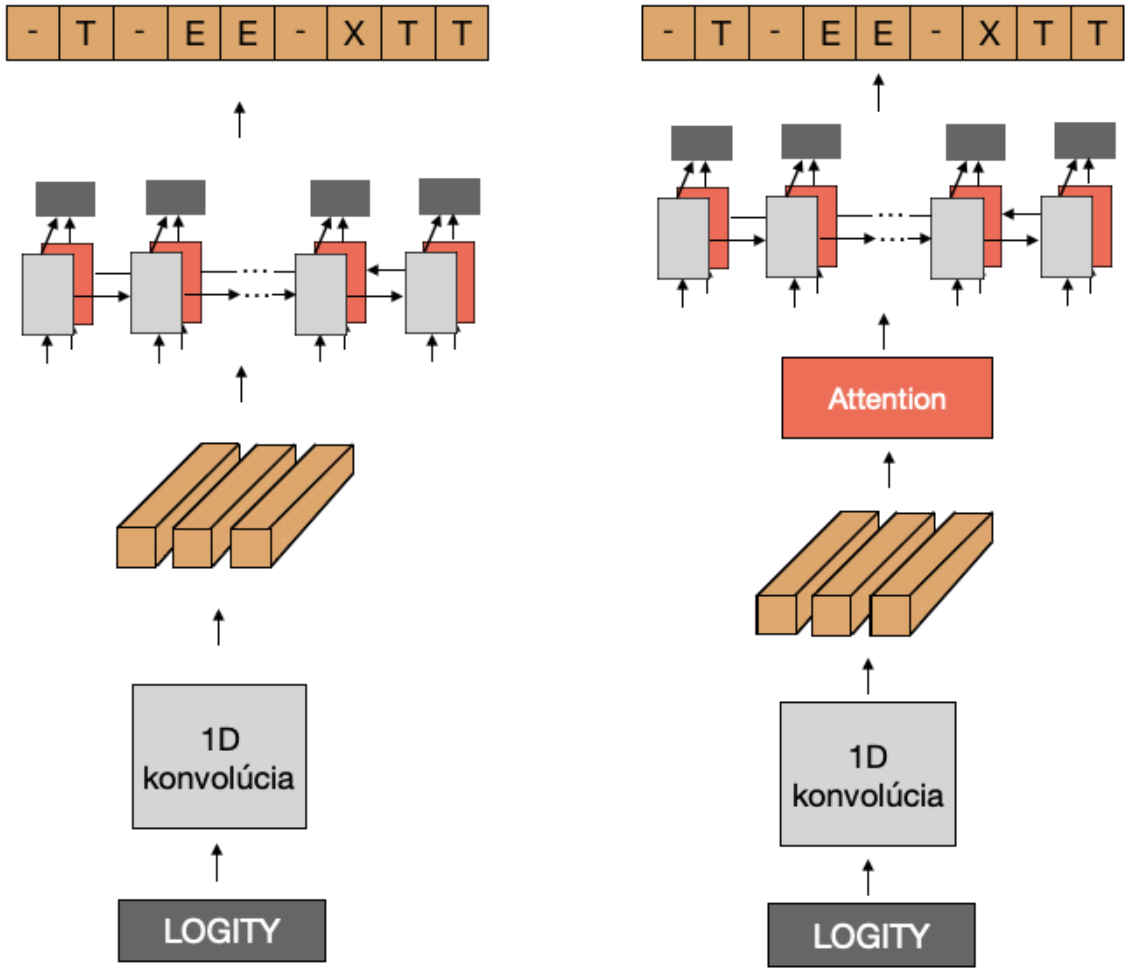
Prehľadávanie s použitím logitov pre neisté znaky. Vstupom budú logity oboch modalít. Z logitov vytvoríme prepisy, ktoré zarovnáme pomocou minimalizácie levenshtainovej vzdialenosti. Začneme iterovať pozíciami jednej z modalít. V prípade že pravdepodobnosť znaku na danej pozícii bude menšia ako nami definovaná hodnota, začneme prehľadávať logity druhej modality na pozícii, pre ktorú bol zarovnaný znak prvej modality. V prípade že sa spoločný znak nenájde, do výsledného prepisu bude použitý znak prvej modality, ktorou budeme iterovať.

Prepojenie pomocou neurónovej siete. Finálny výpis vytvoríme z neurónovej siete, ktorej vstup budú tvoriť logity oboch modalít. Ako neurónovú sieť plánujeme použiť konvolučnú neurónovú sieť pre extrakciu čŕt. Výstup tejto siete bude tvoriť vstup pre LSTM sieť. Celá sieť by bola trénovaná pomocou objektívnej CTC funkcie. Výstup siete by bol vo forme logitov. Logity následne prevedieme do textovej podoby.

Alternatívou by mohol byť použitie vlastnej pozornosti(angl. attention) medzi výstupom z jedna d konvolúcie a LSTM sieťou, ktorý by mohol pomôcť pri výbere dôležitých informácií. Tento návrh môžeme vidieť na obrázku 3.2.
V prípade sequence to sequence modelu by ako enkóder bola použitá konvolučná neurónová sieť, podobne ako v práci [20] spolu s LSTM neurónovou sieťou. Pre každú modalitu by bol vytvorený vlastný enkóder. V dekóderi by sa mohla počítať pozornosť z jednotlivých enkóderov. Výstupy enkóderov by sa vynásobili spolu s váhami pozornosti čím by sme dostali kontextový vektor. Kontextové vektory oboch modalít by boli konkatenované a spojené do jedného vektora, ktorý by bol vstupom LSTM alebo GRU neurónovej siete.

