TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Využití neuronových sítí pro automatickou fonetickou transkripci

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **František Kynych**
Vedoucí práce: Ing. Petr Červa, Ph.D.
Konzultant: Ing. Jiří Málek, Ph.D.

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

The use of neural networks for automatic phonetic transcription

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **František Kynych**
Supervisor: Ing. Petr Červa, Ph.D.
Consultant: Ing. Jiří Málek, Ph.D.

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**František Kynych**
Osobní číslo:	**M15000036**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Využití neuronových sítí pro automatickou fonetickou transkripci**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s problematikou fonetické transkripce češtiny a dalších několika vybraných jazyků.
2. Ve zvoleném frameworku natrénujte automatický generátor fonetické transkripce využívající neuronové sítě.
3. Experimentálně vyhodnoťte a porovnejte přesnost vytvořených generátorů pro různé vnitřní architektury i nastavení hyperparametrů.
4. Finální dosažené výsledky porovnejte pro češtinu i několik dalších jazyků se stávajícím generátorem používaným na TUL.
5. Pomocí cross-validace zkuste odhalit a opravit chyby v trénovacích datech pro češtinu, která jsou reprezentována poloautomaticky vytvořeným a postupně ručně opravovaným seznamem několika set tisíc slov s výslovnostmi.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30-40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Nishant Shukla: Machine Learning with TensorFlow, Manning Publications Company, 2017**

[2] **Online kurz "Natural Language Processing with Deep Learning", dostupný na http://web.stanford.edu/class/cs224n/syllabus.html**

Vedoucí bakalářské práce: **Ing. Petr Červa, Ph.D.**
Ústav informačních technologií a elektroniky

Konzultant bakalářské práce: **Ing. Jiří Málek, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **19. října 2017**

Termín odevzdání bakalářské práce: **14. května 2018**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 19. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

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

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 11.5. 2018

Podpis: 

Abstrakt

Tato bakalářská práce je zaměřena na automatickou fonetickou transkripci pomocí neuronových sítí. Hlavním cílem bylo využít neuronové sítě a výsledky porovnat s chybovostí dosud používaného systému Baseline pro automatické generování fonetické transkripce. K řešení je použit Sequence-to-Sequence G2P toolkit, který je vyvíjen skupinou CMU Sphinx přímo pro tuto úlohu a dále byl upraven Neural Machine Translation toolkit, ten původně sloužil pro překlad z jednoho jazyka do jiného, ale poskytoval řadu dalších metod, které bylo možné vyzkoušet. Pomocí těchto toolkitů se postupně hledala architektura rekurentní neuronové sítě s nejmenší chybovostí.

Výsledky experimentování byly porovnány na stejné sadě dat se systémem Baseline. Hlavním dosaženým výsledkem je menší chybovost tohoto systému, u češtiny se podařilo relativně snížit chybovost o 41,5 %, u angličtiny o 22 % a u švédštiny o 33,5 %. Pomocí nejlepšího modelu byly hledány chyby v české slovní zásobě, používané na ústavu ITE. Našlo se 10 515 potenciálních chyb, které se musely ručně kontrolovat, zhruba u 10 % z nich se jednalo o chyby skutečné.

Klíčová slova

automatická fonetická transkripce, rekurentní neuronové sítě, Sequence-to-Sequence G2P toolkit, Neural Machine Translation toolkit

Abstract

This bachelor thesis is focused on automatic phonetic transcription using neural networks. The main goal was to use neural networks and compare the results with the error rate of the Baseline system used to automatically generate phonetic transcription. Sequence-to-Sequence G2P toolkit, that was developed by CMU Sphinx group for this task, is used to solve the problem. Additionally, the Neural Machine Translation toolkit, which is used for translation from one language to another, was modified, because it provides a number of other methods that could be tested. Using these tools, the architecture of the recurrent neural network with the lowest error rate was searched.

The experimental results were compared on the same set of data with the Baseline system. The main achieved result is the lower error rate of this system, the error in the Czech was relatively reduced by 41.5 %, by 22 % in English and by 33.5 % in Swedish. Using the best model, mistakes were found in the Czech phonetic dictionary used at the ITE department. There were found 10,515 potential mistakes that had to be manually checked, roughly 10 % of them were real mistakes.

Keywords

automatic phonetic transcription, recurrent neural networks, Sequence-to-Sequence G2P toolkit, Neural Machine Translation toolkit

Obsah

Seznam tabulek

Seznam zkratek

EOS	End of Sequence
GD	Gradient Descent
GNMT	Google's Neural Machine Translation
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
NMT	Neural Machine Translation
PAC	Phonetic Alphabet for Czech
WER	Word Error Rate
WERR	Word Error Rate Reduction

Úvod

Tato bakalářská práce se věnuje automatické fonetické transkripci s použitím neuronových sítí. Jedná se o automatické generování výslovnosti slov, kdy se model rekurentní neuronové sítě automaticky učí pravidla pro transkripci z již používané slovní zásoby na ústavu ITE. Tato slovní zásoba je používána pro mnoho aplikací zabývajících se zpracováním řeči, např. nepřetržitý monitoring.

Přepis slov do jejich výslovnosti je složitý proces, například kvůli výslovnostem příjmení, měst, některých převzatých výrazů a cizích slov. Dosud se pro tuto úlohu používal systém Baseline, který nevyužívá neuronových sítí, ale statistického modelování. Pomocí rekurentních neuronových sítí by se mohlo dosáhnout lepších výsledků, než kterých dosahuje současný systém.

Cíle práce

Cílem bylo seznámit se s problematikou fonetické transkripce a pro tuto úlohu vybrat vhodné nejnovější toolkity. Vybrán byl Sequence-to-Sequence G2P toolkit, který je vyvíjen Carnegie Mellon Universitou a je určen přímo pro úlohu fonetické transkripce. Dalším je Neural Machine Translation (NMT) toolkit, ten je vyvíjen Google Research týmem a původně sloužil pro automatický překlad z jednoho jazyka do jiného, ale po úpravě šel použít i pro tuto úlohu. NMT toolkit byl vybrán z důvodu většího množství funkcí a některé z nich zatím nebyly aplikovány v této oblasti. První bylo potřeba tyto toolkity zprovoznit, aby se pomocí nich mohly provádět experimenty.

Dále bylo s jejich pomocí potřeba hledat optimální nastavení hyperparametrů rekurentní neuronové sítě, kdy se postupně experimentuje s různými hodnotami

se záměrem minimalizace Word Error Rate (WER) pro automatické generování transkripce daného jazyka. Motivací bylo překonání současně používaného systému Baseline a zjistit limit automatické transkripce pro různě složité jazyky.

Posledním cílem bylo nalezení chyb v české slovní zásobě pomocí modelu s nejmenší chybovostí. Chyby se hledaly cross validací, kdy je potřeba rozdělit data vždy na trénovací a testovací části a postupně hledat chyby v každé testovací části dat. Nalezené chyby bylo potřeba ručně projít, mohlo se jednat o chybu modelu, novou alternativní výslovnost nebo chybu ve slovní zásobě.

Současný stav automatické fonetické transkripce

Do roku 2016 se pro automatickou fonetickou transkripci používaly statistické přístupy, poté se s rozvojem hlubokých neuronových sítí začaly objevovat nové toolkity. Pokroku v rekurentních neuronových sítích pomohly články, ve kterých byly představeny attention mechanismy.

Carnegie Mellon Universita vyvíjí nástroj Sequence-to-Sequence G2P, který byl použit pro základní experimenty v této práci. Tento toolkit je od roku 2016 dostupný na portálu Github v repozitáři CMU Sphinx a dodnes se zde objevují nové aktualizace. Využitím Neural Machine Translation toolkitu bylo možné testovat na úloze fonetické transkripce i nejnovější metody, které se dosud používaly pouze pro strojový překlad.

Na ústavu ITE se dosud neexperimentovalo s využitím rekurentních neuronových sítí pro automatickou fonetickou transkripci slovních zásob.

Struktura práce

První část práce seznámí čtenáře s pravidly pro fonetickou transkripci českého jazyka a jejím použitím. Jsou zde také zmíněny některé odlišnosti dalších použitých jazyků a jejich složitost.

Další kapitola obsahuje teorii týkající se rekurentních neuronových sítí, popisuje princip fungování, metody používané pro generování různých výstupů sítě a učení

pomocí zpětné propagace gradientu. Také se věnuje metodám a architekturám, které byly použity v experimentálním trénování různých struktur modelů. Byly použity metody z nejnovějších publikací v této oblasti.

Poté práce již popisuje toolkity použité k experimentování, potřebné transformace slovní zásoby a samotné experimenty prováděné v toolkitu Sequence-to-Sequence G2P, kde se trénovaly modely za použití různých architektur, metod a jejich kombinací. Druhý Neural Machine Translation toolkit poskytoval více druhů attention mechanismů, dropout, různé architektury sítě a další metody. Jednotlivé experimenty jsou popsány, jejich výsledky jsou uvedeny v tabulkách a dosažené přesnosti dané metody jsou postupně porovnávány. Modely se celou dobu ladily na validační sadě a na konci každého experimentování je pro daný jazyk použita testovací sada, která stanovuje výslednou přesnost. Další částí je porovnání chybovosti modelu s naivním přístupem (odborný odhad parametrů) a optimalizací parametrů. Poslední částí je porovnání systému Baseline s dosaženými výsledky v této práci, kde je hlavním dosaženým údajem relativní snížení chybovosti, uváděném v procentech. Hodnoty pro toto porovnání byly získány na stejných datech pro češtinu, angličtinu a švédštinu.

Poslední část práce se zabývá odhalením chyb ve slovní zásobě pro český jazyk pomocí cross-validace. Slovník obsahující několik set tisíc slov byl rozdělen vždy na trénovací a testovací část. Po natrénování každého modelu se na testovací části hledaly potenciální chyby a celý seznam nalezených potenciálních chyb se musel ručně kontrolovat.

1 Problematika fonetické transkripce

Fonetická transkripce se zabývá přepisem textu do fonémů a v této práci se přepisují jednotlivá slova do odpovídajícího řetězce fonémů. Pro přepis slov do jejich výslovnosti jsou dána fonologická pravidla.

Aby bylo možné výslovnosti zapisovat, tak je potřeba fonetická abeceda, zde ve slovníku je použita fonetická abeceda pro češtinu (PAC).

Zastoupeny zde jsou tři různé jazyky a každý z nich má různou složitost pro úlohu fonetické transkripce.

1.1 Čeština

Před definicí pravidel je potřeba hlásky rozdělit na souhlásky a samohlásky, dále pak na souhlásky znělé a neznělé (párové nebo jedinečné). Tento přehled je uveden v tabulce 1.1 pomocí fonémů.

Tabulka 1.1: Dělení českých hlásek, zdroj: [1]

Samohlásky	a, á, e, é, i, í, o, ó, u, ú									
Znělé párové souhlásky	b	d	ď	g	z	ž	v	h	dz (C)	dž (Č)
Neznělé párové souhlásky	p	t	ť	k	s	š	f	ch (X)	c	č
Jedinečné souhlásky (znělé)	m, n, ň, l, j, r, ř									

1.1.1 Základní pravidla

V tabulce 1.2 jsou uvedena některá základní pravidla pro přepis písmen nebo spojení na foném dle české fonetické abecedy PAC.

