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

UMĚLÝ BÁSNÍK

ARTIFICIAL POET

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL BANČÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. KAREL BENEŠ**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Bančák Michal**
Obor: Informační technologie
Téma: **Umělý básník**
Artificial Poet
Kategorie: Zpracování řeči a přirozeného jazyka

Pokyny:

1. Seznamte se rekurentními neuronovými sítěmi jako nástrojem pro modelování sekvencí.
2. Sestavte korpus básní jako datovou sadu a natrénujte na něm síť
3. Navrhněte a implementujte postup vzorkování z natrénovaného modelu
4. Navrhněte a implementujte sadu heuristik pro zvýšení kvality generovaných básní
5. Ověřte kvalitu generovaných básní

Literatura:

- podle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beneš Karel, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L. S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Dokument predstavuje prácu na automatickom generovaní poézie, pomocou Long Short-Term Memory rekurentnej neurónovej siete. Cieľom práce je vytvoriť aplikáciu, ktorá imituje písanie básní. Jedná sa o jazykové modelovanie na úrovni znakov v slovenskom jazyku. Model neurónovej siete použitý v práci sa skladá z troch vrstiev LSTM so 400 skrytými jednotkami. K tejto práci bola taktiež vytvorená zbierka básní v slovenskom jazyku vo veľkosti 900k znakov. Výsledkom práce je generovanie textu, ktorý má prvky básne. Dosahovaná presnosť generovania je 41.85 %.

Abstract

The paper presents a work on automatic poetry generation using the Long Short-Term Memory recurrent neural network. The aim of this work is to create an application that imitates the writing of poems. This is a character-level language modeling in the Slovak language. The neural network model used in the work consists of three layers of LSTM, with 400 hidden units. A collection of poems in the Slovak language with a size of $900k$ characters was also created for this work. . The final model is generating text that has poem elements. Achieved accuracy of generation is 41.85 %.

Kľúčové slová

Neurónová sieť, Long Short-Term Memory, Automatické generovanie básní, Jazykové modelovanie na znakovej úrovni

Keywords

Neural Network, Long Short-Term Memory, Automatic Poem Generation, Character-Level Language modeling

Citácia

BANČÁK, Michal. *Umělý básník*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Karel Beneš

Umělý básník

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne po vedením pána Ing. Karla Beneša. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Bančák
17. mája 2017

Poďakovanie

Rád by som poďakoval pánovi Ing. Karlovi Benešovi za trpezlivosť a veľa užitočných rád a pripomienok k mojej práci.

Obsah

Kapitola 1

Úvod

Poézia je odpradávna súčasťou ľudskej kultúry. Písanie básní slúžilo napr. ako dar pre blízkych ľudí, alebo ako text pre mnohé hudobné žánre. Avšak schopnosťou písať básne nedisponujú všetci ľudia. Pre takých je možné aspoň čiastočne problém odstrániť vďaka strojovému učeniu.

Automatické generovanie básní bolo už známou témou výskumov. Väčšina využívala šablóny pre vytvorenie básne podľa sady pravidiel (napr. rým, frekvencia slova) v kombinácii s lexikografickými zdrojmi [12].

Druhý rad výskumov používal genetické algoritmy na generovanie básní [9]. Tieto výskumy sa riadili tým, že na základnej úrovni musí každá báseň spĺňať podmienky, a to gramatiku (každá báseň musí byť gramaticky bezchybná), zmysluplnosť (báseň vyjadruje nejakú správu, zväčša pocit autora, ktorý niečo interpretuje) a poetickosť (musí byť jasné, že sa jedná o poéziu, jasne odlíšiteľnú od jednoduchého textu). Vo výsledku model vygeneroval viacero básní, z ktorých vybraná bola tá, ktorá tieto podmienky spĺňa všetky.

Tretí rad výskumov vychádza zo štatistického strojového prekladu a existujúcich text-generujúcich aplikácií [13]. Napríklad, pracujú viac s dopytom ľudí. Ich model žiada na vstup zopár slov a načíta najviac relevantné básne zo zbierky. Načítané básne sú rozdelené na základné slová, ktoré sú zoskupené do zhlukov. Básne sú generované iteratívnym vyberaním slov zo zhlukov podľa určitých pravidiel.

Moje riešenie je podobné v nutnosti mať vstupné dáta vo forme básní, ktoré slúžia ako trénovacie. Pravidlá básne, ako napríklad rým, verše a slohy, alebo správna gramatika sú implicitne pri takomto trénovaní zaobstarané sieťou. Samotné generovanie básne je pre sieť jednoduché. Na generovanie je nutná určitá interakcia s užívateľom: Po užívateľovi program vyžaduje vstupný text, napr. jeden verš. Na základe tohto textu, vyráta pravdepodobnosť aký znak by mal nasledovať. Nie je teda nutné pri každom generovaní kontrolovať či text spĺňa pravidlá poézie, hľadať podobnosť zadaného slova v zbierke básní, stačí sieť raz natrénovať.

V tejto práci sa využívajú už existujúce básne od známych spisovateľov na zostavenie a natrénovanie neurónovej siete pre jazykové modelovanie na úrovni znakov. Zo slovenských básní bola vytvorená zbierku, ktorá obsahuje diela rôznych autorov, vďaka čomu sa neurónová sieť nebude upínať na štýl jedného konkrétneho spisovateľa. Tým je vytvorená báseň jedinečná a použiteľná pre kohokoľvek.

Kapitola 2

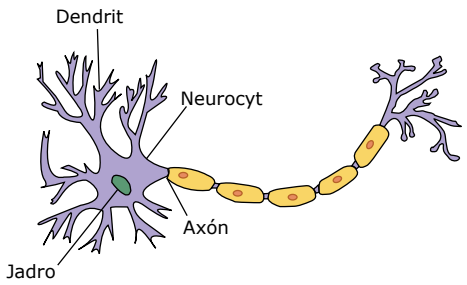
Neurónová sieť pre modelovanie textu

Neurónová sieť (NN) je výpočtový model používaný v umelej inteligencii, ktorý vznikol na základe biologickej predlohy. Je stvorená prepojením jednotlivých neurónov. NN je možné trénovať pre zdokonalenie jej vlastností. Trénovaním NN je napríklad možné dosiahnuť, že bude schopná klasifikovať prvky do dvoch alebo viacerých tried.

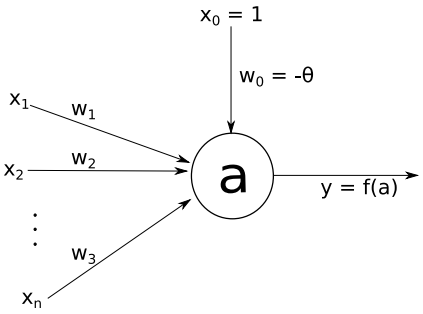
Pri používaní NN pre modelovanie textu je požadované, aby rozpoznávala jednotlivé písmená a ich sled tak, aby po natrénovaní bola schopná takýto sled písmen (slová) tvoriť a nimi tvoriť zmysluplné vety. Všetky tieto vlastnosti a schopnosti NN sú dané práve jej štruktúrou a zložením.

2.1 Neurón

Základným prvkom NN je formálny neurón, ktorého predlohou bol biologický neurón. Formálny neurón má niekoľko vstupov x_1, x_2, x_3, ... , x_n (obrázok 2.1b), ktoré predstavujú dendrity v biologickej predlohe (obrázok 2.1a). V biologickom neuróne slúžia dendrity ako vstupy. Každý vstup je ohodnotený príslušnou váhou w_1, w_2, w_3, ... , w_n.



(a) Biologický neurón[1]



(b) Formálny neurón vzniknutý na základe vlastností biologického neurónu

Obr. 2.1: Porovnanie formálneho a biologického neurónov. Vstupy x_i predstavujú dendrity. V jadre sa sústreďuje vnútorný potenciál, ktorý je týmito vstupmi určený. Vo formálnom neuróne je zastúpený hodnotou a.

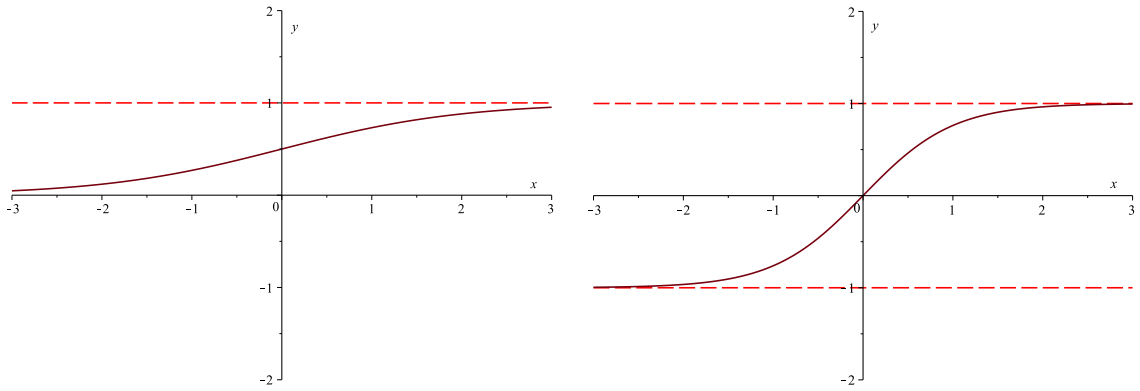
Ohodnotené vstupy vo formálnom neuróne určujú vnútorný potenciál a. Vnútorný potenciál je hodnota, ktorá spolu s prahovou hodnotou θ určujú výstup. Ak je vnútorný potenciál menší než prahová hodnota, neurón nereaguje a na výstupe je tak 0. Vnútorný potenciál je daný rovnicou:

$$a = \vec{w_i}^T \vec{x_i} + \theta \tag{2.1}$$

Výstup y je daný prenosovou (aktivačnou) funkciou.

$$y = f(a) \tag{2.2}$$

Existuje viacero typov prenosových funkcií, pre moje potreby v tejto práci sú dôležité sigmoidné funkcie logistická sigmoida 2.2a a hyperbolický tangens 2.2b:



(a) Logistická sigmoida

(b) Hyperbolický tangens

Obr. 2.2: Bežne používané prenosové (aktivačné) funkcie.