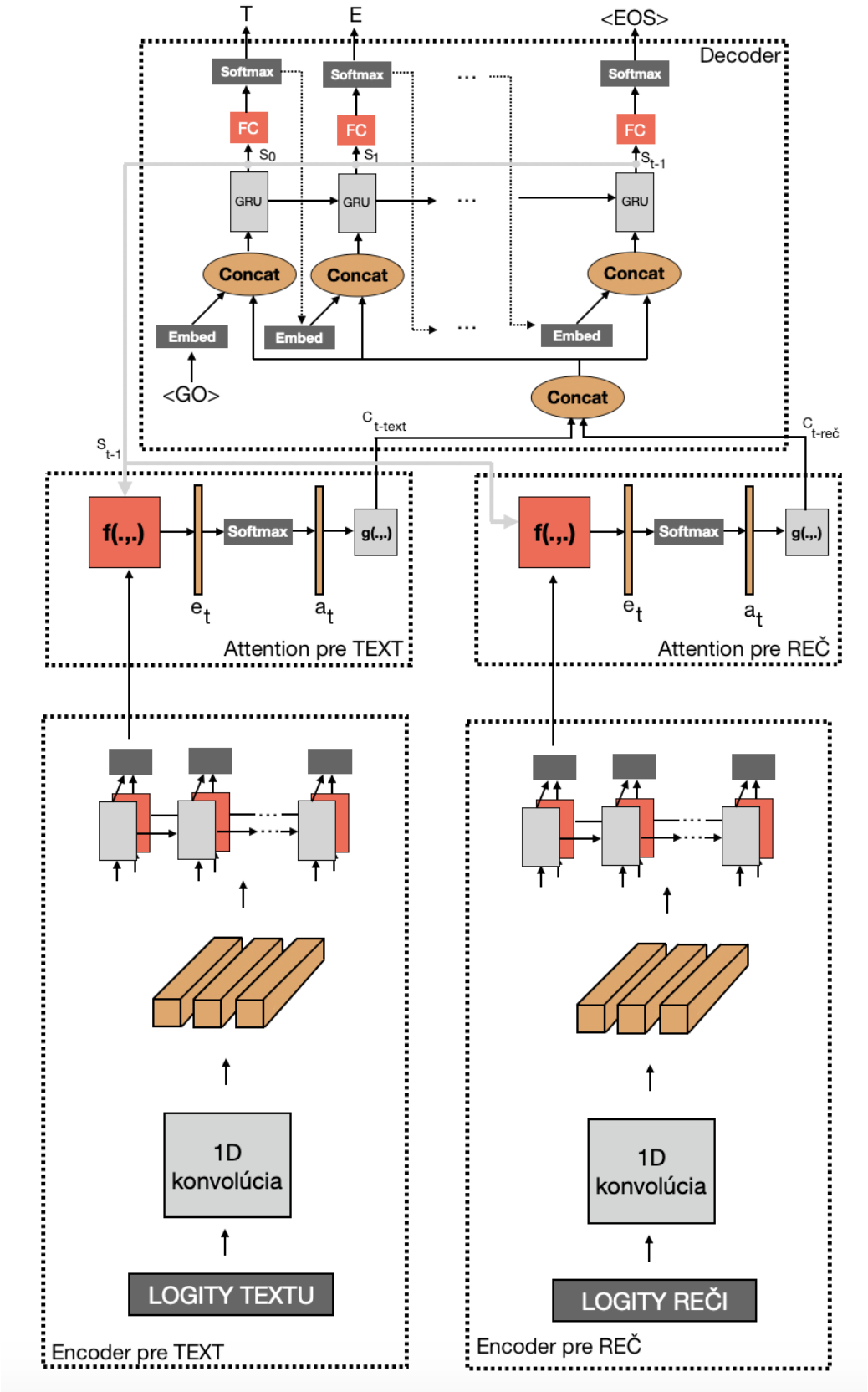
Problémy pri mapovaní sekvencií

Pri rozpoznávači textu môžu byť chyba pri prepise v interpunkcii. Problém potom nastáva v spôsobe ako takúto chybu pomocou rečového systému opraviť. Oprava takejto chyby môže



Obr. 3.2: Návrh modelov pre prepojenie logitov z oboch modalít do textu. Vľavo model s LSTM, v pravo model s LSTM a vlastnou attention.

byť náročná, nakoľko vyjadriť interpunkciu v rečovom systéme nie je ľahké. Ďalší problém môže nastať v prípade chybných, alebo chýbajúcich mäkčeňov a dĺžňov. V prípade že rečový systém nerozpozná v hlase dĺžeň, správny prepis môže byť opravený na nesprávny, alebo môže označiť nesprávny prepis za korektný. Otázka nastáva aj v probléme namapovania čísel z výstupu rečového systému na výstup systému rozpoznávania textu, nakoľko číslice a čísla v textovej podobe môžu mať iný význam a tvar ako číslice v rečovej nahrávke. Taktiež ich forma môže mať iný tvar, nakoľko čísla môžu byť zapísané slovne alebo pomocou symbolov. Ďalší problém ktorý pri mapovaní vzniká sú rôzne symboly, napríklad symbol %, alebo symboly platidiel ako € alebo $ a rôzne iné.



Obr. 3.3: Navrhnutý seq2seq model na prepojenie výstupov rozpoznávača reči a rozpoznávača písaneho textu.

Kapitola 4

Implementácia

4.1 Algoritmy spojovania

```
[{'t': 1.0},
 {'e': 1.0},
 {'r': 1.0},
 {'k': 1.0},
 {'a': 1.0},
 {' ': 0.0010085433320346663, None: 0.9989914566796534},
 {'d': 0.611028846330596061, 'n': 0.3889711537013215},
 {'a': 1.0}]
```

Obr. 4.1: Ukážka vytvorenej siete zmätenia pre jednu modalitu používaná v algoritmus1.

Algoritmus 1. Vstupom algoritmu sú logity pre každú modalitu. Z logitov sa vytvorí prepis k rôznych najpravdepodobnejších viet, pomocou prefixového dekódera. Z jeho výstupu je vytvorená sieť zmätenia. Príklad takejto siete znätenia sa nachádza na obrázku 4.1. Sieť zmätenia je zoznam slovníkov, kde každý slovník obsahuje kľúč - v tomto prípade najpravdepodobnejší znak a hodnotu, ktorá reprezentuje pravdepodobnosť istoty znaku. Viac znakov sa na danej pozícii nachádza v prípade neistoty siete. Na obrázku je to napríklad znak d a znak n, kde je pri každom znaku uložená jeho pravdepodobnosť.

Sieť zmätenia je vytvorená pre logity reči a logity textu. Vytvorená sieť zmätenia je uložená do vopred definovanej triedy: index - reprezentuje pozíciu na ktorej sa znak v sieti zmätenia nachádza, origin - nadobúda hodnotu ASR - rozpoznávač reči a OCR - rozpoznávač textu. Atribút character, ktorý nadobúda hodnotu not_sure v prípade väčšieho počtu znakov pre danú pozíciu. Ak je na pozícii iba jeden znak, predspracuje sa rovnakým spôsobom ako trénovacie dáta rozpoznávača reči a uloží sa do triedy. Posledný atribút je označený ako original, kde je uložená reprezentácia znaku pred predspracovaním.

Po uložení siete zmätenia do triedy vzniknú dva zoznamy tried, jeden pre reč a druhý pre text. Zoznamy sú zarovnané pomocou levenshtainového zarovnania do jedného spoločného zoznamu. V prípade že sa niektoré znaky zo siete zmätenia nezhodujú a sieť zmätenia pre

jednu modalitu je dlhšia ako sieť zmätenia pre tú druhú, levenshtainové zarovnanie na takúto pozíciu priradí hodnotu None.

Následne sa vykoná iterácia spoločným zarovnaným zoznamom. V prvom kroku sa zapamätajú pozície, kde si je sieť jedným alebo druhým znakom neistá. V prípade že sú neisté oba znaky, zapamätajú sa pozície oboch modalít. Spolu s pozíciami neistých znakov sa ukladajú aj pozície znakov zarovnané na hodnotu None. Pre znaky, ktoré sa zhodujú a majú pravdepodobnosť 1 je vytvorená kotva. Kotva má pevnú pozíciu a po jej vytvorení sa už nemení. Následne sa spracujú neisté znaky a znaky zarovnané na hodnotu None. Postupne sa iteruje neistými pozíciami pre sieť zmätenia prvej modality. Ku každej pozícii prvej modality sa hľadá najdlhšia postupnosť neistých pozícií druhej modality taká, aby sa tam vyskytovala pozícia prvej modality pre ktorú postupnosť hľadáme. V prípade že sa žiadna postupnosť nenájde, vráti sa pôvodná hľadaná pozícia. Toto je vykonávané z možného posunu jednotlivých znakov pri zarovnaní siete, ktoré je potrebné napraviť. Po nájdení najdlhšej postupnosti pozícií sa maximalizuje spoločná pravdepodobnosť rovnakých znakov. Znak s najväčšou spoločnou pravdepodobnosť je pridaný ku kotvám. Algoritmus končí po preiterovaní celej dĺžky spoločného zarovnania.

Algoritmus 2. Vstupom algoritmu sú logity pre modalitu reči a modalitu textu. Algoritmus vráti pre logity každej modality najpravdepodobnejšie znaky. Následne sú do zoznamu uložené pozície najpravdepodobnejších znakov, ktorých najpravdepodobnejší znak nie je hodnota blank. Pre každú pozíciu zo zoznamu sa vytvorí nový zoznam s K najpravdepodobnejšími znakmi spolu s ich pravdepodobnosťami. Týmto je docielené to, že ak sa znaky na niektorej pozícii nezhodnú, môžu byť prehľadávané zvyšné znaky pre túto pozíciu. Začne sa iterovať oboma zoznamami pozícií súčasne. Pre každú pozíciu sa maximalizuje pravdepodobnosť spoločného znaku. V prípade že najpravdepodobnejší znak je nájdený, vytvorí sa kotva. Ak sa spoločný znak nenájde, prehľadávajú sa logity od poslednej pozície znaku, pre ktorý bola vytvorená kotva, po aktuálnu pozíciu znaku. V prípade nájdenia najpravdepodobnejšieho znaku je vytvorená kotva. Ak pri prehľadávaní nie je nájdený žiadny spoločný znak, algoritmus sa presunie na ďalšiu pozíciu tej modality, pre ktorú sa znak nenašiel. Pozícia druhej modality ostáva nezmenená. Algoritmus končí po preiterovaní celej sekvencie, aspoň jednej modality.

Algoritmus 3. Algoritmus 3 je kombinácia algoritmu 1 a algoritmu 2. Z algoritmu 1 bolo prevzaté levenshtainové zarovnanie a z algoritmu 2 bolo prevzaté prehľadávanie pozícií v logitoch. Toto by malo viesť k väčšiemu množstvu znakov, pre ktoré môžeme maximalizovať spoločnú pravdepodobnosť a zároveň by to malo zlepšiť výsledky v prípade že sa na niektorej pozícii nenájde spoločný znak.