Dále se přepisuje ě na [je], ale pouze v případě, že následuje souhlásku b, f, p nebo v.

Písmeno x můžeme přepsat na [gz], pokud za ním následuje jakákoliv znělá souhláska, nebo se může přepsat na [ks], pokud je za ním neznělá souhláska, ale zároveň nachází-li se před nebo mezi samohláskami. Jestliže je na počátku slova spojení ex a po tomto spojení následuje samohláska, tak se zapisuje jako [egz].

Tabulka 1.2: Základní pravidla pro přepis na foném

Písmeno / Spojení	**Foném**
ch	X
ů	ú
w	v
q	kv
y	i
ý	í
dě	ďe
tě	ťe
ně	ňe
di	ďi
ti	ťi
ni	ňi

1.1.2 Spodoba znělosti

Znělé souhlásky se v některých případech mohou vyslovovat jako neznělé a je to možné i v opačném případě. Prvním případem je výskyt znělé souhlásky na konci slova (např. led se přepíše na [let]) a druhým je vliv okolí párové souhlásky, tyto párové souhlásky jsou uvedeny v tabulce 1.1.

Může se změnit znělá souhláska na neznělou (např. hlou**b**ka → [hlou**p**ka], protože za písmenem b se nachází neznělé k), neznělá souhláska na znělou (např. lé**č**ba → [lé**Č**ba], po č následuje znělé b) nebo se může změnit skupina souhlásek (např. le**ck**do → [le**Cg**do], po písmenech c a k následuje znělé d).

Výjimka je u písmene v, kde dívka se přepíše do tvaru [díf ka], ale nezpůsobuje změnu, například u slova svatba.

1.2 Ostatní jazyky

Kromě češtiny se také použila slovní zásoba pro angličtinu a švédštinu. Důvodem k použití dalších jazyků byla jejich odlišná složitost pro úlohy fonetické transkripce. Angličtina je oproti češtině složitější, například transkripce pro samohlásky sousedící s písmenem *R* má 7 různých druhů výslovností (tzv. R-Colored vowels) a obsahuje řadu dalších pravidel. Naopak u švédštiny se zde například vyskytuje více samohlásek (ä, ö, å).

U češtiny se slovní zásoba vytvářela pomocí definovaných fonetických pravidel a dále se poloautomaticky doplňovala. U angličtiny a švédštiny jsou tyto slovní zásoby převzaté a vytvořeny byly odborníky na fonetickou transkripci.

Díky různé složitosti pravidel pro jednotlivé jazyky je možné porovnat jednotlivé architektury neuronových sítí a zjistit, jak se mění jejich hyperparametry a tím určit, v jakém rozsahu hodnot je doporučeno experimentovat s dalšími jazyky.

Pravidla všech použitých jazyků zde nejsou podrobně popsána, jelikož jejich rozbor není cílem této práce. Neuronová síť se tato pravidla učí přímo z poskytnutých dat pro trénování, bez jakéhokoliv definování pravidel uživatelem.

2 Základní principy rekurentních neuronových sítí

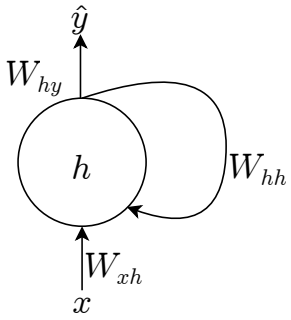
Rekurentní neuronové sítě jsou používány pro zpracování sekvencí dat $x_1, ..., x_t$. Strukturu mají podobnou klasickým dopředným neuronovým sítím, ale na rozdíl od nich mají zpětnou vazbu, díky ní jsou schopné predikovat hodnotu nejenom na základě příznaků na vstupní vrstvě, ale i z předchozích hodnot těchto příznaků. Tato architektura si uchovává uloženou informaci o kontextu ve svém skrytém stavu h_t, který se mění v každém kroku t.

$$h_t = f(h_{t-1}, x_t; \theta) \tag{2.1}$$

Skrytý stav je funkcí předchozího stavu h_{t-1} a současného vstupu x_t, θ jsou parametry této funkce f. Jedná se o vektor fixní délky, ten si nemusí uchovávat všechny vstupní hodnoty, stačí mít dostatek informací pro predikci další hodnoty sekvence.

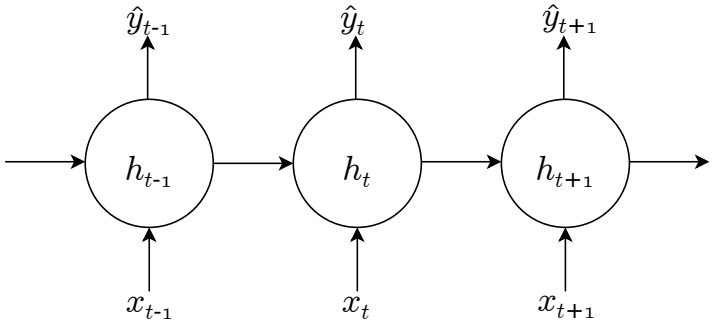
Na obrázku 2.1 je znázorněn jeden rekurentní neuron, kde:

- x je vstupní vrstva
- h je skrytý stav
- $\hat{y}$ je výstupní vrstva
- W_{xh}, W_{hh} a W_{hy} jsou matice vah



Obrázek 2.1: Rekurentní neuron

Na obrázku 2.2 je schéma rekurentního neuronu, který je pro přehlednost rozvinutý napříč časem, kde v každém časovém kroku t přijde nový vstup x a také je predikován výstup $\hat{y}$.

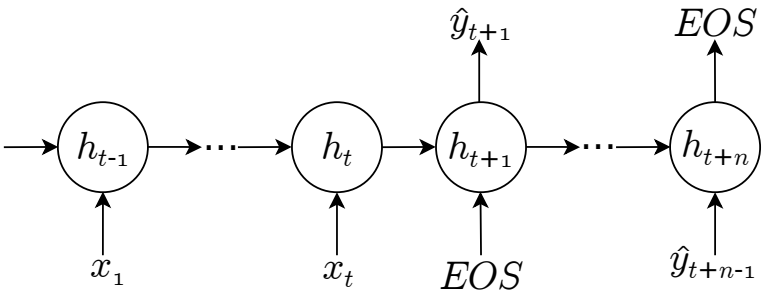


Obrázek 2.2: Rekuretní neuron rozvinutý napříč časem

Rekurentní neuronové sítě se mohou dělit na více druhů podle počtu vstupů a výstupů (viz [2]):

- seq to seq - při každé nové vstupní hodnotě x je predikován $\hat{y}$ (např. vstupem jsou ceny akcií za posledních n dní a výstupem jsou predikce na další den)
- seq to vector - první jsou sítí zpracovány vstupy x, výsledek $\hat{y}$ je použit z posledního kroku (např. vstupem je text a pouze v posledním kroku je výstupem sítě detekce sentimentu)
- vector to seq - v prvním časovém kroku síť obdrží počáteční stav x a nuly nebo $\hat{y}_{t-1}$ v každém dalším kroku (např. generování melodie z počáteční noty)

- delayed seq to seq - na vstupu jsou postupně hodnoty x a výstup $\hat{y}$ je ignorován, dokud není na vstupu end of sequence (EOS) token, který značí konec sekvence, výstupem je sekvence $\hat{y}$, dokud se na vstupu neobjeví tentýž EOS token (např. překlad z jednoho jazyka do jiného, používáno v této práci pro fonetickou transkripci)



Obrázek 2.3: Delayed sequence to sequence model

2.1 Dopředná propagace

U dopředné propagace je potřeba vypočítat skrytý stav h_t jako součin vah W_{hh} se vstupním vektorem x a k tomu je přičten součin vah W_{hh} s předchozím stavem h_{t-1}. Výsledek této operace je vstupem do aktivační funkce $\sigma(x)$. [3]

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1}) \tag{2.2}$$

Aktivační funkcí bývá často hyperbolický tangens (2.3) nebo se také používá funkce ReLU (2.4).

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

$$r(x) = max(0, x) \tag{2.4}$$

Posledním krokem (2.5) pro výpočet výstupu $\hat{y}_t$, je součin vah W_{hy} a skrytého stavu h_t, který je vstupem do funkce $softmax$.

$$\hat{y}_t = softmax(W_{hy}h_t) \tag{2.5}$$

Softmax (2.6) je klasifikátor, který zvýší rozdíly mezi hodnotami tím, že nejvyšší hodnota v exponenciále nejvíce vzroste a poté jsou všechny exponenciály normalizovány jejich součtem, kde C je počet tříd. Výsledkem je třída s nejvyšší pravděpodobností $\hat{P}(x_{t+1}|x_t, ..., x_1)$.

$$softmax(x) = \underset{c}{\operatorname{argmax}} \frac{e^x}{\sum_{c=1}^{C} e^{x_c}} \tag{2.6}$$

2.2 Trénování

Na výstupu se vypočítá chyba $softmax$ pomocí křížové entropie (2.7), snažíme se minimalizovat chybu mezi výstupem klasifikátoru a požadovanou (správnou) hodnotou, ta je reprezentována vektorem y, ve kterém jsou samé nuly kromě jedničky na indexu požadované třídy (tzv. one hot vector).

$$J_t(\theta) = -\sum_{c=1}^{C} y_{t,c} \log \hat{y}_{t,c} \tag{2.7}$$

Chybu je potřeba pomocí gradientu šířit celou neuronovou sítí i zpět v čase, tím dokážeme upravit matice vah W_{xh}, W_{hh} a W_{hy} tak, aby se tato chyba v dalších iteracích postupně snižovala. Pro vypočtení gradientu u každé matice vah je potřeba využít derivací a řetízkového pravidla.

Po získání všech gradientů se mohou metodou Gradient Descent změnit váhy posunutím proti směrnici parciální derivace, kde α je learning rate, ten udává, jak velké kroky se mají dělat. U ostatních matic vah probíhá tato změna stejně.

$$W_{hy} = W_{hy} - \alpha \frac{\partial J_t(\theta)}{\partial W_{hy}} \tag{2.8}$$

Problémem u zpětného šíření gradientu může být opakované násobení gradientu maticí vah, čímž může zanikat nebo naopak explodovat gradient. Pro ošetření zanikajícího gradientu se nejčastěji používá složitější vnitřní architektura rekurentního neuronu, pomůže i správná inicializace vah spolu s výběrem aktivační funkce. Explodující gradient může být redukován použitím aktivační funkce ReLu, regularizací vah, složitější architekturou sítě nebo za použití metody Gradient Clipping. Tato metoda kontroluje, zda hodnota gradientu není vyšší než hodnota zvoleného prahu, pokud je, tak se hodnota gradientu pouze nastaví na nižší předem specifikovanou hodnotu.

2.3 Optimalizace trénování

Pro zlepšení trénování existují metody, které například zrychlují Gradient descent, omezují oscilace při trénování nebo další metody pomáhají v tom, aby se model nepřetrénoval na trénovací množině dat, díky čemuž bude mít podobnou úspěšnost i na validační sadě dat.