2.2 Viacvrstvová neurónová sieť

Formálne neuróny sa zväčša v NN nachádzajú vo vrstvách. Vrstva je časť NN, ktorej všetky neuróny sú pospájané s každým neurónom predchádzajúcej vrstvy. V základnom princípe jednoduchej NN sú dve vrstvy, vstupná a výstupná. Takáto sieť má pomerne jednoduchú štruktúru a zložitejšie výpočty sú pre ňu ťažko dosiahnuteľné. Pre získanie zložitejšej NN, ktorá je schopná dosahovať lepších výsledkov v zložitejších výpočtoch, je možné využiť viacvrstvovú neurónovú sieť (obrázok 2.3). Takáto sieť obsahuje okrem vstupnej a výstupnej vrstvy aj skryté vrstvy. Vrstva viacvrstvovej neurónovej siete sa riadi rovnicou:

$$\vec{h}^l = f(\mathrm{W}\vec{h}^{l-1} + \vec{b}) \tag{2.3}$$

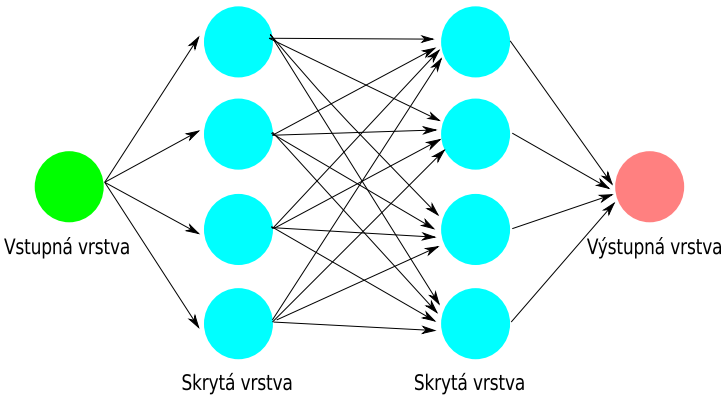
kde $\vec{h}^{l-1}$ je vstup (výstup predchádzajúcej vrstvy), $\vec{h}^l$ je výstup, W je matica váh, ktorá ohodnocuje vstupy a $\vec{b}$ je bias vektor.

V matematickej teórii umelých neurónových sietí, teorém univerzálnych aproximácií stanovuje, že sieť s jednou skrytou vrstvou obsahujúcou konečný počet neurónov môže aproximovať spojité funkcie na kompaktných podmnožinách R^n, za miernych predpokladov

[1] Obrázok 2.1a prevzatý z https://upload.wikimedia.org/wikipedia/commons/thumb/b/b5/Neuron.svg/300px-Neuron.svg.png

pre aktivačnú funkciu. Teorém teda stanovuje, že jednoduchá neurónová sieť môže reprezentovať širokú škálu funkcii, ak je správne zostavená. Jedna z prvých verzií [4] teorému bola predstavená pre sigmoidnú aktivačnú funkciu.

V roku 1991 však bolo ukázané, že nezáleží na špecifickej voľbe aktivačnej funkcie [6], architektúra viacvrstvovej NN sama o sebe má potenciál byť univerzálnym aproximátorom.



Obr. 2.3: Viacvrstvová neurónová sieť, ktorá sa skladá zo vstupnej a výstupnej vrstvy a dvoch skrytých vrstiev, kde neuróny každej vrstvy sú prepojené s každým neurónom nasledujúcej vrstvy. Vstupy do takejto siete sú ohodnotené váhami a predávané do skrytých vrstiev kde dochádza k výpočtom. Následne z poslednej skrytej vrstvy sú odovzdané výstupy do výstupnej vrstvy.

2.3 Dopredná sieť Nn-gram

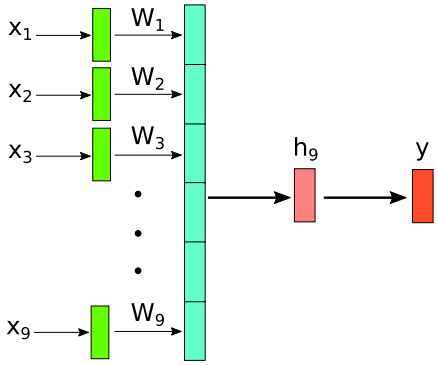
Jedným typom siete pre jazykové modelovanie, ktorá sa používala z dôvodu menšej náročnosti na výpočetnú techniku je dopredná sieť Nn-gram (Neural n-gram, obrázok 2.4). Nn-gram je jednoduchá sieť, ktorá obsahuje jednoduchú skrytú vrstvu s prepojením neurónov. Keďže sa jedná o jazykové modelovanie, je nutné aby znaky mali na sebe určitú závislosť napríklad v slovách, alebo slová navzájom vo vetách v prípade modelovania na úrovni slov. Preto vstupom do tejto siete je sekvencia o dĺžke n po sebe idúcich prvkov (znakov, slov). Sieť sa na takýchto dátach trénuje a učí sa ako sú jednotlivé prvky v sekvenciách usporiadané. Po natrénovaní by mala byť potom schopná tvoriť nový text.

Na výstup doprednej siete je používaná funkcia softmax, nazývaná tiež normalizovaná exponenciálna funkcia. Je daná rovnicou:

$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.4}$$

Softmax stláča hodnoty do intervalu $[0, 1]$ a taktiež aby ich súčet bol 1. Tým normalizuje výstup, s ktorým je následne možné pracovať ako s pravdepodobnosťami. Výstupom siete je teda pravdepodobnostné pole obsahujúce pravdepodobnosť pre všetky prvky.

Tento typ siete na túto prácu nie je vhodný, pretože Nn-gram je primárne stvorený pre modelovanie na úrovni slov a na úrovni znakov nedosahuje tak efektívne výsledky ako rekurentné neurónové siete [7]. Toto tvrdenie bolo overené aj experimentom (5.1).

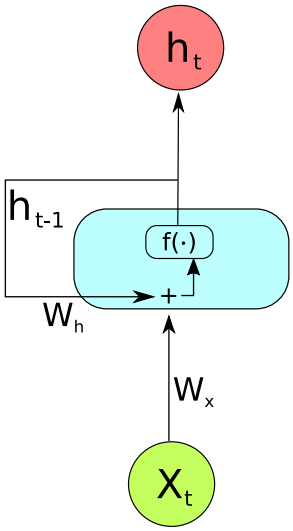


Obr. 2.4: Dopredná sieť Nn-gram. Vstupom je určitý počet znakov, v tomto prípade 9, každý vstup je ohodnotený príslušnou váhou. So vstupných znakov si sieť trénuje postupnosti znakov.

2.4 Rekurentné neurónové siete

Pre tradičná NN, hoci viacvrstvovú, je náročné pracovať so závislosťami znakov, ktoré k modelovaniu textu patria. Je sice možné, aby si klasická NN dokázala nejaké závislosti natrénovať, avšak je to veľmi časovo náročné, preto je vhodné využiť rekurentné neurónové siete (RNN).

RNN je sieť, ktorá sa vyznačuje tým, že si uchováva predchádzajúce informácie, čo znamená, že vystúp nie je ovplyvnený len príslušným vstupom, ale aj predchádzajúcimi. Každý výstup teda ovplyvňuje ďalší, čiže závislosť predchádzajúcich výpočtov je zachovaná. Jednoduchšie povedané RNN obsahuje cyklus, ktorý každému výpočtu dáva na vstup aj históriu predchádzajúcich dát (2.5). Vstupom je vektor $\vec{x}$ a výstupom vektor $\vec{h}$, kde vektor $\vec{h}$ je zároveň vložený do histórie. Tým sú v RNN vyrátané výstupy aj na základe histórie predchádzajúcich stavov.



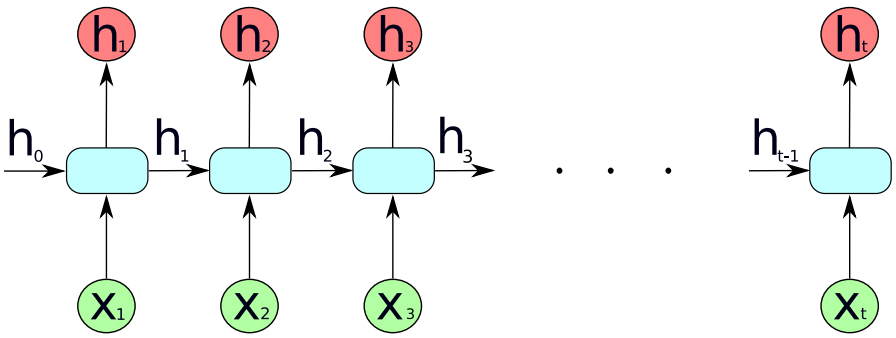
Obr. 2.5: Rekurentná neuronová sieť (RNN). Výpočet výstupu $\vec{h}_t$ je ovplyvnený nie len vstupom $\vec{x}_t$, ale aj minulým výstupom $\vec{h}_{t-1}$.

Výstup vrstvy RNN je daný rovnicou:

$$\vec{h}_t = f(\mathrm{W}_x\vec{x}_t + \mathrm{W}_h\vec{h}_{t-1} + \vec{b}) \tag{2.5}$$

kde $\vec{x}_t$ je vstup, $\vec{h}_{t-1}$ je predchádzajúci výstup, W_x je matica váh, ktorá ohodnocuje vstupy, W_h je matica váh, ktorá ohodnocuje predchádzajúce výstupy.

Pri pohľade na RNN je zrejmé, že história sa cyklicky aktualizuje a prikladá na vstup ďalším výpočtom. Ak sa tento cyklus rozbalí v čase, je možné RNN zobraziť, ako spojenie viacerých častí prepojených práve zotrvávajúcou históriou (obrázok 2.6).



Obr. 2.6: Rozložená RNN. Závislosť vstupov $\vec{x}_t$ a výstupov $\vec{h}_t$ je zobrazená, ako prepojenie po sebe idúcich výpočtov, ktoré si odovzdávajú reprezentáciu histórie. Je možné vidieť, že napr. výstup $\vec{h}_2$ môže byť ovplyvnený nie len vstupom $\vec{x}_2$, ale aj vstupom $\vec{x}_1$. Taktiež je možné vidieť, že je to veľmi hlboká dopredná sieť Nn-gram.

