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

NAPOVÍDAČ TEXTU POMOCÍ NEURONOVÉ SÍTĚ V PROHLÍŽEČI

NEURAL NETWORK FOR AUTOCOMPLETE IN THE BROWSER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JÁN JAKUB KUBÍK**
AUTHOR

VEDOUCÍ PRÁCE — **MARTIN KOLÁŘ, M.Sc.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Kubík Ján Jakub**
Program: Informační technologie
Název: **Napovídač textu pomocí neuronové sítě v prohlížeči**
Neural Network for Autocomplete in the Browser
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se rekurentními neuronovými sítěmi pro generování textu, a problematikou napovídače textu
2. Implementujte a natrénujte rekurentní neuronovou síť v tensorflow
3. Seznamte se s programováním Google Chrome Extension
4. Implementujte evaluaci rekurentní síte v tensorflow.js, a využití modelu pro napovídání během psaní textu
5. Zveřejněte webovou implementaci pro demonstrační účely

Literatura:

- https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
- https://karpathy.github.io/2015/05/21/rnn-effectiveness/
- https://www.tensorflow.org/js/
- http://deeplearning.stanford.edu/tutorial/
- https://cs230.stanford.edu/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

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

Vedoucí práce: **Kolář Martin, M.Sc.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cieľom tejto práce je vytvoriť a natrénovať neurónovú sieť, ktorá sa následne bude používať v internetovom prehliadači pre napovedanie sekvencií anglických slov v priebehu písania textu používateľom. Zámerom je zjednodušenie písania častých slovných obratov. Zvolený problém je vyriešený pomocou rekurentnej neurónovej siete schopnej predpovedať textové sekvencie zo vstupného textu. Natrénovaná neurónová sieť je použitá v rozšírení pre prehliadač Google Chrome. Na základe normalizovaného výstupu neurónovej siete, následného výberu tokenov pomocou samplovacieho dekódovacieho algoritmu a ich spájaním je rozšírenie schopné generovať sekvencie anglických slov, ktoré sú zobrazované používateľovi ako navrhovaný text. Výsledná neurónová sieť je optimalizovaná pomocou výberu vhodného počtu rekurentných vrstiev a neurónov v jednotlivých vrstvách, stratovej funkcie a počtu trénovacích epôch. Prínosom tejto práce je použitie neurónovej siete na predpovedanie sekvencií anglických slov v prehliadači, ktoré zjednodušuje menej zdatným používateľom každodennú prácu s písaním textu na internete.

Abstract

The goal of this thesis is to create and train a neural network and use it in a web browser for English text sequence prediction during writing of text by the user. The intention is to simplify the writing of frequent phrases. The problem is solved by employing a recurrent neural network that is able to predict output text based on the text input. Trained neural network is then used in a Google Chrome extension. By normalized ouput of the neural network, text choosing by sampling decoding algorithm and connecting, the extension is able to generate English word sequences, which are shown to the user as suggested text. The neural network is optimized by selecting the right loss function, and a suitable number of recurrent layers, neurons in the layers, and training epochs. The thesis contributes to enhancing the everyday user experience of writing on the Internet by using a neural network for English word sequence autocomplete in the browser.

Kľúčové slová

neurónová sieť, jazykový model, rekurentná neurónová sieť, generovanie prirodzeného jazyka, rozšírenie pre prehliadač Google Chrome, automatické dopĺňanie textu

Keywords

neural network, language model, recurrent neural network, natural language generation, Google Chrome extension, text autocomplete

Citácia

KUBÍK, Ján Jakub. *Napovídač textu pomocí neuronové sítě v prohlížeči.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Kolář, M.Sc.

Napovídač textu pomocí neuronové sítě v prohlížeči

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Martina Kolaŕa, M.Sc. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Ján Jakub Kubík
30. júla 2020

Poďakovanie

Rád by som sa poďakoval môjmu vedúcemu bakalárskej práce pánovi Martinovi Kolářovi M.Sc. za odborné vedenie, trpezlivosť a ochotu. Taktiež by som sa veľmi rád poďakoval svojej rodine, ktorá ma podporovala v priebehu tvorby a písania tejto práce.

Obsah

Kapitola 1

Úvod

Spracovanie prirodzeného jazyka sa zaoberá pochopením, interpretovaním a generovaním prirodzeného jazyka. Jeho hlavnou úlohou je zjednodušiť komunikáciu medzi ľuďmi a strojmi. Už v 50. rokoch 20. storočia vytvoril Alan Turing test založený na spracovaní jazyka [31]. Tento test slúži na preukázanie inteligentného správania sa systému. Pôvodne bolo spracovanie prirodzeného jazyka založené na pravidlách a neskôr na štatistických metódach. Spolu s vytvorením umelého neurónu a s rozvojom umelých neurónových sietí sa stalo spracovanie prirodzeného jazyka jedným z cieľov umelej inteligencie s využitím neurónových sietí. V súčasnosti sa stretávame so spracovaním prirodzeného jazyka pomocou neurónových sietí v podobe prekladačov, chatbotov, napovedačov textu, syntetizátorov reči na text a syntetizátorov textu na reč.

Cieľom bakalárskej práce je vytvoriť rekurentnú neurónovú sieť schopnú predpovedať výsledný text na základe vstupného textu. Následne je potrebné použiť neurónovú sieť v rozšírení pre prehliadač Google Chrome za účelom navrhovania sekvencií anglických slov používateľovi.

Tému práce som si vybral z dôvodu rozšírenosti neurónových sietí a ich obrovskej dôležitosti a integrovanosti v každodennom ľudskom živote. Chcel som zistiť, čo sú neurónové siete a ako fungujú.

V kapitole 2 sú analyzované základné koncepty neurónových sietí. Ide o základné koncepty ako umelý neurón, neuronové siete spolu s aktivačnými funkciami, trénovanie neurónových sietí a problémy s ním spojené, rekurentné neurónové siete a podrobný popis GRU neurónu. Kapitola 3 sa zaoberá generovaním prirodzeného jazyka. Najskôr rozoberá základný popis jazykového modelu. Ďalej popisuje n-gramy a neurónové siete pre modelovanie prirodzeného jazyka. Z neurónových sietí rozoberá ich najmodernejšiu architektúru. Na záver sú dekódované algoritmy pre generovanie sekvencie a úprava textu pre spracovávanie pomocou neurónovej siete. V kapitole 4 je technológia rozšírenia prehliadača so zameraním na rozšírenie pre prehliadač Google Chrome. V tejto kapitole sú jednotlivé komponenty Google Chrome rozšírenia, ich komunikácia pomocou zasielania asynchrónnych správ a spôsoby pre ukladanie dát v Google Chrome rozšírení. Hlavnou náplňou kapitoly 5 je návrh riešenia - od rekapitulácie zadania cez Gmail Smart Compose a stručný popis môjho riešenia sa prechádza k podrobnej architektúre, spôsobu generovania sekvencie a úprave hyperparametrov neurónovej siete. V kapitole 6 je implementácia riešenia. Najskôr sú predstavené použité technológie, dôležité implementačné časti, finálne trénovanie a výber dekódovacieho algoritmu pre generovanie sekvencie. V závere kapitoly sú uvedené problémy pri implementácii a jej publikácii.

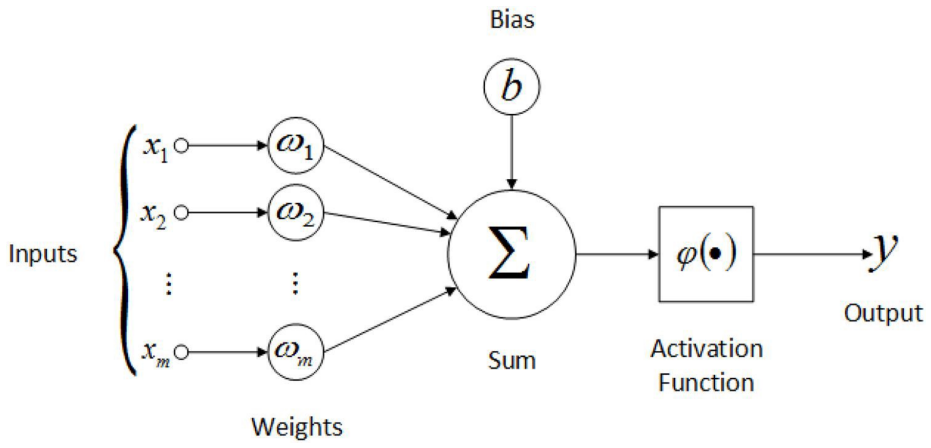
Kapitola 2

Základné koncepty neurónových sietí

Umelé neurónové siete sú výpočtové modely, ktoré sú inšpirované nervovým systémom živých bytostí. Skladajú sa z umelých neurónov, ktoré sú spolu s aktivačnou funkciou analyzované v sekcii 2.1. Sekcia 2.2 popisuje vrstvy neurónových sietí, trénovanie, rozdelenie a ich hyperparametre. V ďalšej časti 2.3 sú predstavené problémy neurónových sietí. V poslednej sekcii 2.4 sú opísané jednoduché rekurentné neurónové siete spolu s ich trénovaním a podrobný popis GRU neurónu.

2.1 Umelý neurón

Základnou výpočtovou jednotkou neurónových sietí je umelý neurón. Umelý neurón je zjednodušeným matematickým modelom biologického neurónu. Zjednodušeným preto, že ani v dnešnej dobe sa presne nevie, ako biologické neuróny fungujú. Následujúci popis je inšpirovaný článkom [33]. Na obrázku 2.1 je matematický model umelého neurónu.



Obr. 2.1: Umelý neurón, prevzaté z [2].

Vstupom neurónu je vektor x o m prvkoch s príslušnými váhami w. Ich hodnoty sú reálnymi číslami. Taktiež bias b je reprezentovaný reálnym číslom. Výstupom neurónu je reálne číslo y.

$$y = \varphi(P) \tag{2.1}$$

Rovnica 2.1 hovorí, že výstup neurónu y sa vyráta aplikáciou aktivačnej funkcie φ na vnútorný potenciál P neurónu.

$$P = \sum_{i=1}^{m} w_i * x_i + b \tag{2.2}$$

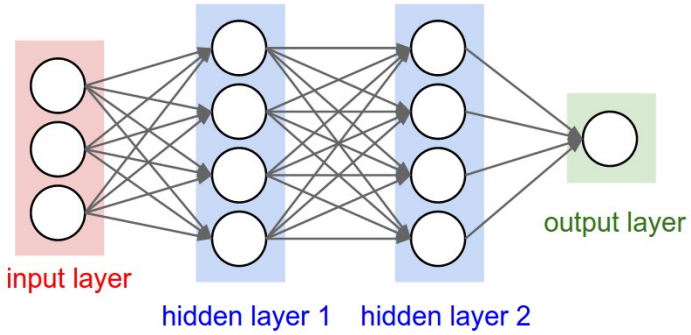
Rovnica 2.2 vyjadruje vnútorný potenciál neurónu. Znamená, že vnútorný potenciál P je súčtom všetkých vstupov x prenásobených príslušnými váhami w a k tomuto sa priráta bias b. Bias zvykne byť modelovaný ako jeden zo vstupov, kde $x_0 = 1$ a $w_0 = b$. V tomto prípade je vnútorný potenciál P vyjadrený rovnicou 2.3.

$$P = \sum_{i=0}^{m} w_i * x_i \tag{2.3}$$

2.2 Umelé neurónové siete

Umelé neurónové siete slúžia na transformovanie vstupných dát na požadované výstupné dáta alebo na klasifikáciu vstupných dát do rôznych tried. Táto transformácia alebo klasifikácia sa prispôsobuje riešeným úlohám pomocou zmeny topológie neurónových sietí a ich trénovaním. Trénovanie neurónových sietí je analyzované v podsekcii 2.2.

Umelé neurónové siete sú zložené z umelých neurónov, usporiadaných do vrstiev. Celkovo sú 3 druhy vrstiev, pri čom každá vrstva má svoj špecifický účel. Následujúci popis je podľa článku [34]. Jednotlivé vrstvy sú zobrazené na obrázku 2.2.



Obr. 2.2: Vrstvy neurónovej siete, prevzaté z [34].

Všetky neuróny jednotlivých vrstiev prijímajú na svojich vstupoch reálne čísla a výstupom je taktiež vždy reálne číslo.

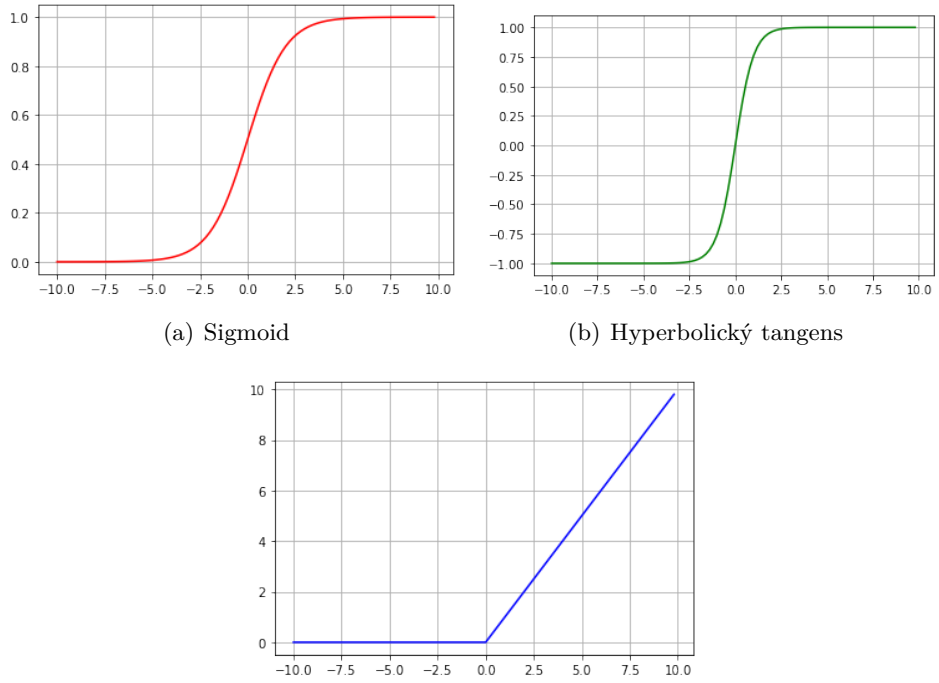
Vstupná vrstva zodpovedá za získavanie informácií, signálov, vlastností alebo nameraných hodnôt z okolitého prostredia. Vstupná vrstva je vždy jedna. Neuróny vstupnej vrstvy nemajú váhy ani aktivačnú funkciu. Ich úlohou je posielanie predpripravených dát ďalším vrstvám.

Ďalšia vrstva v poradí je skrytá. Skrytá vrstva musí byť vždy minimálne jedna, ale môže ich byť aj viac. Záleží to na architektúre neurónovej siete. Skrytá vrstva je zložená z neurónov, ktoré majú vstupy s pridelenými váhami. Každý neurón skrytej vrstvy používa aktivačnú funkciu. Skrytá vrstva je z pohľadu vrstiev najdôležitejšia pre celú neurónovú sieť.