2.3.1 Inicializace vah

Váhy modelu W je potřeba před začátkem trénování správně inicializovat, ty se inicializují náhodně, čímž se naruší symetrie, aby nebyl gradient pro všechny váhy stejný a každý neuron reprezentoval jiný příznak. Nejčastěji se používá Xavier (označována také Glorot) inicializace, která zároveň pomáhá s problémem zanikajícího a explodujícího gradientu.

2.3.2 Dropout

Tato jednoduchá technika se využívá k regularizaci modelu [6], kde má každý neuron pravděpodobnost p, se kterou nebude pro danou iteraci v učení modelu použit (výstup tohoto neuronu se nastaví na nulu). Tím v každé iteraci trénujeme mírně odlišný model s jiným počtem neuronů. Díky tomuto řešení se neuronová síť přímo

nespoléhá na konkrétní neuron (příznak) a snaží se, aby každý z nich poskytoval co nejlepší příznaky.

2.3.3 Learning rate decay

Learning rate decay pomáhá při konvergenci modelu tím, že se zmenšuje learning rate α ve chvíli, kdy již v dalších epochách neklesá hodnota chybové funkce $J(\theta)$, někdy se využívá i opětného zvyšování parametru α.

Dále uvedené metody pomáhají se zrychlením konvergence při trénování neuronových sítí, kde metoda Gradient descent (GD) dělá kroky pouze podle velikosti learning rate α.

2.3.4 Momentum

Momentum bere v ohled předchozí gradienty a ty vynásobené hyperparametrem β přičítá k aktuálně vypočítanému gradientu. Hodnota β se většinou nastavuje na 0,9 a α je learning rate, kterému se musí najít optimální hodnota.

$$\begin{aligned} v_{t+1} &= \beta v_t + \alpha \nabla_\theta J(\theta) \\ \theta &= \theta - v_{t+1} \end{aligned} \tag{2.9}$$

2.3.5 RMSProp

RMSProp zrychluje konvergenci tím, že omezuje oscilace. Pokud je nějaký z gradientů stále vyšší než ostatní, tak je zmenšen a naopak pro dimenze, kde jsou gradienty menší, se tyto gradienty snaží zvětšovat (2.10). Parametr ϵ se nastavuje na nízkou hodnotu (často na 10^{-8}), ten pouze zabraňuje dělení nulou. Symbol $\otimes$ označuje násobení po prvcích.

$$\begin{aligned} u_{t+1} &= \beta u_t + (1-\beta) \nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta) \\ \theta &= \theta - \alpha \frac{\nabla_\theta J(\theta)}{\sqrt{u_{t+1} + \epsilon}} \end{aligned} \tag{2.10}$$

2.3.6 Adam

Adam (Adaptive moments estimation) je kombinací metod Momentum (v_{t+1}) a RMSProp (u_{t+1}). Hodnota β_1 se často inicializuje na 0,9 a β_2 na 0,999. Symbol $\otimes$ označuje násobení po prvcích.

$$\begin{aligned} v_{t+1} &= \beta_1 v_t + (1-\beta_1)\nabla_\theta J(\theta) \\ u_{t+1} &= \beta_2 u_t + (1-\beta_2)\nabla_\theta J(\theta) \otimes \nabla_\theta J(\theta) \\ \hat{v}_{t+1} &= \frac{v_{t+1}}{1-\beta_1^T} \\ \hat{u}_{t+1} &= \frac{u_{t+1}}{1-\beta_2^T} \\ \theta &= \theta - \alpha \frac{\hat{v}_{t+1}}{\sqrt{\hat{u}_{t+1}+\epsilon}} \end{aligned} \tag{2.11}$$

2.4 Vícevrstvé neuronové sítě

Dosud byla zmíněna pouze síť s jednou skrytou vrstvou, ale těchto vrstev lze použít více. Vstup x_t bude vstupem do první skryté vrstvy, zde se z něj vypočítá skrytý stav h_t. Skrytý stav první vrstvy nebude použit jako vstup do funkce $softmax$, ale bude vstupem do další vrstvy této sítě, funkce $softmax$ bude využita až u poslední skryté vrstvy. Díky více vrstvám si dokáže neuronová síť vytvářet vlastní příznaky a tím často dosahuje lepší úspěšnosti v dané úloze. Sítě s více než jednou skrytou vrstvou jsou nazývány jako hluboké neuronové sítě.

2.5 LSTM

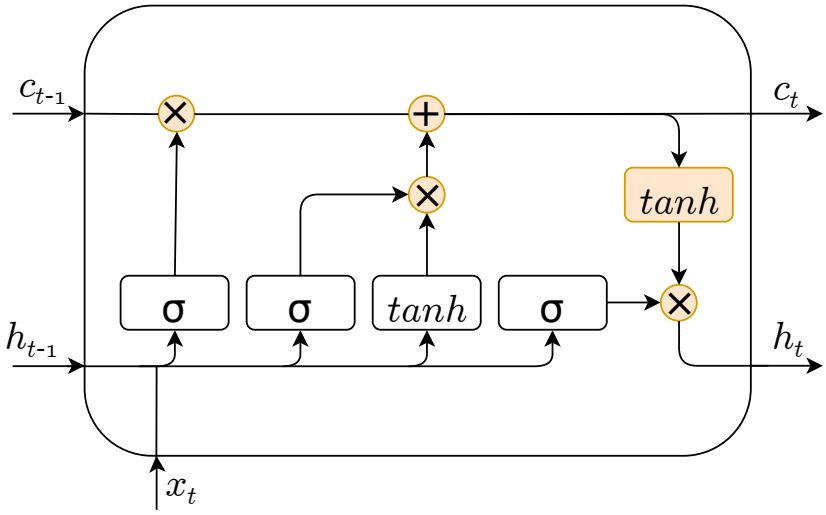
LSTM (Long Short-Term Memory) neuron se používá stejně jako obyčejný rekurentní neuron, ale nabízí lepší přesnost, rychlejší konvergenci a pamatuje si lépe dlouhodobé závislosti [7]. Tento neuron si uchovává dva skryté stavy h_t a c_t, kde h_t si uchovává krátkodobé a c_t dlouhodobé stavy.

Tento neuron si vybírá (2.12), které informace si má ze vstupu uchovat, které má zapomenout nebo použít pro výstup $\hat{y}_t$. Stav c_t vypočteme pomocí hodnoty f_t, ta je označována jako forget gate, protože určí, jak moc z předchozího stavu se promítne

do dalšího a k ní se přičte to, co bylo vybrané pomocí input gate i_t. Output gate o_t určuje kolik ze stavu c_t bude použito v h_t pro výstup neuronu. Symbol $\otimes$ označuje násobení po prvcích.

$$
\begin{aligned}
i_t &= \sigma(W_{xi}^T x_t + W_{hi}^T h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}^T x_t + W_{hf}^T h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}^T x_t + W_{ho}^T h_{t-1} + b_o) \\
g_t &= tanh(W_{xg}^T x_t + W_{hg}^T h_{t-1} + b_g) \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes g_t \\
\hat{y}_t &= h_t = o_t \otimes tanh(c_t)
\end{aligned}
\tag{2.12}
$$

Tato architektura pomáhá omezit problém zanikajícího gradientu tím, že nyní gradient může plynout zpět v čase pouze přes stav c_t, k němu je přičítána hodnota $i_t \otimes g_t$ a stav c_t je pouze násoben po prvcích hodnotou f_t.



Obrázek 2.4: Vnitřní schéma LSTM neuronu, barevně odlišené bloky označují operace prováděné po prvcích

2.6 GRU

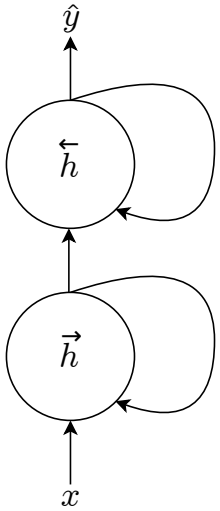
GRU (Gated Recurrent Unit) je zjednodušená (2.13) verze LSTM neuronu, která dosahuje podobných výsledků [8].

Oba skryté stavy, které byly u LSTM, jsou zde v jednom stavu h_t. Dále je použita pouze jedna hodnota z_t jako input i forget gate, pokud nabývá hodnoty 1, tak je input gate plně otevřen a forget gate zavřen. Nenachází se tu output gate, na výstupu je vždy celý stav h_t. Symbol $\otimes$ opět označuje násobení po prvcích.

$$\begin{aligned} z_t &= \sigma(W_{xz}^T x_t + W_{hz}^T h_{t-1}) \\ r_t &= \sigma(W_{xr}^T x_t + W_{hr}^T h_{t-1}) \\ g_t &= tanh(W_{xg}^T x_t + W_{hg}^T (r_t \otimes h_{t-1})) \\ h_t &= (1 - z_t) \otimes tanh(W_{xg}^T h_{t-1} + z_t \otimes g_t) \end{aligned} \tag{2.13}$$

2.7 Bidirekční rekurentní sítě

U standardních architektur rekurentní neuronové sítě může být problém v tom, že lze vstupní data zpracovat pouze v jednom směru tak, jak přichází. Bidirekční rekurentní neuronové sítě (viz 2.5) spočívají v tom, že obsahují skrytý stav $\overrightarrow{h_t}$ pro jeden směr a stav $\overleftarrow{h_t}$ pro druhý směr sekvence [9]. Nemusí být použit pouze standardní neuron, lze využít i LSTM nebo GRU.



Obrázek 2.5: Bidirekční architektura rekurentní neuronové sítě

2.8 Attention model

Tento model se chová podobně jako člověk, kdy při určování výstupu v daném časovém kroku počítá váhy, které říkají, na kterou část vstupu se má soustředit a díky tomu je výstup modelu při delších sekvencích lepší než bez použití attention mechanismu.

V každém kroku t se počítají váhy $\alpha_{t,t'}$, ty značí množství pozornosti, kterou by měl věnovat model pro predikci y_t, stavu $h_{t'}$, kde t' je časový krok, ve kterém byla na vstupu hodnota $x_{t'}$. Z těchto vah je vypočtena hodnota kontextu c_t a attention vektor a_t.

$$\begin{aligned} \alpha_{t,t'} &= \frac{e^{score(h_t,h_{t'})}}{\sum_{t'=1}^{T_x} score(h_t, h_{t'})} \\ c_t &= \sum_{t'=1}^{T_x} \alpha_{t,t'} h_{t'} \\ a_t &= tanh(W_{ca} c_t + W_{ha} h_t) \end{aligned} \tag{2.14}$$

K vypočtení vah $\alpha_{t,t'}$ je zapotřebí $score(h_t, h_{t'})$, to je získáno tím, že se naučí malá neuronová síť, která se učí spolu s rekurentní neuronovou sítí, kde je použit attention mechanismus. Tato neuronová síť má dva různé druhy podle autorů Luong (2.15) a Bahdanau (2.16), podrobněji jsou popsané v článcích Luong [10] a Bahdanau [11].

$$score(h_t, h_{t'}) = h_t^T W h_{t'} \tag{2.15}$$

$$score(h_t, h_{t'}) = v_a^t tanh(W_1 h_t + W_2 h_{t'}) \tag{2.16}$$

2.9 Beam search

Beam search pomáhá určit nejlepší možný výstup ze sítě tím, že se snaží vybrat takovou sekvenci znaků, která má dohromady nejvyšší pravděpodobnost (2.17).