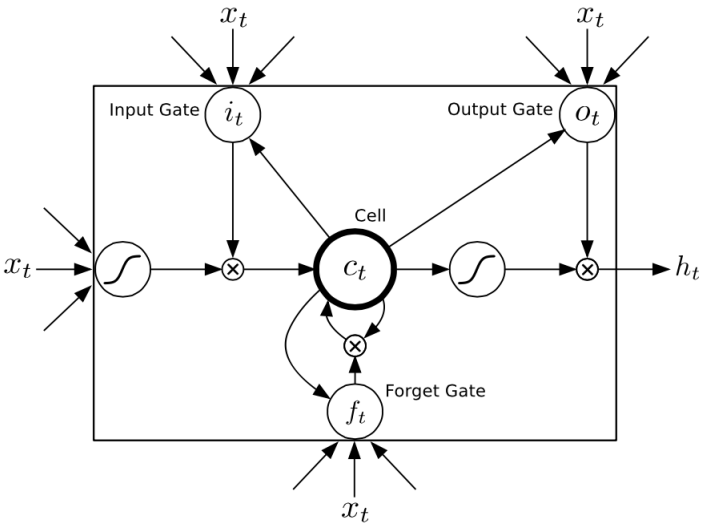
Pri tomto zobrazení RNN je možné vidieť ako sa táto história prenáša z jedného výpočtu na druhý. Problém tohto typu NN je ten, že je veľmi náročné dostatočne natrénovať čo i

len jednoduchú RNN [10]. Pri pohľade na obrázok (2.6), vstup $\vec{x}_1$ nemusí ovplyvniť výstup $\vec{h}_{21}$ a tým pádom dôležitá závislosť môže byť sieťou zahodená a dochádza k nežiadanému vyhodnoteniu výstupu $\vec{h}_{21}$. Existuje veľa typov verzií RNN stavaných na dlhé závislosti, ktoré taktiež môžu závislosť odstrániť, s tým rozdielom, že majú mechanizmy na udržanie informácie po dlhú dobu.

2.4.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) [5] je špeciálna verzia RNN stvorená na pracú s dlhými závislosťami v dátach.

Štruktúra LSTM je oveľa zložitejšia než štruktúra štandardnej rekurentnej siete. LSTM sa skladá zo štyroch spolupracujúcich častí (obrázok 2.7). Štruktúra LSTM je teda tvorená vstupnou bránou (input gate), zabúdacou bránou (forget gate), výstupnou bránou (output gate) a pamäťovou bunkou, ktorá uchováva históriu.



Obr. 2.7: Štruktúra LSTM[2]. Skladá sa zo štyroch častí vstupná brána (input gate), výstupná brána (output gate), zabúdacia brána (forget gate) a bunka s históriou (Cell), ktorá v sebe uchováva históriu. Tieto štyri časti zaobstarávajú možnosť uchovať si dlhšie závislosti. Každá z brán je vektor a je daná funkciou $\sigma(\cdot)$ so vstupom $\vec{x}_t$ a výstupom $\vec{h}_{t-1}$.

Prvou z častí je zabúdacia brána $\vec{f}_t$, ktorá slúži na rozhodovanie, ktoré dáta sa zahodia. Riadi sa rovnicou:

$$\vec{f}_t = \sigma(\mathrm{W}_{fx}\vec{x}_t + \mathrm{W}_{fh}\vec{h}_{t-1} + \vec{b}_f) \tag{2.6}$$

kde $\vec{b}_f$ je bias vektor a W_{fx} a W_{fh} sú matice váh, $\vec{x}_t$ je vstupný vektor, $\vec{h}_{t-1}$ je vektor predchádzajúcich výstupov. Aktivačnou funkciou je logistická sigmoida $\sigma(\cdot)$.

Zabúdacia brána zohľadňuje vstup $\vec{x}_t$ a predchádzajúci výstup $\vec{h}_{t-1}$. Výstupom sú hodnoty v intervale [0,1], jedna pre každý možný výstup LSTM. Hodnoty vektoru $\vec{f}_t$ značia, čo sa z pamäti odstráni a čo zanechá, kde hodnota $\vec{1}$ značí, že všetky pôvodné hodnoty sa zachovajú a hodnota $\vec{0}$ všetky pôvodné hodnoty sa odstránia.

[2]Obrázok 2.7 prevzatý z http://russellsstewart.com/assets/img/lstm.png

Ďalšou časťou je vstupná brána:

$$\vec{i_t} = \sigma(\mathrm{W}_{ix}\vec{x}_t + \mathrm{W}_{ih}\vec{h}_{t-1} + \vec{b}_i) \tag{2.7}$$

Tá rozhoduje o tom, či sa budú do pamäti pričítať nové dáta. Tieto nové hodnoty, ktoré sa uchádzajú o vstup do pamäti sú uložene vo vektore $\vec{g_t}$:

$$\vec{g}_t = tanh(\mathrm{W}_{gx}\vec{x}_t + \mathrm{W}_{gh}\vec{h}_{t-1} + \vec{b}_g) \tag{2.8}$$

Vektor $\vec{i_t}$ rozhoduje o tom, ktoré hodnoty z vektoru $\vec{g}_t$ sa pričítajú do pamäti, a ktoré naopak budú nepoužité.

Súčasťou LSTM je aj pamäťová bunka $\vec{C_t}$. Pamäťová bunka je zodpovedná za udržiavanie histórie. Je to stav pamäti s informáciami. V každom čase sa vykonáva aktualizácia tejto pamäťovej bunky pomocou zabúdacej a vstupnej brány zo stavu C_{t-1} do stavu C_t. Aktualizácia je daná rovnicou:

$$\vec{C_t} = \vec{C}_{t-1} \otimes \vec{f_t} + \vec{g}_t \otimes \vec{i_t} \tag{2.9}$$

kde operácia $\otimes$ značí element-wise násobenie, vzťah $\vec{C}_{t-1} \otimes \vec{f_t}$ značí odstránenie dát určených vektorom $\vec{f_t}$ a vzťah $\vec{g}_t \otimes \vec{i_t}$ značí nové hodnoty zapisujúce sa do stavu C_t.

Poslednou časťou je výstupná brána $\vec{o}_t$, ktorá je filtrom pre výstup z pamäťovej bunky. Tento vektor je určený rovnicou:

$$\vec{o}_t = \sigma(\mathrm{W}_{ox}\vec{x}_t + \mathrm{W}_{oh}\vec{h}_{t-1} + \vec{b}_o) \tag{2.10}$$

Výstup LSTM je:

$$\vec{h}_t = \vec{o}_t \otimes tanh(C_t) \tag{2.11}$$

Výstup je určený výstupom pamäťovej bunky za použitia funkcie $tanh()$, ktorá výstupné hodnoty upraví do intervalu [-1,1] a potom vektorom (filtrom) $\vec{o}_t$ sa na výstup dostávajú zvolené hodnoty.

Výstupom modelu je pravdepodobnostné pole obsahujúce pravdepodobnosť pre každý možný výstupný prvok. Úspešnosť týchto výstupoch je možné sledovať pomocou chyby. Chyba je teda cross entropia, riadi sa rovnicou:

$$E = -\sum_{i=1}^{m} t_i \log x_i \tag{2.12}$$

kde x_i je í-ty vstup a t_i je pravý výstupný cieľ.

Výstup je tiež v rovnakej forme ako výstup doprednej siete Nn-gram, je riadený funckiou softmax.

Kapitola 3

Dáta ako vstup neurónovej siete

Na natrénovanie jazykového modelu je nutné mať text, ktorý bude slúžiť ako vstup pre NN, aby bola schopná natrénovať postupnosť písmen a tvoriť slová. Vstupný text je tým pádom neodmysliteľnou časťou mojej práce. Text však nemôže byť náhodný sled slov. Keďže chcem aby vo výsledku môj program dokázal generovať básne, je nutné ako vstupný text použiť poéziu.

Dáta, ktoré som v mojej práci použil ako vstupy na trénovanie modelu, sa dajú rozdeliť na dve časti a to prózu, ktorou sa model učí čo najviac slov a na časť poézie, ktorá slúži na to, aby model bol schopný vytvoriť báseň, a to tak, že sa model dotrénováva na dátach poézie, kde sa učí napríklad ako dlhý priemerne je jeden verš, že na konci veršov sa slová rýmujú, apod.

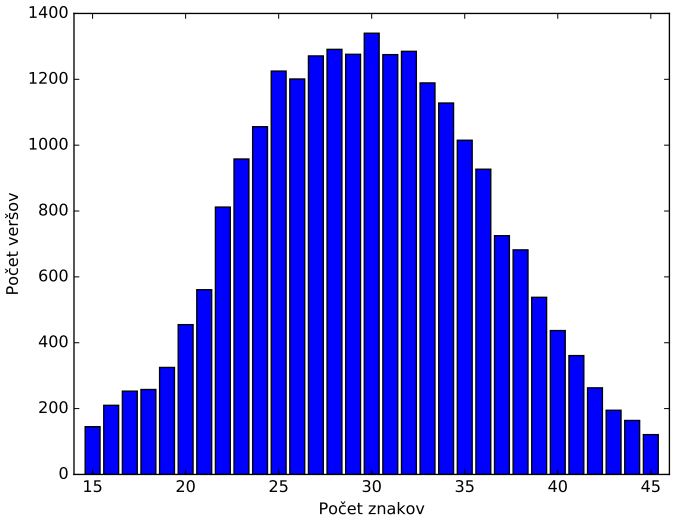
3.1 Hľadanie a zbieranie dát pre vstup modelu

Pre dosiahnutie dobrých výsledkov je dôležité mať dostatočné množstvo básní. Model neobmedzuje vstupné dáta, je teda možné dať na vstup rôzne množstvo dát, platí čím viac vstupných dát, tým kvalitnejších výsledkov sa dosahuje. Dostačujúce množstvo nie je možné vyčísliť, záleží od výsledných generovaných básní, podľa ktorých je možné sa rozhodnúť, či je množstvo dostačujúce.

Priemerne báseň obsahuje 5 strof, kde každá má 4 až 8 veršov po približne $25 - 35$ znakov (histogram 3.1), čo je približne 2100 znakov na báseň. Ak cieľ je aspoň $1M$ znakov, je nutné nájsť 477 básní. Z tohto plynie, že hľadať básne po jednom je veľmi neefektívne, čo sa týka pomeru čas/zisk.