Výstupná vrstva je vždy jedna. Rovnako ako u skrytej vrstvy majú jej neuróny pridelené váhy a používajú aktivačnú funkciu. Aktivačná funkcia výstupnej vrstvy môže byť úplne iná ako aktivačná funkcia skrytej vrstvy.

Aktivačné funkcie

Aktivačné funkcie sú veľmi dôležité pre neurónové siete na naučenie sa nelineárnych závislostí medzi vstupom a výstupom neurónovej siete. Medzi najznámejšie aktivačné funkcie patrí sigmoid, hyperbolický tangens a ReLU. Sigmoid je na obrázku 2.3(a), má rovnicu $\varphi(x) = \frac{1}{1+e^{-x}}$. Hyperbolický tangens na obrázku 2.3(b), má rovnicu $\varphi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. ReLU je na obrázku 2.3(c) a má rovnicu $\varphi(x) = \max(0, x)$.



(a) Sigmoid

(b) Hyperbolický tangens

(c) ReLU

Obr. 2.3: Aktivačné funkcie.

Rozdelenie neurónových sietí

Neurónové siete sa delia podľa rôznych kritérií. Jedným z kritérií môže byť počet skrytých vrstiev. Neurónová sieť s jednou skrytou vrstvou sa nazýva plytká neurónová sieť. Neurónová sieť s viacerými skrytými vrstvami sa nazýva hlboká neurónová sieť.

Ďalším kritériom pre delenie neurónových sietí je spôsob, akým sa v nich šíria dáta. Neurónová sieť s dopredným šírením dát sa nazýva konvolučná neurónová sieť. Tento typ neurónovej siete sa používa na úlohy zamerané na klasifikáciu. Neurónové siete so spätným šírením dát sa nazývajú rekurentné neurónové siete. Tento typ neurónových sietí sa používa na transformáciu vstupných dát na výstupné dáta vzhľadom na predchádzajúce vstupné dáta.

Ďalej sa neuronové siete delia podľa typu trénovania. Trénovanie bez učiteľa, trénovanie s učiteľom a trénovanie posilňovaním.

V práci je použitá rekurentná neurónová sieť (sekcia 2.4), ktorá používa trénovanie s učiteľom. Preto je ďalej definované len trénovanie s učiteľom.

Trénovanie neurónových sietí

Trénovanie alebo učenie neurónových sietí je vo svojej podstate minimalizácia celkovej chyby neurónovej siete pomocou zmeny váh a biasov jednotlivých neurónov vo vrstvách neurónovej siete. Pred trénovaním sa incializujú váhy a biasy jednotlivých neurónov. Spôsoby ich inicializácie sú stále predmetom výskumov. Najčastejšie sa inicializujú náhodnými hodnotami. Prvým krokom v trénovaní je nastavenie neurónov vstupnej vrstvy prvkom z trénovacej sady dát. Nasleduje dopredný prechod všetkými vrstvami, kde sa výstupy neurónov v jednotlivých vrstvách rátajú pomocou rovnice 2.1. Na výstupnej vrstve neurónovej siete je výsledný vektor. Ďalším krokom je výpočet celkovej chyby za použitia stratovej funkcie na výsledný vektor a požadovaný vektor z trénovacej sady dát. Pre rôzne typy úloh boli vytvorené rôzne stratové funkcie. Ako príklad je uvedená rovnica jednoduchej štvorcovej stratovej funkcie 2.4 pre výpočet chyby neurónovej siete.

$$E(k) = \sum_{n=1}^{m} [t_i(k) - o_i(k)]^2 \tag{2.4}$$

Chyba sa určuje ako celková chyba u všetkých výstupných neurónov pomocou stratovej funkcie. Chyba E predstavuje chybu medzi k-tym výsledným vektorom t a k-tym požadovaným vektorom o. N je index neurónu výstupnej vrstvy, ktorá je zložená z m neurónov.

Minimalizácia celkovej chyby sa zabezpečuje zmenou váh a biasov pomocou zápornej hodnoty gradientu stratovej funkcie. Pre jednoduchosť sa pre rovnice 2.5, 2.6 a 2.7 predpokladá, že bias je súčasťou váh ako v rovnici 2.3. V rovnici 2.5 predstavuje ∇E gradient, ktorý je vektorom parciálnych derivácii celkovej chyby E vzhľadom na jednotlivé váhy w neurónovej siete, pri čom má neurónová sieť n váh.

$$\nabla E = [\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, ..., \frac{\partial E}{\partial w_7}, \frac{\partial E}{\partial w_n}] \tag{2.5}$$

Na výpočet gradientu potrebného na úpravu váh sa používa algoritmus spätného šírenia chyby (anglicky backpropagation). Tento algoritmus využíva reťazové pravidlo derivácie (anglicky chain rule).

$$\frac{\partial E}{\partial w_j} = \frac{\partial E}{\partial z_j} \frac{\partial z_j}{\partial w_j} \tag{2.6}$$

Rovnica reťazového pravidla derivácií 2.6 hovorí, že parciálna derivácia chyby E vzhľadom na váhu w_j sa vyráta ako súčin parciálnej derivácie chyby E vzhľadom na aktivačnú funkciu z_j a parciálnej derivácie aktivačnej funkcie z_j vzhľadom na váhu w_j.

Posledným krokom je upravenie váh. Jednotlivé váhy sú upravované podľa vzorca 2.7.

$$w_{j_{new}} = w_j - \alpha \frac{\partial E}{\partial w_j} \tag{2.7}$$

Tento vzorec hovorí, že od aktuálnej hodnoty váhy wj je odrátaná zmena veľkosti chyby E vzhľadom na váhu w_j, ktorá je prenásobená konštantou α. α je učiaci koeficient. Správne určenie tohto koeficientu je veľmi dôležité. Pretože veľmi malé hodnoty by spôsobovali malé

kroky pri optimalizácii stratovej funkcie a veľmi dlho by sa neurónová sieť učila. Veľké kroky by boli tiež nevhodné, pretože by nikdy nedoviedli k minimalizácii celkovej chyby (zdroj [23] časť 1).

Jednotlivé trénovacie kroky sa opakujú pre každý prvok trénovacej sady dát. Opakujú sa pre pevne daný počet epôch. Epocha je jedna iterácia cez všetky prvky trénovacej sady dát. Aktualizácia váh a biasov pre každý jeden prvok trénovacej sady dát je výpočtovo veľmi náročné. Preto boli vyvinuté optimalizačné metódy.

Základným optimalizačným algoritmom je dávkový zostup gradientu (anglicky batch gradient descent). Princíp tohto algoritmu je založený na aktualizácii váh až po spracovaní všetkých prvkov dátovej sady. To znamená, že pre výpočet nárastu každej váhy je potrebné sčítať chyby všetkých prvkov dátovej sady. Toto je časovo aj priestorovo veľmi náročné. Z toho dôvodu vznikol algoritmus minidávkový zostup gradientu (anglicky minibatch gradient descent), ktorý si pri každom opakovaní vyberie iba časť dátovej sady pevnej veľkosti (zdroj [20] kapitola 3).

Pretrénovanie a zmiernenie jeho dopadu

Pretrénovanie patrí k jedným z najčastejších problémov neurónových sietí. Neurónová sieť je pretrénovaná vtedy, keď pre dáta z trénovacej sady dosahuje veľmi dobré výsledky a pre dáta mimo trénovacej sady dát dosahuje veľmi slabé výsledky. Znamená to, že neurónová sieť nevie dobre generalizovať. Riešením tohto problému sú L2 regularizácia a dropout regularizácia.

V práci je použitá dropout regularizácia pomocou skrytých dropout vrstiev neurónovej siete. Dropout regularizácia je podrobne popísaná v článku [27]. Dropout slúži pri trénovaní k náhodnému odstaveniu neurónov v jednotlivých vrstvách podľa nastavenej pravdepodobnosti p. Týmto spôsobom sa zamedzuje prílišnému prispôsobeniu neurónovej siete trénovacím dátam.

Hyperparametre a ich úprava

Hyperparametre sú rôzne premenné alebo funkcie použité pri tvorbe neurónovej siete a počas jej trénovania. Hyperparamentre ovplyvňujú úspešnosť neurónovej siete. Medzi najznámejšie hyperparametre patria počet skrytých vrstiev, stratová funkcia, koeficient učenia, počet neurónov v jednotlivých vrstvách a počet trénovacích epôch.

Existuje mnoho spôsobov optimalizácie hyperparametrov. Medzi najčastejšie používané patria manuálny prístup, grid search, random search a bayesova optimalizácia. V poslednej dekáde je optimalizácia hyperparametrov súčasťou rozsiahlych výskumov. Viac informácií sa nachádza v knihe [9], ktorá je zdrojom tejto podsekcie.

2.3 Problémy neurónových sietí spojené s gradientom

Gradient má dôležitú úlohu pri trénovaní dopredných a rekurentých neurónových sietí. S gradientom je spojený problém explodujúceho gradientu (anglicky exploding gradient problem) a problém miznúceho gradientu (anglicky vanishing gradient problem). Problém miznúceho gradientu sa rieši omnoho zložitejšie ako problém explodujúceho gradientu. Tieto problémy sa môžu objaviť pri hlbokých dopredných neurónových sieťach a pri rekurentných neurónových sieťach, ktoré majú veľký počet krokov alebo hĺbku. Zdrojom tejto sekcie je [16] kapitola 1.

Explodujúci gradient

Explodujúci gradient môže byť spôsobený, ak sú parciálne derivácie celkovej chyby vzhľadom k jednotlivým váham neurónov väčšie ako 1. Toto spôsobuje veľké zmeny váh jednotlivých neurónov a vedie to k nestabilite neurónovej siete. V extrémnom prípade to môže spôsobiť pretečenie dátového typu váh a ich nastavenie na NaN. Všeobecným riešením je orezávanie gradientu (anglicky gradient clipping). Pre orezávanie gradientu je nastavená hranica, ktorú keď gradient prekročí, je orezaný. Toto obmedzí maximálnu zmenu váh neurónov pri trénovaní.

Miznúci gradient

Miznúci gradient nastáva, ak sú parciálne derivácie celkovej chyby vzhľadom k jednotlivým váham neurónov menšie ako 1. Ich násobenie v algoritme spätnej propagácie vedie k veľmi malým hodnotám gradientov, ktoré sú používané na aktualizáciu váh. To znamená, že neurónová sieť sa nevie naučiť správne transformovať vstup na výstup podľa trénovacej sady dát. V extrémnom prípade môže viesť miznúci gradient až k podtečeniu a úplnému zastaveniu trénovania.

2.4 Rekurentné neurónové siete

Vďaka rekurentným neurónovým sieťam boli dosiahnuté veľké pokroky v oblasti prekladania jazyka, syntézy reči, generovania textu a mnoho ďalšieho. Rekurentné neurónové siete boli vytvorené pre potrebu zachytávania závislostí vo vstupných dátach, ako je napríklad súvislý text alebo zvuková stopa. Najskôr boli vytvorené jednoduché rekurentné neurónové siete, z ktorých sa vďaka ich nedostatkom vyvinuli rekurentné neurónové siete s komplexnejšiou štruktúrou neurónu ako je GRU a LSTM.

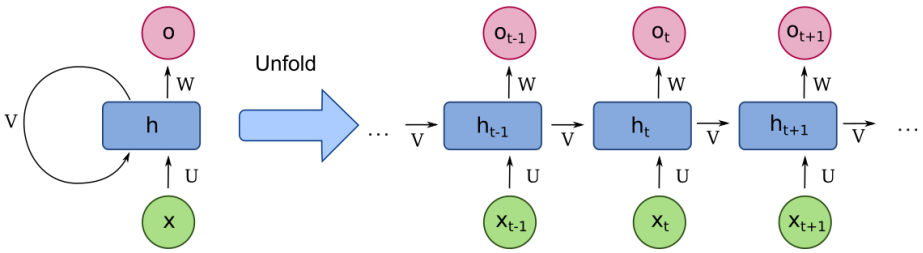
Trénovanie rekurentných neurónových sietí

Na trénovanie rekurentných neurónových sietí sa používa algoritmus spätnej propagácie v čase (anglicky backpropagation through time). Algoritmus je založený na algoritme spätnej propagácie, ktorý je popísaný v sekcii 2.2. Rozdiel je v tom, že zmeny váh a biasov pre minimalizáciu celkovej chyby neurónovej siete sa musia prispôsobovať aj vzhľadom ku všetkým predchádzajúcim vstupom. Toto je výpočtovo aj pamäťovo veľmi náročné a mohlo by to spôsobovať miznúci alebo explodujúci gradient (sekcia 2.3). Preto sa používa skrátená spätná propagácia v čase (anglicky truncated backpropagation through time). Skrátená spätná propagácia v čase je modifikovaná verzia algoritmu spätnej propagácie v čase, kde sa pri trénovaní vypočítavajú gradienty pre úpravu váh a biasov len pre fixný počet prvkov do minulosti a nie pre celú sekvenciu (zdroj [16] kapitola 2).

Jednoduché rekurentné neurónové siete

Jednoduché rekurentné neurónové siete sú podobné klasickým dopredným neurónovým sieťam, s tým rozdielom, že ich neuróny majú pamäťový prvok, ktorý sa nazýva skrytý stav. Skrytý stav nesie informáciu o predchádzajúcich vstupoch neurónovej siete. To znamená, že aktuálny výstup neurónovej siete závisí na všetkých predchádzajúcich vstupoch (reprezentovaných skrytým stavom) a na aktuálnom vstupe. Skrytý stav je v jednoduchej rekurentnej neurónovej sieti reprezentovaný funkciou hyperbolický tangens (obrázok 2.3(b)).

Výsledok jednoduchej rekurentnej neurónovej siete sa vyráta pomocou skrytého stavu a vstupného prvku, ktorý svojím dopredným priechodom cez neurónovú sieť pozmení skrytý stav. Následne je tento skrytý stav použitý pre ďalší vstup. Celý tento proces sa opakuje pre všetky vstupy rekurentnej neurónovej siete (zdroj [16] kapitola 1).



Obr. 2.4: Rekurentný neurón v čase t, prevzaté z [29].

Vľavo na obrázku 2.4 je neurón jednoduchej rekurentnej neurónovej siete so skrytým stavom v, ktorý nesie informáciu o predchádzajúcich vstupoch bunky. Vpravo na obrázku 2.4 je ten istý rekurentný neurón rozložený v čase 1 až $t+1$ za účelom zobrazenia vplyvu predchádzajúcich vstupov x_1 až x_t reprezentovaných skrytým stavom v a aktuálneho vstupu x_{t+1} na aktuálny výstup o_{t+1}.

GRU neurón

Z dôvodu miznúceho gradientu (sekcia 2.3), ktorý obmedzuje schopnosť zapamätania si dlhších závislostí v sekvenciach dát u jednoduchých rekurentných neurónových sietí boli vytvorené rekurentné neuronové siete so zložitejšou stavbou neurónu.

