BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2017
http://excel.fit.vutbr.cz

Umelý Básnik

Michal Bančák*

Abstrakt
Článok 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 dvoch vrstiev LSTM so 400 skrytými jednotkami. Pre prácu bola vytvorená zbierka básní v slovenskom jazyku vo veľkosti 900k znakov. Výsledkom práce je generovanie textu, ktorý má prvky básne.

Kľúčové slová: Neuronová sieť — Long Short-Term Memory — Automatické generovanie — Jazykové modelovanie na znakovej úrovni

Priložené materiály: Stiahnuteľné zdrojové súbory — Zbierka básní — Vygenerované básne

*xbanca00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

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. Pomocou neurónovej siete pre jazykové modelovanie tento problém nemusia riešiť a môžu si vytvoriť svoju báseň.

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 [1].

Druhý rad výskumov používal genetické algoritmy na generovanie básní [2]. 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 odlišiteľnú 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 [3]. 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.

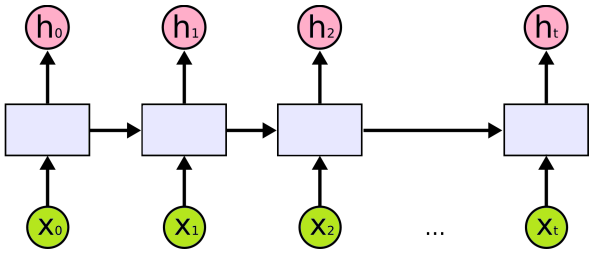
Článok popisuje zostavenie a natrénovanie neurónovej siete pre jazykové modelovanie, ktorá je schopná vytvoriť báseň. Na natrénovanie neurónovej siete je použitá zbierka básní rôznych slovenských spisovateľov, vďaka čomu sa neurónová sieť nebude upínať na štýl jedného konkrétneho spisovateľa. Tým by mala byť vytvorená báseň jedinečná a použiteľná pre kohokoľvek.

Moje riešenie sa odvíja od predchádzajúcich, v podobe nutnosti mať vstupné dáta vo forme básní, avšak za použitia neurónovej siete, konkrétne rekurentnej neurónovej siete. Pri jazykovom modelovaní, pravidlá básne, ako napríklad rým, verše a slohy, alebo správna gramatika sú implicitne zaobstarané sieťou. Sieť si dokáže tieto pravidlá natrénovať. Nájsť správnu konfiguráciu a natrénovať neurónovú sieť je časovo veľmi náročné. Samotné generovanie básne je pre sieť jednoduché. Na základe textu zadaného užívateľom, vyráta pravdepodobnosť každého znaku. Nie je teda nutné pri každom generovaní kontrolovať či text spĺňa pravidlá poézie, hľadať podobnosť zadaného slova v zbierke básní, stačí sieť raz natrénovať. Taktiež je nutná určitá interakcia s užívateľom. Po užívateľovi program vyžaduje vstupný text, napr. jeden verš.

2. LSTM pre modelovanie textu

2.1 Rekurentné neurónové siete

Pre jazykové modelovanie na úrovni znakov je nevyhnutná závislosť jednotlivých znakov. Slovo je sled zasebou idúcich písmen. Je dôležité, aby pre neurónovú sieť (NN) nebolo ťažké tieto závislosti si natrénovať. Preto v tejto práci využívam rekurentnú neurónovú sieť (RNN). Tradičná NN si taktiež dokáže určité závislosti natrénovať, ale RNN je pre túto prácu vhodnejšia: RNN má výhodu v tom, že si udržiava históriu predchádzajúcich vstupov. Udržiavanie histórie je umožnené cyklicky sa aktualizujúcej reprezentácie histórie, ktorá je v každom čase výpočtu na vstupe. Ak sa táto sieť rozbalí v čase (obrázok (1)), je možné vidieť prenášajúcu sa reprezentáciu histórie z jedného výpočtu k nasledujúcemu, čiže výstup h_i nie je ovplyvnený len aktuálnym vstupom x_i, ale aj predchádzajúcimi. Problém tohto typu NN je ten, že je veľmi náročné natrénovať čo i len jednoduchú RNN [4].