Databáza Zlatý Fond [1] poskytuje veľa slovenských diel. Vďaka tejto databáze už nebol problém nájsť slovenskú poéziu. Použitie básní známych slovenských autorov pre vedecké účely nepodlieha autorskému zákonu. Použitý dataset je zverejnený[1]. Pre uľahčenie práce som si hneď ako prvé vybral známe dlhšie slovenské básne. Nazývam ich dlhými, pretože jedno takéto dielo obsahuje 291 strof, kde 286 strof je desaťveršových a 5 strof je osemveršových. Ak by sa vstupné dáta skladali len z týchto dlhých básní, výsledok by nebol kvalitný, pretože model, po natrénovaní pomocou týchto dát, by sa priveľmi upínal na štýl jediného autora, čo by bol nežiaduci účinok. Preto ďalšie básne boli krátke a od rôznych autorov.

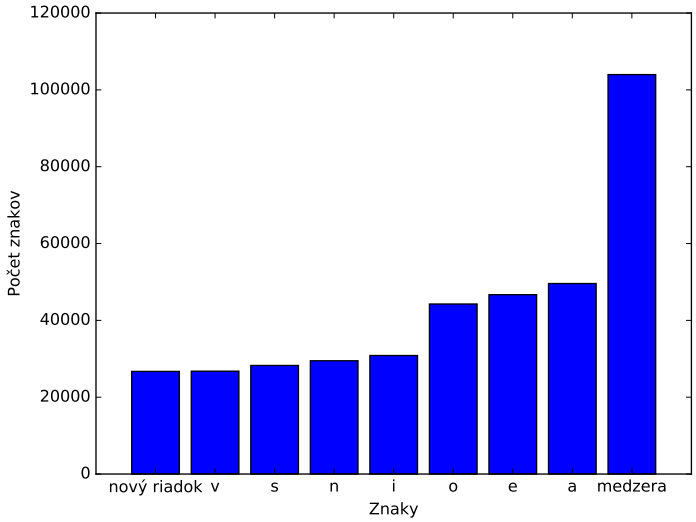
[1] Dataset je dostupný na http://www.stud.fit.vutbr.cz/~xbanca00/dataset/



Obr. 3.1: Histogram počtu znakov vo verši. Počet nie je daný pravidlom a tak básne môžu mať rôzny počet znakov vo verši. Avšak väčšina obsahuje $25 - 35$ znakov.

3.1.1 Dáta pre trénovanie modelu

Slovenská abeceda obsahuje 46 písmen. Model rozlišuje veľké a malé písmená, číslice a iné rôzne znaky. Maximálny počet znakov v slovenskom jazyku je 124 znakov. Maximálny, pretože model nemusí vždy pracovať so všetkými znakmi, ale pracuje vždy len s tými, ktoré sa nachádzajú v texte. V mojej zbierke sa nachádza 121 unikátnych znakov. Početnosť najčastejších znakov zobrazuje histogram 3.2.



Obr. 3.2: Histogram výskytu znakov v zbierke básní. Zobrazené nie sú všetky znaky, ale len najčastejšie sa vyskytujúce.

Zo vstupných dát je vytvorený takzvaný slovník, ktorý obsahuje všetky znaky nachádzajúce sa v texte, dátach. Pomocou tohto slovníka je indexovaný každý znak, teda každému znaku je pridelený práve jeden, jedinečný index. Pre sieť neexistuje pojem slovo alebo znak, pracuje s indexami.

Súčasťou sú aj validačné dáta, ktoré sú taktiež poézia, avšak neslúžia na trénovanie, ale na overovanie správnosti trénovania. Nie je nutné mať validačných dát rovnaké množstvo ako trenovacích, ale taktiež platí čím viac dát tým lepšie. Moja zbierka obsahuje celkovo $900k$ znakov, z čoho je $800k$ trénovacích a $100k$ je validačných.

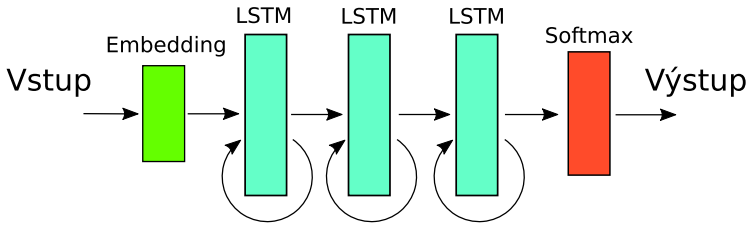
Kapitola 4

Implementácia

Na modelovanie jazyka na znakovej úrovni som vytvoril aplikáciu v jazyku *Python 2.7* s využitím high-level knižnice Keras [3], ktorá sa zaoberá modelovaním neurónových sietí. Ako backend pre túto knižnicu využívam Theano [11]. Aplikácia sa skladá zo štyroch skriptov, a to zostavenie modelu, trénovanie, generovanie a overenie správnosti vygenerovanej básne, čo však patrí do logického celku generovania.

4.1 Zostavenie modelu

Pred zostavením modelu sú upravené vstupné dáta. Zbierka je otvorená v kódovaní UTF-8, čo umožňuje bezproblémovo pracovať so slovenským jazykom. Následne je zo zbierky vytvorený slovník všetkých znakov nachádzajúcich sa v texte. Každému znaku je priradený jedinečný index, ktorý ho reprezentuje. Pre zostavenie modelu je nutné definovať vrstvy modelu. Prvou je vrstva Embedding, ktorá z indexov znakov vytvorí vektory, ktoré predáva na vstup LSTM vrstve. Nasledujú vrstvy LSTM a poslednou je vrstva Dense, ktorá spája neuróny medzi vrstvami. Vyobrazenie modelu je na obrázku 4.1.



Obr. 4.1: Neurónová sieť používaná v programe. Skladá sa z vrstvy Embedding, troch vrstiev LSTM a vrstvy Dense.

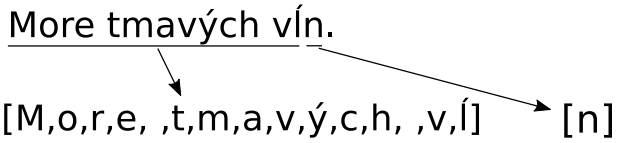
Pre optimalizáciu parametrov modelu využívam algoritmus *Adam* [8]. Tomuto algoritmu je nutné zadať hodnotu learning rate, ktorá ovplyvňuje, aké veľké kroky v n-dimenzionálnom priestore model robí pri trénovaní. Taktiež je možné využiť argument *decay*, ktorý definuje, ako sa learning rate bude počas trénovania v každej iterácii meniť vzťahom:

$$\alpha_t = \frac{\alpha_0}{1 + tK} \tag{4.1}$$

Vďaka tomuto je možné dosahovať lepších výsledkov. K získaní týchto hodnôt som urobil experimenty, viac v kapitole 5.

4.2 Trénovanie

Po zostavení modelu sa prechádza na trénovanie. Na začiatku je nutné upraviť vstupné dáta. Následne je vstupný text navzorkovaný a to tak, že je rozdelený na sekvencie. Dĺžka sekvencie môže byť rôzna a na zistenie vhodnej bol vykonaný experiment (Kapitola 5). Trénovací príklad potom pozostáva z dvoch častí. Príklad trénovacej vzorky je zobrazený na obrázku 4.2. Tento postup je použitý aj na validačné dáta. Takto navzorkované dáta sú privedené modelu na vstup trénovania.



Obr. 4.2: Príklad vzorkovania vstupný dát. Dáta sú rozdelené do sekvencií v dvoch premenných, kde v jednej sú po sebe idúce znaky a v druhej jeden nasledujúci za sekvenciou.

Trénovanie prebieha iteratívne, iteruje prechod dátami. Jedno takéto prejdenie sa nazýva epocha. Doba trvania jednej epochy, nutný počet epoch a teda celková doba trénovania sa odvíja od konfigurácie.

Samotné trénovanie je zapuzdrené v metóde knižnice Keras. Úspešnosť trénovania je sledovaná hodnotami validačnou a trénovacou chybou. Pri klesajúcej trénovacej chybe, ale stúpajúcej validačnej sa dostávame do situácie, kedy sa model pretrénováva, snaží sa naučiť trénovacie dáta naspamäť. Takáto situácia je v tomto prípade nežiadúca a je nutné sledovať, kedy model dosahuje najmenšej validačnej chyby. Na kontrolu trénovania je možné využiť tzv. callbacky, ktoré umožňujú ovplyvňovať trénovanie za behu. Konkrétne na predídenie predtrénovania slúži callback EarlyStopping, ktorý sleduje validačnú chybu. Ak sa validačná chyba zhoršuje dlhšie než je počet epoch zadaných v argumente, callback trénovanie zastaví.

4.3 Generovanie

Keď je model natrénovaný, je možné ho použiť na generovanie básne. Pre generovanie básne je nutné zadať vstupný text vo forme, napr. prvého veršu. Tento text slúži na inicializáciu skrytých stavov modelu. Vstupný text je navzorkovaný ako pri trénovaní. Každá pravdepodobnosť z výstupu je upravená hodnotou α, ktorá ovplyvňuje veľkosť pravdepodobností (rovnica 4.2).

$$p'_i = \frac{p_i^{\frac{1}{\alpha}}}{\sum_{k=0}^{n} p_k} \tag{4.2}$$

kde p je pravdepodobnosť znaku. S rastúcou hodnotou α sa zväčšujú pravdepodobnosti znakov a to tak, že najmenšie pravdepodobnosti sa zväčšia najviac. To činí program viac samostatným a vygenerovaný text je tak viacej odlišný od trénovacích predlôh a menej sa riadi výstupom siete. Implicitne je $\alpha = 0.5$, avšak užívateľ si môže túto hodnotu zmeniť.

Pravdepodobnosti po úprave hodnotou α sú opäť v intervale $[0, 1]$ a normalizované aby ich súčet bol 1. Následne sa náhodne vyberá index podobne ako pri hode kockou, s tým, že by počet strán bol rovnaký ako počet znakov a každý znak mal inú pravdepodobnosť padnutia. Znaky s väčšími pravdepodobnosťami majú väčšiu šancu, avšak znak sa vyberá náhodne. Index je prevedený na znak a pripísaný k počiatočnému textu. Generovanie ďalšieho znaku je teda vždy ovplyvnené aj predchádzajúcimi vygenerovanými znakmi. Na zlepšenie gene-

rovaných básní som vytvoril aj skript, ktorý kontroluje vygenerované básne. Na kontrolu je použitá zbierka všetkých dát, poézia aj próza. Skript potom kontroluje každé vygenerované slovo, či sa nachádza v zbierke. Ak sa v zbierke nenachádzajú dve vygenerované slová, generuje sa nová báseň. Povolená chyba aplikácie je teda maximálne jedno chybné slovo na verš.