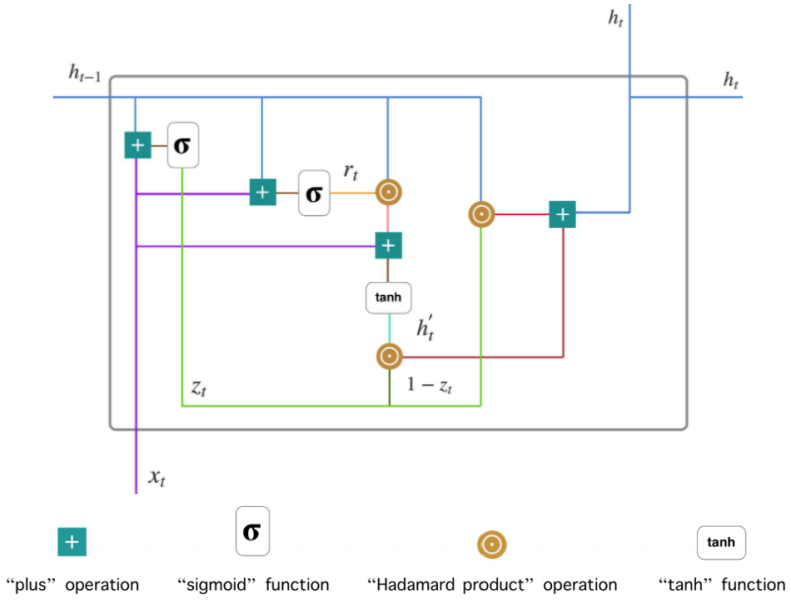
V článku [12] z roku 1997 je opísaný model neurónu dlhá krátka pamäť (anglicky long short-term memory, ďalej len LSTM). LSTM má na určovanie skrytého stavu vnútorný mechanizmus brán. Brány regulujú tok informácií a vedia rozlíšiť, ktoré dáta sú dôležité a je potrebné ich zachovať a ktoré dáta sa môžu zahodiť. LSTM má vstupnú, výstupnú a zabúdajúcu bránu.

V článku [7] z roku 2014 je predstavená alternatíva k LSTM s menším počtom brán. Nazýva sa bránová rekurentná jednotka (anglicky gated recurrent unit, ďalej len GRU). GRU má len aktualizačnú a resetovaciu bránu. Vďaka menšiemu počtu brán (menej výpočtov) a porovnateľným výsledkom s LSTM je v tejto práci použitý GRU neurón. Preto je ďalej popísaný matematický model GRU neurónu.

Na obrázku 2.5 je model GRU neurónu. Popis GRU neurónu je inšpirovaný [16] kapitolou 3. GRU neurón sa skladá zo vstupu x_t, skrytého stavu h_{t-1}, výstupu h_t, ktorý je totožný so skrytým stavom použitým v ďalšej iterácii, aktualizačnej brány z_t a resetovacej brány r_t. Skrytý stav sa ráta pomocou vstupu, predchádzajúceho skrytého stavu a brán.

Rovnica 2.8 je matematickým vyjadrením **aktualizačnej brány** z_t. Jej úlohou je určiť, aké množstvo predchádzajúcich informácií bude propagované do budúcna. Aktualizačná brána z_t sa vyráta prenásobením vstupu x_t príslušnou váhou $W^{(z)}$. K tomu sa priráta predchádzajúci skrytý stav h_{t-1} prenásobený jeho váhou U^z. A na toto sa aplikuje sigmoid funkcia σ, ktorá je na obrázku 2.3(a).

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{2.8}$$



Obr. 2.5: Model GRU neurónu, prevzaté z [16] kapitola 3.

Rovnica 2.9 je matematickým vyjadrením **resetovacej brány** r_t, ktorá zabezpečuje zabúdanie časti nepodstatných informácií. Resetovacia brána r_t sa vyráta rovnako ako aktualizačná brána z_t. Rozdiel je len vo váhach W a U.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{2.9}$$

Rovnica 2.10 predstavuje **dočasný vnútorný stav** h'_t, ktorý používa resetovaciu bránu na ukladanie relevantných informácii z minulosti. Ráta sa ako súčet násobku vstup x_t s jeho váhou W a hadamardov produkt [1] na resetovaciu bránu r_t a násobok predchádzajúceho skrytého stavu h_{t-1} s jeho váhou U. Na tento celok sa aplikuje hyperbolický tangens tanh, ktorý je popísaný na obrázku 2.3(b).

$$h'_t = \tanh(Wx_t + r_t \circ Uh_{t-1}) \tag{2.10}$$

Posledná rovnica 2.11 počíta **skrytý stav** h_t ako súčet dvoch hadamardových produktov. V provom produkte je použitá aktualizačná brána z_t a predchádzajúci skrytý stav h_{t-1}. V druhom hadamardovom produkte je použitý dočasný vnútorný stav h'_t s aktualizačnou bránou z_t, ktorej hodnota je odrátaná od 1.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h'_t \tag{2.11}$$

[1] https://medium.com/linear-algebra/part-14-dot-and-hadamard-product-b7e0723b9133

Kapitola 3

Generovanie prirodzeného jazyka

Generovanie prirodzeného jazyka (anglicky natural language generation, ďalej len NLG) je podmnožinou spracovania prirodzeného jazyka (anglicky natural language processing, ďalej len NLP). NLG sa zaoberá vytváraním zrozumiteľných textov v rôznych jazykoch. Táto práca sa zaoberá anglickým jazykom. Pre NLG je zásadne vytvoriť jazykový model, ktorý predpovedá následujúci token z predchádzajúcich. Následne je jazykový model pomocou dekódovacích algoritmov použitý na generovanie textových sekvencií zložených z tokenov. Jazykové modely prešli od modelov založených na pravidlách cez pravdepodobnostné modely ako sú napr. n-gramy až po jazykové modely založené na neurónových sieťach. Jazykový model spolu s n-gramami sú analyzované v sekcii 3.1. V ďalšej sekcii 3.2 je vývoj jazykových modelov od RNN cez Sequence-to-sequence architektúru až po Transformera. V sekcii 3.3 sú dekódovacie algoritmy používané pre generovanie prirodzeného jazyka. Posledná sekcia 3.4 sa zaoberá úpravou textu pre jeho spracovávanie pomocou neurónových sietí.

3.1 Jazykový model

Cieľom jazykového modelu je vypočítať pravdepodobnosť sekvencie tokenov alebo pravdepodobnosť následujúceho tokenu v závislosti od všetkých predchádzajúcich tokenov. Rovnica 3.1 vyjadruje pravdepodobnosť sekvencie W, ktorá je zložená z tokenov w_1 až w_n.

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 ... w_n) \tag{3.1}$$

Rovnica 3.2 predstavuje výpočet pravdepodobnosti sekvencie. Postupne sa rátajú pomocou reťazového pravidla podmienené pravdepodobnosti jednotlivých tokenov w_1 až w_n a následne sú tieto pravdepodobnosti prenásobené za účelom získania pravdepodobnosti výslednej sekvencie.

$$P(w_1, w_2, w_3, w_4, w_5 ... w_n) = \prod_{i=1}^{n} P(w_i | w_1 ... w_{i-1}) \tag{3.2}$$

Tento spôsob je veľmi nevýhodný z dôvodu potreby zahŕňania do výpočtu pravdepodobnosti výslednej sekvencie pravdepodobnosti všetkých predchádzajúcich tokenov. Preto vznikli n-gramy.

N-gramy

N-gramy berú v úvahu len určitý počet tokenov z minulosti, nie celú sekvenciu tokenov. Rovnica 3.3 vyjadruje, že pravdepodobnosť celej sekvencie zloženej z tokenov w_1 až w_n sa

neráta zo všetkých predchádzajúcich tokenov, ale len z predchádzajúcich $k-1$ tokenov.

$$P(w_1, w_2, w_3, w_4, w_5 ... w_n) = \prod_{i=1}^{n} P(w_i | w_{i-k} ... w_{i-1}) \tag{3.3}$$

Ak $k = 1$, tak sa hovorí o unigrame, kde sa pre každý token v sekvencii vyráta pravdepodobnosť a násobením sa skladá. Unigram je preto veľmi nepresný. V bigrame ($k = 2$) sa berie do úvahy prechádzajúci token a v trigram ($k = 3$) sa berú do úvahy 2 predchádzajúce tokeny. Čím viac tokenov z minulosti sa použije, tým je jazykový model presnejší, ale taktiež je výpočtovo náročnejší z hľadiska používania pamäte.

N-gramy nie sú schopné vytvárať kombinácie tokenov, ktoré sa nenachádzajú v trénovacej sade. Existujú techniky na zmiernenie tohoto problému, ale nie sú dostačujúce. Ďalším problémom je, že kontext generovanej sekvencie je určovaný len pomocou pár predchádzajúcich tokenov (zdroj [10] kapitola 9).. Riešením týchto problémov sú jazykové modely založené na neurónových sieťach (sekcia 3.2).

3.2 Vývoj jazykových modelov a ich architektúr

Jazykové modely prešli postupným vývojom. Zo začiatku sa používali jazykové modely založené na pravidlách. Od nich sa prešlo k štatistickým modelom ako napr. n-gramy (sekcia 3.1). Aktuálne sú najpokročilejšou metódou pre tvorbu jazykových modelov neurónové siete, ktoré prešli taktiež vývojom.

Riešením problému generovania sekvencií, ktoré sa nenachádzajú v trénovacej sade dát sú dopredné neurónové siete.

Rekurentné neurónové siete

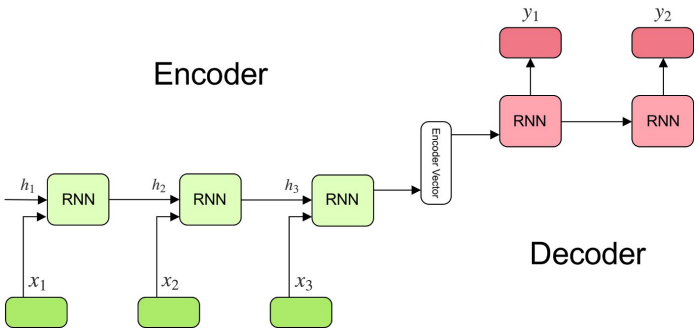
Od dopredných neurónových sietí sa postupne prešlo k rekurentným neurónovým sieťam s komplexnejšou štruktúrou neurónu GRU alebo LSTM. RNN riešia problém nedostatočného kontextu sekvencie, ktorý je v n-gramoch. Rekurentné neurónové siete sú schopné rozpoznávať závislosti v dlhých sekvenciách a stali sa na určitú dobu najpokročilejšou a najlepšou metódou vo všetkých oblastiach NLG (tiež v NLG). Veľmi dobrým článkom o použití rekurentných neurónových sietí na generovanie textu je článok „The Unreasonable Effectiveness of Recurrent Neural Networks“ od Andreja Karpathyho [15]. Viac informácií o rekurentných neurónových sieťach je v sekcii 2.4.

Najzásadnejšie architektúry neurónových sietí pre spracovávanie prirodzeného jazyka sú v súčasnosti (jún 2020) Sequence-to-sequence (ďalej len seq2seq) a Transformer.

Sequence-to-sequence

Seq2seq architektúra bola predstavená v roku 2014 v článku [28]. Na obrázku 3.1 je jednovrstvová seq2seq architektúra. Skladá sa z enkódera a dekódera, ktoré sú zložené z rekurentných neurónov (LSTM alebo GRU). Pomocou rekurentných neurónov a ich skrytých stavov určuje seq2seq kontext vstupnej sekvencie. Úlohou enkódera je načítanie sekvencie zloženej z tokenov $x_1, x_2, ..., x_n$ a jej reprezentácia pomocou vektora Encoded Vector, ktorý slúži ako vstup pre dekóder. Dekóder slúži na vygenerovanie sekvencie zloženej z tokenov $y_1, y_2, ..., y_n$.

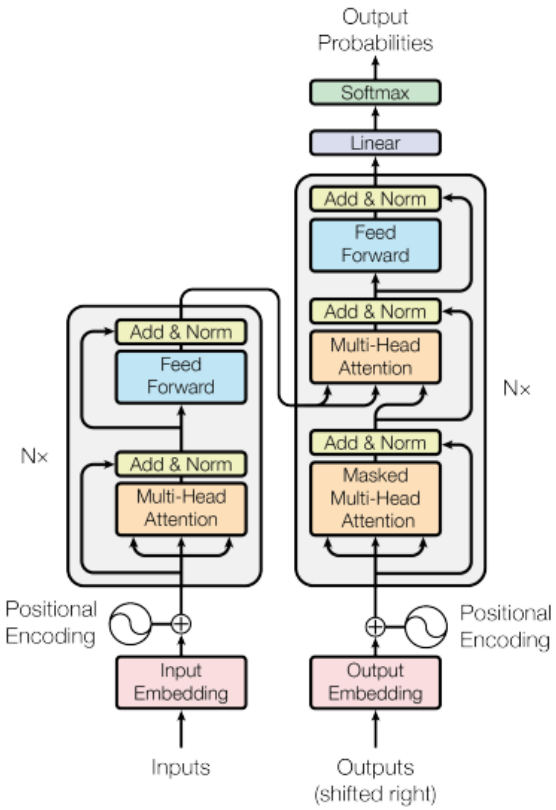
Seq2seq architektúra nahradila v roku 2016 štatistický prístup pre Google Translator (článok [35]).



Obr. 3.1: Architektúra Sequence-to-sequence, prevzaté z [17].

Transformer

Transormer je alternatívou k rekurentným sieťam a ich architektúram. Bol predstavený v roku 2017 článkom [32]. Skladá sa z endódera, dekódera a je založený na attention mechanizme. Attention mechanizmus sa pozerá na vstupnú sekvenciu a v každom kroku sa rozhoduje, ktorá časť sekvencie je dôležitá a tak určuje kontext vstupnej sekvencie.



Obr. 3.2: Architektúra Transformeru, prevzaté z [32].

Z architektúry Transformera, ktorá je na obrázku 3.2 je zrejmá dôležitosť attention mechanizmu pre túto architektúru. Naľavo na obrázku je enkóder a napravo dekóder. Pre

oba je dôležité pozičné zakódovanie vstupných slov, keďže Tranformer nepoužíva rekurentné neuróny.

V dnešnej dobe sú najlepšie jazykové modely založené na Transoformerovi a jeho rôznych variáciách. Medzi najlepšie jazykové modely v súčasnosti (jún 2020) patria:

- BERT [8]
- GPT-2 [22]
- MegatronLM [26]
- T-NLG [24]

3.3 Obecný popis generovania sekvencie pomocou neurónových sietí

Pre generovanie textovej sekvencie pomocou neurónovej siete je potreba vstupný text, jazykový model (natrénovanú neurónovú sieť), dekódovací algorirmus a ukončovaciu podmienku.

$$sekvencia = t_1 + t_2 + t_3 + ... + t_n \tag{3.4}$$

Rovnica 3.4 popisuje generovanie sekvencie. Pred samotným generovaním sekvencie je potrebné spracovať vstupný text rozdelený na tokeny a nastaviť tak jazykový model (skryté stavy u rekurentných neurónových sietí) pre generovanie sekvencie. Následne sa pomocou dekódovacieho algoritmu vygeneruje token t_1, ktorý je použitý ako vstup jazykového modelu pre generovanie tokenu t_2. Generovanie tokenov sa vykonáva po ukončovaciu podmienku. Ukončovaciou podmienkou môže byť počet vygenerovaných tokenov, koniec vety, normalizovaný súčet pravdepodobností generovaných tokenov a mnoho ďalších. Generovaná sekvencia je spojením vygenerovaných tokenov t_1 až t_n.