Vstupom algoritmu sú logity pre modalitu reči a modalitu textu. Zistia sa najpravdepodobnejšie znaky z logitov. Podobne ako v algoritme 2 sa odfiltrujú najpravdepodobnejšie znaky blank. Pre dané znaky sa vytvorí zoznam pozícií v logitoch a je vytvorená veta z najpravdepodobnejších znakov. Táto veta je vytvorená pre modalitu reči aj modalitu textu a uložená to zoznamu tried rovnako ako v algoritme1 s tým rozdielom, že namiesto označenia *not_sure* pre neistý znak bol vždy uložený najpravdepodobnejší znak. Po zarovnaní sa iteruje výsledným spoločným zarovnaním. Ak sa najpravdepodobnejšie znaky rovnajú, predpokladá sa že budú mať najväčšiu spoločnú pravdepodobnosť a je vytvorená kotva. V prípade nerovnosti najpravdepodobnejších znakov sa pre danú pozíciu v logitoch vyberie k najpravdepodobnejších znakov spolu s ich pravdepodobnosťami. Následne sa v oboch zoznamoch hľadá spoločný najpravdepodobnejší znak. V prípade že sa žiadny spoločný znak

nenájde, znaky sú zahodené a algoritmus sa posúva na ďalší znak. Ak sa niektorý zo znakov pri levenshtainovom zarovnaní namapoval na hodnotu None, prehľadávajú sa všetky pozície v logitoch od poslednej kotvy, po ďalší najbližší znak v zarovnaní. Pre každú pozíciu z tohoto rozmedzia je vytvorený spoločný zoznam k najpravdepodobnejších znakov spolu s ich pravdepodobnosťami. V prípade nenájdenia znaku je tento znak z druhej modality odstránený. V prípade že sa špeciálny znak na danej pozícii nachádza ako najpravdepodobnejší, vytvorí sa kotva so špeciálnym znakom, nakoľko v abecede reči sa špeciálne znaky nenachádzajú.

Algoritmus 4. Vstupom tohoto prístupu sú logity oboch modalít. Z logitov každého rozpoznávača sa vytvorí prepis a zapamätá sa pozícia v sekvencii, kde sa daný znak nachádza. Prepis, spolu s pozíciami sa zarovná pomocou levenshtainového zarovnania. Po zarovnaní sa iteruje zapamätanými pozíciami a porovnáva sa, či je znak na pozícii pravdepodobnejší, ako predom zvolená hranica. V prípade že pravdepodobnosť znaku je istejšia ako zvolená hranica, vytvorí sa kotva, ktorá obsahuje informácie o znaku, jeho pozícii a pozícii na ktorú sa zarovnal v rámci druhého rozpoznávača. V prípade že je pravdepodobnosť znaku nižšia ako hranica, vyberie sa z danej pozície k najpravdepodobnejších znakov. Následne sa z poslednej vytvorenej kotvy vyberie pozícia znaku z druhého rozpoznávača zvýšená o hodnotu jedna. Vyberie sa tiež pozícia ďalšieho najbližšieho znaku z druhého rozpoznávača znížená o hodnotu jedna. Tieto dve pozičné hodnoty vytvárajú interval v logitoch druhého rozpoznávača, ktorý sa bude prehľadávať. Z intervalu sa vyberie k najpravdepodobnejších znakov pre každú pozíciu intervalu a maximalizuje sa spoločná pravdepodobnosť k najpravdepodobnejších znakov prvého a druhého rozpoznávača. V prípade že sa nájde spoločný najpravdepodobnejší znak, vytvorí sa kotva, do ktorej sa uloží najpravdepodobnejší znak spolu s pozíciami výskytu znaku pre každý rozpoznávač. V prípade nenájdenia najpravdepodobnejšieho znaku sa daný znak vo výslednom prepise nachádzať nebude.

Konvolučno-rekurentná neurónová sieť. Dĺžka vstupných logitov z rozpoznávača reči a rozpoznávača textu bola rôzna, preto bola potrebné urobiť doplnenie (angl. padding) na dĺžku dlhšej sekvencie. Hodnota ktorou sa doplňovalo bola mínus sto. Sekvencie sa po doplnení spojili cez spoločnú dĺžku sekvencie na 555 hodnôt pre každú sekvenciu. Takto spojené sekvencie tvorili vstup jednej dávky (angl. batch). Keďže dĺžka sekvencií v rámci jednej dávky musí byť rovnaká, vybrala sa najdlhšia sekvencia z dávky a ostatné sekvencie sa zarovnali na jej dĺžku. Na dávkach sa trénovala neurónová sieť, ktorá obsahovala 1D konvolúciu, LSTM vrstvy a lineárnu vrstvu.

Vytvorenie anotačného servera Pre tvorbu rečovej dátovej sady bol vytvorený anotačný server[1]. Anotačný server bol naprogramovaný v jazyku *Python* vo verzii 3.8.13. Na implementáciu anotačného servera bola použitá knižnica *cgi*. CGI je súbor štandardov, ktoré vysvetľujú ako sa údaje vymieňajú medzi webovým serverom a scriptom. Script CGI je vyvolaný HTTP serverom a po jeho vyvolaní sa spusti ako program. Nahrávanie bolo vykonávané pomocou programovacieho jazyka *javascript*, vizuálna časť stránky bola naprogramovaná pomocou *HTML* a upravená pomocou *CSS* a *bootstrap*.

Stránka anotačného servera sa zobrazí po zadaní URL adresy. V strede stránky sa nachádza veta, ktorú má používateľ za úlohu prečítať. Pod vetou sú tlačidla, pomocou ktorých vie používateľ vykonávať jednotlivé úkony. Tlačidlo Start recording slúži na začatie

[1] https://www.stud.fit.vutbr.cz/~xkabac00/voice_recorder.py

```
"id": "a712af21-d382-4025-b936-f32b52172b18",
"deleted": false,
"text_lines": [
    {
        "annotated": true,
        "annotations": [
            {
                "id": "9860feda-d73e-4d26-a03c-cb53fc",
                "text_original": "pinalei, trogim",
                "text_edited": "pinlei, trogim",
                "created_date": "2021-12-19 10:27:23.9387"
            }
        ],
    }
    ...
]
```


Obr. 4.2: Ukážka JSON súboru z ktorého boli vygenerované vety pre anotačný server.

nahrávania, Stop and save recording zastaví a uloží nahrávanie. Po stlačení tohoto tlačidla sa na obrazovku vygeneruje nová veta. Pause umožní používateľovi prestať nahrávať. Po stlačení tlačidla pause sa tlačidlo zmení na resume a v nahrávaní je možné opäť pokračovať. Tlačidlo reset recording slúži na opätovné nahrávanie. Tlačidlo next recording vygeneruje novú vetu na prečítanie. Pod tlačidlami sa nachádzajú body ako správne nahrávať vety.

V prípade prepájania reči a písma je potrebné mať k dispozícii obe modality s rovnakým textovým obsahom. Aby sme nemuseli vytvárať obe modality súčasne, pre písaný text sme mali výstup riadkov textu spolu s prepismi zo servera PERO-OCR. Výstup riadkov bol vo formáte JSON, ktorého štruktúru môžeme vidieť na obrázku 4.2.