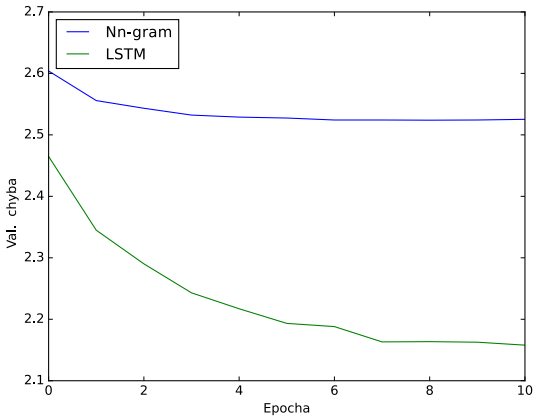
Kapitola 5

Experimenty

Pre dosiahnutie čo najlepších výsledkov je nutné vhodne konfigurovať model. Vhodnú konfiguráciu pre zostavenie modelu nie je zväčša možné predpovedať vopred. Je nutné s konfiguráciou experimentovať. Touto konfiguráciou sa myslí počet LSTM vrstiev, šírka týchto vrstiev, learning rate (parameter riadiaci rýchlosť učenia), dĺžka vstupných sekvencií.

5.1 LSTM vs Nn-gram

Jednoduchá sieť bez LSTM vrstvy (Nn-gram) si taktiež dokáže určité závislosti natrénovať, avšak zďaleka nedosahuje výsledkov LSTM. Tento rozdiel zobrazuje obrázok 5.1. Aj keď Nngram pomalým tempom trénuje a pri konci už takmer netrénuje, model s LSTM dosahuje lepšie výsledky oveľa skôr.

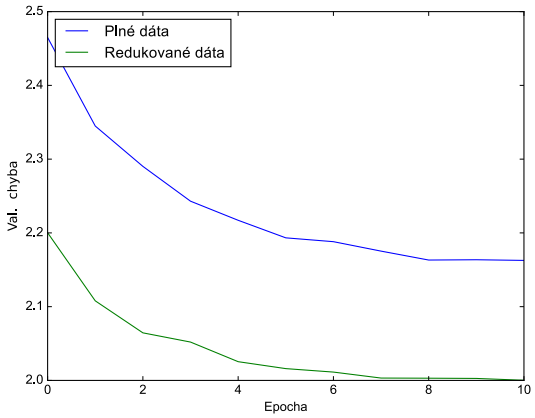


Obr. 5.1: Rozdiel medzi modelom s LSTM a bez LSTM vrstvy. Sieť Nn-gran nedokáže za rovnaký čas konkurovať modelu s LSTM vrstvou.

5.2 Architektúra modelu

Na začiatok bolo nutné nájsť predvolený bod siete, tzv. odrazový mostík, od ktorého by sa všetky experimenty odvíjali. Pre začiatok som teda použil jednoduchý model s jednou vrstvou LSTM. Taktiež boli na vstup priložené básne bez veľkých písmen a bez diakritiky.

Takáto sieť sa dokáže na takýchto dátach úspešne natrénovať. Avšak program potom dokáže generovať len text s takýmito znakmi, čo nie je cieľ. Preto môj ďalší postup bol skúsiť model, ktorý správne pracuje s jednoduchými dátami, natrénovať na plných dátach, t.j. veľké písmena aj diakritika (graf 5.2). Takýto model pri normálnych dátach je už nedostačujúci



Obr. 5.2: Rozdiel medzi modelom s jednou vrstvou LSTM natrénovaný na normálnych a model natrénovaným na redukovaných, jednoduchých dátach. Keďže je približne 3-násobný počet znakov než v jednoduchých dátach, model je príliš jednoduchý aby dosahoval lepších výsledkov.

a ďalším experimentom bol rôzny počet vrstiev LSTM.

5.2.1 Počet vrstiev LSTM

Tabuľka 5.1 zobrazuje výsledky s 1, 2, 3 vrstvami LSTM. Tri vrstvy LSTM v modely dosahujú lepších výsledkov validačnej chyby. V grafe 5.3 je zobrazený celý priebeh trénovania

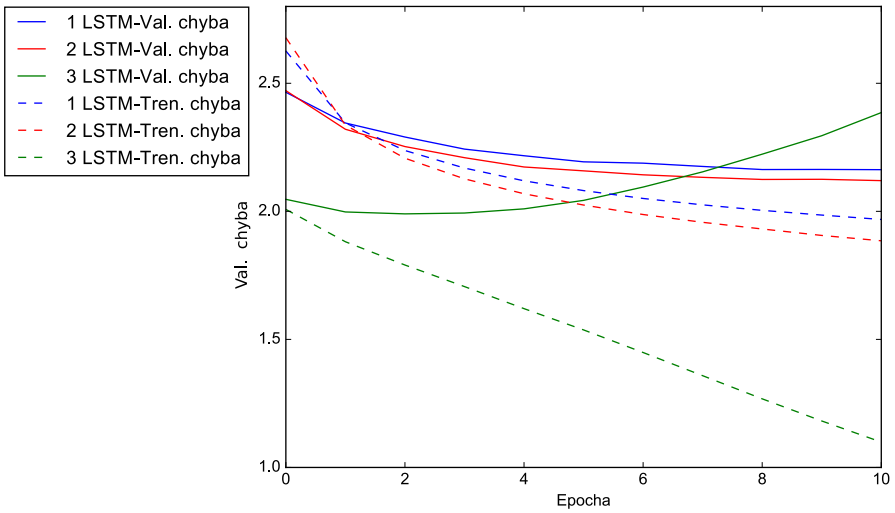
Tabuľka 5.1: Experiment - počet vrstiev

Počet vrstiev	tren. chyba	val. chyba	Epocha
1	1.9281	2.1548	14
2	1.8454	2.1195	13
3	**1.7904**	**1.9901**	**3**

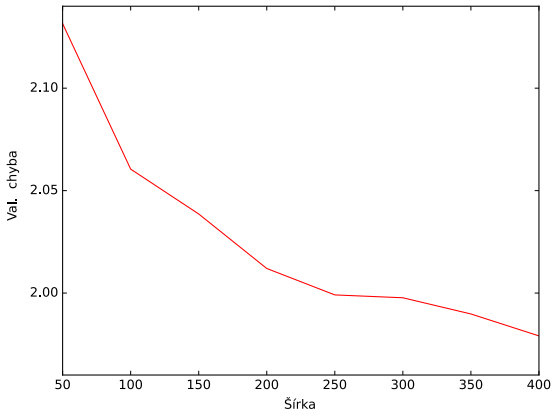
s validačnou a trénovacou chybou modelov s 1, 2 a 3 vrstvami LSTM. Je možné si všimnúť, že model s 3 vrstvami sa s rastúcim počtom epoch zhoršuje. To je zapríčinené tým, že model je už dosť zložitý nato, aby sa učil trénovacie dáta naspamäť, čiže sa pretrénovával. Graf 5.3 túto situáciu popisuje. Túto situáciu je možné vyriešiť použitím callbacku EarlyStopping. Výsledný model teda pracuje s tromi vrstvami LSTM, keďže takýto model dosahoval najlepšie výsledky.

5.2.2 Šírka LSTM vrstiev

Po zistený dostačujúceho počtu vrstiev LSTM je vhodné zistiť šírku týchto vrstiev. Graf 5.4 zobrazuje najlepšie výsledky jednotlivých šírok.

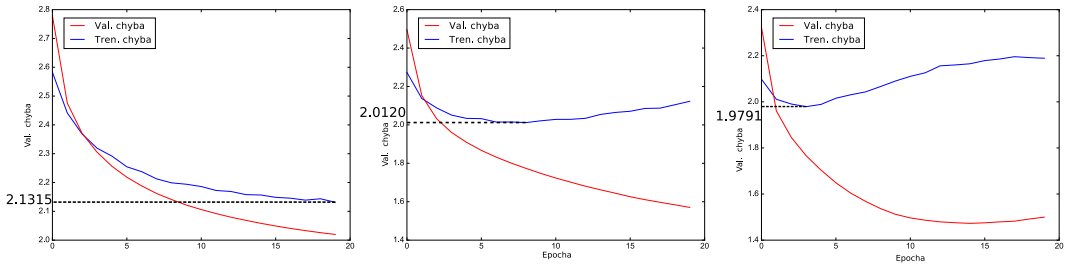


Obr. 5.3: Graf celého priebehu trénovania modelu s 1, 2 a 3 vrstvami LSTM. Validačná chyba v modely s 3 vrstvami klesá do svojho minima následne sa zhoršuje, zatiaľ čo trénovacia chyba stále klesá - vzniká pretrénovanie. V modely s 1 a 2 vrstvami validačná aj trénovacia chyba klesajú do ich limít.



Obr. 5.4: Experiment č. 2 - šírka vrstiev LSTM. S rastúcou šírkou je dosahovaná menšia validačná chyba.

S rastúcou šírkou sa taktiež zvyšuje doba trénovania, no najmä sa model skorej dostane do fázy pretrénovania (graf 5.5). Keďže pretrénovanie už nie je problém a šírka 400 skrytých jednotiek dosahuje najlepších výsledkov, rozhodol som sa túto šírku prijať ako výslednú a použiť ju vo výslednom modely.



(a) 50 skrytých jednotiek. (b) 200 skrytých jednotiek. (c) 400 skrytých jednotiek.