Dekódovacie algoritmy

Dekódovacie algoritmy sú skupinou algoritmov, ktoré sa používajú na generovanie textových sekvencií z jazykového modelu. Zdrojom tejto sekcie sú slidy [25]. Dekódovacie algoritmy sú:

- **Greedy dekódovací algoritmus** vyberá v každom kroku z pravdepodobnostného rozloženia cez jednotlivé tokeny určeného jazykovým modelom najpravdepodobnejší token (argmax). Tento algoritmus slúži na vygenerovanie jedného tokenu, ktorý sa potom s ďalšími vygenerovanými tokenmi používa ako vstup jazykového modelu pre vygenerovanie celej sekvencie. Algoritmus sa vykonáva po vygenerovanie koncového tokenu alebo inú ukončovaciu podmienku, ako napr. počet tokenov, zložená pravdepodobnosť generovaných tokenov atď.
- **Beam search dekódovací algoritmus** má za cieľ nájsť sekvenciu s najväčšou pravdepodobnosťou vytvorením viacerých sekvencií súčasne. Hlavnou myšlienkou je, že sa v každom kroku dekódovania sleduje k najpravdepodobnejších čiastočných sekvencií. Po dosiahnutí ukončovacieho kritéria algoritmu sa vyberie sekvencia s najväčšou pravdepodobnosťou. V prípade, že $k = 1$, tak ide o dekódovací algoritmus greedy.

Vyššia hodnota konštanty k približuje generovanú sekvenciu písanému textu, ale zvyšuje taktiež výpočetnú náročnosť. Čo nie je vhodné a z tohoto dôvodu sa táto metóda nepoužíva ani neporovnáva s ostatnými metódami.

- **Samplovanie** sú algoritmy založené na náhodnom výbere podľa pravdepodobnostného rozloženia tokenov z jazykového modelu. Existujú rôzne variácie. Najjednoduchšie je **čisté samplovanie (anglicky pure sampling)**, ktoré je podobné ako greedy search. S tým rozdielom, že namiesto argmax sa používa samplovanie podľa pravdepodobnostného rozloženia tokenov z jazykového modelu.

3.4 Úpravy textu pre spracovávanie neurónovou sieťou

V textových dátach sú rôzne vzťahy, ktoré sú určené vlastnosťami ako napr. počet výskytov jednotlivých písmen, slov alebo vetných celkov, druh textových prvkov (podstatné meno, prídavné meno,...), dôležitosť a mnoho ďalších. V tejto práci sa prevádzajú textové dáta do číselnej podoby a následne sa používajú pre spracovávanie neurónovou sieťou. Neurónová sieť sa v prípade, že má dostatok trénovacích dát požadované vlastnosti naučí sama počas trénovania.

Spôsoby tokenizácie textových dát

Informácie v tejto podsekcii sú z [30]. Tokenizácia textových dát je proces rozdeľovania textových dát na menšie celky ako sú:

- **Písmena** ako tokeny sú vhodné a celkom presné, ale neefektívne z hľadiska počtu predpovedí neurónovej siete. A to tak, že pre predpovedanie celej sekvencie je potrebné veľké množstvo predpovedí jednotlivých písmen, ktoré sa spájajú do slov a slová do sekvencie.

- **Slová** sú sekvencie písmen oddelených medzerami. Pri použití slov na tokenizáciu textových dát nastáva problém slov mimo slovníka (anglicky out of vocabulary problem, ďalej len OOV). Slovník obsahuje všetky slová, ktoré sú obsiahnuté v trénovacej sade dát, na ktorej je neurónová sieť trénovaná. OOV problém znamená, že neurónová sieť vie pracovať len so slovami zo slovníka. Všetky ostatné slová sú reprezentované pomocou 1 tokenu, ktorý sa zvyčajne nazýva OOV token. OOV problém môže byť vyriešený použitím hybridného modelu. Hybridný model používa na tokenizáciu slová, ale keď sa objaví slovo mimo slovníka, tak sa použije tokenizácia založená na písmenách.

- **Podslovná** tokenizácia je ďalší možný spôsob tokenizácie textu. Pre tvorbu podslov sa používa algoritmus Byte Pair Encoding [13] s jeho rôznymi variantami. Algoritmus Byte Pair Encoding bol pôvodne vytvorený na kompresiu dát.

Metódy prevodu tokenov na čísla

Informácie v tejto podsekcii sú z [4] a [10] kapitola 8. Neurónové siete nevedia pracovať s textovými dátami reprezentovanými tokenmi. Preto musia byť tokeny prevedené z textovej podoby do podoby čísel, s ktorými neurónové siete vedia pracovať. Existujú rôzne metódy na prevod. Medzi najznámejšie metódy patria:

- **Integer encoding** je zakódovanie každého tokenu pomocou iného čísla. Dôležité pre túto metódu je, že vzťahy medzi jednotlivými tokenmi sú dopredu známe. V zakódovanej podobe sú vzťahy určené číselným rozdielom jednotlivých zakódovaných tokenov číslami.

- **One hot encoding** reprezentuje každý token binárným vektorom o dimezii veľkosti počtu všetkých kódovaných tokenov zo slovníka. Pre každý token je v tomto vektore nastavený jeden prvok na 1 a všetky ostatné na 0. Tento spôsob kódovania je vhodný, ak nie sú prítomné žiadne vzťahy medzi vstupnými tokenmi a je k dispozícii dostatočne veľká pamäť.

- **Embeding** reprezentuje podobné tokeny pomocou podobných vektorov. Takýmto spôsobom reprezentuje metóda embeding vzťahy medzi jednotlivými tokenmi. Metóda embeding kóduje tokeny na n-dimenzionálny vektor, pri čom jeho dimenzionalita nie je ničím obmedzená. Hodnoty čísel vo vektoroch sú upravované počas trénovania, čo znamená, že na konci trénovania sú podobné tokeny reprezentované podobnými vektormi. Algoritmy pre embeding metódu sú word2vec [19] a glove [21].

Kapitola 4

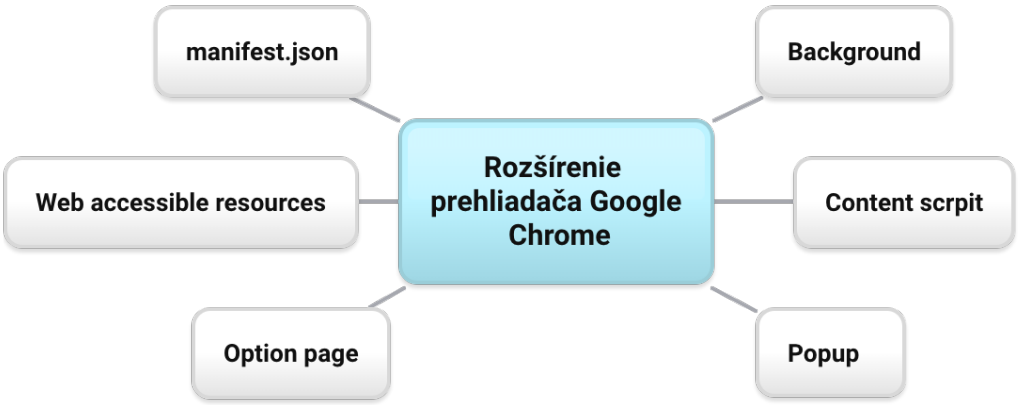
Rozšírenie pre prehliadač Google Chrome

Rozšírenia prehliadačov sú malé programy, ktoré slúžia na prispôsobovanie prehliadačov užívateľom za účelom zlepšenia ich užívateľského zážitku. Aktuálne je bezpečnosť jednou z hlavných priorít rozšírení prehliadačov. Preto je štruktúra rozšírení pevne daná a zasahovanie do prehliadača je do značnej miery obmedzené. V technológii rozšírení prehliadačov nemôže samotný kód rozšírenia priamo zasahovať do kódu stránky. Toto je spôsobené oddelenosťou procesov. Rozšírenie prehliadača beží v jednom procese a samotný prehliadač beží v druhom procese. Výhodou oddelenosti procesov je bezpečnosť a zlepšenie užívateľského zážitku, pretože napríklad zle naimplementované rozšírenie bežiace v jednom procese s prehliadačom by spôsobilo svojím pádom pád celého prehliadača. Do kódu stránky sa dá zasahovať pomocou kódu, ktorý je vložený priamo do aktuálnej webovej stránky pomocou rozšírenia. Za predchodcu technológie rozšírení prehliadačov je považovaná technológia XUL[1]. XUL bola používaná Firefoxom. Táto technológia povoľovala značné zásahy do Firefoxu, čo bolo na jednej strane dobré pre prispôsobovanie prehliadačov individuálnym potrebám používateľov, ale na druhej strane to bolo veľké bezpečnostné riziko. Na dnešnom trhu je viacero veľkých prehliadačov ako Opera, Firefox, Safari a Google Chrome. Všetky uvedené prehliadače podporujú s malými rozdielmi technológiu rozšírenia prehliadačov. V bakalárskej práci je použité rozšírenie pre prehliadač Google Chrome, a preto je ďalej rozoberané ako jediné. Informácie v tejto kapitole sú čerpané z [6]. Sekcia 4.1 popisuje štruktúru rozšírenia prehliadača Google Chrome a jeho jednotlivé komponenty. V ďalšej sekcii 4.2 je analyzovaná komunikácia medzi jednotlivými komponentami. Možnosti ukladania dát v rozšírení prehliadača Google Chrome sú rozpísané v poslednej sekcii 4.3.

4.1 Komponenty

Na obrázku 4.1 sú stavebné komponenty rozšírenia pre prehliadač Google Chrome. Celkovo je ich 6 a každý komponent má svoj špecifický účel. Väčšina používateľov prichádza do priameho kontaktu len s option page a popup komponentom.

[1] https://developer.mozilla.org/en-US/docs/Archive/Mozilla/XUL



Obr. 4.1: Štruktúra rozšírenia pre prehliadač Google Chrome.

Manifest.json

Manifest.json je vstupný bod technológie rozšírenia prehliadača Google Chrome. Tento súbor musí byť umiestnený v koreňovom adresári rozšírenia prehliadača. Manifest súbor obsahuje informácie o všetkých ostatných komponentoch prostredníctvom ciest k ich JavaScript (ďalej js) a html súborom. Obsahuje taktiež množstvo ďalších informácií, ako sú metainformácie o rozšírení a rôzne povolenia prístupu k externým zdrojom a dátovým úložiskám. Súbor manifest.json je vo formáte Javascript Object Notation (JSON) [2]. Prístup k externým zdrojom je cez položky v poli s kľúčom permissions. Súbor manifest musí povinne obsahovať aspoň 3 kľúče (manifest_version, name a version). V prípade nesprávnej implementácie týchto základných kľúčov, syntaktickej chyby alebo nesprávnej cesty k html alebo js súborom jednotlivých komponentov, prehliadač Google Chrome odmietne nainštalovať toto rozšírenie.

Web accessible resources

Web accessible resources je v manifest súbore pod kľúčom web_accessible_resources. Je to pole, ktoré špecifikuje cesty k rôznym súborom, ktorých použitie je očakávané v kontexte rôznych navštívených webových stránok. Napríklad rozšírenie prehliadača vie pomocou neho zmeniť webovú stránku na obrázok. Jednotlivé položky poľa sú zadávané relatívnymi cestami vzhľadom ku koreňovému adresáru, kde je manifest súbor. V položkách môžu byť wildcards, čo znamená, že môžu byť napríklad jednou položkou povolené všetky obrázky s príponou jpg a nemusia sa zadávať do poľa web_accessible_resources cesty ku každému jpg súboru zvlášť.

Option page

Option page je v manifest súbore prvok pod kľúčom option_page. Slúži na prispôsobenie rozšírenia a zmenu rôznych nastavení, ktoré sú poskytnuté užívateľovi rozšírenia prehliadača Google Chrome. Option page je dostupné užívateľovi pomocou pravého kliku myškou na ikonku rozšírenia v paneli nástrojov a následným výberom option položky zo zoznamu alebo z manažéra rozšírení ako jedna z možností pre konkrétne rozšírenie.

[2] https://www.json.org/json-en.html

Background

Background je komponent, ktorý povinne obsahuje js súbor - handler udalostí pre rozšírenie. Background môže obsahovať voliteľný html súbor. Background je v manifest súbore uložený pod kľúčom background s jednotlivými položkami v poli. Background script beží od spustenia prehliadača až po jeho zatvorenie (ak používateľ manuálne nevypne celé rozšírenie cez manažéra rozšírení).

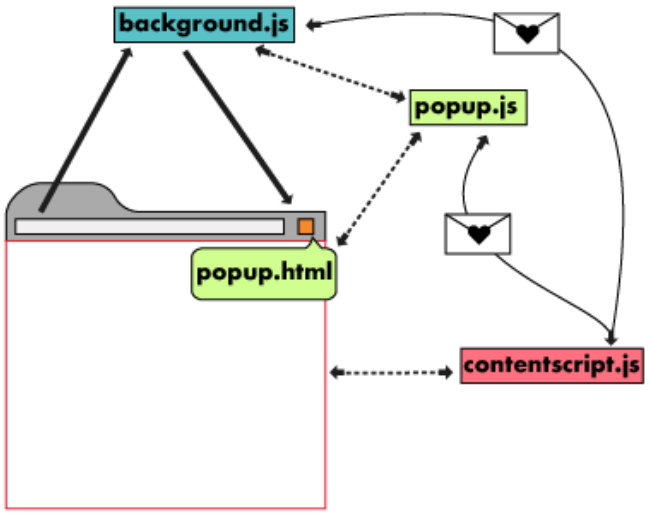
Content skript

Content skript má kľúč content_script a obsahuje pole js súborov spolu s URL adresami [3], kde je content skript pre rozšírenie spúšťaný. Content scripty slúžia na čítanie Document Object modelu (DOM) [4] konkrétnej webovej stránky a jeho zmenu spojenú s možnosťou vkladania js kódu priamo do stránky za účelom jeho vykonávania. Content skripty bežia v kontexte načítanej webovej stránky. To znamená, že vždy, keď sa načíta nová stránka, spustí sa content skript nanovo.

Popup

Posledným komponentom je popup. Popup sa skladá z html a js súboru, ktoré sú uložené v manifest súbore ako pole pod kľúčom popup. Popup väčšinou slúži na rôzne odkazy ako napríklad na domovskú stránku a zvyčajne obsahuje rôzne informácie o rozšírení. Obsah popup komponentu sa zobrazí po kliknutí na ikonku konkrétneho rozšírenia v paneli nástrojov pre prehliadač Google Chrome.

4.2 Komunikácia medzi komponentami



Obr. 4.2: Komunikácia medzi jednotlivými komponentmi rozšírenia prehliadača Google Chrome, prevzaté z [6].