Standardně bychom vybrali jako výstup v každém časovém kroku t znak s nejvyšší pravděpodobností (Greedy search). Beam search má nastavenou hodnotu B, která v každém kroku t vybere B nejpravděpodobnějších znaků a v dalším kroku hledá ke každému vybranému znaku dalších B znaků. Výsledkem je poté sekvence znaků, která má dohromady tuto pravděpodobnost nejvyšší [12].

$$\underset{y}{\operatorname{argmax}} \prod_{t=1}^{T_y} P(y_t|x, y_1, ..., y_{t-1}) \tag{2.17}$$

Tato metoda se využívá u delayed seq2seq modelů, kde se první ze vstupu vypočítají skryté stavy a poté se hledá B nejpravděpodobnějších znaků, ty jsou pak dány na vstup v dalším časovém kroku a hledají se další znaky stejným způsobem, dokud na výstupu není pro každou sekvenci EOS (end of sequence) token.

3 Provedené práce a dosažené výsledky

Pro hledání optimální architektury rekurentní neuronové sítě, která bude používána pro automatickou fonetickou transkripci byly vybrány dva frameworky, prvním je Sequence-to-Sequence G2P toolkit, který vyvíjí CMUSphinx, a druhý je nástroj pro tvorbu sequence to sequence modelů Neural Machine Translation (NMT), který poskytuje Google. Oba tyto nástroje jsou postaveny na frameworku Tensorflow společnosti Google.

3.1 Transformace slovníků

Před začátkem trénování bylo potřeba přetransformovat slovníky obsahující slovní zásobu. Nejprve se musely odstranit kolokace a speciální výslovnosti, dále bylo potřeba rozdělit tuto slovní zásobu na trénovací, validační a testovací sadu dat. Některá slova mají více možných výslovností, u této transformace se muselo zajistit, aby všechny výslovnosti k danému slovu byly ve stejné sadě dat.

3.2 Použití Sequence-to-Sequence G2P toolkitu

Sequence-to-Sequence G2P toolkit (viz [4]) je naprogramován s použitím Tensorflow frameworku, v době experimentování byla použita verze, která s několika úpravami fungovala na Tesnroflow verze 1.0. V době sepisování této práce již byly udělány změny, díky kterým funguje na verzi Tensorflow 1.5. Poskytován je skupinou CMU Sphinx, což je skupina vyvíjející systémy pro zpracování řeči na Carnegie Mellon univerzitě. G2P toolkit je součástí CMUSphinx toolkitu, ten byl vyvinut pro zpracování

řeči, dá se využít pro velké množství aplikací a G2P používá k vytvoření vlastního fonetického slovníku. Zpřístupnili také state-of-the-art model, který má architekturu s nejlepšími výsledky pro fonetickou transkripci angličtiny, ale používají jiný formát zápisu fonetické formy, proto se musela hledat optimální architektura i pro angličtinu.

Tento framework umožňuje experimentování s parametry rekurentní neuronové sítě, kde je možné měnit šířku a hloubku sítě, learning rate α, learning rate decay, max gradient norm (hodnota používána metodou Gradient clipping pro ošetření explodujícího gradientu), batch velikost (kolik dat je při trénování zpracováváno v jedné iteraci), počet iterací trénování, výběr optimalizace (GD, RMSProp, Adam) a je možné použít LSTM nebo GRU neuron.

Při nastavení počtu iterací na nulu se po každé iteraci kontroluje hodnota chybové funkce modelu $J(\theta)$, když neklesala poslední tři iterace, tak je snížen learning rate α vynásobením nastavené hodnoty pro learning rate decay, pokud ani v příštích třech epochách neklesla hodnota chybové funkce, tak se trénování tohoto modelu ukončí.

Během trénování modelu se vypisuje perplexita, ta je vypočítána jako $2^{J(\theta)}$ a hodnota blíže k 1 znamená menší chybu modelu. Přesnost se vypisuje až po skončení trénování modelu.

3.2.1 Základní model

Model, který tento toolkit používá, je možné modifikovat mnoha způsoby, základní architektura tohoto modelu je tvořena jako delayed sequence to sequence model, jehož dekodér využívá attention mechanismu a na výstupu může být oproti obyčejné softmax funkci tzv. sampled softmax. Sampled softmax se používá, pokud je velký počet výstupních tříd a s touto metodou není potřeba počítat výstupní pravděpodobnosti pro každou třídu ve slovníku, vybere se pouze podmnožina, se kterou se bude počítat chybová funkce $J(\theta)$.

Před trénováním modelu byla potřeba transformace dat slovníku na tři části - trénovací, validační a testovací část. Na validační sadě se testovala úspěšnost růz-

ných architektur a testovací sada se použila pouze jednou a to na konci trénování, kdy už byl na validační sadě nalezen nejlepší možný model. Tyto slovníky bylo potřeba přetransformovat do formátu, kde je na každém řádku slovo a mezerou odděleny fonémy daného slova.

3.2.2 Základní architektura a optimalizace jejích parametrů

3.2.2.1 Čeština

Jako první byl hledán nejlepší model pro český jazyk. Po transformaci slovníku měla validační i testovací sada 10 000 slov a trénovací sada obsahovala 594 922 slov.

Začalo se hledáním optimálního learning rate α na modelu se 2 vrstvami a 64 neurony v každé vrstvě, pro učení se využívalo metody Gradient descent, který je poté porovnán s optimalizací RMSProp a Adam. Hledání se realizovalo pomocí skriptu napsaného v Pythonu, postupně testoval různé hodnoty parametru α a s nimi trénoval zvlášť modely, kde se muselo natrénovat několik iterací, vyhodnotit chybu modelu, zaznamenat ji do XML souboru, takto dál pokračovat po dobu 10 000 iterací a s batch velikostí 128. Z výsledného XML souboru jsou udělány grafy, které zobrazují snižování chyby modelu.

Průběh chyby v závislosti na iteracích je zobrazen v grafu 3.1, jsou zde znázorněny všechny testované metody, ale pouze část jejich průběhu. Označení WER na svislé ose znamená Word Error Rate, tato hodnota udává chybu, která se vypočítá počtem chybných slov vydělených celkovým počtem slov ve slovníku. Přesnost modelu je 1 - WER a s použitím metody Gradient descent byla nejlepší dosažená přensnost $95,2\ \%$.

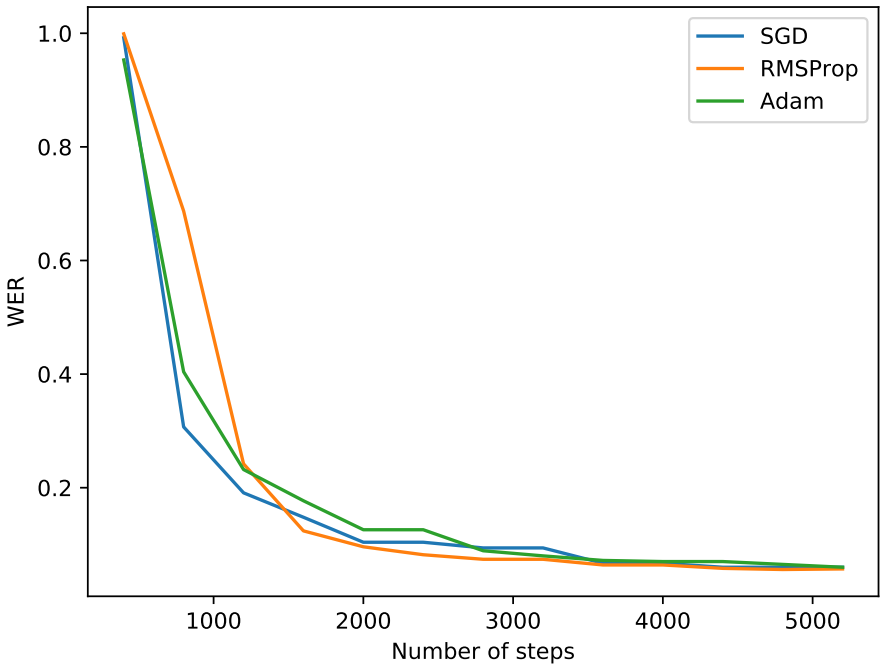
Optimální learning rate se hledal v rozsahu $\langle 0{,}0001;\ 1\rangle$, ve kterém se vybralo 10 hodnot, se kterými se trénovalo. Ale například u Gradient descentu byl jako minimální learning rate vybrán 0,05, tato hodnota již byla příliš malá, chyba modelu prvních 2400 iterací stagnovala a až při dalších iteracích se tato chyba pomalu začala snižovat, to je signál pro zvýšení hodnoty learning rate.

Po metodě Gradient descent byla vyzkoušena optimalizace RMSProp, se kterou

by měla být rychlejší konvergence, takže by výsledné grafy měly mít se stejným počtem iterací menší chybovost. Pro tyto další metody se znovu musel hledat optimální learning rate.

Learning rate se nyní hledal pouze na rozsahu $\langle 0{,}0001;\ 0{,}0512\rangle$, zde při výběru hodnoty 0,0001 se model neučil a stagnoval po celou dobu trénování (příliš malá hodnota). Naopak při nastavení learning rate na 0,0128 již WER osciloval a při zdvojnásobení tohoto parametru se WER minimálně snížil a poté začal divergovat (100% chybovost).

Dále se zkoušela optimalizace Adam, která by měla mít stejně jako RMSProp rychlejší konvergenci než Gradient descent. S použitím této optimalizace se dosáhlo nejmenší chyby ze všech testovaných metod, proto byla vybrána pro pokračování v hledání nejlepšího modelu. Zde již learning rate 0,0032 byl příliš velký, proto WER pouze osciloval, nejlepšího výsledku se dosáhlo s nastavením parametru α na 0,0005 a WER byl 4,2 %.



Graf 3.1: Průběh chyby modelu v závislosti na iteracích

Po nalezení nejlepší funkce a jejího parametru α se pokračovalo v hledání velikosti modelu. Šířka a počet vrstev modelu se hledaly postupně, kdy se natrénovala každá architektura s learning rate 0,0005, optimalizací Adam a batch velikostí 128, výsledky se ukládaly do XML souboru a poté z nich byl vybrán nejlepší model. Některé architektury byly svou úspěšností velmi blízko sebe, proto se vybral model, který měl nejméně chybných transkripcí a zároveň se přihlíželo i na úspěšnost na trénovací sadě. Zde jsou v tabulce 3.1 uvedeny pouze některé modely, ty nebyly omezeny počtem iterací, ale trénovaly se, dokud chybová funkce klesala, jakmile zůstala na stejné hodnotě, tak se po třech epochách trénování ukončilo.