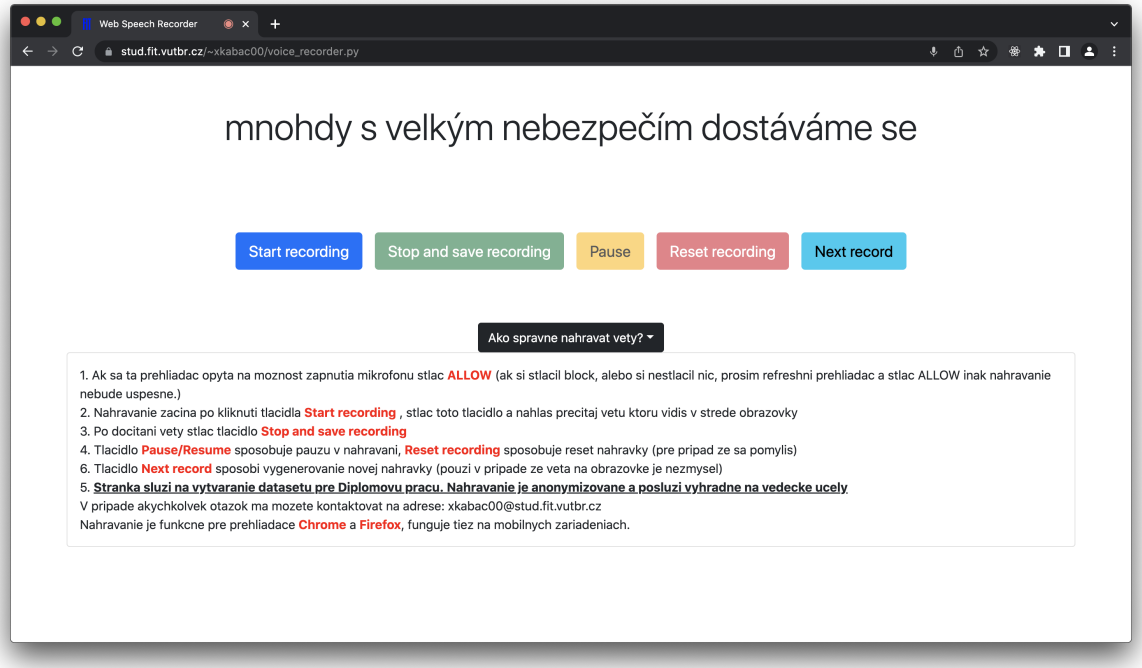
Z JSON súboru boli vybraté riadky, ktoré obsahovali opravený prepis. K dispozicii bol aj pôvodný výstup z rozpoznávača reči. Pre anotačný server boli vybraté riadky ktorých chybovosť výpočítaná pomocou levenshtainovej vzdialenosti bola z intervalu (0;25) a celkový počet písmen tvoril aspoň 70% riadka.

Pre anotačný server bol náhodne vybratý riadok, ktorý sa užívateľovi zobrazil v strede vygenerovanej stránky.

4.2 Rozpoznávače

Pre rozpoznávanie reči boli dátové sady predspracovaná rovnako ako v práci [9]. Odstránila sa interpunkcia, špeciálne znaky, diakritika a veľké písmená boli zmenené na malé. Všetky nahrávky boli prevedené do frekvencie 16.000 kilohertzov. Pri trénovaní bol použitý predtrénovaný model wav2vec2[2]. Na predtrénovaný model sa urobila projekcia a pridala sa lineárna vrstva. Pre jednotlivé dávky sa dĺžky sekvencií doplnili do dĺžky najdlhšej sekvencie hodnotou mínus sto. Celý model bol dotrénovaný na vyššie uvedených datasetoch.

[2] https://huggingface.co/arampacha/wav2vec2-large-xlsr-czech



Obr. 4.3: Ukážka vygenerovanej stránky anotačného servera pre tvorbu dátovej sady.

Kapitola 5

Experimenty

5.1 Dátové sady

V rámci diplomovej práce boli použité dátové sady mozilla common voice a dátová sada politických nahrávok.

Dátová sada mozilla common voice Z dátovej sady mozilla common voice boli vybraté dva datasety - slovenský rečový dataset a český rečový dataset. Český rečový dataset obsahoval 54 hodín reči. Dataset je reprezentovaný v mp3 súboroch spolu s textovým súborom. Textový súbor obsahuje informácie ako napríklad text v nahrávke, vek, prízvuk a pohlavie.

Dataset pre slovenský jazyk obsahoval 18 hodín nahrávok a bol rovnakého formátu ako český dataset.

Dátová sada politických nahrávok. Je sada rečových nahrávok z poslaneckej snemovne českej republiky. Dátová sada obsahuje 88 hodín nahrávok z rôznych zasadaní. K týmto nahrávkam sú k dispozícií textové prepisy vo formáte XML. Príklad prepisu pre nahrávky sa nachádza na obrázku 5.1.

Každý prepis obsahuje nasledujúce tagy: Trans, v tomto tagu je uložený názov nahrávky a dátum jej vytvorenia. Nasleduje tag Speakers, v ktorom je zoznam všetkých rečníkov, ktorí v nahrávke vystupujú. Ku každemú rečníkovi sú priradené informácie ako dialekt, meno rečníka, prízvuk a id rečníka. Tag episode, v ktorom sú uložené všetky prepisy z daného zasadnutia. Tag Section obsahuje informáciu o časovom rozmedzí pre danú sekciu, sú tu 2 základné informácie začiatok nahrávania sekcie a koniec nahrávania sekcie. Turn - v tomto tagu sú uložené informácie začatí a konci nahrávania pre jedného rozprávača. Nasledujú samotné prepísané vety. Medzi prepismi viet sa môžu vyskytnúť tagy Event a Sync. Tag event nesie v sebe informáciu o type udalosti (ako napríklad zle vyslovené slovo, hluk v miestonosti). V prípade zle vysloveného slova je v tagu Event napísané ako bolo slovo vyslovené spolu s tým ako by sa dané slovo vyslovovať malo. Tag Sync je synchronizačná časová značka. Tento tag obsahuje v sebe čas a pomocou tohoto tagu sa dajú jednotlivé nahrávky orezávať.

CER (character error rate) bola zvolená ako hlavná metrika pre vyhodnocovanie úspešnosti jednotlivých prístupov. Metrika CER sa vypočíta ako

$$\frac{(S + D + I)}{N}, \tag{5.1}$$

```
<Turn endTime="279.210" speaker="spk1" mode="spontan" startTime="0.000">
        konecne ta posledni kapitola
        <Event desc="speaker" type="noise" extent="instantaneous" />
        <Sync time="2.530" />
        je tak jak byla
        <Event desc="dycky" type="pronounce" extent="begin" />
        vzdycky
        <Event desc="dycky" type="pronounce" extent="end" />
        ...
</Turn>
```

Obr. 5.1: Ukážka XML súboru ku rečovým nahrávkam politikov.

kde S je počet substitúcií, D je počet vymazaní a I je počet doplnení znakov predikovanej sekvencie do správnej sekvencie ktorej dĺžka je N znakov [10].

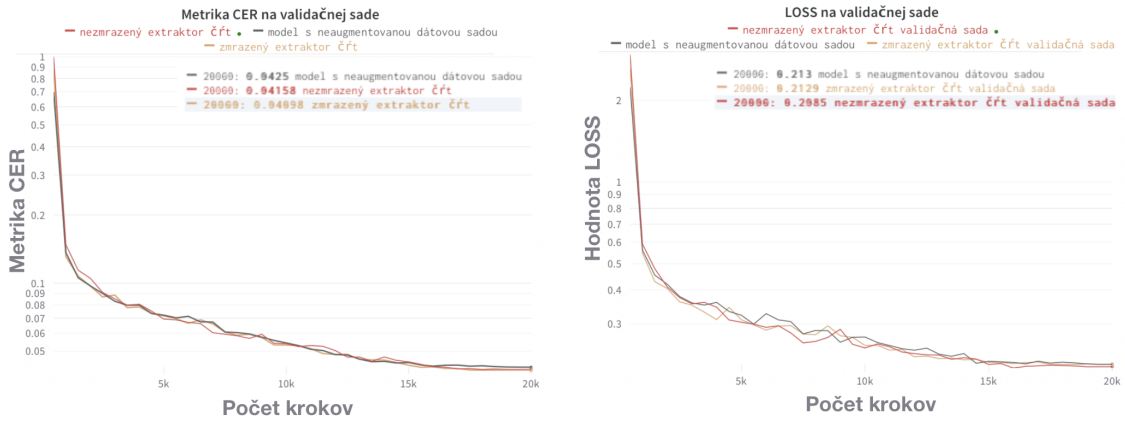
5.2 Rozpoznávač reči

Pre natrénovanie rozpoznávača reči bolo použitých 9 773 nahrávok z mozilla common voice pre český jazyk, 2 991 nahrávok z mozilla common voice pre slovenský jazyk a 51 553 nahrávok z politických prejavov. V evaluačnej sade bolo 4 144 nahrávok mozilla common voice pre češtinu, 2 271 nahrávok mozzila common voice pre slovenčinu a 5 729 politických nahrávok. Nahrávky na trénovanie a evaluáciu boli vybrané náhodne.