[3] https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL
[4] https://developer.mozilla.org/enUS/docs/Web/API/Document_Object_Model

Každý komponent má svoj špecifický účel. To znamená, že napríklad background rovnako ako popup nemôže pristupovať k DOMu otvorenej webovej stránky alebo content skript nemôže mať nastavené handlery pre celý internetový prehliadač Google Chrome. Jednotlivé komponenty môžu spolu komunikovať pomocou asynchrónneho zasielania správ, ktoré je zobrazené na obrázku 4.2. Správy môžu byť zasielané jednorázovo alebo môžu byť použité dlhotrvajúce spojenia. Komunikácia pomocou správ je možná aj medzi rôznymi rozšíreniami prehliadača Google Chrome.

4.3 Možnosti ukladania dát

Na ukladanie dát pre rozšírenie prehliadača Google Chrome môže byť použitá localStorage [5], IndexedDB [6] a chrome storage [7]. Chrome storage musí byť povolená pre rozšírenie v manifest súbore pomocou kľúčového slova storage v poli permission. LocalStorage a IndexedDB sú defaultne povolené.

Chrome storage je súčasťou technológie rozšírenia prehliadača Google Chrome. Zachováva dáta medzi jednotlivými spusteniami prehliadača (rovnako ako LocalStorage a IndexedDB). Dáta nie sú viazané na žiadnu doménu, ale ku konkrétnemu rozšíreniu prehliadača Google Chrome. LocalStorage je viazaná na konkrétnu doménu. No v prípade jej použitia pre background script (ktorý má doménu chrome://URI), je možné používať uložené dáta cez background script alebo pomocou asynchronnej komunikácie v content script. Chrome storage je úložisko s asynchrónnym prístupom k dátam, pri čom dáta môžu byť uložené lokálne alebo synchrónne. Lokálne znamená, že dáta sú uložené len na konkrétnom stroji. Synchrónne znamená, že dáta sú automaticky synchronizované s užívateľovým Google účtom naprieč rôznymi zariadeniami. Chrome storage pre jednotlivé rozšírenia prehliadača Google Chrome je limitované veľkosťou dátového úložiska a počtom zápisov dát. Počet zápisov je maximálne 120 za minútu a 1800 za hodinu. Citlivé informácie by nemali byť ukladané bez šifrovania v chrome storage, pretože nepoužíva žiadnu šifrovaciu metódu na zabezpečenie dát.

[5] https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
[6] https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API
[7] https://developer.chrome.com/extensions/storage

Kapitola 5

Návrh riešenia

V tejto kapitole je uvedený návrh riešenia pre napovedanie textových sekvencií pomocou rozšírenia pre prehliadač Google Chrome za použitia rekurentnej neurónovej siete. V sekcii 5.1 je rozobrané zadanie bakalárskej práce. V sekcii 5.2 je popis existujúceho riešenia Gmail Smart Compose. V nasledujúcej sekcii 5.3 je stručne zhrnutý postup riešenia. Ďalšia sekcia 5.4 je architektúra riešenia s podrobným popisom jednotlivých komponentov. V sekcii 5.5 je spôsob výberu a skladanie tokenov z výstupu neurónovej siete. Ďalej je v tejto sekcii popis pravidla pre zobrazovanie generovanej sekvencie používateľovi. V poslednej sekcii 5.6 je rozobraté upravovanie hyperparametrov neurónovej siete.

5.1 Zrekapitulovanie zadania

Cieľom tejto bakalárskej práce je vytvoriť napovedač sekvencií anglických slov v prehliadači Google Chrome. Napovedač má byť použiteľný na rôznych internetových stránkach. Má používať rekurentnú neurónovú sieť, ktorá sa vyhodnocuje na strane používateľa v prehliadači Google Chrome pomocou rozšírenia pre prehliadač Google Chrome. Neurónová sieť má byť schopná predpovedať následujúci token na základe vstupných tokenov. V rozšírení je potrebné implementovať logiku na generovanie textových sekvencií za použitia natrénovanej neurónovej siete. Ná záver je potrebné vytvoriť webovú stránku na demonštračné účely a publikovať výsledné rozšírenie pre prehliadač Google Chrome na Chrome Web Store.

5.2 Existujúce riešenie - Gmail Smart Compose

Gmail Smart Compose (článok [5], ďalej len GSC) je systém na dopĺňanie textových sekvencií v reálnom čase pre Gmail. Jeho predchodcom je Google Smart Reply (článok [14]), ktorý slúži na automatické generovanie krátkych odpovedí na prichádzajúce emaily.

Jadro GSC je tvorené neurónovou sieťou, ktorá je natrénovaná na veľkom množstve emailových dát. Každým stlačením klávesy užívateľom sa zasiela request na streamovací RPC server, kde je GSC systém produkčne umiestnený. Odpoveďou zo serveru je generovaná sekvencia, ktorá je následne zobrazená. GSC systém používa beam search na vygenerovanie n sekvencií pomocou neurónovej siete. Z n sekvencií sa vyberie sekvencia s najväčšou pravdepodobnosťou. Táto sekvencia je zaslaná používateľovi na zobrazenie v prípade, ak je GSC systém dostatočne "spokojný"s vygenerovanou sekvenciou. Spokojnosť je určená ako dĺžkou normalizovaný logaritmický súčet podmienených pravdepodobností všetkých tokenov tvo-

riacich výslednú sekvenciu. Jazykový model je dostatočne spokojný, ak výsledná hodnota "spokojnosti"prekročí prahovú hodnotu.

Hlavné výzvy, s ktorými sa stretli vývojári GSC sú:

- **Latencia** v generovaní sekvencií. Pretože celý systém je umiestnený na streamovaciom RPC serveri.
- **Škálovatelnosť** pre 1.5 miliardy používateľov, ktorí boli v roku 2019 aktívni na Gmaili.
- **Personalizáciu** pre každého používateľa, pretože každý používateľ má svoj typický štýl v písaní emailov. Perzonalizácia GSC zabezpečuje sekundárnym odľahčeným jazykovým modelom, s ktorým počas generovania sekvencie intereaguje globálny jazykový model. Odľahčený jazykový model je natrénovaný na používateľových emailoch. Tento model sa periodicky pretrénováva.
- **Ochrana súkromia.** Systém nemôže nikomu sprístupňovať používateľove privátne dáta (ani počas trénovania).

Google používa TPU [1] na trénovanie aj na produkčné nasadenie systému GSC. TPU je AI ASIC vyvinutý Googlom pre trénovanie a produkčné nasadenie systémov s neurónovými sieťami.

5.3 Stručný popis návrhu riešenia

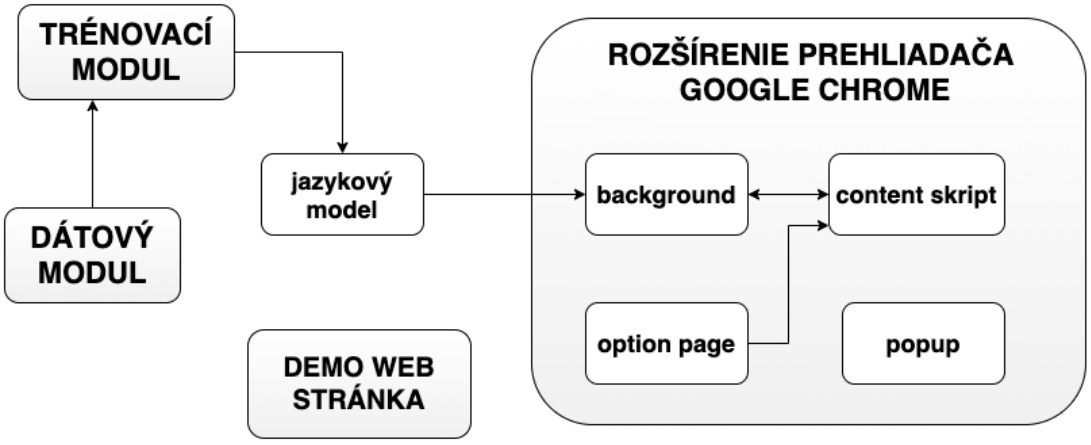
Najskôr je potrebné vytvoriť a natrénovať neurónovú sieť, ktorá bude použitá v rozšírení pre prehliadač Google Chrome. Pre trénovanie je potrebná dátová sada. Dátová sada je rozdelená na trénovaciu, validačnú a testovaciu časť. Na trénovanie neurónovej siete je použitá len časť trénovacej časti dátovej sady. Trénovacia, testovacia a validačná časť dátovej sady sú pomocou podslovnej tokenizácie rozdelené na menšie časti (podslovné tokeny). Z nich je vytvorený slovník na mapovanie jednotlivých tokenov na čísla a naopak. Ďalej sú podslovné tokeny prevedené pomocou slovníka na čísla, z ktorých sú vytvorené dvojice vstupných a cieľových tokenov. Tieto dvojice sa používajú priamo na trénovanie, validovanie a testovanie neurónovej siete. Výstupom neurónovej siete je pravdepodobnostné rozloženie cez všetky tokeny zo slovníka. Z pravdepodobnostného rozloženia sa pomocou algoritmov na generovanie sekvencie (sekcia 3.3) vygeneruje sekvencia a tá je po splnení určitých podmienok (sekcia 5.5) zobrazená používateľovi.

Nová sekvencia sa začne generovať až po uplynutí časového úseku bez užívateľovej aktivity vo vybranom HTML elemente. Časový úsek je defaultne 0.5 sekundy.

Každý užívateľ si po inštalácii SPT automaticky stiahne a lokálne uloží predtrénovanú neurónovú sieť a slovník na mapovanie tokenov. Takýmto spôsobom je zabezpečená škálovatelnosť systému SPT.

SPT nerieši žiadnym spôsobom personalizáciu konkrétnym používateľom.

Ochranu súkromia používateľov nie je taktiež potrebné riešiť. Pretože sa nimi písaný text nikam neodosiela, ale len sa lokálne u nich spracováva pomocou výsledného riešenia, ktoré sa volá Sequence Prediction Tool (ďalej len SPT).



Obr. 5.1: Architektúra riešenia

5.4 Architektúra riešenia

Na obrázku 5.4 je architektúra môjho riešenia. Skladá sa zo 4 hlavných komponentov:

- **Dátový modul** slúži na stiahnutie dátovej sady, jej vyčistenie, tvorbu slovníka, podslovnú tokenizáciu, prevod tokenov na čísla a tvorbu dvojíc vstupných a výstupných tokenov na trénovanie, validovanie a testovanie.
- **Trénovací modul** slúži na zostavenie a trénovanie rekurentnej neurónovej siete pomocou predpripravených číselných dvojíc z dátového modulu. Natrénovaním neurónovej siete sa vytvorí jazykový model, ktorý generuje tokeny zo vstupných tokenov. Jazykový model je následne konvertovaný do formátu vhodného na použitie v rozšírení pre prehliadač Google Chrome.
- **Demonštračná webová stránka** slúži na vyskúšanie si výsledného riešenia SPT.
- **Rozšírenie prehliadača Google Chrome** je zložené z background, content skript, option page a popup. Rozšírenie prehliadača získava písaný text užívateľom na internetových stránkach a používa ho ako vstup pre jazykový model. Z jazykového modelu je použitím algoritmov na generovanie sekvencie (sekcia 3.3) vygenerovaná sekvencia, ktorú rozšírenie prehliadača po splnení určitých podmienok (sekcia 5.5) zobrazí používateľovi ako napovedaný text.

Dátový modul

Dátový modul slúži na spracovanie dátovej sady, ktorá je vybraná priamo z TensorFlow frameworku datových sád na spracovanie textu [2]. Použitá datová sada je gigaword [11]. Obsahuje približne 4,000,000 článkov, pri čom trénovacia časť obsahuje 3,803,957 článkov, validačná 189,651 článkov a testovacia 1,951 článkov.

Pred samotným spracovaním dátovej sady bolo potrebné vyčistiť text. Následne je text rozdelený na tokeny pomocou podslovnej tokenizácie a pomocou vytvoreného slovníka sú

[1] https://cloud.google.com/tpu
[2] https://www.tensorflow.org/datasets/catalog/overview

tokeny prevedené na čisla. Z čísel sú vytvorené dvojice vstupných a výstupných tokenov, ktoré sa používajú priamo na trénovanie neurónovej siete.