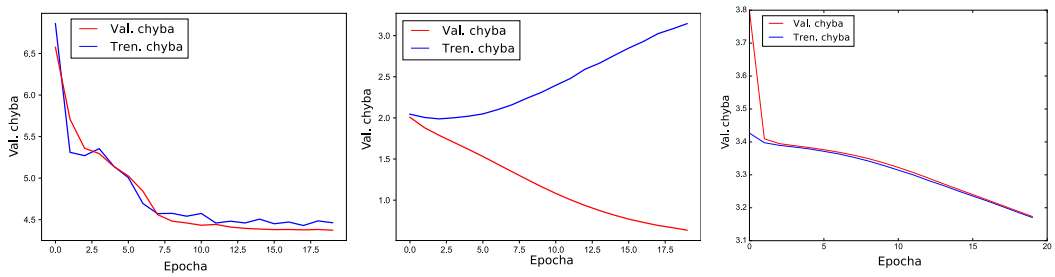
Obr. 5.5: Rôzne šírky LSTM vrstiev. S rastúcou šírkou je možné vidieť kedy model dosahuje najlepších výsledkov. Pri väčších šírkach dosahuje model najlepšej validačnej chyby skorej. Ak však trénovanie nie je včas zastavené, môže dôjsť k pretrénovaniu.

5.3 Parametre ovplyvňujúce trénovanie

Zo začiatku bolo dôležité nájsť predvolený bod, od ktorého by sa ďalšie experimenty odvíjali. K tomu patrí aj nájsť vhodný algoritmus optimalizácie parametrov. Najprv som používal RMSprop, ktorý však nepracoval správne. Algoritmus Adam si už so slovenskými dátami poradiť vedel. Dôležitý nie je však len samotný algoritmus, ale tiež jeho parametre, ktoré veľkou mierou ovplyvňujú trénovanie.

5.3.1 Learning rate

Ďalšou úlohou bolo správne optimalizovať rýchlosť trénovania, parameter learning rate. Tento parameter ovplyvňuje rýchlosť učenia. Ak je príliš vysoký (graf 5.6a), model robí priveľmi veľké kroky počas učenia a nič sa nenaučí. Ak je learning rate priveľmi malý (graf 5.6c), model robí malé kroky, doba trénovania sa zvyšuje a tým pádom je dosiahnutie správneho natrénovania taktiež nemožné. Vhodný learning rate je potom možné vidieť na grafe 5.6b.

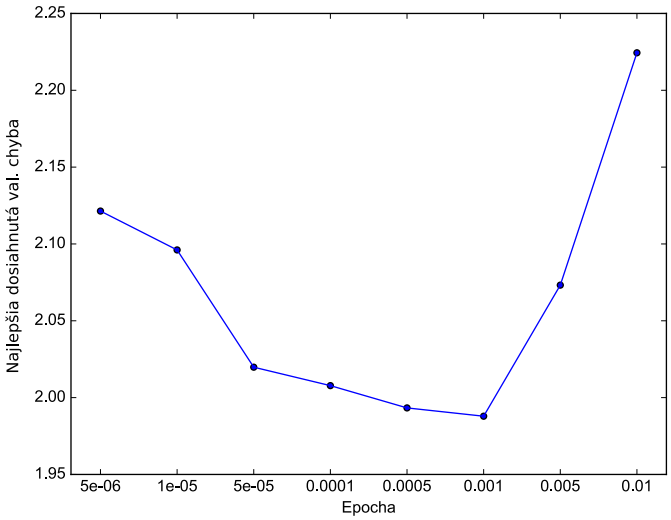


(a) Learning rate $5 \cdot 10^{-3}$. (b) Learning rate 10^{-3}. (c) Learning rate 10^{-6}.

Obr. 5.6: Rozdielny parameter riadiaci rýchlosť učenia. Príliš veľký (5.6a) alebo príliš malý (5.6c) learning rate spôsobí, že sa model nič nenaučí.

Pri porovnaní všetkých experimentov s learning rateom (graf 5.7) je vidieť, že stredne veľký learning rate dosahuje najlepších výsledkov.

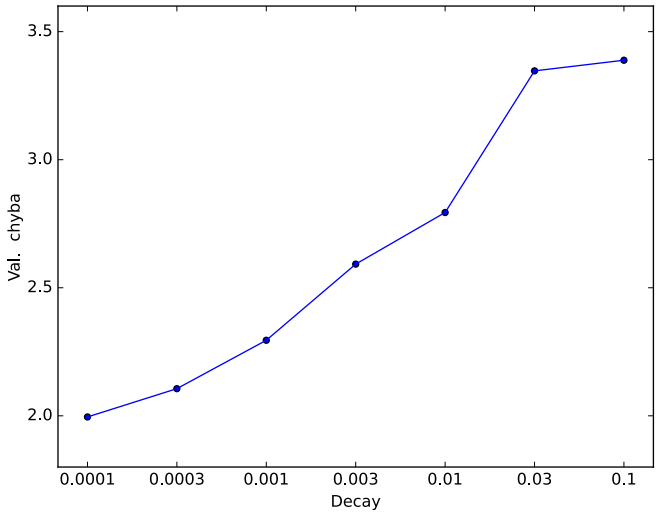
Pre ďalšie použitie som teda vybral hodnotu 10^{-3} ako najvhodnejšiu a preto táto hodnota je vo výslednom modely.



Obr. 5.7: Graf popisuje hodnoty validačnej chyby pre rôzne hodnoty learning rate. Najlepší výsledok dosahuje hodnota 10^{-3}.

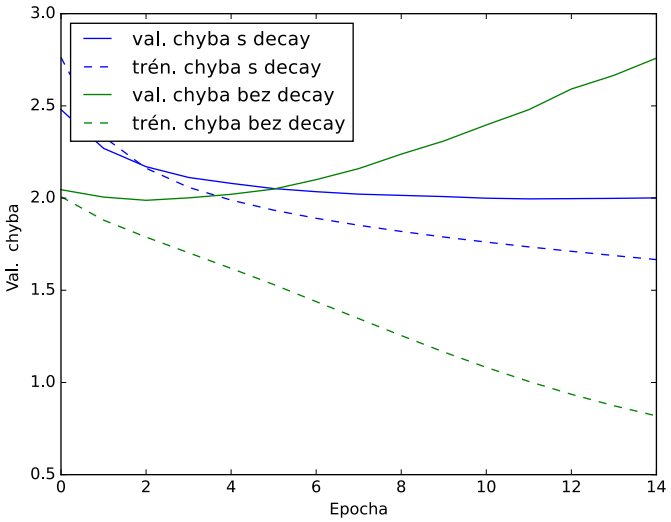
5.3.2 Decay - parameter na zmenenie learning rate za behu

Statický learning rate je celkom neprispôsobivý a tým pádom trénovanie menej efektívne. Ak by sa learning rate menil počas trénovania menil mohol by sa viac prispôsobovať a tým pádom dostať k lepšiemu výsledku s menšou validačnou chybou. Toto je umožnené pomocou argumentu decay, ktorý slúži nato, aby sa learning rate za behu trénovania menil. Počiatočný learning rate je rovný 10^{-3}.



Obr. 5.8: Vyobrazenie všetkých hodnôt decay, pre ktoré bol vykonaný experiment. Hodnota 0.0001 dosahuje najlepšie výsledky.

Graf (5.8) zobrazuje výsledky jednotlivých hodnôt. Hodnota 10^{-4} dosahuje najlepšieho výsledku. Ak si však porovnáme model s decay a model bez decay (graf 5.9), je možné si všimnúť, že podobnú najlepšiu hodnotu dosiahne oveľa neskôr. Tak isto je však oddialené pretrénovanie.



Obr. 5.9: Porovnanie modelu s decay a bez decay. Je možné vidieť, že dosahujú podobne úspešný výsledok, avšak za iný počet epoch. Model bez decay však už lepší výsledok dosiahnuť nemôže, pretože dochádza k pretrénovávaniu. Model s decay dosahuje síce minimálne, ale lepší výsledok.

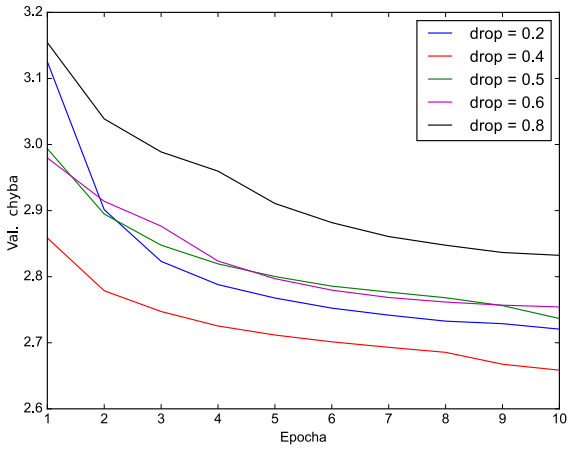
Lepšieho výsledku sa však dosahuje za dlhšiu dobu trénovania, čo znamená, že nie je isté, či je použitie decay efektívne. V ďalších experimentoch však pracujem s decay 10^{-4}, keďže táto hodnota dosahuje najlepší výsledok a to 1.9856.

5.4 Dropout - parameter na zdokonalenie trénovania

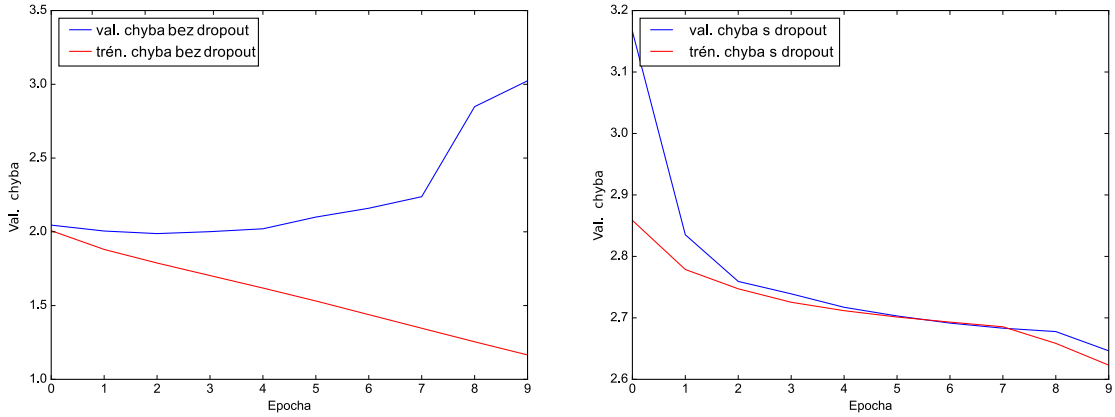
Ako sa dalo vidieť v predchádzajúcich experimentoch pri takejto konfigurácii, model dosahuje svojho minima v chybe pomerne skoro a následne sa model začína pretrénovávať. Výsledky, ktoré v tomto minime dosahuje sú celkom slušné, ale je vhodné zistiť, či by model nedosahoval ešte lepšie výsledky, ak by sa tak skoro nepretrénovával. Túto možnosť ponúka vrstva Dropout, ktorá vypína pri trénovaní určitý počet neurónov (počet sa zadáva v argumente v intervale $[0, 1]$, značí percentá vypnutých neurónov). Každú epochu sa vypínajú iné neuróny a tým je každou epochou pre model trénovanie odlišné, čo zabraňuje pretrénovaniu. Graf 5.10 zobrazuje tento experiment. Najlepšej hodnoty v najkratšom čase dosahuje dropout, kde sa každú epochu vypne 40 % neurónov.