Tabulka 3.1: Natrénovaný model pro češtinu s různým počtem vrstev a různou šířkou, $\text{WER}_{\text{valid}}$ udává chybnost na validační části dat a $\text{WER}_{\text{train}}$ na trénovací části

Počet vrstev	**Šířka**	**Chybných fonémů**	**$\text{WER}_{\text{valid}}$ [%]**	**$\text{WER}_{\text{train}}$ [%]**
1	256	231	2,3	1,8
2	128	234	2,3	2,0
3	128	207	2,1	1,0
3	256	208	2,1	1,0
4	128	215	2,1	1,5
4	256	212	2,1	1,0

Z výsledků byl vybrán model se třemi vrstvami a 128 neurony v každé vrstvě, s tímto modelem byly prováděny další experimenty. Model se čtyřmi vrstvami a 128 neurony je dále také testován u dalšího frameworku, jelikož má blízké výsledky nejlepšímu modelu, ale s menším rozdílem mezi úspěšností na validační a trénovací sadě.

Posledním experimentem, který se dal pomocí tohoto frameworku udělat, bylo hledání optimální batch velikosti. Tento parametr udává, kolik slov se zpracovává najednou v jedné iteraci a váhy modelu θ se mění až po zpracování celé dávky. U předchozích experimentů byla použita velikost dávky 128 a v této části se zkoušely menší i větší hodnoty. Výsledky jsou uvedeny v následující tabulce 3.2 a pro trénování byl použit model, který byl vybrán v předchozí části.

Batch velikost 512 dosahovala nejlepší úspěšnosti 98,1 %, výsledky tohoto modelu na testovací sadě jsou uvedeny v části 3.2.3.

Tabulka 3.2: Výsledky při hledání batch velikosti na modelu se třemi vrstvami a 128 neurony v každé vrstvě (čeština)

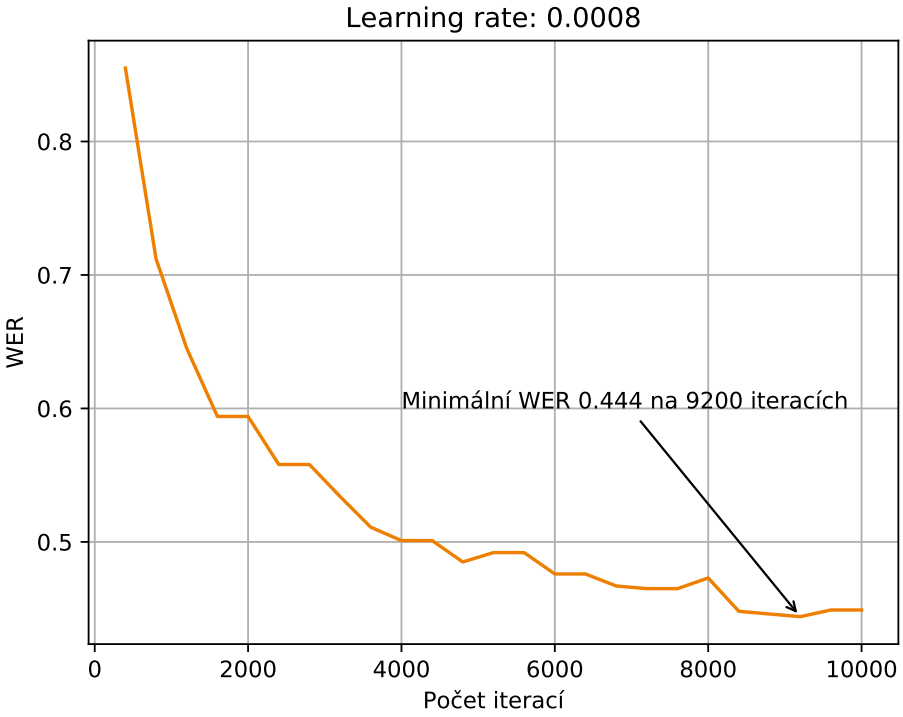
Batch velikost	**Chybných fonémů**	**$\mathbf{WER_{valid}}$** [%]
32	220	2,2
64	214	2,1
128	207	2,1
256	209	2,1
512	193	1,9

3.2.2.2 Angličtina

Opět bylo potřeba najít optimální velikost modelu, který bude mít největší úspěšnost při fonetické transkripci angličtiny. Slovník je zde oproti češtině menší, trénovací sada má zatím pouze 110 472 slov a velikosti validační i testovací sady zůstaly stejné jako u předchozího jazyka (10 000 slov). Testovací sada byla uložena stranou a použila se pouze pro stanovení závěrečné přesnosti.

Prvním krokem bylo opět hledání parametru learning rate α, ale použita byla jen Adam metoda pro optimalizaci. Tento parametr se hledal na intervalu $\langle 0{,}0001;0{,}0032\rangle$, kdy hodnota 0,0001 byla příliš malá a model na začátku trénování stagnoval, poté WER klesal příliš pomalu a naopak při hodnotě 0,0032 začínal WER mírně oscilovat. Ke hledání byl použit model se 2 vrstvami, 64 neurony v každé vrstvě a batch velikost byla 128.

Na grafu 3.2 je zobrazen průběh chyby model WER v závislosti na počtu iterací, nejmenší chybu se podařilo získat s hodnotou learning rate α 0,0008. Zároveň je na grafu vidět, že chyba neklesá tak rychle jako u modelu pro český jazyk, to je zde způsobeno menší velikostí trénovací sady a zároveň složitějšími pravidly pro transkripci angličtiny. Zatím byla nejlepší dosažená úspěšnost 55,6 %.



Graf 3.2: Průběh chyby modelu v závislosti na iteracích s použitím metody Adam

Dále bylo potřeba hledat velikost modelu, hloubka sítě se hledala postupně od 1 po 5 vrstev a u každého počtu vrstev se zkoušel různý počet neuronů. V tabulce 3.3 bylo vybráno několik modelů z prováděných experimentů. Některé jsou svými výsledky blízko, ale rozdíly zde jsou větší než u českého jazyka. Nejlepšího výsledku dosáhl model se třemi vrstvami a 128 neurony v každé z nich, úspěšnost měl 65,5 %.

Tabulka 3.3: Natrénovaný model pro angličtinu s různým počtem vrstev a různou šířkou

Počet vrstev	**Šířka**	**Chybných fonémů**	**$\mathbf{WER_{valid}}$ [%]**	**$\mathbf{WER_{train}}$ [%]**
1	256	3561	35,6	24,9
2	256	3589	35,9	25,3
3	128	3451	34,5	21,1
4	128	3478	34,8	21,2
4	256	3529	35,3	21,8
5	256	3891	38,9	24,2

Pokračovalo se hledáním optimální batch velikosti, u předchozích experimentů byla použita velikost dávky 128, zde se opět zkoušel větší i menší počet slov pro zpracování v jedné iteraci, výsledky jsou uvedeny v tabulce 3.4.

Tabulka 3.4: Výsledky při hledání batch velikosti na modelu se třemi vrstvami a 128 neurony v každé vrstvě (angličtina)

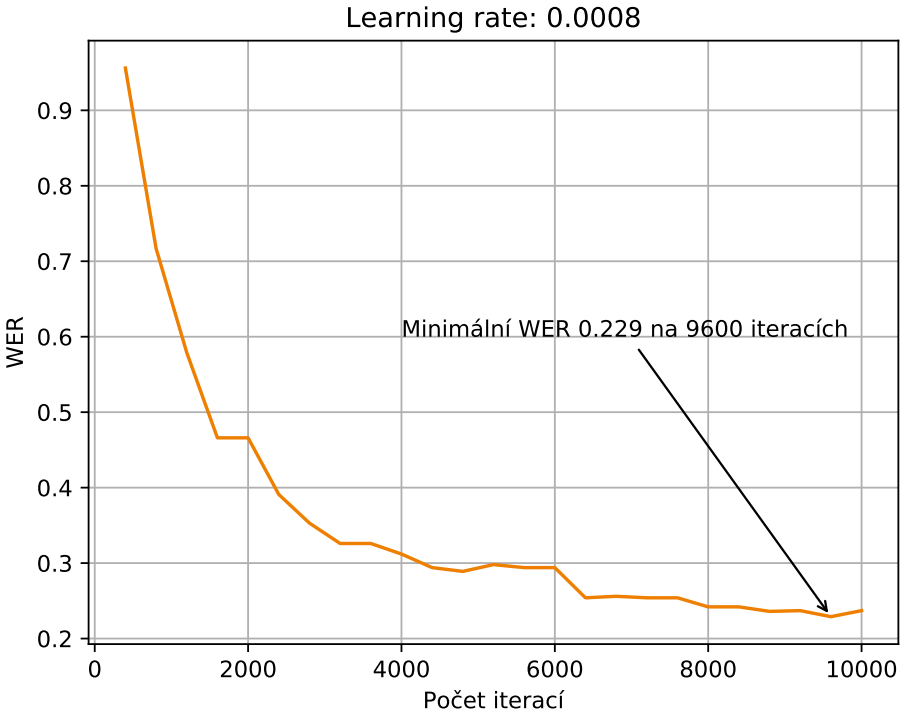
Batch velikost	**Chybných fonémů**	**$\mathbf{WER_{valid}}$ [%]**
32	3450	34,5
64	3396	34,0
128	3405	34,1
256	3519	35,2
512	3548	35,5

Nejvyšší dosažená přesnost pro angličtinu byla 66 % s batch velikostí 64, v této části se experimentovalo i s parametrem learning rate α a tento výsledek by získán s hodnotou 0,0006, ta byla v průběhu trénování snižována pomocí learning rate decay. Nízká přesnost je zde dána menší velikostí slovníku a složitějšími pravidly pro transkripci.

3.2.2.3 Švédština

Posledním jazykem, pro který se hledala nejlepší architektura, byla švédština, zde byla velikost slovníku větší než u angličtiny, trénovací množina obsahovala 280 522 slov a validační i testovací 10 000 slov.

Začalo se opět hledáním hodnoty learning rate α, se kterou bude model rychle konvergovat, použit byl jako u předchozích jazyků model se 2 vrstvami, 64 neurony v každé vrstvě a batch velikostí 128. Hledalo se na intervalu $\langle 0{,}0001; 0{,}0128 \rangle$, průběh nejlepší hodnoty 0,0008 je zobrazen v grafu 3.3, s tímto parametrem model dosáhl WER 22,9 %.



Graf 3.3: Průběh chyby modelu v závislosti na iteracích s použitím metody Adam

Po nalezení learning rate se experimentovalo s velikostí modelu, vybrané výsledky jsou uvedeny v tabulce 3.5. Nejlepšího výsledku dosáhla architektura se čtyřmi vrstvami, 128 neurony v každé z nich a dosáhla úspěšnosti 83,5 %.

Tabulka 3.5: Natrénovaný model pro švédštinu s různým počtem vrstev a různou šířkou

Počet vrstev	**Šířka**	**Chybných fonémů**	**$\mathbf{WER_{valid}}$ [%]**	**$\mathbf{WER_{train}}$ [%]**
1	128	1675	16,8	11,8
2	128	1615	16,2	10,3
3	128	1653	16,5	10,0
4	64	1719	17,2	12,6
4	128	1502	15,0	9,5

Naposledy se hledala batch velikost, která by mohla dosáhnout lepších výsledků než ta, která byla použita pro předchozí experimenty, výsledky jsou uvedeny v tabulce 3.6.