Obrázok 1. Rekurentná neurónová sieť rozbalená v čase. Prenášajúca sa reprezentácia histórie ovplyvňuje nasledujúci výpočet a je s každým výpočtom aktualizovaná.

2.2 Long Short-Term Memory

Jednou z verzií RNN, ktoré sú vytvorené na prácu s dlhými závislosťami je Long Short-Term Memory (LSTM) [5]. Štruktúra LSTM je oveľa zložitejšia ako štruktúra štandardnej RNN. LSTM sa skladá zo štyroch navzájom spolupracujúcich častí (obrázok (2)).

LSTM sa riadi rovnicami:

$$\vec{v} = \sigma(W_{vx}\vec{x}_t + W_{vh}\vec{h_{t-1}} + \vec{b}_v) \quad (1)$$

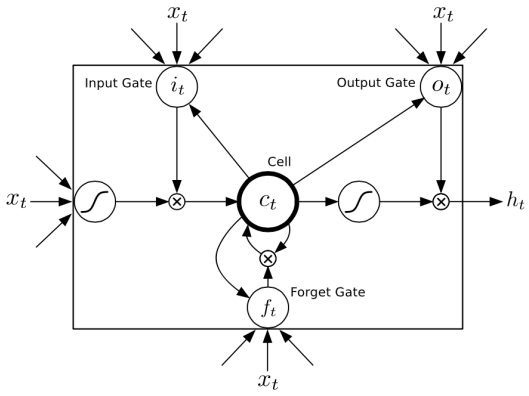
$$\vec{g}_t = tanh(W_{gx}\vec{x}_t + W_{gh}\vec{h_{t-1}} + \vec{b}_g) \quad (2)$$

$$C_t = C_{t-1} \otimes f_t + g_t \otimes i_t \quad (3)$$

$$h_t = o_t \otimes tanh(C_t) \quad (4)$$

kde b_v je vektor vychýlenia a W_v je matica váh.

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



Obrázok 2. [1]Štruktúra LSTM. Skladá sa zo štyroch častí zo vstupnej brány (input gate), výstupnej brány (output gate), zabúdacej brány (forget gate) a bunky s históriou (Cell), ktorá v sebe uchováva históriu. Tieto štyri časti zaobstarávajú možnosť uchovať si dlhšie závislosti. Každú z brán predstavuje vektor, ktorý je daný funkciou $\sigma()$ so vstupom x_t a výstupom h_{t-1}.

Zabúdacia brána $\vec{f}_t$ (rovnica (1)), rozhoduje, ktoré hodnoty sa z pamäti odstránia.

Vstupná brána $\vec{i}_t$ je daná rovnicou (1) a rozhoduje, ktoré hodnoty sa do pamäti pričítajú. Tieto hodnoty, ktoré sa uchádzajú o vstup do pamäti sú uložené vo vektore $\vec{g}_t$ (rovnica (5)).

Ďalšou časťou LSTM je pamäťová bunka C_t, ktorá je zodpovedná za udržiavanie histórie. V každom čase sa táto bunka aktualizuje pomocou zabúdacej brány $\vec{f}_t$, vstupnej brány $\vec{i}_t$ a vektoru hodnôt $\vec{g}_t$. Aktualizácia je daná rovnicou (3).

Poslednou časťou je výstupná brána $\vec{o}_t$, daná rovnicou (1), ktorá je filtrom pre výstup h_t. Výstup je daný rovnicou (4) a filtrom $\vec{o}_t$ sa na výstup dostávajú potrebné hodnoty. . Výstupom je pole obsahujúce pravdepodobnosti pre každý znak v slovníku. Na základe týchto pravdepodobností, je programom vybraný znak s najväčšou pravdepodobnosťou.