Trénovací modul

Úlohou trenovacieho modulu je vytvoriť rekurentnú neurónovú sieť, natrénovať ju a uložiť. Natrénovaná neurónová sieť sa ukladá vo formáte pre TensorFlow. Z tohoto formátu sa konvertuje do formátu pre TensorFlow.js, ktorý sa používa v rozšírení pre prehliadač Google Chrome.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (1, None, 200)            1206400
_________________________________________________________________
gru_2 (GRU)                  (1, None, 340)            551820
_________________________________________________________________
dropout_2 (Dropout)          (1, None, 340)            0
_________________________________________________________________
gru_3 (GRU)                  (1, None, 340)            694620
_________________________________________________________________
dropout_3 (Dropout)          (1, None, 340)            0
_________________________________________________________________
dense_1 (Dense)              (1, None, 6032)           2056912
=================================================================
Total params: 4,509,752
Trainable params: 4,509,752
Non-trainable params: 0
```

Obr. 5.2: Architektúra použitej rekurentnej neurónovej siete.

Na obrázku 5.2 je štruktúra výslednej neurónovej siete. K jej štruktúre sa dospelo úpravou hyperparametrov neurónovej siete, ktorých výber je vysvetlený v sekcii 5.6. Neurónová sieť sa skladá z:

- **Vstupnej embedding vrstvy** s 200 neurónmi.
- **2 dropout vrstiev** na zamedzenie pretrénovania.
- **2 skrytých rekurentných GRU vrstiev** s 340 neurónmi na naučenie sa závislostí v trénovacích dátach.
- **Výstupnej dense vrstvy** obsahujúcej 6032 neurónov, čo je presne počet prvkov v slovníku plus počet prvkov v byte. Pretože TensorFlow subword tokenizácia používa bytové kódovanie na OOV tokeny.

Graf prepojenia jednotlivých vrstiev u výslednej neurónovej siete sa dá pozrieť v logoch z finálneho trénovania. Popis ich spustenia v TensorBoard je v prílohách.

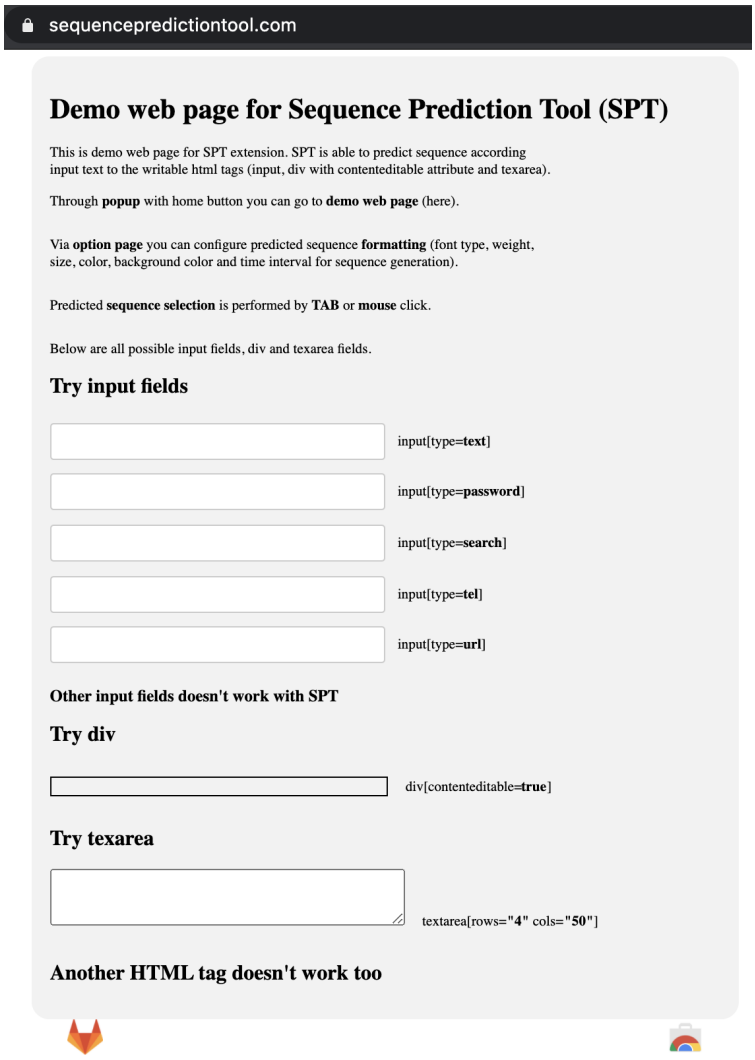
Jazykový model

Jazykový model je natrénovaná, uložená a konvertovaná neurónová sieť z Tesorflow do Tensorflow.js formátu. Konverovaná do tf.js formátu je z dôvodu jej používania v rozšírení

pre prehliadač Google Chrome. Kvôli sťahovaniu jazykového modelu v rozšírení je jazykový model spolu so slovníkom na mapovanie tokenov umiestnený na serveri.

Demonštračná webová stránka

Na obrázku 5.3 je demonštračná webová stránka [3], ktorá slúži na vyskúšanie si funkcionality SPT. Na webovej stránke je popis a spôsoby používania SPT. V spodnej časti stránky je odkaz na zdrojové kódy a odkaz do Chrome Web Store pre inštaláciu SPT rozšírenia.



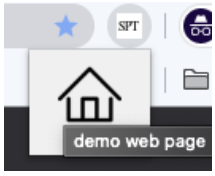
Obr. 5.3: Demonštračná webová stránka.

Rozšírenie prehliadača Google Chrome

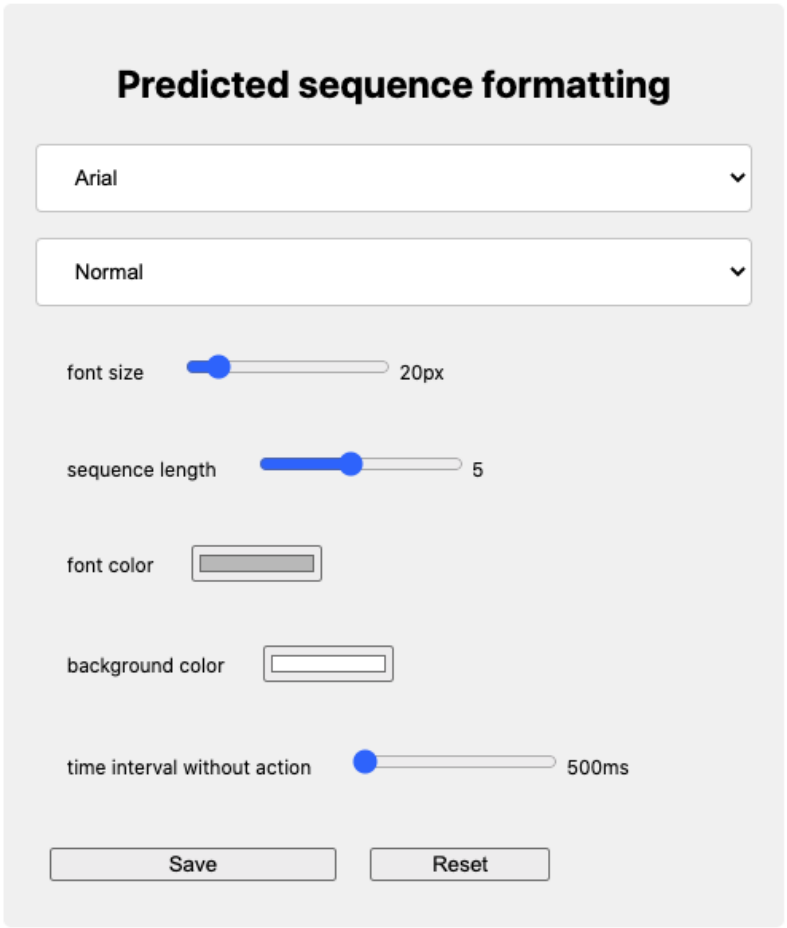
Rozšírenie pre prehliadač Google Chrome sa skladá z: popup, option page, background a content skript.

Na obrázku 5.4 je **popup** komponent, ktorý slúži len ako odkaz na demonštračnú webovú stránku. Načítava sa vždy po svojom otvorení.

[3] https://sequencepredictiontool.com/

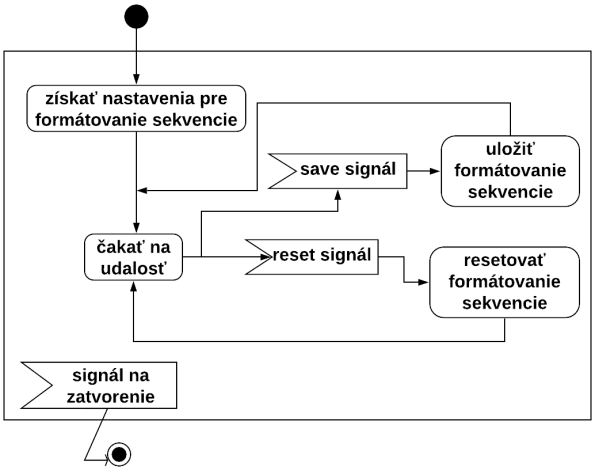


Obr. 5.4: Popup komponent rozšírenia pre prehliadač Google Chrome.



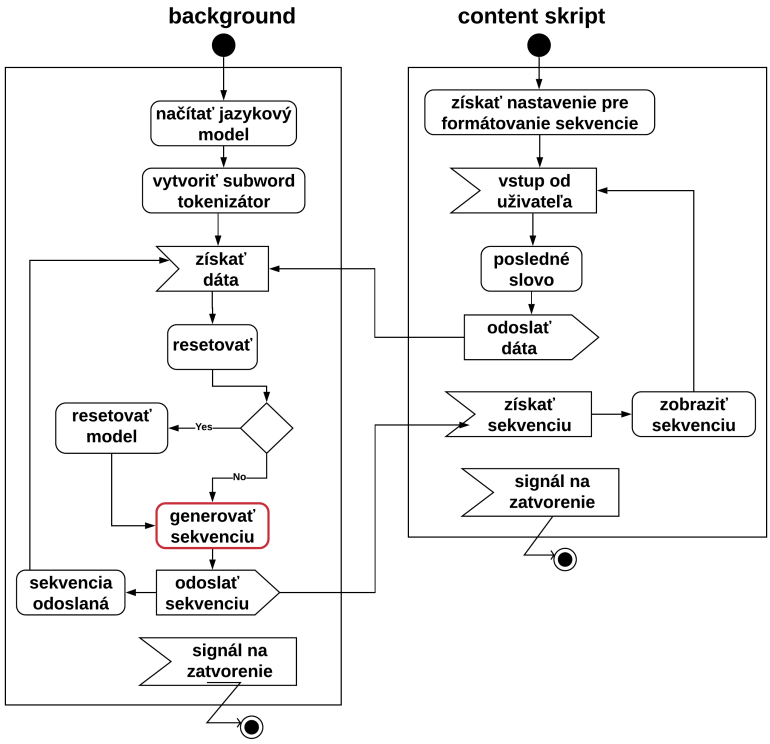
Obr. 5.5: Option page rozšírenia prehliadača Google Chrome pre formátovanie predpovedanej sekvencie.

Na obrázku 5.5 je **option page**, ktoré slúži na prispôsobovanie si formátovania predpovedanej sekvencie. Umožňuje zmeniť typ písma, jeho tvar, veľkosť, maximálnu dĺžku generovanej sekvencie, farbu, farbu pozadia a časový interval užívateľovej nečinnosti, po ktorej sa začne generovať textová sekvencia. Každú zmenu treba potvrdiť uložením hodnôt pomocou tlačítka Save. Po nainštalovaní rozšírenia sú parametre na formátovanie nastavené na defaultné hodnoty (Arial, Normal, 20px, 5 slov, silver, white, 500ms). Do defaultného nastavenia formátovania sa dá vrátiť pomocou tlačítka Reset.



Obr. 5.6: Diagram aktivít pre option page.

Na obrázku 5.6 je diagram aktivít pre option page. Skript pre option page sa spúšťa vždy po otvorení tohto komponentu. Z obrázku je zrejmé, že najskôr sa získajú aktuálne nastavenia pre formátovanie predpovedanej sekvencie a následne sa čaká na jednu zo save alebo reset udalostí. Save uloží nastavenia zadané užívateľom do chrome storage a reset obnoví pôvodné nastavenia a uloží ich tiež do chrome storage. Vykonávanie option page sa ukončí jeho zatvorením.



Obr. 5.7: Diagram aktivít pre background a content skript.

Na obrázku 5.7 je diagram aktivít pre background a content skript. **Background** slúži na generovanie sekvencie a komunikáciu s content skriptom. Spúšťa sa po otvorení prehliadača Google Chrome. Najskôr si načíta jazykový model. Vytvorí podslovný tokenizátor a čaká na dáta od content skriptu. Z obdržaných dát sa rozhodne, či má resetovať skryté stavy v GRU vrstvách neurónovej siete a zistí si z dát maximálnu dĺžku pre generovanie sekvencie. Ďalej vygeneruje sekvenciu slov zo vstupnej sekvencie z dát. Spôsob generovania sekvencie je v sekcii 5.5. Túto sekvenciu pošle content skriptu a vráti sa do čakajúceho stavu na dáta. Vykonávanie background skriptu môže byť ukončené len zatvorením prehliadača.

Content skript slúži na získavanie písaného textu používateľom, komunikáciu s background skriptom a zobrazovanie predpovedanej sekvencie. Beží v kontexte webovej stránky a spúšťa sa vždy pri načítaní novej stránky alebo pri znovu načítaní pôvodnej stránky.

Content skript si najskôr získa uložené vlastnosti na formátovanie sekvencie a maximálnu dĺžku generovanej sekvencie (po ich uložení cez option page sa tieto nastavenia prejavia v content skript až pri jeho novom spustení). Ďalej content skript čaká na vstup od používateľa. Po napísaní textu užívateľom vyberie tento text a pošle background skriptu dáta. V dátach je zahrnutý napísany text, maximálna dĺžka pre generovanie sekvencie a informácia o tom, či má background skritp resetovať skryté stavy v GRU vrstvách neurónovej siete. Resetovanie je potrebné vykonať, ak si užívateľ nevybral predchádzajúcu generovanú sekvenciu a v prípade, ak užívateľ začne písať do iného HTML elementu ako doteraz. Ďalej content skript čaká na predpovedanú sekvenciu od backgroundu, ktorú následne zobrazí a vráti sa do čakajúceho stavu na užívateľov vstup.

Content skript vie pracovať s textom z HTML elementov:

- **input** s atribútom *type* pre hodnoty $text, password, search, tel, url$
- **div** s atribútom $contenteditable = "true"$
- **texarea**

Content skript vie pracovať so zmienenými HTML elmentmi len ak nie sú dynamicky vytvárane. To znamená, že tieto elementy musia byť vytvorené a dostupné na stránke pred spustením content skriptu.

Content skript posiela background skriptu text, ktorý používateľ napísal pred pauzou do niektorého z HTML elementov. Tento text je následne použitý na generovanie sekvencie v background skripte. Použitý text na generovanie spolu s generovaným textom z aktuálne používaného HTML elementu, ktorý používateľ zvolil, sa ukladá. Ukladajú sa pre to, aby sa content skript vedel pri ďalšom zasielaní textu rozhodnúť, ktorý text už bol použitý na generovanie a ktorý text ešte nie a treba ho použiť.

Beh content skriptu môže byť ukončený načítaním novej stránky, znovu načítaním pôvodnej stránky alebo zatvorením prehliadača.

5.5 Generovanie sekvencie

Generovanie sekvencie prebieha v background pomocou predtrénovanej rekurentnej neurónovej siete a dekódovacieho algoritmu. Výstupom neurónovej siete je pravdepodobnostné rozloženie výskytu ďalšieho tokenu s rozložením cez všetky tokeny zo slovníka. Postupným výberom a spájaním tokenov (rovnica 3.4) z pravdepodobnostného rozloženia pomocou dekódovacieho algoritmu samplovanie sa generuje celá sekvencia. Výber dekódovacieho algoritmu je zdôvodnený v sekcii 6.4.

Pre vygenerovanú sekvenciu sa ďalej požíva pravidlo spoľahlivosti, ktoré určí, či sa vygenerovaná sekvencia zobrazí používateľovi alebo nie. Použitie spoľahlivosti je inšpirované z GSC (sekcia 5.2). Rovnica 5.1 predstavuje spokojnosť, ktorej výpočet je prevzatý z článku o GSC [5]. Spokojnosť je určená ako dĺžkou T normalizovaný logaritmický súčet n pravdepodobností P_i generovaných tokenov.

$$spolahlivost = \frac{1}{T} \sum_{i=1}^{n} \log_{10} P_i \tag{5.1}$$

Ak je spoľahlivosť väčšia ako $log_{10}0.5$, tak background pošle content skriptu vygenerovanú sekvenciu. V opačnom prípade mu pošle prázdny reťazec a content skript nezobrazí vygenerovanú sekvenciu. Hodnotu pre zlogaritomovanie 0.5 je určená ako hraničná, pretože predstavuje pravdepodobnosť 50 percent pre celú sekvenciu, čo považujem za dostatočnú spoľahlivosť.

5.6 Upravovanie hyperparametrov neurónovej siete

V bakalárskej práci je použitý manuálny prístup na úpravu hyperparametrov neurónovej siete. Upravované hyperparametre sú:

- **Počet rekurentných vrstiev** bol určený na 2 podľa experimentov v článku o GSC [5].
- **Počet neurónov skrytej vrstvy** bol taktiež určený podľa experimentov v článku GSC [5]. V GSC experimentoch Google používa 1024 a 2048 rekurentných neurónov. V práci je použitých len 340, z dôvodu obmedzených výpočetných možností na trénovanie v porovnaní s Google.
- **Počet trénovacích epôch** bol určený na 29 podľa priebehu trénovania (sekcia 6.3).
- **Učiaci koeficient**. S učiacim koeficientom je úzko spojený optimalizátor, ktorý rôznymi spôsobmi mení veľkosť učiaceho kroku počas trénovania. Optimalizátor Adam bol vybraný na základe článku [18]. Z článku vyplýva, že Adam patrí k jedným z najlepších optimalizátorov.
- **Stratová funkcia**. Výstupom neurónovej siete je pravdepodobnostné rozloženie cez všetky tokeny zo slovníka. Preto spadá do triedy úloh spojených s klasifikáciou do viacerých tried. Podľa článku [3] je pre tento typ úloh vhodná sparse categorical crossentropy starová funckia, ktorá je použitá.

Kapitola 6

Implementácia riešenia

Táto kapitola obsahuje popis implementácie a jej najdôležitejších častí. Sekcia 6.1 sa zaoberá použitými technológiami a ich popisom. V sekcii 6.2 sú popísané najdôležitejšie časti kódu pre vytvorenie a trénovanie neurónovej siete a pre jej použitie v rozšírení pre prehliadač Google Chrome. V sekcii 6.3 je rozobrané finálne trénovanie neurónovej siete. V ďalšej sekcii je spôsob výberu dekódovacieho algoritmu pre generovanie sekvencie. Posledná sekcia 6.5 popisuje problémy pri implementácii a publikácii výsledného riešenia.

6.1 Použité technológie

Implementácia sa skladá z dvoch častí. Prvá časť sa zaoberá neurónovou sieťou. Konkrétne jej konštrukciou, trénovaním a ukladaním. Druhá časť je rozšírenie pre prehliadač Google Chrome, ktoré sa stará o generovanie textových sekvencií pomocou natrénovanej neurónovej siete a ich zobrazovanie na rôznych webových stránkach. Boli použité nasledujúce technológie:

- **pre neurónovú sieť**:
  - **Python 3.6** bol použitý v spojení s TensorFlow frameworkom a NumPy knižnicou na vytvorenie, trénovanie a konverziu neurónovej siete do formátu pre TensorFlow.js.
  - **NumPy 1.16.4** bol použitý na prípravu trénovacej sady dát.
  - **TensorFlow 1.15.0** podporuje 2 prístupy: sekvenčný a funkcionálny. Funkcionálny prístup je variabilnejší. V tomto prístupe si môže užívateľ definovať presný postup trénovania neurónovej siete, jej štruktúru a mnoho ďalšieho. Sekvenčný prístup sa dá na rozdiel od funkcionálneho konvertovať do TensorFlow.js, a preto je použitý v tejto práci.
- **pre rozšírenie prehliadača**:
  - **JavaScript ES6**
  - **Tesorflow.js 1.4.0** sú časti TensorFlow frameworku prepísané do JavaScriptu. Tensorlfow.js sa v práci používa prácu s natrénovanou neurónovou sieťou. TensorFlow.js neobsahuje všetky prvky Tensorlfow, a preto bolo potrebné prepísať podslovný tokenizátor [1] pre TensorFlow z Pythonu do JavaScriptu.

[1] https://www.tensorflow.org/datasets/api_docs/python/tfds/features/text/SubwordTextEncoder

- **jQuery 3.4.1**
- **jQuery UI Autocomplete 1.12.1** slúži na dopĺňanie písaného textu z dopredu definovaného listu. V bakalárskej práci je použitý na zobrazovanie predpovedaných sekvencií užívateľom na rôznych internetových stránkach.
- **Technológia rozšírenia pre prehliadač Google Chrome** slúži na získavanie písaného textu používateľom, z ktorého sa generujú pomocou jazykového modelu textové sekvencie. Tieto sekvencie sú zobrazované používateľovi s možnosťou ich doplnenia k už napísanému textu. Podrobné informácie k rozšíreniu sú v kapitole 4.

6.2 Dôležité implementačné časti

Dôležité implementačné časti pre neurónovú sieť sú: trieda Dataset a trieda RNN.

Trieda Dataset

Táto trieda sa stará o stiahnutie dátovej sady gigaword, ktorá je rozdelená na trénovaciu, validačnú a testovaciu časť. Všetky časti sa očistia odstránením HTML br tagov, všetkých čísel, dvojitých úvodzoviek, zátvoriek a viacnásobných interpunkčných znamienok. Ďalej vytvorí slovník pomocou TensorFlow triedy SubwordTextEncoder a vytvorí podslovné tokeny. Vytvorený slovník sa preformátuje do JSON formátu pre používanie v TensorFlow.js a uloží sa. Následne sa vytvoria dvojice predstavujúce vstupný a cieľový token. Posledným krokom je rozdelenie týchto dvojíc na trénovacie dávky o veľkosti 32 dvojíc (vstupný a výstupný token).

Trieda RNN

Trieda RNN si pripraví trénovaciu, validačnú a testovaciu časť pomocou triedy Dataeset a uloží si slovník pre TensorFlow.js taktiež pomocou triedy Dataset. Následne si vytvorí neurónovú sieť a spustí jej trénovanie. Po dotrénovaní konvertuje uloženú neurónovú sieť z TensorFlow do TensorFlow.js a uloží ju.

Dôležité implementačné časti pre rozšírenie prehliadača sú: background, trieda SubWordTokenizer, trieda SequencePredictor a JQuery UI Autocomplete v content skripte.

Background

Background si pri prvom spustení rozšírenia pre prehliadač zo serveru stiahne natrénovanú neurónovú sieť a slovník na kódovanie tokenov. Neurónovú sieť si uloží do IndexedDB a slovník do LocalStorage. Pri ďalších spusteniach používa uložené inštancie. Ďalej si vytvorí objekt SequencePredictor a komunikačný kanál na prenos dát medzi SequencePredictorom a JQery UI Autocomplete v content skripte.

Trieda SubWordTokenizer

Trieda SubWordTokenizer je prepísaná TensorFlow trieda SubwordTextEncoder z Pythonu do JavaScriptu. Konkrétne len metódy encode a decode. Encode sa stará o zakódovanie slova do listu tokenov pomocou uloženého slovníka. Decode sa stará o dekódovanie listu

tokenov do sekvencie slov taktiež pomocou uloženého slovníka. Metóda encode používa hashovaciu funkciu, ktorá je prevzatá z intetnetu [2].

Trieda SequencePredictor

SequencePredictor sa vytvára v background skripte. Načíta si uloženú neurónovú sieť pomocou TensorFlow.js a vytvorí si objekt SubWordTokenizer na zakódovavanie generovaniej sekvencie tokenov na slová a dekódovanie vstupnej sekvencie slov od content skriptu na tokeny. Ďalej tento objekt čaká na dáta od content skriptu. Na základe týchto dát sa rozhoduje, či má resetovať skryté stavy v skrytých neurónových GRU vrstvách a generuje novú sekvenciou podľa pravidiel popísaných v sekcii 5.5. Vygenerovaná sekvencia je zaslaná content skriptu. Ďalej SequencePredictor objekt čaká na ďalšie dáta od content skriptu. Tento proces sa opakuje, pokiaľ nie je vypnuté rozšírenie SPT alebo pokiaľ nie je zatvorený webový prehliadač. Táto trieda používa prevzatú funkciu z internetu na výber tokenu pomocou samplovania[3] z pravdepodobnostného rozloženia výstupu neurónovej siete.

JQery UI Autocomplete v content skripte

JQery UI Autocomplete je prispôsobené pre moje potreby načítavania písaneho textu používateľom a zobrazovania generovanej sekvencie v objekte SequencePredictor, ktorý je vytvorený v backgrounde.

Vytvára sa spustením content skriptu a vie pracovať s HTML elementami popísanými v sekcii 5.4. Pred jeho vytvorením si content skript načíta z chrome storage vlastnosti pre formátovanie generovaných sekvencií.

V content skripte je pomocná funkcia measure_text, ktorá je prevzatá z internetu[4]. Táto funkcia je na zistenie veľkosti textu pre správne vyrátanie veľkosti pozadia pre generovanú sekvenciu.