Pri porovnaní modelov (graf 5.11) s dropout (graf 5.11b) a bez dropout (graf 5.11a) môžeme vidieť, že výsledok nie je zlomový a dosiahnuté výsledky nie sú výrazne lepšie. Avšak nedochádza k pretrénovaniu, čo môže mať za následok, že po dlhšom čase trénovanie dosiahne kvalitnejšie výsledky.

Využitie dropout však výrazne predlžuje dobu trénovania, čo takisto ako pre decay kladie otázku, či využitie tohto parametru je výhodne, a či naozaj dosahuje lepších výsledkov.



Obr. 5.10: Zobrazenie všetkých experimentov s Dropout. Výsledky sú horšie ako bez dropout, ale je možné si všimnúť, že s každou epochou sa validačná chyba zlepšuje, čo znamená, že dosahuje, síce za dlhší čas, lepších výsledkov. Experiment obsahoval 5 rôznych hodnôt dropout. Najlepší výsledok v desiatich epochách dosiahol dropout 0.4.



(a) Graf bez dropout.

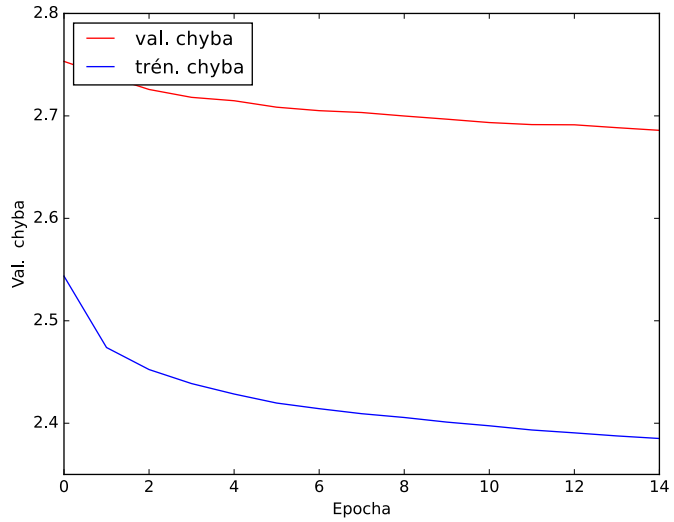
(b) Graf s dropout 40%

Obr. 5.11: Grafy zobrazujúci rozdiel medzi modelom s dropout (graf (5.11b)) a bez dropout (graf (5.11a)). Model (5.11a) dosahuje lepší výsledok v pomerne krátkom čase, avšak následne dochádza k pretrénovaniu, zatiaľ čo model (5.11b) sa naďalej trénuje. Nedosahuje však porovnateľné výsledky v rovnakom čase, avšak za dlhší čas môže dosiahnuť lepšie.

5.5 Trénovanie prózy

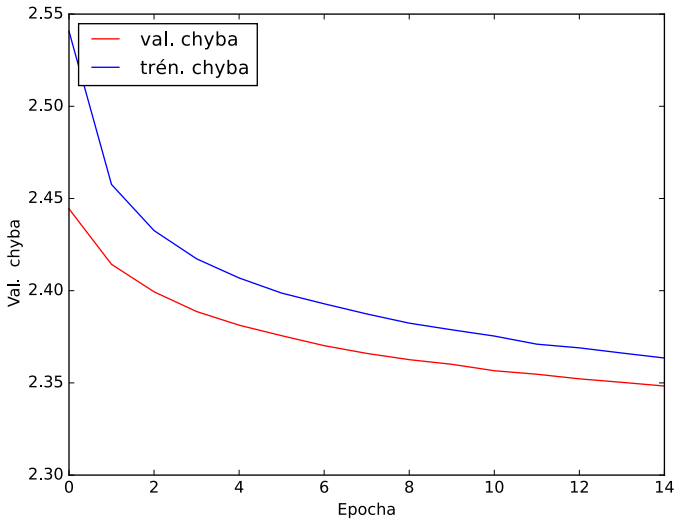
Pre získanie kvalitnejších výsledkov finálneho modelu som experimentálne skúšal natrénovať model najprv na prózy, alebo inom, rôznom texte mimo poézie. Od tohto experimentu som čakal, že modelom generovaný text bude výrazne lepší, keďže klasický text je jednoduchší, model by sa mal trénovať lepšie. Trénovanie na próze zobrazuje graf 5.12.

Trénovanie na próze nedosahuje také výsledky, aby model bolo možné používať, avšak sa jedná len o predtrénovanie a je nutné model natrénovať ešte na poézii. Graf 5.13 zobrazuje



Obr. 5.12: Trénovanie modelu prózou. Validačná chyba je vysoká nedosahuje takých hodnôt ako v iných experimentoch, avšak jedná sa o trénovanie prózy, ktorá obsahuje takmer dvojnásobný počet znakov. Keďže sa jedná o taký počet znakov a bol použitý dropout chyba sa s každou epochou zmenšuje, ale veľmi pomaly. Avšak sa jedná len o predtrénovanie prózou.

dotrénovanie na poézii. Na grafe je zaujímavé, že validačná chyba je lepšia ako trénovacia. Takáto situácia je spôsobená použitím dropout. Zodpovedá to tomu, že sa predchádza pretrénovaniu.



Obr. 5.13: Dotrénovanie modelu poéziou. Learning rate je rovný 10^{-3}, za použitia decay 10^{-4} a dropout 0.4. Keďže sa jedná o finálny model, dosiahnuté výsledky sú veľmi nedostačujúce. Klesanie validačnej chyby je veľmi pomalé a rozdiel chyby v po sebe idúcich epochách je minimálny a trénovanie tak vyžaduje dlhý čas trénovania.

Takto natrénovaný model by číselne nemusel vykazovať dobré výsledky a preto je nutné sa pozrieť na vygenerovaný text 5.14a. Tučným písmom je zvýraznený počiatočný text.

Ako sen čo sa tvári smutne *podena,*
v nebod sa mi sa stradno sa sa zastali,
v prade sa na prediam sa sa pozdenie sa
v sade sa pred svet sa
stratne sa sa to stralne sa sa stralený som sa
ten sa strane sa
stratne sa postene sa stranie sa v obej sa
sloven sa podom sa
zastali sa v priestala sa v straste sa na sa
strale mi sa pod sa to
svet sa pred sa sa slet mi sa na tvoji sa
stradia v duše strala
nediasti sa postalene.

(a) Vygenerovaný text modelom predtrénovaným prózou.

Stretol som raz krásnu ženu,
na nebo v tej svetu srdce,
tebe sa svätým pohoja,
a tak je kraj dobrotov,
barky život krásny snával

(b) Vygenerovaná báseň modelom bez dropout a prózy.

Obr. 5.14: Rozdiel vygenerovaných textov modelom bez prózy a decay 5.14a a modelom s prózou a s dropout 5.14b. Výsledok je jasne viditeľný a je jasné, že model 5.14b dosahuje oveľa lepšie výsledky.

Vygenerovaný text je nepoužiteľný pri trénovaní v čase približne dvoch týždňov. Po dlho trvajúcom trénovaní by však model mohol dosahovať lepších výsledkov, avšak keďže priemerne je dĺžka jednej epochy $2h\,45m$, dosiahnuť lepších výsledkov by vyžadovalo veľa času na trénovanie.

5.6 Záver experimentov

Po zohľadnení všetkých experimentov je vo finálnom modeli používaný model bez dropout a bez prózy. Takýto model dosahuje ako aj číselné tak aj textové lepšie výsledky. Vygenerovaný text s použitím decay a bez dropout a (text 5.14b v predchádzajúcej podkapitole) prózy to dokazuje. Ak sa pozrieme na takto vygenerovaný text je viditeľné, že text pripomína poéziu v slovenčine. Pre finálny model preto používam model bez dropout a bez prózy, len za pomoci parametru decay.

Kapitola 6

Testovanie užívateľmi

Po vytvorení jednoduchej terminálovej aplikácii, ktorá dokáže generovať poéziu, alebo text podobný poézii, je nutné vykonať testovanie. Ak sa pozrieme na vygenerovaný text 6.1 jednoduchým modelom s jednou vrstvou LSTM so šírkou 100 skrytých jednotiek a learning rate rovným 0.001, dosahovaný výsledok nie je kvalitný a nedá sa považovať za úspech.

Čo sa to deje s týmto svetom*,lr kbXlj mla osbpls o mlhiWa*
kW pbXW oW s pslnfW viWpó osÉpóZe mlhlgW'
s nlvmWakó gbel opWnlpr gbakbg vWo kbXlj lXgÓjWiW'
fZe oW s plj oplgÓ oW hWŕaé hnÁoku vWsfbsW'.

Obr. 6.1: Vygenerovaná báseň jednoduchým modelom.

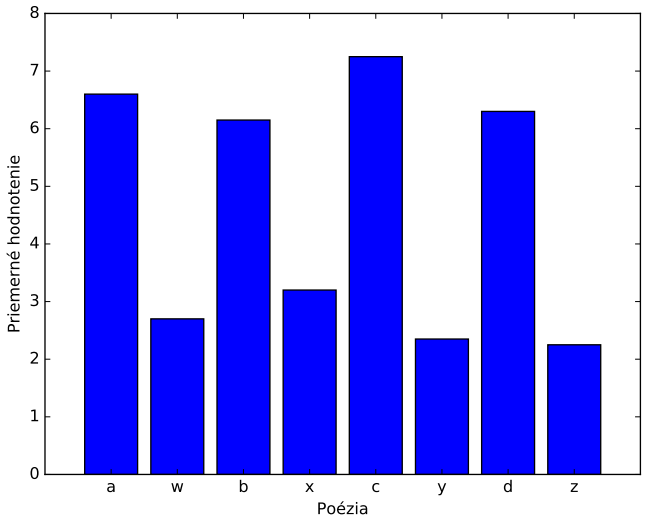
Zatiaľ čo vygenerovaný text 6.2 modelom s tromi vrstvami LSTM so šírkou 400 skrytých jednotiek a rovnakým learning rateom za použitia parametru decay, ktorý je rovný 10^{-4}, pripomína poéziu a vygenerované slová sú správne.