3. Dáta ako vstup pre model

Natrénovanie modelu vyžaduje podklad, na ktorom model trénuje. V tejto práci sa jedná o zbierku básní. Tieto dáta boli dôsledne vyberané a sú výhradne v slovenskom jazyku. Je dôležité, aby zbierka obsahovala básne od rôznych autorov, aby sa model priveľmi neupínal na štýl jedného spisovateľa. Program by tak nebol objektívny a imitoval by konkrétneho spisovateľa, čo je nežiadúci efekt. Väčšina básní bola použitá z elektronickej databázy Zlatý fond [6], ktorá obsahuje tvorbu rôznych slovenských spisovateľov.

Priemerný počet znakov vo verši je 30 – 40 a prie-

merne obsahuje strofa 6 veršov. Dáta v zbierke sa rozdeľujú do dvoch skupín: trénovacie a validačné. Trénovacie dáta slúžia na natrénovanie modelu, validačné na overenie trénovania. Celkový počet trénovacích dát je 800k znakov a 100k znakov validačných dát. Celá zbierka je kódovaná znakovou sadou UTF-8, čo umožňuje pracovať so slovenskou abecedou.

Zo zbierky básni je pre model vytvorený slovník, obsahujúci všetky znaky nachádzajúce sa v texte. Týmto znakom sú priradené indexy. Každý znak má jedinečný index, ktorý ho reprezentuje. Modelu tak nie sú predávané znaky, ale ich indexy. V zbierke je 124 rôznych znakov.

Pre trénovanie modelu je najprv nutné básne navzorkovať. Vzorky sú vstupy modelu na trénovanie. Tieto vzorky sa rozdeľujú na dve časti. Prvá časť je sekvencia niekoľkých znakov (kapitola 5), druhá časť je jeden znak nasledujúci za sekvenciou. Takto navzorkované básne sa odovzdávaju modelu na vstup. Dĺžka sekvencie je parameter ovplyvňújúci kvalitu trénovania. Na dĺžku sekvencie bol vykonaný experiment (viac v kapitole (5)).

4. Implementácia

Program je implementovaný v jazyku *Python* s využitím knižnice Keras [7], ktorá sa zaoberá modelovaním neurónových sietí. Ako backend pre túto knižnicu využívam Theano [8].

Program je rozložený do troch častí. Zostavenie modelu, trénovanie a generovanie. Môj model je momentálne zostavený z dvoch vrstiev LSTM so šírkou 400 skrytých jednotiek. Detaily k zostavovaniu modelu popíšem v kapitole (5).

Pre optimalizáciu modelu využívam algoritmus [9]. Tomuto algoritmu je nutné zadať hodnotu learning rate, ktorá ovplyvňuje, aké veľke kroky v n-dimenzionálnom priestore model robí pri trénovaní. K získaní tejto hodnoty som urobil experiment. (Viac v kapitole (5))

Trénovanie prebieha iteratívne. Doba trvania iterácie, nutný počet iterácii a teda celková doba trénovania sa odvíja od konfigurácie. S rastúcim počtom vrstiev LSTM a šírkou sa doba trvania iterácie zväčšuje, avšak najlepší výsledok je dosiahnutý za menej iterácií. Výsledný čas pre dosiahnutnie najlepšieho natrénovania tak môže byť nižší.

Úspešnosť trénovania je sledované hodnotami validačnou a trénovacou stratou (ang. loss). Najlepší výsledok je symbolizovaný najnižšou hodnotou validačnej straty. Pri klesajúcej trénovacej strate, 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 najlepšej validačnej straty.