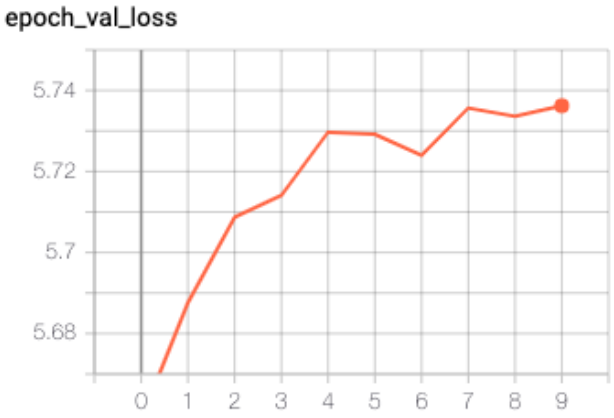
6.3 Priebeh trénovania

Finálne trénovanie prebiehalo na platforme GCP cez vytvorenú inštanciu Jupyter Notebooku [5] v termináli za pomoci prenajatého GPU Nvidia T4 [6].

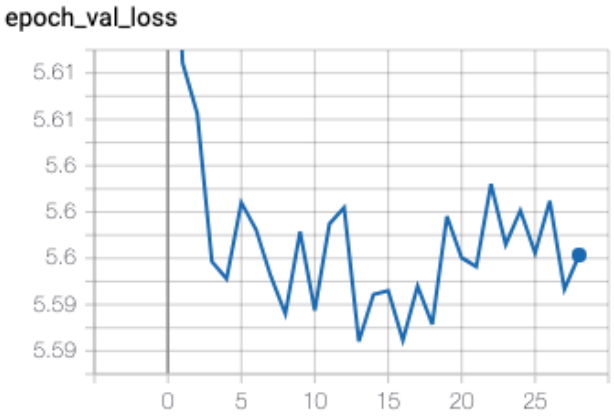
Pôvodne bolo použitých 10 epôoch s 30-timi percentami dátovej sady gigaword, čo je približne 1,300,000 článkov. Vývoj validačnej stratovej funkcie pre toto trénovanie je na obrázku 6.1 (TesorBoard z nejakého dôvodu nezobrazil priebeh medzi 1. a 2. epochou). Na obrázku je vidieť nárast validačnej stratovej funkcie, čo znamená, že trénovanie neprebiehalo dobre. Pravdepodobne to bolo kôli veľkému množstvu trénovacích dát a malému počtu neurónov v skrytých GRU vrstvách.

Z dôvodu nárastu stratových funkcii a finančnej náročnosti trénovania bola zmenšená trénovacia datová sada na 5 percent a koeficient v dropout vrstvách z 0.2 na 0.1. 5 percent trénovacej sady gigaword je stále veľké množstvo dát. Je to približne 200,00 článkov. Taktiež bol zmenený počet trénovacích epôch z 10 na 29. Obrázok 6.2 je validačná stratová funkcie pre tento prípad. Z vývoja validačnej stratovej funkcie je zrejmé, že to pomohlo.

[2] https://www.measurethat.net/Benchmarks/Show/7018/0/string-hashcode
[3] https://stackoverflow.com/questions/41654006/numpy-random-choice-in-javascript
[4] https://stackoverflow.com/questions/118241/calculate-text-width-with-javascript
[5] https://jupyter.org/
[6] https://www.nvidia.com/en-us/data-center/tesla-t4/



Obr. 6.1: Vývoj validačnej stratovej funkcie počas finálneho trénovania 1.

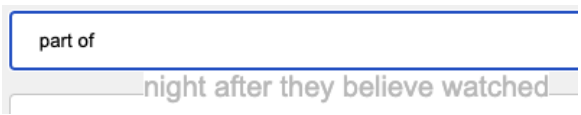


Obr. 6.2: Vývoj validačnej stratovej funkcie počas finálneho trénovania 2.

Vizualizácia oboch príadov trénovania sa dá pozrieť cez TesorBoard. Postup pre spustenie je v prílohách.

6.4 Generovanie sekvencie a výber dekódovacieho algoritmu

Dĺžka generovanej sekvencie je defaultne určená na maximálne 5 slov. Cez option page komponentu rozšírenia si to môže používateľ prispôsobiť od jedného slova do 10. 10 slov je zvolených ako maximum, pretože podľa článku [1] je priemerná dĺžka anglickej vety 14 znakov. Za predpokladu, že používateľ musí napísať nejaký text pre vygenerovanie sekvencie je 10 slov vhodná maximálna dĺžka.



Obr. 6.3: Príklad vygenerovanej sekvencie.

Na obrázku 6.3 je príklad vygenerovanej a zobrazenej textovej sekvencie používateľovi. Jej doplnenie sa potvrdzuje klávesou TAB alebo stlačením myšky.

Výber dekódovacieho algoritmu pre generovanie sekvencie

Pre generovanie sekvencie pomocou neurónovej siete je dôležitý výber dekódovacieho algoritmu. Rozhodovanie prebiehalo medzi jednoduchým samplingom a greedy algoritmom (sekcii 3.3). Z pokusov, ktoré sú pre greedy algoritmus reprezentované obrázkom 6.5 a pre sampling obrázkom 6.4 vyplýva, že sampling je lepší ako greedy. Text generovaný samplingom sa viac približuje reálnemu textu písaného používateľom, nemá tendenciu opakovať sa a častejšie je zobrazovaný používateľovi.

Vstupný text	Tokeny generované samplingom	Pravdepodobnosti v %	Vygenerovaný text	Zobrazí sa
"What do you want to "	["be ", "even ", "with ", "the ", "past "]	[32.837, 22.642, 30.131, 38.79, 42.946]	"be even with the past "	NIE
"we are living "	[" ", "house ", "about ", "nine ", "people", "."]	[35.139, 47.628, 81.069, 28.356, 86.58, 72.024]	" house about nine people."	ANO
"new mobile "	[" ", "on ", "tuesday", "."]	[12.222, 66.995, 60.791, 56.121]	" on tuesday."	NIE
"Where are you "	[" ", "millions ", "of ", "border", ", ", "spar", "king "]	[56.888, 46.689, 79.496, 7.797, 83.701, 39.028, 32.515]	" millions of border, sparking "	NIE
"life is like "	[" ", "refugees ", "in ", "the ", "cab", "ine", "nt "]	[67.767, 11.378, 33.352, 88.846, 29.445, 56.467, 70.428]	" refugees in the cabinent "	ANO
"he would "	[" ", "works ", "last ", "bid", "'", "t ", "scored "]	[96.692, 61.466, 85.508, 81.939, 94.214, 62.441, 86.213]	" works last bid't scored "	ANO
"The price of oil "	[" ", "unk", "now", ", ", "the ", "green ", "light "]	[12.41, 94.961, 16.356, 10.223, 1.448, 81.157, 38.195]	" unknow, the green light "	NIE

Obr. 6.4: Príklady generovania sekvencie pomocou samplingu.

Vstupný text	Tokeny generované greedy	Pravdepodobnosti v %	Vygenerovaný text	Zobrazí sa
"What do you want to "	[" ", "hours ", "after ", "the ", "u", "."]	[63.069, 7.103, 35.783, 23.097, 5.31, 94.161]	" hours after the u."	NIE
"we are living "	[". "]	[34.122]	"."	NIE
"new mobile "	["to ", "the ", "u", "."]	[9.054, 17.99, 7.575, 96.009]	"to the u."	NIE
"Where are you "	[" ", "compared ", "with ", "the ", "u", "."]	[82.053, 9.553, 34.891, 35.328, 6.68, 95.532]	" compared with the u."	NIE
"life is like "	[" ", "compared ", "with ", "the ", "u", "."]	[68.139, 5.754, 26.177, 37.643, 6.205, 96.073]	" compared with the u."	NIE
"he would "	[" ", "hours ", "after ", "the ", "u", "."]	[71.477, 6.87, 29.56, 23.094, 5.984, 93.975]	" hours after the u."	NIE
"The price of oil "	[" ", "compared ", "with ", "the ", "u", "."]	[69.396, 5.99, 24.231, 36.453, 6.205, 94.944]	" compared with the u."	NIE

Obr. 6.5: Príklady generovanie sekvencie pomocou greedy algoritmu.

6.5 Problémy pri implementácii a jej publikácii

- **Trénovanie neurónovej siete.** Zo začiatku bola neurónovú sieť trénovaná na platforme Google Colaboratory [7]. Ponúka výhodu možnosti trénovania pomocou GPU. Zásadná nevýhoda je session timeout a maximálna doba trénovania. Session timeout je 90 min, to znamená, že po maximálne 90 min (niekedy aj skôr) bez aktivity je trénovanie automaticky ukončené. Maximálna doba trénovania je 12 hod v klasickej verzii a 24 hod v platenej Pro verzii. Tento problém bol vyriešený pomocou Google Cloud AI platformy[8] vytvorením Jupyter notebooku s prenájmom 1 GPU Nvidia Tesla T4. Vytvorený Jupyter notebook z nejakého dôvodu tiež pracoval chybne, a tak sa cezeň otvoril terminál, v ktorom sa trénovala neurónová sieť.

- **Konvertovanie natrénovaného modelu** z TensorFlow do TensorFlow.js a jeho následná deserializácia v TensorFlow 2 a vyššie má chybu v deserializácii GRU vrstvy. Podarilo sa to vyriešiť použitím TensorFlow 1.15, ktorý túto chybu nemá. Celý popis problému a jeho riešenie je v GitHub issue [9].

- **Umiestnenie jazykového modelu a slovníka na serveri**. Jazykový model je spolu so slovníkom umiestnený v GCP storage buckete, v ktorom je nastavený prístup na čítanie pre používateľa s odkazom na jazykový model alebo na slovník. Ich umiestnenie na serveri je z dôvodu, nenájdenia spôsob, akým by sa dal načítať natrénovaný model priamo z adresára umiestnenom v balíčku rozšírenia pre prehliadač Google Chrome pomocou TensorFlow.js.

- **Správne pochopenie funkcionality komponentov rozšírenia** a ich komunikácia boli kľúčové pre riešenie. Pôvodne bolo získavanie vstupu od používateľa a generovanie sekvencie v content skripte, čo vyžadovalo pri každom načítaní webovej stránky stiahnutie neurónovej siete a slovníka na kódovanie tokenov, následne bolo generovanie sekvencie pomocou neurónovej siete a slovník presunuté do background skriptu. To spôsobilo, že model bol sťahovaný len pri otvorení prehliadača. Na záver bolo pridané ukladanie stiahnutého modelu do IndexedDB a slovníka do LocalStorage. Čo spôsobilo, že jazykový model a slovník sa stiahnú len po nainštalovaní SPT. Pri ďalších spusteniach SPT sa neurónová sieť načítava z IndexedDB a slovník z LocalStorage.

- **Publikovanie výsledného rozšírenia na Chrome Web Store**. Chrome Web Store vyžaduje vytvorenie účtu pre možnosť publikovania rozšírení. Pre samotnú publikáciu rozšírenia vyžaduje prezentačné obrázky, popis produktu, rôzne bezpečnostné a iné obmedzenia. Pred samotnou publikáciou je rozšírenie vždy poslané na kontrolu do Google, ktorý ho môže zamietnuť a vyžadovať zmeny v rozšírení. Pre odpublikovanie SPT na Chrome Web Store bolo potrebné niekoľko takýchto zmien zapracovať.

[7] https://colab.research.google.com/
[8] https://cloud.google.com/gcp
[9] https://github.com/tensorflow/tfjs/issues/2442

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť systém na napovedanie textových sekvencií pre používateľov so zámerom zjednodušenia písania častých slovných obratov. Systém mal bežať na strane používateľa v prehliadači Google Chrome.

Výsledný systém, ktorý sa volá Sequence Prediction Tool, bol vytvorený pomocou technológie pre rozšírenie prehliadača Google Chrome. Sequence Prediction Tool získava písaný text používateľom na rôznych internetových stránkach. Získaný text sa používa na generovanie textových sekvencií pomocou rekurentnej neurónovej siete a samplovacieho algoritmu na generovanie sekvencií. Vygenerovaná sekvencia sa musí ešte skontrolovať. Kontrola je porovnanie logaritmického súčtu pravdepodobností tokenov tvoriacich generovanú sekvenciu normalizovaného počtom tokenov vo vygenerovanej sekvencii s konštantnou hodnotou spoľahlivosti. Po jej presiahnutí je generovaná sekvencia zobrazená používateľovi s možnosťou doplnenia po jej potvrdení. Výsledné riešenie je publikované na Chrome Web Store, z ktorého si ho môže stiahnuť každý používateľ s prehliadačom Google Chrome. Na záver bola vytvorená demonštračná internetová stránka na vyskúšanie si výsledného riešenia.

Vďaka tejto práci som získal vedomosti o tom čo sú neurónové siete, ako fungujú a ako sa dajú používať v spolupráci s inými technológiami - v mojom prípade s rozšírením pre prehliadač Google Chrome. Ďalej som získal prehľad o pokroku v oblasti spracovania prirodzeného jazyka a komplexnosti existujúcich modelov na generovanie prirodzeného jazyka.

V rámci ďalšej práce by bolo vhodné skúsiť zlepšiť generovanie sekvencií a vytvoriť personalizáciu pre konkrétnych používateľov.

Pre zlepšenie generovania sekvencií navrhujem vyskúšať architektúru Sequence to sequence a architektúru jednoduchého Transformera. Pre ďalšie zlepšenie generovania sekvencie by bolo vhodné skúsiť presunúť neurónovú sieť s generovaním sekvencie na stranu servera a spraviť z toho službu, ktorá by komunikovala s rozšírením u používateľov. To by umožnilo použitie komplexnejších modelov založených na Transformerovi, ktoré sú state of the art v súčasnosti (jún 2020).

Pre personalizáciu navrhujem vytvoriť sekundárny jazykový model pre každého používateľa, ktorý by sa natrénoval na užívateľom písaných textoch a pravidelne by sa pretrénovaval.

Literatúra

[1] Acropolitan, T. *Sentence Length Has Declined 75% in the Past 500 Years* [online]. August 2017 [cit. 2020-05-10]. Dostupné z: https://medium.com/@theacropolitan/sentence-length-has-declined-75-in-the-past-500-years-2e40f80f589f.

[2] Ahire, J. B. *The Artificial Neural Networks Handbook: Part 4* [online]. Medium, november 2018 [cit. 2020-03-20]. Dostupné z: https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e.

[3] Brownlee, J. *How to Choose Loss Functions When Training Deep Learning Neural Networks* [online]. machinelearningmastery.com, január 2019 [cit. 2020-04-10]. Dostupné z: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[4] Brownlee, J. *3 Ways to Encode Categorical Variables for Deep Learning* [online]. 2020 [cit. 2020-03-30]. Dostupné z: https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/.

[5] Chen, M. X., Lee, B. N., Bansal, G., Cao, Y., Zhang, S. et al. Gmail Smart Compose: Real-Time Assisted Writing. *CoRR*. 2019, abs/1906.00080. Dostupné z: http://arxiv.org/abs/1906.00080.

[6] Chrome, G. *Extend the Browser* [online]. [cit. 2020-04-01]. Dostupné z: https://developer.chrome.com/extensions/overview.

[7] Chung, J., Gulcehre, C., Cho, K. a Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[8] Devlin, J., Chang, M., Lee, K. a Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. 2018, abs/1810.04805. Dostupné z: http://arxiv.org/abs/1810.04805.

[9] Feurer, M. a Hutter, F. *Hyperparameter Optimization*. Springer, 2018. 3-33 s. ISBN 978-1789132335.

[10] Goldberg, Y. *Neural Network Methods for Natural Language Processing*. Morgan Claypool Publishers, 2017. ISBN 978-1627052986.

[11] Graff, D., Kong, J., Chen, K. a Maeda, K. English gigaword. *Linguistic Data Consortium, Philadelphia*. 2003, roč. 4, č. 1, s. 34.

[12] Hochreiter, S. a Schmidhuber, J. Long short-term memory. *Neural computation*. MIT Press. 1997, roč. 9, č. 8, s. 1735–1780.

[13] Jaswal, A. S. *Byte Pair Encoding — The Dark Horse of Modern NLP* [online]. [cit. 2020-03-29]. Dostupné z: https://towardsdatascience.com/byte-pair-encoding-the-dark-horse-of-modern-nlp-eb36c7df4f10.

[14] Kannan, A., Kurach, K., Ravi, S., Kaufmann, T., Tomkins, A. et al. Smart Reply: Automated Response Suggestion for Email. *CoRR*. 2016, abs/1606.04870. Dostupné z: http://arxiv.org/abs/1606.04870.

[15] Karpathy, A. *The Unreasonable Effectiveness of Recurrent Neural Networks* [online]. Máj 2015 [cit. 2020-03-26]. Dostupné z: https://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[16] Kostadinov, S. *Recurrent Neural Networks with Python Quick Start Guide: Sequential learning and language modeling with TensorFlow*. Packt, 2018. ISBN 978-1789132335.

[17] Kostadinov, S. *Understanding Encoder-Decoder Sequence to Sequence Model* [online]. Február 2019 [cit. 2020-06-30]. Dostupné z: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346.

[18] Mack, D. *How to pick the best learning rate for your machine learning project* [online]. medium.com, apríl 2018 [cit. 2020-04-10]. Dostupné z: https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2.

[19] Mikolov, T., Chen, K., Corrado, G. S. a Dean, J. *Efficient Estimation of Word Representations in Vector Space*. 2013. Dostupné z: http://arxiv.org/abs/1301.3781.

[20] Nielsen, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.

[21] Pennington, J., Socher, R. a Manning, C. Glove: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Október 2014, s. 1532–1543. Dostupné z: https://www.aclweb.org/anthology/D14-1162.

[22] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. et al. Language Models are Unsupervised Multitask Learners. 2018. Dostupné z: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[23] Rashid, T. *Make Your Own Neural Network*. CreateSpace Independent Publishing Platform, 2016. ISBN 978-1530826605.

[24] Rosset, C. *Turing-NLG: A 17-billion-parameter language model by Microsoft* [online]. Február 2020 [cit. 2020-06-30]. Dostupné z: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

[25] See, A. *Lecture 15: Natural Language Generation* [online]. Január 2020 [cit. 2020-06-30]. Dostupné z: https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture15-nlg.pdf.

[26] Shoeybi, M., Patwary, M. A., Puri, R., LeGresley, P., Casper, J. et al. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *ArXiv*. 2019, abs/1909.08053.

[27] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. Jún 2014, roč. 15, s. 1929–1958.

[28] Sutskever, I., Vinyals, O. a Le, Q. V. Sequence to Sequence Learning with Neural Networks. *CoRR*. 2014, abs/1409.3215. Dostupné z: http://arxiv.org/abs/1409.3215.

[29] Tiwari, P. *Recurrent Neural Network* [online]. MyTechWorld, december 2018 [cit. 2020-03-27]. Dostupné z: https://tekworld.org/2018/12/28/day-47-100-days-mlcode-recurrent-neural-network/#page-content.

[30] Tomáš Mikolov , Ilya Sutskever , Anoop Deoras , Hai-Son Le , Stefan Kombrink , Jan Černocký. SUBWORD LANGUAGE MODELING WITH NEURAL NETWORKS. 2011.

[31] TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*. Október 1950, LIX, č. 236, s. 433–460. Dostupné z: https://doi.org/10.1093/mind/LIX.236.433. ISSN 0026-4423.

[32] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention Is All You Need. *CoRR*. 2017, abs/1706.03762. Dostupné z: http://arxiv.org/abs/1706.03762.

[33] Visual Recognition, C. C. N. N. for. *Modeling one neuron* [online]. 2020 [cit. 2020-03-20]. Dostupné z: https://cs231n.github.io/neural-networks-1/.

[34] Visual Recognition, C. C. N. N. for. *Neural Network architectures* [online]. 2020 [cit. 2020-03-20]. Dostupné z: https://cs231n.github.io/neural-networks-1/.

[35] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv preprint arXiv:1609.08144*. 2016.

Príloha A

Obsah priloženého CD