Pre rozpoznávanie reči boli natrénované 3 modely. Prvý model bol trénovaný so zmrazeným extraktorom čŕt a neaugmentovanou dátovou sadou. Zvyšné dva modely boli trénované s augmentovanou dátovou sadou a so zamrazeným a nezamrazeným extraktorom čŕt. Augmentácia bola vykonaná modifikovaním pôvodných nahrávok gaussovským šumom, zvýšovaním a znižovaním hlasitosti a menením výšky tónu. Všetky modifikácie boli vykonané s pravdepodobnosťou 80%. Vývoj metriky cer pre validačnú sadu sa nachádza na obrázku 5.2. Z obrázka je vidieť, že rozdiely vo výsledkoch sú zanedbateľné, z toho dôvodu boli nasledujúce experimenty vyhodnotené na modely trénovanom so zamrazeným extraktorom čŕt a augmentovanou dátovou sadou, ktorý mal pre validačnú dátovú sadu najnižšiu hodnotu metriky CER 4.09%.

5.3 Korekcia rozpoznávača textu pomocou rozpoznávača reči

Pri vyhodnocovaní výsledkov korekcie modality textu pomocou modality reči boli použité nahrávky z anotačného servera. Prostredníctvom servera bolo vyzbieraných 1932 nahrávok. Nahrávky boli rozdelené do 3 dátových sád. Prvá dátová sada obsahovala jednoduché nahrávky, ktorých počet bol 400, jednalo sa o nahrávky v ktorých sa nevyskytovali číslice a špeciálne znaky. Druhá dátová sada obsahovala komplexné nahrávky, ktorých počet bol 200. Táto dátová sada obsahovala nahrávky, ktoré neboli vyberané na základe žiadnych kritérií a mohli obsahovať čísla a špeciálne znaky. Tretia dátová sada o veľkosti 1 332 nahrávok slúžila pre natrénovanie neurónových sieti pomocou ktorých sme z logitov oboch modalít chceli získať finálny prepis vety.



Obr. 5.2: Zobrazenie loss a metriky CER na validačnej sade počas trénovania modelu. Môžeme vidieť že modely majú podobnú hodnotu metriky chybovosti CER. Najmenšiu hodnotu však dosiahol model so zamrazeným enkóderom čŕt pri augmentovanej dátovej sade.

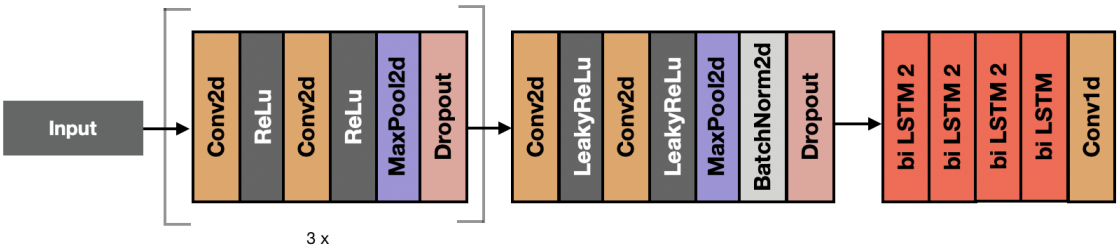
Tabuľka 5.1: Metrika CER pre rozpoznávač reči a rozpoznávač textu na výsledných dátových sadách.

	jednoduchá	komplexná	jednoduchá + komplexná
Rozpoznávač reči - predsprac.	6.77	16.62	10.46
Rozpoznávač reči	15.72	32.32	21.82
Rozpoznávač písaného textu	3.02	2.36	2.78

Pred vyhodnotením experimentov pre korekciu, boli dátové sady vyhodnotené rozpoznávačom textu a rozpoznávačom reči. Výsledky vyhodnotenia sa nachádzajú v tabuľke 5.1. Prvý riadok v tabuľke Rozpoznávač reči - predsprac. reprezentuje rozpoznávač reči vyhodnotený na dátovej sade predspracovanej rovnakým spôsobom ako trénovacia sada pre rozpoznávač reči. Druhý riadok v tabuľke Rozpoznávač reči reprezentuje rozpoznávač reči vyhodnotený na nepredspracovanej testovacej dátovej sade. Posledný riadok reprezentuje Rozpoznávač písaného textu. Pre rozpoznávač písaného textu bola prevzatá konvolučná rekurentná neurónová sieť[1], ktorá bola trénovaná pomocou CTC. Náčrt architektúry tejto siete sa nachádza na obrázku **??**. Ukázalo sa že rozpoznávač písaného textu dosiahol najlepšie výsledky na uvedených dátových sadách. úspešnosť rozpoznávača reči mohla byť ovplyvnená špeciálnymi znakmi a skratkovými slovami, ktoré boli jednotlivými rečníkmi čítané rôznym spôsobom.

Vyhodnotenie algoritmov spojovania môžeme vidieť v tabuľke 5.2. Ako najlepší spôsob prepojenia sa ukázal algoritmus 4, ktorý sa rozhoduje na základe rozpoznávača reči iba v prípade neistoty znaku. Ako druhý najlepší prístup sa ukázalo rozhodovanie na základe spoločného prehľadávania v logitoch. Prístup inšpirujúci sa článkom [14] dosiahol horšie výsledky v porovnaní s uvedenými dvoma prístupmi. Toto mohlo byť spôsobené tým, že pri sieti zmätenia neboli dostupné všetky znaky spolu s ich pravdepodobnosťami, ale tie najpravdepodobnejšie. Ani jeden algoritmus však nedosahoval lepšie výsledky ako rozpoznávač

[1] https://github.com/DCGM/pero-ocr



Obr. 5.3: Ukážka architektúry modelu, ktorý bol použitý pre rozpoznávanie textu. bi LSTM 2 reprezentuje obojsmernú LSTM sieť s 2 vrstvami.

textu.

Pre algoritme 4 boli vykonané experimenty s hĺbkou prehľadávania k najpravdepodobnejších logitov. Algoritmus mal ako zvolenú hranicu miery istoty znaku 80%. Pre k boli zvolené hodnoty 3,6,9. Výsledky sa nachádzajú v tabuľke 5.4. Algoritmus dosahoval najlepšie výsledky na jednoduchej a jednoduchej + komplexnej dátovej sade pri hodnote $k = 3$. Metrika CER na jednoduchej dátovej sade bola 3,96% a na jednoduchej + komplexnej dátovej sade 5,07%. Toto bolo spôsobené tým, že znaky od 3 pozície mali malú pravdepodobnosť. V prípade že sa málo pravdepodobný znak z výstupu jedného rozpoznávača nachádzal v znakoch druhého rozpoznávača a nebol nájdený žiadny iný spoločný znak z tohoto znaku bola vytvorená kotva, hoci znak mohol dosahovať malú spoločnú pravdepodobnosť. Pre komplexnú dátovú sadu sa ako vhodnejšie ukázala väčšia hĺbka prehľadávania, kde algoritmus dosiahol metriku CER 2,74%. Toto bolo spôsobené tým, že niektoré spoločné znaky najmä pri rozpoznávači reči sa nachádzali v hĺbke 4 a nižšie. Pri voľbe hĺbky prehľadávania $k >= 6$ dosahoval algoritmus rovnaké výsledky.

Experiment na hĺbku prehľadávania bol tiež vykonaný pre algoritmus3. Hĺbka prehľadávania k nadobúdala hodnoty 3,6,9. Výsledky pre algoritmus3 sa nachádzajú v tabuľke 5.3. Pri algoritme 3 sa ako najlepšia hĺbka prehľadávania ukázala hodnota $k = 6$ pre jednoduchú dátovú sadu, na ktorej dosiahla metriku CER 3.78% a pre komplexnú dátovú sadu hodnota $k = 9$, kde dosiahla hodnotu 6,71%. Rozpoznávač reči robil jednu z najčastejších chýb na medzerách medzi slovami. Medzeri preto boli pri rozpoznávači textu veľmi pravdepodobné, zatiaľ čo pri rozpoznávači reči boli málo pravdepodobné. Z tohoto dôvodu algoritmus3 dosahoval lepšie výsledky s väčšou hĺbkou prehľadávania v porovnaní s algoritmom4. V porovnaní s algoritmom 4 dosahoval horšie výsledky pre komplexnú dátovú sadu. Toto je spôsobené najmä číslami a špeciálnymi znakmi.