Stretol som raz krásnu ženu,
na nebo v tej svetu srdce,
tebe sa svätým pohoja,
a tak je kraj dobrotov,
barky život krásny snával

Obr. 6.2: Vygenerovaná báseň zložitým modelom.

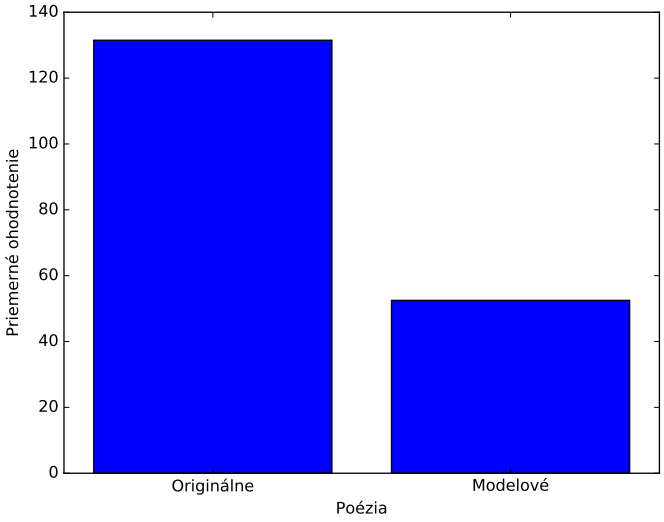
Správna konfigurácia je teda veľmi dôležitá a po vhodnom natrénovaní a získaní výsledkov som prešiel na testovanie verejnosťou. Testovanie nebolo zamerané na prácu užívateľa s aplikáciou, ale na výsledné vygenerované básne. V testovaní teda ide, ako vygenerovaný text vplýva na ľudí, aké z neho majú pocity a hlavne, či vygenerovaný text aspoň pripomína báseň.

Testovania sa zúčastnilo 20 osôb rôznej vekovej kategórie, od mladých až po starších a seniorov. Testovanie prebiehalo formou dotazníka A, kde testovaný mali ohodnotiť báseň hodnotou na stupnici od 1 do 8. V dotazníku boli zmiešané originálne básne od spisovateľov a básne vygenerované aplikáciou. Básne neboli pomenované, ani inak komentované a tak testujúci nevedeli, ktoré básne sú z ruky človeka, a ktoré sú vygenerované aplikáciou. Výsledky testovania zobrazuje graf 6.3.



Obr. 6.3: Graf výsledkov testovania. Všetky básne označené ako 'a,b,c,d' sú vytvorené človekom a básne označene ako 'w,x,y,z' sú vygenerované aplikáciou. Výsledky testovania dopadli jednoznačne v prospech básní od spisovateľov. Je možné si všimnúť, že aj každá báseň vytvorená človekom má rôzne hodnotenie a každá báseň vplýva na ľudí inak. Avšak žiadna vygenerovaná báseň nedosahuje podobné výsledky.

Je možné si všimnúť, že výsledky sú jednoznačné a vygenerované básne nedosahujú takej kvality, ako básne vytvorené človekom. Spriemerované výsledky, ktoré slúžia na zistenie rozdielu úspešnosti originálnych a vygenerovaných básní zobrazuje graf 6.4.



Obr. 6.4: Graf spriemerovaných výsledkov testovania. Priemerné hodnotenie vygenerovaných básní nedosahuje ani polovičné hodnotenie básní originálnych o čom svedčí aj dosahovaná presnosť.

Pozitívum vygenerovaných básní je, že skorej pripomínajú modernú poéziu, kde nie sú až tak dodržiavané pravidlá poézie a význam ako taký je niekedy pre obyčajného človeka ťažko pochopiteľný.

Výsledky testovania jasne dokazujú, že vygenerované básne nedosahujú takej kvality, aby mohli konkurovať, alebo byť porovnávané s ľudským výtvorom.

Kapitola 7

Záver

Cieľom práce bolo generovanie básní v slovenskom jazyku. Vytvoriť neurónovú sieť (NN) pre jazykové modelovanie na úrovni znakov. Pomocou rekurentných neurónových sietí typu Long Short-Term Memory je to možné dosiahnuť. Pri správnom zostavení NN sa dosahuje kvalitných výsledkov. Výsledné generovanie je už jednoduchou časťou vytvorenia básne.

Používaný model neurónovej siete nevyužíva všetky dostupné zlepšovacia nástroje, parametre a možnosti, ktoré by boli schopné výsledný model ešte zdokonaliť. Tieto možnosti sa nepoužívajú, pretože čas, ktorý je nutný na čo najlepšie natrénovanie je neodhadnuteľne dlhý a je teda otázne, či je ešte stále efektívne tieto možnosti využiť. Kvalitnejšie výsledky by taktiež aplikácia dosahovala, ak by modelu bolo poskytnuté viac vstupný dát vo forme básní na trénovanie, avšak slovenských básní nie je také množstvo, ako napr. anglických. Preto je dosahovanie lepších výsledkov v anglickom jazyku jednoduchšie. Taktiež zaváži aj to, že slovenský jazyk je bohatší na znaky než anglický.

Doterajšie výsledky sú považované za čiastočný úspech. Vygenerované básne nedosahujú úrovne porovnateľnej s ľudským výtvorom, avšak program dokáže vygenerovať gramaticky správne slovo a čiastočne dodržiavať rýmy na konci veršov. Taktiež dodržiava niektoré pravidlá pravopisu, napr. veľké písmeno vždy nasleduje za bodkou. Jednoduchá aplikácia na natrénovanie a generovanie poézie bola implementovaná. Dosahovaná presnosť je 41.85 %, čo znamená, že model predikuje nasledujúci znak za sekvenciou znakov s takouto presnosťou. Výsledky tejto práce boli prezentované na študentskej konferencii Excel@FIT [2].

V budúcej práci by som chcel zdokonaliť model dostatočným natrénovaním na próze, čo znamená, dlhšie trénovať na väčších vstupných dátach. Taktiež za použitia dropout, ktorý síce oddaľuje pretrénovanie, ale za cenu časovej náročnosti. Pokračovať aj v zbieraní básní a trénovať model predtrénovaný prózou na väčšom množstve poézie a vytvoriť webové užívateľské rozhranie na generovanie básní.

Literatúra

[1] *Zlatý fond denníka SME - Najväčšia slovenská elektronická knižnica.* [Online; navštíveno 15.12.2016].
URL http://zlatyfond.sme.sk/

[2] Bančák, M.: Umelý Básnik. In *Excel@FIT, Študentská konferencia*, 2017.

[3] Chollet, F.: Keras. https://github.com/fchollet/keras, 2015.

[4] Cybenko, G.: Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, ročník 2, č. 4, 1989: s. 303–314.

[5] Hochreiter, S.; Schmidhuber, J.: Long short-term memory. *Neural computation*, ročník 9, č. 8, 1997: s. 1735–1780.

[6] Hornik, K.: Approximation capabilities of multilayer feedforward networks. *Neural networks*, ročník 4, č. 2, 1991: s. 251–257.

[7] Karpathy, A.; Johnson, J.; Fei-Fei, L.: Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[8] Kingma, D.; Ba, J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Manurung, R.; Ritchie, G.; Thompson, H.: Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, ročník 24, č. 1, 2012: s. 43–64.

[10] Pascanu, R.; Mikolov, T.; Bengio, Y.: On the difficulty of training recurrent neural networks. *ICML (3)*, ročník 28, 2013: s. 1310–1318.

[11] Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, ročník abs/1605.02688, Květen 2016.
URL http://arxiv.org/abs/1605.02688

[12] Wu, X.; Tosa, N.; Nakatsu, R.: New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *International Conference on Entertainment Computing*, Springer, 2009, s. 191–196.

[13] Yan, R.; Jiang, H.; Lapata, M.; aj.: i, Poet: Automatic Chinese Poetry Composition through a Generative Summarization Framework under Constrained Optimization. In *IJCAI*, 2013.

Prílohy

Príloha A

Ukážka testovacieho dotazníku

Jedná sa o jednu stranu testovacieho dotazníku. Kompletný dotazník sa nachádza na priloženom pamäťovom médiu.

Dotazník - Testovanie básní

Bakalárska práca
Michal Bančák

1.

Svet starý v rumoch, zlobou, krivdou podlý
riad… hľa, že pýcha pádu predchodí:
kydajú z krku nevoľníkov modly,
bo slnce vzišlo našej slobody,
omračujúce na smrť Kykymoru.
Sem na prsia mi hneď-hneď trikolóru…

A radostný duch všetko schytá, berie,
unáša kriely jasnej nálady;
hne v pochod, koho v líce pohladí,
mu srdce otvoriac, jak domu dvere.

Len do dier lezú hnusné netopiere,
že oprsklo ich slnce znezrady,
to svetlo, čo už nikto nezradí,
nezvedie, aby zašlo v nočnom šere…

Hej, doba zvrtla bludom vekov, darmo
sa tomu vzpierať… Heslo bez síl, pier:
nám sloboda, nám! — vám? čo nie? jarmo.
Ó, vďaka, Bože, mňa že, i mňa, starca,
osvitlo, v ceste hoci na cmiter;
viac nečujem škrek pätnásteho marca.

1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○

2.

Deň ako tmavá noc sa rúti tieňom,
že i keď ti sa priateľ
a na krásne zostal som neviete,
tam ho veľký zvudla predsa
som ja pri nebol som ľudskej miliónsky verné neba.
A všetko som viac nenasiel sa valia,
a v hrobom sa dušu v svety
a hryštár synov hrobu vody.

A ty si spieva v mojich pannych
jak večne posladný sveta.
Od teba láska v srdci sa viny.

Od slnce v pramene pohlady.
A vy ste na hrobu sa svieti
na zemi starom národný vy.

1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○