```
app_root/
├── docs/ -> text práce v pdf so zdrojovými kódmi pre LaTeX
├── neural_network/
│   ├── models/ -> produkčne nasadený model a slovník pre TensorFlow.js
│   ├── logs/ -> logy z finálneho trénovania
│   ├── dataset.py -> trieda Dataset
│   ├── rnn.py -> trieda RNN
│   ├── train.py -> vstupný bod pre trénovanie
│   └── requirements.txt -> zoznam balíčkov na inštaláciu
└── extension/
    ├── demo_web_page/ -> demo stránka hostovaná na www.sequencepredictiontool.com
    ├── external_modules/ -> externé moduly pre tf.js a JQery
    ├── background.js -> background komponenta rozšírenia
    ├── communication.js -> komunikačný kanál
    ├── content.js -> content skript s JQery UI Autocomplete
    ├── data.js -> pomocné funkcie pre slovník a neurónovú sieť
    ├── manifest.json -> vstupný bod technológie rozšírenia
    ├── options.html a options.js -> nastavovanie parametrov generovanej sekvencie
    ├── sequence_predictor.js -> trieda SequencePredictor
    └── sub_word_tokenizer.js -> trieda SubWordTokenizer
```

Príloha B

Používanie výslednej aplikácie

Výsledná aplikácia sa dá používať troma spôsobmi:

- **Inštalácia z Chrome Web Store.** Pre tento prípad je potrebné mať internetový prehliadač Google Chrome a stiahnúť si SPT z Chrome Web Store[1].
- **Upravovanie neurónovej siete alebo rozšírenia.** Pre lokálne vyskúšanie si rozšírenia je dôležíté mať nastavený prehliadač Google Chrome v Developer móde (More Tools -> Extension -> Developer mode). Rozšírenia sa lokálne inštalujú pomocou More Tools -> Extension -> Load unpacked.
  - **Upravovanie neurónovej siete.** Vyžaduje mať nainštalovaný Python3.6 a nainštalovanie si balíčkov z requirements.txt. Po natrénovaní je potrebné uložiť jazykový model so slovníkom na server s prístupom cez odkaz a zmeniť URL adresu pre jazykový model a slovník v background.js.
  - **Upravovanie rozšírenia.** Nemá žiadne špeciálne požiadavky okrem Developer módu.

[1] https://chrome.google.com/webstore/detail/sequence-prediction-tool/biblhlpmjanngadiholiloifmnmmfohk

Príloha C

Spustenie logov z trénovania v TensorBoard

Pre spustenie logov z finálneho trénovania je požadovaný **Python3.6** a **tensorboard** balíček. Tensorboard balíček je zahrnutý v **requirements.txt**. Po nainštalovaní requirements.txt sa spúšťa TensorBoard s logmi z root adresára aplikácie pomocou príkazu:

```
tensorboard --logdir neural_network/logs/
```