Tabulka 3.6: Výsledky při hledání batch velikosti na modelu se čtyřmi vrstvami a 128 neurony v každé vrstvě (švédština)

Batch velikost	**Chybných fonémů**	**$\mathbf{WER_{valid}}$** [%]
32	1506	15,1
64	1627	16,3
128	1502	15,0
256	1621	16,2
512	1687	16,9

S batch velikostí 128 se dosáhlo úspěšnosti 85 %, kdy se stejně jako u angličtiny experimentovalo s parametrem α, použila se hodnota 0,0006 a ta je v průběhu trénování zmenšována.

3.2.3 Celkové výsledky pro jednotlivé jazyky

V této části jsou uvedeny nejlepší dosažené výsledky pomocí frameworku Sequence-to-Sequence G2P toolkit na těchto jazycích a testovacích sadách, tyto sady dat nebyly do této doby použity a výsledky zde dosažené jsou finální. Testovací sada byla dále použita pouze u vyhodnocení druhého frameworku.

Tabulka 3.7: Nejlepší dosažené výsledky pro každý jazyk na validační a testovací sadě s toolkitem G2P

Jazyk	**Validační WER** [%]	**Testovací WER** [%]
Angličtina	34,0	34,6
Čeština	1,9	1,9
Švédština	15,0	16,0

Výsledky na testovací sadě jsou u angličtiny a švédštiny menší než na validační sadě, ale testovací sada zde představuje data, se kterými se síť ještě neviděla, tudíž se dalo očekávat zhoršení úspěšnosti oproti datům, na kterých se síť učila nebo ladila. U češtiny zůstala přesnost stejná.

3.3 Použití Neural Machine Translation toolkitu

Jedná se o Neural Machine Translation toolkit poskytovaný na stránkách Github skupinou Tensorflow a vyvinut byl Google Research týmem (viz [5]). Používaná verze

pro experimenty byla naprogramována Google týmem s použitím Tensorflow 1.5, na začátku trénování bylo potřeba stáhnout nightly verzi knihovny, protože aktuální stabilní verze byla 1.4, ale v této verzi se nacházela chyba v metodě Beam search, která byla opravena později.

Tento toolkit je určen pro delayed sequence to sequence aplikace, konkrétně pro překládání textů do jiných jazyků. Pro použití na fonetickou transkripci bylo pouze potřeba upravit formát slovníků a přepsat funkci na vyhodnocení přesnosti, protože původní kód nedovoloval více správných možností, naopak u fonetické transkripce mají některá slova ve slovníku více než jeden správný tvar fonémů.

Oproti předchozímu Sequence-to-Sequence G2P toolkitu je zde mnohem větší množství parametrů modelu, které lze měnit a také spousta metod, které předchozí framework neposkytoval. Například zde lze navíc použít bidirekční architekturu sítě nebo je poskytována GNMT architektura, kde je jedna vrstva bidirekční a zbytek vrstev standardní, dále je možné vybrat druh attention mechanismu, také tu je více možností, jak měnit během trénování hodnotu learning rate a inicializaci vah. Obsahuje i všechny metody, které používal předchozí framework.

V předchozím frameworku se nemusel zadat počet iterací a sám dokázal detekovat, kdy je vhodné ukončit trénování, ale zde se musí tento počet zadávat a trénování se zastaví, jakmile se dosáhne tohoto čísla. Na tomto principu jsou ale postaveny metody pro learning rate decay, kdy například s použitím luong5 se learning rate začne snižovat po polovině iterací a sníží se pětkrát před koncem trénování.

Výhodou zde je možnost použití Tensorboard, kde lze ve webovém prohlížeči sledovat grafy, ve kterých je znázorněna přesnost, hodnota chybové funkce, perplexita na trénovací i validační sadě a průběh hodnoty learning rate.

3.3.1 Další navržené architektury

3.3.1.1 Čeština

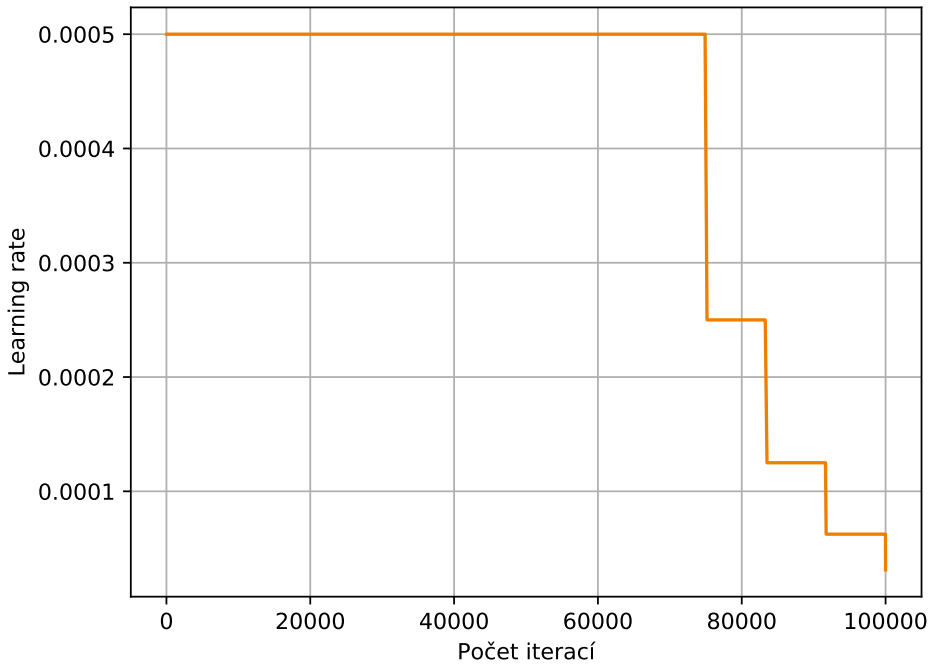
Jako první se zpracovávala znovu čeština, ale kroky, které se prováděly u předchozího frameworku, zde nebyly opakovány. Byly použity hodnoty, které se ukázaly jako

nejlepší a dále se experimentovalo s různými druhy attention mechanismu, bidirekční a GNMT architekturou a použil se Beam search.

Každý experiment byl proveden třikrát, jelikož při každém trénování se dosáhlo mírně odlišného výsledku. Framework poskytuje možnost použití random state, ale i se stejně nastavenou hodnotou se dosahovalo různých výsledků, příčinou mohla být inicializace vah modelu nebo náhodný výběr dávky dat pro trénování z trénovacího setu.

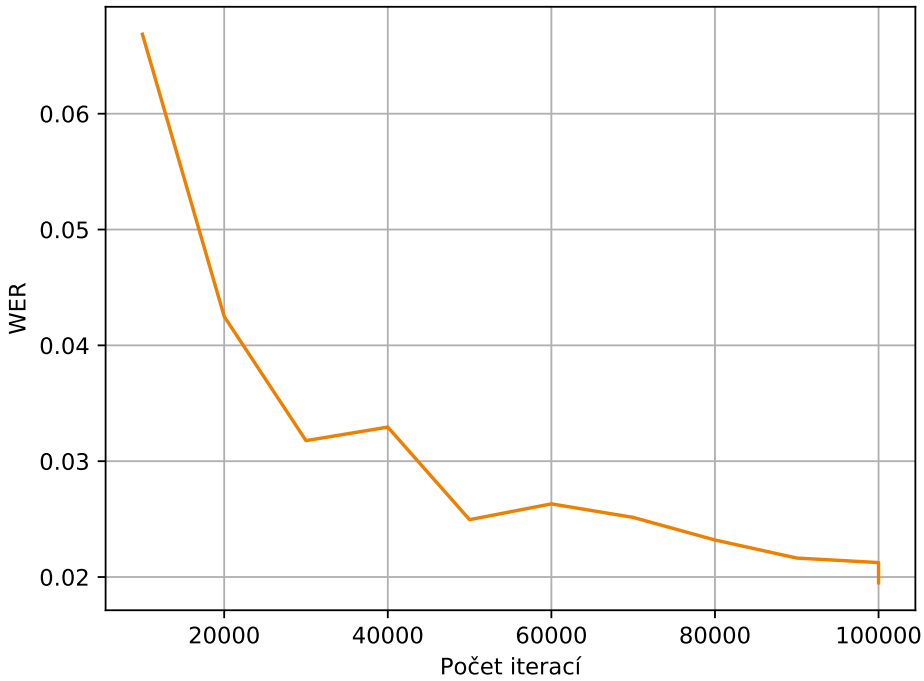
Prvním natrénovaným modelem byl pokus o dosažení stejných výsledků, kterých dosáhl předchozí framework. Learning rate byl nastaven na 0,0005, batch velikost 512, šířka modelu 128 a 3 vrstvy, decay scheme na luong234 (po dvou třetinách trénovacích kroků se do konce trénování learning rate sníží čtyřikrát), optimalizace Adam, attention Luong a trénovalo se 100 000 iterací. Výsledná úspěšnost byla u všech tří pokusů 98 %, ale v druhém pokusu se zvýšil počet iterací na 150 000 a v průměrném výsledku tří pokusů dosáhl model úspěšnosti 98,1 %.

V následujícím grafu 3.4 je zobrazeno, jak se v průběhu trénování snižuje learning rate s použitím metody Luong234.



Graf 3.4: Průběh parametru learning rate α v závislosti na iteracích

V dalším grafu 3.5 je vidět průběh WER při trénování modelu, kterým jsme se snažili dosáhnout stejných nejlepších výsledků jako u předchozího frameworku, pouze stačilo nastavit více iterací. Výsledky na validační sadě jsou ukládány každých 10 000 iterací.



Graf 3.5: Průběh WER modelu v závislosti na délce trénování (iteracích)

V pokusu s přidáním dropout, který pomůže s regularizací rekurentní neuronové sítě a mohl by i zvýšit úspěšnost, se dosáhlo malého zlepšení, výsledky jsou uvedeny v tabulce 3.8. Learning rate byl zvýšen na hodnotu 0,001 a trénovalo se 100 000 iterací, vyšší dropout hodnoty než 0,3 zhoršovaly přesnost.

Tabulka 3.8: Výsledné přesnosti pro různé hodnoty dropout (čeština)

Dropout	**WER** [%]
0,1	1,79
0,2	1,77
0,3	1,80

Pokračovalo se experimentováním s různými druhy attention mechanismů, poskytovány jsou Luong, Scaled Luong, Bahdanau a Normed Bahdanau. Výsledky

těchto experimentů jsou uvedeny v tabulce 3.9, při trénování byl použit dropout 0,2 a trénovalo se 100 000 iterací.

Tabulka 3.9: Výsledné přesnosti pro jednotlivé druhy attention mechanismu (čeština)

Attention	**WER** [%]
Luong	1,77
Scaled Luong	1,67
Bahdanau	1,70
Normed Bahdanau	1,77

Pro další experimenty byl používán Scaled Luong attention mechanismus.

V následujícím trénování je použita metoda Beam search, která v každém kroku na výstupu vybere daný počet možností s nejvyšší pravděpodobností a v každém dalším kroku vybírá opět další možnosti, správný výstup je poté sekvence znaků, která má dohromady nejvyšší pravděpodobnost. V tabulce 3.10 jsou uvedeny výsledky pro různý počet hodnot, které Beam search vybírá. Beam search s hodnotou 1 nemělo smysl testovat, je to stejné, jako kdyby se tato metoda nepoužila.