Pre algoritmus4 bol vykonaný experiment s mierou istoty znaku na výstupe rozpoznávača. Miera istoty znamená rozhodovaciu hranicu. Ako hĺbka prehľadávania bola nastavená konštanta 6. Ak je pravdepodobnosť znaku menšia ako miera istoty, vykoná sa prehľadávanie logitov druhého rozpoznávača. Výsledky experimentu sa nachádzajú v tabuľke 5.5. Pre nižšie nastavené hodnoty bola chybovosť algoritmu menšia. Pri miere istoty 0,3 sa algoritmus rozhodoval len na základe rozpoznávača textu. Algoritmus dosahoval najlepšie výsledky pri miere istoty 0,4. Pri tomto nastavení bola dosiahnutá hodnota metriky CER 2,32%. Samotný rozpoznávač textu dosahoval na komplexnej dátovej sade hodnotu 2,36%. Tento výsledok bol spôsobený veľkou mierou istoty rozpoznávača textu pri špeciálnych znakoch a číslach. Prevažná korekcia bola vykonávaná na znakoch abecedy. Pri jednoduchej dáto-

```
probyl v ČSR
probyt v če eser


Podle údajů MNV
podle údaju emen we


obce složky MV NF, VO KSČ a AV
obce složkyvv v okaes če a a we
```

Obr. 5.4: Ukážka niektorých riadkov z jednoduchej dátovej sady. Kde vrchný riadok reprezentuje výstup rozpoznávača textu a dolný riadok reprezentuje výstup rozpoznávača reči.

Tabuľka 5.2: Metrika CER vyhodnotená na jednotlivých dátových sadách pre navrhnuté algoritmy spojovania.

	jednoduchá	komplexná	jednoduchá + komplexná
Algoritmus 1	8.64	8.32	8.52
Algoritmus 2	22.01	26.18	23.54
Algoritmus 3	3.78	6.76	4.87
Algoritmus 4	**3.03**	**2.32**	**2.77**

vej sade algoritmus nedosiahol lepšie výsledky v porovnaní s rozpoznávačom textu. Lepšie výsledky neboli dosiahnuté ani v porovnaní s jednoduchou dátovou sadou. Pri ručnom prechádzaní jednoduchej dátovej sady sa zistilo, že táto dátová sada obsahuje veľké množstvo skratkových slov. Príklad takýchto slov sa nachádza na obrázku 5.4. Algoritmus mal najväčšiu chybovosť práve na znakoch takýchto slov.

Neurónové siete. Pri korekcii výstupov pomocou neurónových sietí boli natrénované 3 modely - rekurentná neurónová sieť, rekurentná neurónová sieť s 1 krát podvzorkovaným vstupom a rekurentná sieť s augmentovanou dátovou sadou. Pre rekurentnú neurónovú sieť s augmentovanou dátovou sadou, boli z trénovacej sady vybraté nahrávky, ktorých metrika CER z rozpoznávača reči pochádzala z intervalu (0; 25). Vybratá dátová sada bola augmentovaná pomocou modifikácie nahrávok gaussovským šumom. Výsledná veľkosť trénovacej sady po augmentácií bola 3832 vzoriek. Na obrázku 5.5 sa nachádza vývoj CER metriky pre trénovaciu sadu. Model CRNN-aug bol natrénovaný v menšom počte epôch ako zvyšné dva modely. Toto môže byť dôsledok toho, že bol trénovaný na dátovej sade, ktorá neobsahovala veľké množstvo chýb a bola augmentovaná. V rámci rozpoznávača s augmentovanou dátová sada bola táto evaulvácia robená na rovnakej trénovacej sade ako pre zvyšné modely. Tabuľka 5.6 obsahuje vyhodnotenie modelov na testovacích sadách. Model, ktorý bol trénovaný na augmentovanej dátovej sade dosahoval najlepšie výsledky metriky CER 16.74% pre jednoduchú dátovú sadu. Pre komplexnú a jednoduchú + komplexnú dátovú sadu dosahovala najlepšie výsledky rekurentná sieť bez podvzorkovania vstupu a augmentácie s metrikou CER 26.66% pre komplexnú a 20.64% pre jednoduchú + komplexnú dátovú sadu.

Tabuľka 5.3: Metrika CER vyhodnotená pre algoritmus 3 s rôznym počtom prehľadávaní k najlepších logitov.

hĺbka prehľadávania	jednoduchá	komplexná	jednoduchá + komplexná
3	3.96	6.99	5.07
6	**3.78**	6.76	**4.87**
9	3.84	**6.71**	4.90

Tabuľka 5.4: Metrika CER vyhodnotená pre algoritmus 4 s rôznym počtom prehľadávaní k najlepších logitov.

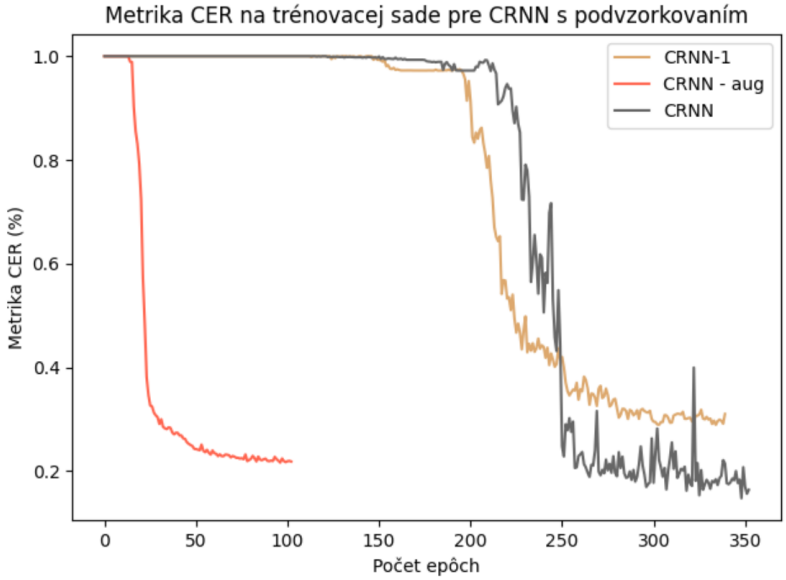
hĺbka prehľadávania	jednoduchá	komplexná	jednoduchá + komplexná
3	3.31	2.78	3.12
6	3.37	2.74	3.14
9	3.37	2.74	3.14

Tabuľka 5.5: Metrika CER vyhodnotená pre algoritmus4 s rôznymi nastaveniami miery istoty. Miera istoty v tomto prípade reprezentuje rozhodovaciu hranicu. Ak je pravdepodobnosť znaku rozpoznávača menšia ako táto hranica, algoritmus prehľadáva výstup druhého rozpoznávača.

miera istoty	jednoduchá	komplexní	jednoduchá + komplexní
0.3	3.03	2.36	2.78
0.4	3.03	**2.32**	**2.77**
0.5	3.03	2.33	2.78
0.6	3.11	2.40	2.84
0.7	3.14	2.56	3.00
0.8	3.37	2.74	3.14
0.9	3.66	2.91	3.38

Tabuľka 5.6: Metrika CER vyhodnotená pre RCNN sieť s rôznymi nastaveniami podvzorkovania.

Typ siete	jednoduchá	komplexná	jednoduchá + komplexná
CRNN	17.03	**26.66**	**20.64**
CRNN-1	24.14	38.23	29.34
CRNN - aug	**16.74**	33.77	23.18



Obr. 5.5: Zobrazenie vývoja hodnoty metriky CER na pôvodnej trénovacej sade. CRNN reprezentuje konvolučnú rekurentnú neurónovú sieť, CRNN-1 reprezentuje sieť, ktorej vstup bol 1 krát podvzorkovaný, CRNN-aug reprezentuje sieť, ktorej trénovacia sada bola augmentovaná pomocou gaussovského šumu.

Správný prepis: Tržba z 1 ha polí osázených bramborami byla 5683,30.
Prepis z rozpoznávača reči: tržba z jednoho hektaru poli osazených pramborami byla pět tisíc šest set osmdesat tři celých třicet
Prepis z rozpoznávača písaného textu: Tržba z 1 ha polí osázených bramborami byla 5683,30

Obr. 5.6: Ukážka výstupu rozpoznávača reči, rozpoznávača textu a správneho prepisu.