Tabulka 3.10: Výsledné přesnosti pro různé hodnoty metody Beam search (čeština)

Beam search	**WER** [%]
3	1,70
5	1,67
10	1,74

Dalším pokusem bylo použití jiné než standardní architektury, testována byla bidirekční a GNMT architektura, ta používá speciální attention mechanismus, zároveň obsahuje jednu bidirekční vrstvu a ostatní vrstvy v modelu jsou standardní. Attention mechanismu u GNMT architektury spojuje spodní vrstvu kodéru s poslední vrstvou dekodéru a tento model navíc používá residuální spoje. Trénovalo se opět 100 000 iterací a byl použit Luong attention mechanismus (u standardní a bidirekční), dropout 0,2 a nebyl použit Beam search. Pro bidirekční architekturu musel být zvolen sudý počet vrstev (vždy jedna vrstva se skrytým stavem $\overrightarrow{h_t}$ a druhá $\overleftarrow{h_t}$), použity byly 4 vrstvy a výsledky jsou uvedeny v tabulce 3.11.

Tabulka 3.11: Výsledné přesnosti pro jednotlivé architektury rekurentní neuronové sítě (čeština)

Architektura	**WER** [%]
Standardní	1,77
Bidirekční	1,70
GNMT	1,67
GNMT v2	1,60

Následovaly pokusy pro bidirekční a GNMT v2 architekturu, použily se dohromady nejlepší výsledky z předchozích kroků - dropout 0,2, attention Scaled Luong a Beam search 5, tyto experimenty se trénovaly 300 000 iterací, ale chybovost se ke konci trénování už nezlepšovala (cca posledních 50 000 iterací). U těchto architektur se i různě měnily další parametry, ale tato uvedená kombinace dosahovala nejlepších výsledků.

Tabulka 3.12: Výsledné přesnosti pro bidirekční a GNMT v2 architektury s použitím nejlepších metod z předchozích experimentů (čeština)

Architektura	**WER** [%]
Bidirekční	1,57
GNMT v2	1,50

Posledním experimentem 3.13 bylo použití GRU neuronu namísto LSTM, rychlost trénování byla téměř stejná a výsledky také velmi podobné. Model byl trénován s bidirekční sítí.

Tabulka 3.13: Porovnání GRU a LSTM neuronu na stejné architektuře modelu (čeština)

Typ neuronu	**WER** [%]
LSTM	1,54
GRU	1,57

Jako nejlepší architektura byla vybrána GNMT uvedená v tabulce 3.12, na této architektuře je na konci této kapitoly spolu s dalšími jazyky uvedena přesnost pro testovací sadu, která opět určí výslednou přesnost tohoto modelu.

3.3.1.2 Angličtina

Zde se podobně jako u předchozího jazyka vycházelo z modelu, který dosahoval nejlepších výsledku za použití toolkitu Sequence-to-Sequence G2P.

Prvním experimentem bylo použitím dropout metody, kde byl prozatím použit Luong attention mechanismus a model byl trénován 250 000 iterací, počet iterací je zde vyšší kvůli menší batch velikosti oproti češtině. Každý model se musel opět trénovat třikrát, aby byl viditelný rozptyl přesnosti modelu, výsledky jsou uvedeny v tabulce 3.14.

Tabulka 3.14: Výsledky po použití různých hodnot dropout (angličtina)

Dropout	**WER** [%]
0,2	31,90
0,3	31,84
0,4	31,87

Pokračovalo se hledáním optimálního attention mechanismu s použitím dropout 0,3 a trénovalo se 250 000 iterací, výsledky jsou uvedeny v tabulce 3.15.

Tabulka 3.15: Výsledné přesnosti pro jednotlivé druhy attention mechanismu (angličtina)

Attention	**WER** [%]
Luong	31,84
Scaled Luong	31,70
Bahdanau	31,74
Normed Bahdanau	31,67

Dále se experimentovalo s použitím metody Beam search. Trénovalo se 250 000 iterací, byl použit dropout 0,3 a Normed Bahdanau attention mechanismus, výsledky tohoto experimentu jsou uvedeny v tabulce 3.16.

Tabulka 3.16: Výsledné přesnosti pro různé hodnoty metody Beam search (angličtina)

Beam search	**WER** [%]
3	31,20
5	31,44
10	31,30

Po experimentování s hodnotou Beam search se hledala nejlepší architektura sítě, kde byla testována bidirekční, GNMT a GNMT v2 architektura. Použit byl Normed Bahdanaou attention mechanismus, dropout 0,3 a Beam search zde ještě nebyl použit, trénovalo se 300 000 iterací a výsledky jsou v tabulce 3.17.

Tabulka 3.17: Výsledné přesnosti pro jednotlivé architektury rekurentní neuronové sítě (angličtina)

Architektura	**WER** [%]
Standardní	31,00
Bidirekční	30,37
GNMT	30,10
GNMT v2	30,20

Pro poslední experiment s angličtinou byly vybrány modely s GNMT a GNMT v2 architekturou, s nimi byla použita metoda Beam search se šířkou 3, dropout 0,3 a Normed Bahdanau attention. Trénovalo se 700 000 iterací s learning rate 0,0006, výsledky jsou uvedeny v tabulce 3.18.

Tabulka 3.18: Výsledné přesnosti pro GNMT a GNMT v2 architektury s použitím nejlepších metod z předchozích experimentů (angličtina)

Architektura	**WER** [%]
GNMT	29,97
GNMT v2	30,37

Nejlepší dosažená úspěšnost na validační sadě byla 70,03 %, porovnání se současně používaným systémem je na konci této kapitoly.

3.3.1.3 Švédština

Posledním jazykem, se kterým se experimentovalo, byla švédština, zde se opět vycházelo z modelu, který dosahoval nejvyšší přesnosti s toolkitem Sequence-to-Sequence G2P.

Prvním experimentem bylo hledání hodnoty dropout (viz 3.19). K tomu byl použit model se 4 vrstvami a 128 neurony v každé z nich, learning rate byl nastaven na 0,0006 a k učení sítě se použila metoda Adam. Trénovalo se 150 000 iterací s batch velikostí 128.

Tabulka 3.19: Výsledné přesnosti pro různé hodnoty dropout (švédština)

Dropout	**WER** [%]
0,2	14,30
0,3	15,04
0,4	16,44

Pokračovalo se hledáním attention architektury, byl použit dropout 0,2 a trénovalo se 150 000 iterací, všechny ostatní hodnoty byly stejné jako u předchozího experimentu. Výsledky jsou uvedeny v tabulce 3.20.

Tabulka 3.20: Výsledné přesnosti pro různé druhy attention mechanismu (švédština)

Attention	**WER** [%]
Luong	14,30
Scaled Luong	14,00
Bahdanau	14,07
Normed Bahdanau	14,04

Všechny hodnoty použité v předchozím experimentu byly použity i zde, pouze attention mechanismus byl použit Scaled Luong. Dále se hledala hodnota Beam search, trénovalo se 150 000 iterací a výsledky jsou v tabulce 3.21.

Tabulka 3.21: Výsledné přesnosti pro různé hodnoty Beam search (švédština)

Beam search	**WER** [%]
3	14,17
5	13,77
10	14,14

Naposledy bylo potřeba najít správnou architekturu modelu, vycházelo se opět z předchozího experimentu, ale nebyl zde použit Beam search. Trénovalo se 150 000 iterací a nejlepších výsledků dosáhla bidirekční architektura (viz 3.22).

Tabulka 3.22: Výsledné přesnosti s různými architekturami rekurentní neuronové sítě (švédština)

Architektura	**WER** [%]
Standardní	14,30
Bidirekční	13,04
GNMT	13,24
GNMT v2	13,40

Pro poslední experiment se vybrala bidirekční a GNMT architektura, kdy se trénovalo 250 000 iterací a byl použit Scaled Luong attention, Beam search 5 a dropout 0,2. Celkově nejnižší přesnosti zde dosáhla bidirekční architektura, výsledky jsou v tabulce 3.23.

Tabulka 3.23: Výsledné přesnosti pro bidirekční a GNMT architektury s použitím nejlepších metod z předchozích experimentů (švédština)

Architektura	**WER** [%]
Bidirekční	12,87
GNMT	13,00

3.3.2 Celkové výsledky pro jednotlivé jazyky

S použitím Neural Machine Translation toolkitu a metod, které poskytuje, se dosáhlo lepších výsledků než u Sequence-to-Sequence G2P toolkitu. Zde jsou v tabulce 3.24 vidět přesnosti jednotlivých toolkitů na validační a testovací sadě. Testovací sada je použita pouze pro finální vyhodnocení přesnosti dané architektury.

Tabulka 3.24: Nejlepší dosažené výsledky pro každý jazyk na validační a testovací sadě s NMT toolkitem

Jazyk	**Validační WER** [%]	**Testovací WER** [%]
Angličtina	29,97	29,60
Čeština	1,50	1,70
Švédština	12,87	12,60

Výsledky z vyhodnocení testovací sady pro angličtinu a švédštinu dosáhly nižší chybovosti než s validační sadou. Díky tomu je vidět, že síť není přeučena a vyladěna pouze na validační množinu dat.

3.4 Shrnutí všech výsledků na testovací sadě

3.4.1 Vliv optimalizace

Všechna porovnání v této kapitole jsou provedena na testovací sadě. V tabulce 3.25 je zobrazen rozdíl výsledné přesnosti u pokusu o odborný odhad hyperparametrů

rekurentní sítě (naivní přístup) a postupné optimalizace těchto parametrů. Je zde vidět, že optimalizace se v této úloze vyplatila a síť s parametry z odhadu by nepřesáhla úspěšnost současného systému. Při odhadu byl použit model se 2 vrstvami a 150 neurony v každé z nich a learning rate 0,001 s použitím metody Adam.

Tabulka 3.25: Porovnání chybovosti (WER) naivního přístupu a po optimalizaci modelu s toolkitem G2P

Jazyk	**Baseline** [%]	**Naivní přístup** [%]	**Optimalizace** [%]
Angličtina	37,96	37,4	34,6
Čeština	2,91	3,1	1,9
Švédština	18,96	19,2	16,0

U toolkitu NMT se s počátečním odhadem hodnot vycházelo z parametrů, které byly získány pomocí optimalizace na G2P toolkitu. Pro každý jazyk se použil optimální počet vrstev, neuronů i learning rate a dále se použil Beam search 10, dropout 0,4 a Luong attention se standardní architekturou sítě, výsledný WER je uveden v tabulce 3.26. Již s použitím těchto metod a odhadnutými parametry se dosáhlo vyšší přesnosti než s toolkitem G2P.

Tabulka 3.26: Porovnání chybovosti (WER) naivního přístupu a po optimalizaci modelu s toolkitem NMT

Jazyk	**Baseline** [%]	**Naivní přístup** [%]	**Optimalizace** [%]
Angličtina	37,96	31,8	29,6
Čeština	2,91	1,8	1,7
Švédština	18,96	15,6	12,6

3.4.2 Porovnání použitých toolkitů se systémem Baseline

V této části jsou porovnány výsledky použití Sequence-to-Sequence G2P toolkitu, Neural Machine Translation toolkitu a systému Baseline.