Zhodnotenie výsledkov. Najlepšie výsledky pre korekciu rozpoznávača textu pomocou rozpoznávača reči boli dosiahnuté pomocou algoritmu4 (2.77%). Tento algoritmus bol založený na princípe prehľadávania logitov druhého rozpoznávača v prípade, že pravdepodobnosť znaku pre rozpoznávač textu bol pod zvolenou mierou istoty. Ako najlepšia miera istoty sa ukázala hodnota 0,4. Tento algoritmus mohol najlepšie výsledky dosahovať z dôvodu nízkej hodnoty metriky CER pri rozpoznávači textu (2.78%). Ako druhý najlepší prístup sa ukázal algoritmus3 (4,87%), ktorý iteroval logitmy zarovnanými pomocou levenshtainového zarovnania a naximalizoval spoločnú pravdepodobnosť znaku. V porovnaní s algoritmom1 (8.52%), ktorý mal na vstupe zarovnané siete zmätenia, mohol dosahovať lepšie výsledky na základe väčšieho počtu znakov, pre ktoré sa snažil maximalizovať spoločnú pravdepodobnosť. Pri experimentovaní s väčším počtom znakov v sieti zmätenia sa do siete začali pridávať hodnoty None. Najhoršie výsledky boli dosiahnuté pri použití algoritmu2 (22.01%). Algoritmus robil najväčšiu chybovosť v prípade nezhodnutia sa oboma modalitami na niektorom znaku.

Pri korekcii výstupov pomocou neurónových sietí a algoritmov, neurónové siete dosahovali vyššiu chybovosť. Táto chybovosť môže byť spôsobená rozdielnou dĺžkou sekvencií a rozdielnou abecedou rozpoznávačov. Zatiaľ čo pri algoritmoch sa ako vhodné ukázalo zarovnanie sekvencie pomocou minimalizácie levenshtainovej vzdialenosti a vyhodnocovanie zarovnaných výstupov, pre ktoré boli definované pravidlá v prípade špeciálnych znakov, neurónové siete sa tieto súvislosti museli naučiť.

Vplyv na výsledky mohla mať dátová sada, na ktorej boli neurónové siete trénované. Z dôvodu komplexnosti jazyka mohlo byť náročné prepojenie niektorých súvislostí. Príkladom je obrázok 5.6 na ktorom sa nachádzajú výstupy jednotlivých rozpoznávačov. Ťažko mapovateľné boli skratkové slová, symboly a číslice vyjadrené v jednej modalite iným spôsobom ako v modalite druhej. Príkladom môže byť číslica 3, ktorá bola na výstupe rozpoznávača textu reprezentovaná číslom a na výstupe rozpoznávača reči reprezentovaná slovom. Pri vytváraní nahrávok, nebol jednotný postup nahrávania u všetkých rečníkov. Niektorí rečníci prečítali skratky ako boli napísané v texte - napríklad "ha", iní toto slovo prečítali ako "hektár". Rovnako aj pri špeciálnych symboloch, kde niektorí rečníci tieto symboly úplne vynechávali, zatiaľ čo iní ich v texte čítali. Niektoré nahrávky boli príliš zašumené a nebolo na nich jasne počuť, čo rečník v danej nahrávke hovorí. Tieto dôvody mohli mať vplyv pri učení neurónových sieti. Lepšie výsledky ako samotný rozpoznávač textu dosiahol algoritmus4, pre komplexnú dátovú sadu, ktorý bol založený na princípe prehľadávania logitov pre zarovnané sekvencie v prípade, že znak na výstupe rozpoznávača textu je málo pravdepodobný.

Pri ručnej kontrole sa ako problematické ukázalo, že rozpoznávač reči robil najčastejšie chyby pri určovaní medzier v texte. V niektorých prípadoch bol text prečítaný rýchlo a rozpoznávač nevedel určiť, že sa jednalo o medzeru medzi dvomi slovami. Na testovacej sade robil rozpoznávač časté chyby v znaku ř, namiesto ktorého dával znak ž. Toto mohlo byť spôsobené tým že v rámci vytvárania testovacej dátovej sady sa pri vytváraní podieľali aj slovenský občania, ktorý mali problém pri čítaní niektorých českých znakov a slov. Toto mohlo mať taktiež vplyv na celkové výsledky.

Kapitola 6

Závěr

Cieľom tejto práce bolo vytvorenie métódy aplikovateľnej pre korekciu výstupov rozpoznávača textu pomocou rozpoznávača reči. V rámci práce bolo navrhnutých niekoľko možných návrhov a niektoré z nich boli implementované. Navrhnuté riešenia boli založené na algoritmických prístupoch a prístupoch riešených pomocou neurónových sietí.

V práci mali menšiu chybovosť prístupy založené na algoritmoch, čo môže byť spôsobené tým, že boli presne definované kroky, na základe ktorých sa má algoritmus správať. Toto malo výhodu najmä pri špeciálnych znakoch a číslach, zatiaľ čo neurónové siete sa jednotlivé vzťahy museli naučiť. Vplyv na výsledky mohla mať aj vytvorená dátová sada, kde niektoré slová neboli rečníkmi čítané jednotnou formou. Iba jeden z navrhnutých prístupov dosiahol lepšie výsledky ako samotný rozpoznávač textu na komplexnej dátovej sade. Bol to prístup založený na prehľadávaní logitov, ktorých znaky boli zarovnané pomocou levenshtainového zarovnania a algoritmus sa pozeral na výstup rozpoznávača reči len v prípade neistého znaku na výstupe rozpoznávača textu. Tento prístup dosiahol hodnotu metriky CER 2.32% zatiaľ čo CER rozpoznávača textu bola 2.36%. Slabé výsledky mohli byť spôsobené komplexnosťou jazyka a ťažkým mapovaním rôznych reprezentácií výrazov. Nakoľko výstup z rozpoznávača textu bol v prípade špeciálnych znakov a čísel reprezentovaný 1 znakom, zatiaľ čo pri rozpoznávači textu to bolo jedno slovo.

Súčasťou práce bolo predspracovanie troch dátových sád, ktoré slúžili na dotrénovanie rozpoznávača reči použitého pre korekciu výstupov rozpoznávača textu. Dve dátové sady obsahovali nahrávky v českom a jedna v slovenskom jazyku. Rozpoznávač textu bol prevzatý. Pri vytváraní vlastnej dátovej sady boli prevzaté textové prepisy, ku ktorým bolo potrebné vytvoriť nahrávky. Pre získanie nahrávok bol implementovaný anotačný server. Vytvorená dátová sada nahrávok, by do budúcna mohla slúžiť aj na iné účely, ako napríklad rozpoznávanie a identifikáciu rečníka. Na získaných prepisoch a nahrávkach boli vyhodnotené implementované metódy pre korekciu rozpoznávača textu pomocou rozpoznávača reči. Opísané boli tiež ďalšie faktory, ktoré mohli mať vplyv na dosiahnuté výsledky a možný budúci vývoj práce.

Pre budúci vývoj práce by bolo vhodné urobiť experimenty pre korekciu výstupov s modelmi ktoré používajú attention. Príkladom takéhoto modelu je seq2seq model opísaný v návrhu, ktorý pracoval na princípe enkódera a attention pre logity každého rozpoznávača osobitne. Výstupom attention by boli kontextové vektory ktoré sú skonkatenované a tvoria vstup dekódera.

Overiť by sa mohla korekcia výstupov rozpoznávačov pre iný jazyk, napríklad angličtinu, ktorá má jednoduchšie pravidla a skloňovanie ako čeština. Vhodné by bolo natrénovanie modelu na zaobstaranej väčšej dátovej sade, ktorá by neobsahovala veľké množstvo špe-

ciálnych znakov, čísel a skratiek. Pri rozširovaní dátovej sady do budúcna by bolo vhodné oboznámiť rečníkov so správnym čítaním jednotlivých znakov, symbolov a čísel.

Za pokus by tiež stálo špecifikovať sa na korekciu výstupu rozpoznávača textu pomocou rozpoznávača reči pre nejakú úzku doménu.

Taktiež by mohla byť vyskúšaná korekcia pre jedného rečníka, na ktorého písmo a reč by boli rozpoznávače natrénované.

Literatúra

[1] Abdel Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G. et al. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2014, zv. 22, č. 10, s. 1533–1545. DOI: 10.1109/TASLP.2014.2339736.

[2] Albawi, S., Mohammed, T. A. a Al Zawi, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, s. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[3] Alex Graves, A.-r. M. a Hinton, G. Speech Recognition with Deep Recurrent Neural Networks , Department of Computer Science, University of Toronto. *Department of Computer Science, University of Toronto*. IEEE. 2013, zv. 3, č. 3, s. 45–49.

[4] Arica, N. a Yarman Vural, F. T. An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*. 2001, zv. 31, č. 2, s. 216–233. DOI: 10.1109/5326.941845. ISSN 10946977.

[5] Arica, N. a Yarman Vural, F. One-dimensional representation of two-dimensional information for HMM based handwriting recognition. *Pattern Recognition Letters*. Elsevier BV. jun 2000, zv. 21, 6-7, s. 583–592. DOI: 10.1016/s0167-8655(00)00023-4. Dostupné z: https://doi.org/10.1016%2Fs0167-8655%2800%2900023-4.

[6] B. H. Juang, L. R. R. The Segmental K-means Algorithm for Estimating Parameters of Hidden Markov Models. *IEEE Trans. A.S.S.P.* 1999, zv. 38, č. 9, s. 1639–1641.

[7] Baevski, A., Zhou, H., Mohamed, A. a Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*. 2020, 2020-Decem, Figure 1, s. 1–19. ISSN 10495258.

[8] Chowdhury, A. a Vig, L. An efficient end-to-end neural model for handwritten text recognition. *British Machine Vision Conference 2018, BMVC 2018*. 2019, s. 1–11.

[9] Chowdhury, S. A., Hussein, A., Abdelali, A. a Ali, A. *Towards One Model to Rule All: Multilingual Strategy for Dialectal Code-Switching Arabic ASR*. arXiv, 2021. DOI: 10.48550/ARXIV.2105.14779. Dostupné z: https://arxiv.org/abs/2105.14779.

[10] Chung, J. S., Senior, A., Vinyals, O. a Zisserman, A. Lip Reading Sentences in the Wild. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, s. 3444–3453. DOI: 10.1109/CVPR.2017.367.

[11] Darmatasia a Fanany, M. I. Handwriting recognition on form document using convolutional neural network and support vector machines (CNN-SVM). *2017 5th International Conference on Information and Communication Technology, ICoIC7 2017*. 2017, April. DOI: 10.1109/ICoICT.2017.8074699.

[12] Dey, R. a Salem, F. M. Gate-variants of Gated Recurrent Unit (GRU) neural networks. In: *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2017, s. 1597–1600. DOI: 10.1109/MWSCAS.2017.8053243.

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

[14] Granell, E. a Martinez Hinarejos, C. D. Combining handwriting and speech recognition for transcribing historical handwritten documents. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. 2015, 2015-Novem, s. 126–130. DOI: 10.1109/ICDAR.2015.7333739. ISSN 15205363.

[15] Granell, E., Romero, V. a Martínez Hinarejos, C. D. Image–speech combination for interactive computer assisted transcription of handwritten documents. *Computer Vision and Image Understanding*. Elsevier Inc. 2019, zv. 180, February, s. 74–83. DOI: 10.1016/j.cviu.2019.01.009. ISSN 1090235X. Dostupné z: https://doi.org/10.1016/j.cviu.2019.01.009.

[16] Graves, A., Fernández, S., Gomez, F. a Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *ACM International Conference Proceeding Series*. 2006, zv. 148, s. 369–376. DOI: 10.1145/1143844.1143891.

[17] Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R. a Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*. 2017, zv. 28, č. 10, s. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924. ISSN 21622388.

[18] Hibat Allah, M., Ganahl, M., Hayward, L. E., Melko, R. G. a Carrasquilla, J. Recurrent neural network wave functions. *Physical Review Research*. American Physical Society. 2020, zv. 2, č. 2, s. 1–17. DOI: 10.1103/PhysRevResearch.2.023358. ISSN 26431564.

[19] Kanai, S., Fujiwara, Y. a Iwamura, S. Preventing Gradient Explosions in Gated Recurrent Units. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, sv. 30. Dostupné z: https://proceedings.neurips.cc/paper/2017/file/f2fc990265c712c49d51a18a32b39f0c-Paper.pdf.

[20] Kang, L., Toledo, J. I., Riba, P., Villegas, M., Fornés, A. et al. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2019, 11269 LNCS, s. 459–472. DOI: 10.1007/978-3-030-12939-2$_3$2. $ISSN$16113349.

[21] Kolen, J. F. a Kremer, S. C. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. *A Field Guide to Dynamical Recurrent Networks*. 2010. DOI: 10.1109/9780470544037.ch14.

[22] Le, X. H., Ho, H. V., Lee, G. a Jung, S. Application of Long Short-Term Memory (LSTM) neural network for flood forecasting. *Water (Switzerland)*. 2019, zv. 11, č. 7. DOI: 10.3390/w11071387. ISSN 20734441.

[23] Martínek, J., Lenc, L. a Král, P. Building an efficient OCR system for historical documents with little training data. *Neural Computing and Applications*. 2020, zv. 32, č. 23, s. 17209–17227. DOI: 10.1007/s00521-020-04910-x. ISSN 14333058.

[24] Memon, J., Sami, M., Khan, R. A. a Uddin, M. Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR). *IEEE Access*. 2020, zv. 8, s. 142642–142668. DOI: 10.1109/ACCESS.2020.3012542. ISSN 21693536.

[25] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M. a Shaalan, K. Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access*. 2019, zv. 7, s. 19143–19165. DOI: 10.1109/ACCESS.2019.2896880. ISSN 21693536.

[26] Nurseitov, D., Bostanbekov, K., Kanatov, M., Alimova, A., Abdallah, A. et al. Classification of handwritten names of cities and handwritten text recognition using various deep learning models. *Advances in Science, Technology and Engineering Systems*. 2020, zv. 5, č. 5, s. 934–943. DOI: 10.25046/AJ0505114. ISSN 24156698.

[27] Poder, T. G., Fisette, J. F. a Déry, V. Speech Recognition for Medical Dictation: Overview in Quebec and Systematic Review. *Journal of Medical Systems*. Journal of Medical Systems. 2018, zv. 42, č. 5. DOI: 10.1007/s10916-018-0947-0. ISSN 1573689X.

[28] Sancho García, I. a Piñero Zapata, M. Aumento de eventos cardiovasculares en pacientes tratados con clopidogrel y omeprazol simultáneamente: una revisión sistemática. *Evidentia*. 2021, zv. 18, č. 2, s. e12814. DOI: 10.1371/journal.pmed1000097. Dostupné z: http://ciberindex.com/p/ev/e12814.

[29] Shi, B., Bai, X. a Yao, C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017, zv. 39, č. 11, s. 2298–2304. DOI: 10.1109/TPAMI.2016.2646371. ISSN 01628828.

[30] S.Kranthi. Automated Number Plate Recognition. 2011, zv. 2, č. 1, s. 102. DOI: 10.47524/tjst.21.6.

[31] Sri. Yugandhar Manchala, Jayaram Kinthali, Kowshik Kotha a Kanithi Santosh Kumar, Jagilinki Jayalaxmi. Handwritten Text Recognition using Deep Learning with TensorFlow. *International Journal of Engineering Research and*. 2020, V9, č. 05, s. 594–600. DOI: 10.17577/ijertv9is050534.

[32] Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B. et al. Start, follow, read: End-to-end full-page handwriting recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2018, 11210 LNCS, s. 372–388. DOI: 10.1007/978-3-030-01231-$1_2$3. $ISSN$16113349.

[33] Yi, C., Wang, J., Cheng, N., Zhou, S. a Xu, B. Applying wav2vec2.0 to Speech Recognition in various low-resource languages. *ArXiv*. 2020, abs/2012.12121.

Príloha A

Obsah DVD

Obsah priloženého DVD:

- annotation_server/ - skripty ku vytvoreniu anotačného servera
- asr/- adresár so skriptami pre natrénovanie rozpoznávača reči
- correction_outputs/ - adresár so skriptami pre korekciu výstupu rozpoznávačov
- politic_voices/ - adresár so skriptami pre spracovanie politických nahrávok
- data_example/ - adresár s ukážkou dátových sád
  - logits/ - ukážka logitov
  - records/ - ukážka získaných nahrávok
  - lov_records/ - ukážka nahrávok v zlej kvalite
- text/ - textová časť práce
- video/ - adresár s videom
- requirement.txt - súbor obsahujúci jednotlivé knižnice použité v práci
- Readme.txt - súbor popisujúci jednotlivé skripty