Současný systém Baseline používá Phonetisaurus G2P Toolkit od společnosti Google, ten nevyužívá neuronové sítě, ale statistické modelování a vážený převodník s konečným počtem stavů.

Již G2P toolkit po optimalizaci dosáhl lepších výsledků než Baseline, ale s využitím metod a architektur NMT toolkitu, které se současně vyskytovaly pouze v úloze

strojového překladu, se podařilo získat ještě vyšší přesnost.

S použitím rekurentních neuronových sítí se u všech testovaných jazyků dosáhlo lepších výsledků a znatelně se redukovala chybovost úlohy fonetické transkripce.

V tabulce 3.27 je uvedena chybovost systému Baseline, dosažené chybovosti v této práci a hodnota relativní redukce chyby WERR (Word Error Rate Reduction), ta značí relativní snížení chyby systému Baseline nejlepšími výsledky této práce.

Tabulka 3.27: Porovnání chybovosti současného systému s výsledky této práce

Jazyk	**Baseline** [%]	**G2P** [%]	**NMT** [%]	**WERR** [%]
Angličtina	37,96	34,6	29,6	22,0
Čeština	2,91	1,9	1,7	41,5
Švédština	18,96	16,0	12,6	33,5

3.4.3 Doporučení parametrů pro experimenty s dalšími jazyky

Pro experimenty s dalšími jazyky je možné z výsledků provedených experimentů doporučit použití kolem 3 až 4 vrstev, v každé z nich okolo 128 neuronů a pro učení použít metodu Adam. Toto doporučení platí pro oba použité toolkity.

U NTM toolkitu se vyplatí testovat Beam search do hodnoty 10, attention mechanismus Scaled Luong nebo Normed Bahdanau a použití dropout s pravděpodobností do 0,5. Všechny nejlepší výsledky používaly alespoň jednu bidirekční vrstvu (bidirekční architektura obsahovala 2, GNMT architektura jednu).

Počet iterací i learning rate decay je vhodné zvolit podle průběhu chybové funkce v grafu (loss), který je možné zobrazit pomocí Tensorboard.

4 Odhalení chyb ve slovníku

V této části se pomocí cross-validace hledaly chyby ve slovní zásobě pro český jazyk, který byl poloautomaticky vytvořen a ručně opravován.

Nejdříve bylo potřeba rozdělit slovní zásobu na více částí, vždy na trénovací a testovací část, celkem bylo vytvořeno 62 trénovacích částí a stejný počet testovacích. Po rozdělení se, pomocí skriptu napsaném v Pythonu, postupně trénovalo 62 modelů na každé trénovací sadě. Poté další skript byl vytvořen pro převedení slov v testovací sadě na fonémy a poslední skript připravil data pro kontrolu.

Výstupní data obsahovala celkem 10 515 slov, která byla natrénovaným modelem špatně převedena do fonetické formy, některá slova mají více možných výslovností a u modelu stačilo, aby se výstup shodoval s jednou z těchto výslovností. U těchto špatně dekódovaných slov je možné, že se jednalo pouze o chybu modelu, novou alternativní výslovnost nebo se jedná o chybnou transkripci ve slovníku, která mohla vzniknout u předchozího modelu používaného pro automatickou transkripci.

Všechna potenciálně chybná slova bylo potřeba nyní ručně kontrolovat a rozlišit, kde se skutečně jedná o chybu a kde nikoliv. Data byla rozdělena do tří sloupců, kde v prvním bylo slovo, v druhém fonémy dekódované modelem a v třetím sloupci byla uvedena výslovnost, kterou obsahuje slovní zásoba.

U některých slov je vidět na první pohled, že jsou špatně dekódovaná modelem, pár příkladů je uvedeno v tabulce 4.1 a několik příkladů správně nalezených chyb je v tabulce 4.2. Zhruba u 10 % špatně dekódovaných slov se jednalo o chybu skutečnou.

Tabulka 4.1: Chybně dekódovaná slova modelem

Slovo	Výslovnost ve slovní zásobě	Výslovnost dekódovaná modelem
anti	anti	aňťi
Bille	bile	bil
Camp	kemp	kamp
Dijkstrův	dijkstrúf	ďijkstrúf
Herz	herc	hers

Tabulka 4.2: Nalezené chyby ve slovní zásobě

Slovo	Výslovnost ve slovní zásobě	Výslovnost dekódovaná modelem
Alzheimerem	alCheimerm	alChajmrem
antijaderné	aňťijaderné	antijaderné
glykoalkaloidy	glikoalkalojidi	glikoalkalojdi
IVP	ivépé	ívépé
kratinký	kratiNkí	kraťiNkí
neodtáhli	neotáhli	neottáhli
nestudilo	nestudilo	nestuďilo
oddělenej	oďelenej	oďelenej
protibiologické	protibijologické	proťibijologické

5 Závěr

Cílem práce bylo ověřit možnost použití rekurentních neuronových sítí pro automatickou fonetickou transkripci a dosažené výsledky porovnat se současně používaným systémem Baseline. K tomu byly použity Sequence-to-Sequence G2P a Neural Machine Translation toolkity. Oba tyto toolkity se podařilo zprovoznit a udělat na nich řadu experimentů. Díky tomu se podařilo vyzkoušet state-of-the-art architektury rekurentních neuronových sítí z posledních dvou let.

Celkem bylo provedeno přes 300 experimentů, nejdelší z nich trval cca 20 hodin (jeden z posledních experimentů pro češtinu) a musel se trénovat třikrát, aby se zjistil rozptyl výsledného WER. Trénovalo se na počítači s operačním systémem Ubuntu, dále obsahuje procesor Intel Core i5-6500 a grafickou kartu NVIDIA GeForce GTX 1070.

Postupně se podařilo zlepšovat úspěšnost těchto systémů a nalezly se optimální hyperparametry a architektury modelu pro každý jazyk. Trénování proběhlo se třemi jazyky (čeština, angličtina, švédština), kde každý z nich zastupoval různou obtížnost v úloze automatické fonetické transkripce. Různá složitost těchto jazyků pomohla určit pásmo hodnot, ve kterém by se měly hledat hyperparametry modelu i pro další jazyky.

Nejlepších výsledků dosahovaly modely se 3 až 4 vrstvami a s počtem neuronů okolo 128 v každé z nich. U učení rekurentních neuronových sítí se dosahovalo nejrychlejší konvergence s použitím metody Adam. Dále je vhodné použít Scaled Luong nebo Normed Bahdanau attention mechanismus, Beam search s maximální šířkou 10 a dropout s pravděpodobností do 0,5. Ve všech nejlepších architekturách byla minimálně jedna bidirekční vrstva.

Postupným experimentováním s různými architekturami a hledáním hyperparametrů modelu rekurentní neuronové sítě se podařilo redukovat chybovost systému. Ukázalo se, že využitím NMT toolkitu a metod, které byly dosud používány pouze pro úlohu strojového překladu z jednoho jazyka do jiného, se docílilo menší chybovosti než s toolkitem G2P i systémem Baseline. S použitím a úpravou NMT toolkitu se mohly aplikovat různé druhy attention mechanismu, použít Beam search a také jiné než standardní architektury (bidirekční, GNMT).

Chybovost s NMT toolkitem byla relativně snížena u češtiny o 41,5 %, u angličtiny o 22 % a u švédštiny o 33,5 %. Díky tomuto zlepšení v přesnosti se bude tento systém používat místo Baseline a poskytne tak lepší kvalitu pro doplňování slovní zásoby.

Nejlepší model pro češtinu byl použit pro hledání potenciálních chyb ve slovní zásobě pro český jazyk, která se používala v této práci. Pomocí tohoto modelu se nalezlo 10 515 slov, ta byla buď špatně dekódována modelem, mohlo se jednat o novou alternativní výslovnost, která nebyla doposud uvedena ve slovní zásobě, nebo šlo o skutečnou chybu. Tento seznam slov se musel ručně projít a výslovnosti bylo potřeba doplnit nebo opravit. Zhruba u 10 % z nich se jednalo o chybu skutečnou a opravou se zvýšila kvalita slovní zásoby pro češtinu.

Zvýšení kvality systému pro automatickou fonetickou transkripci a slovní zásoby je důležité pro řadu dalších aplikací v oblasti zpracování řeči. Například to pomůže zlepšit funkčnost systémů pro diktování a u nepřetržitého online monitoringu televizního a rozhlasového vysílání.

Pokračování je možné v testování dalších metod a architektur, které přinese výzkum neuronových sítí.

Literatura

[1] NOUZA, Jan, ed. Počítačové zpracování řeči: cíle, problémy, metody a aplikace: sborník článků. Liberec: Technická univerzita v Liberci, 2001. ISBN 80-708-3551-6.

[2] GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. Tokyo: O'Reilly, 2017. ISBN 978-1-4919-6229-9.

[3] Online kurz "Natural Language Processing with Deep Learning", dostupné z: http://web.stanford.edu/class/cs224n/syllabus.html

[4] GitHub - cmusphinx/g2p-seq2seq: G2P with Tensorflow. GitHub [online]. Copyright © 2018 [cit. 14.04.2018]. Dostupné z: https://github.com/cmusphinx/g2p-seq2seq

[5] Luong, Minh-Thang, Eugene BREVDO a Rui ZHAO. Neural Machine Translation (seq2seq) Tutorial [online]. 2017 [cit. 2018-04-14]. Dostupné z: https://github.com/tensorflow/nmt

[6] SRIVASTAVA, Nitish, et al. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014, 15.1: 1929-1958.

[7] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. Neural computation, 1997, 9.8: 1735-1780.

[8] CHUNG, Junyoung, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[9] SCHUSTER, Mike; PALIWAL, Kuldip K. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 1997, 45.11: 2673-2681.

[10] Luong, Minh-Thang; PHAM, Hieu; MANNING, Christopher D. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

[11] Bahdanau, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[12] WISEMAN, Sam; RUSH, Alexander M. Sequence-to-sequence learning as beam-search optimization. arXiv preprint arXiv:1606.02960, 2016.

A Obsah přiloženého CD

- text bakalářské práce
  - bakalarska_prace_2018_Frantisek_Kynych.pdf
  - bakalarska_prace_2018_Frantisek_Kynych/
    * obsahuje zprávu bakalarska_prace_2018_Frantisek_Kynych.tex a všechny použité obrázky
  - kopie_zadani_bakalarska_prace_2018_Frantisek_Kynych.pdf
- Neural Machine Translation toolkit
  - verze NMT, která byla použita v této práci
- návod na instalaci použité verze G2P-seq2seq toolkitu

B Architektura nejlepších modelů

Tabulka B.1: Přehled parametrů nejlepších modelů pro každý jazyk

Jazyk	Angličtina	Čeština	Švédština
Počet vrstev	3	3	4
Počet neuronů ve vrstvě	128	128	128
Learning rate	0,0006	0,0005	0,0006
Počet iterací (trénování)	700 000	300 000	250 000
Optimizer	Adam	Adam	Adam
Batch size	64	512	128
Dropout	0,3	0,2	0,2
Learning rate decay	Luong234	Luong234	Luong234
Attention	Normed Bahdanau	Scaled Luong	Scaled Luong
Architektura	GNMT	GNMT v2	Bidirekční
Beam search	3	5	5
WER [%] (test data)	**29,60**	**1,70**	**12,60**