5. Experimenty

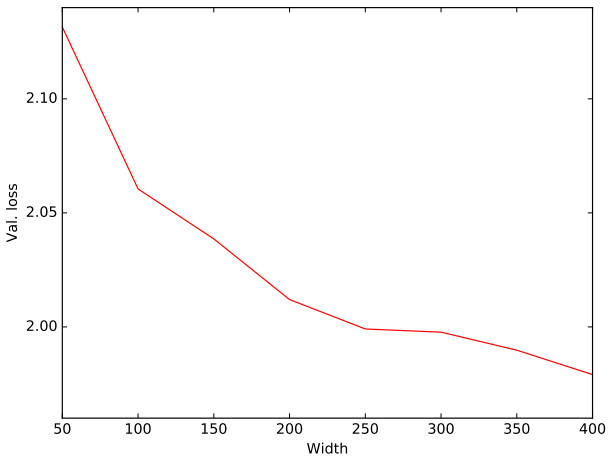
Správnu konfiguráciu pre zostavenie modelu nie je zväčša možné predpovedať vopred. Je nutné, pre dosiahnutie kvalitných výsledkov, s konfiguráciou experimentovať. Touto konfiguráciou sa myslí počet LSTM vrstiev, šírka týchto vrstiev, alebo learning rate (parameter riadiaci rýchlosť učenia).

Začiatkom experimentovania bolo zistiť počet vrstiev. Experiment (tabuľka 1) preukázal, že pre moje účely je lepšie mať minimálne 2 vrstvy. Avšak priveľký rozdiel vidieť nie je. Tri vrstvy narozdiel od dvoch, dosahujú lepšej hodnoty trénovacej straty, avšak vzniká pretrénovanie. Výsledky experimentov sú vyobrazená v iterácii s najlepšou validačnou stratou.

Tabuľka 1. Experiment - počet vrstiev

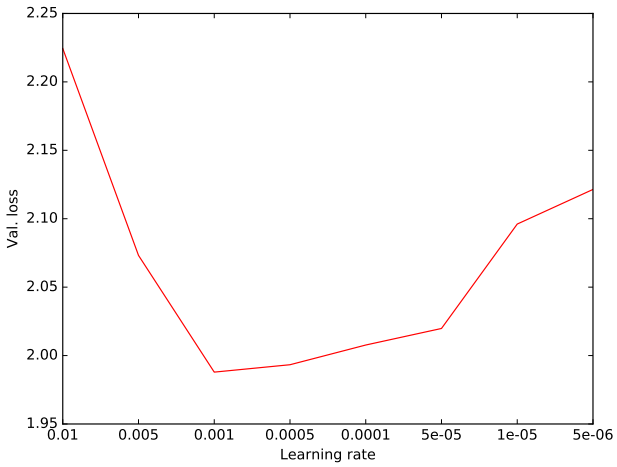
Počet vrstiev	val. strata	tren. strata
1	2.1548	1.9281
2	2.1195	**1.8454**
3	**1.7904**	1.9901

Po zistený dostačujúceho počtu vrstiev LSTM je vhodné zistiť šírku vrstiev. (Experimente č. 2 graf (3). Experimentov som vykonal celkovo osem s ôsmimi rôznymi šírkami. V grafe (3) je možné vidieť aký rozdiel šírka LSTM vytvorí.



Obrázok 3. Experiment č. 2 - šírka vrstiev LSTM. S rostúcou šírkou je dosahovaná presnosť väčšia.

Ďalšou úlohou bolo správne optimalizovať trénovanie. Experiment č. 3 (graf 4) sa zaoberal veľkosťou learning rate. Priveľmi malá hodnota spôsobí, že sa model nič nenaučí, avšak to isté spôsobí aj hodnota veľká.



Obrázok 4. Experiment č. 3 - hodnota learning rate. Priveľmi vysoká, alebo nízka hodnota learning rate nedosahuje dostatočnej presnosti.

Ďalším možným prvkom na ovplyvnenie trénovania a výsledku je dĺžka vstupnej sekvencie. Tento parameter predstavuje počet znakov daný modelu na vstup. Experiment č. 4 na zistenie správnej dĺžky sekvencie bol vykonaný, avšak zmena dĺžky nepredstavovala výrazné rozdiely v učení. Preto pracujem s krátkou sekvenciou, ktorá má kratší čas učenia.

6. Generovanie básne

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. Model s nulovými skrytými stavmi by prakticky generoval vždy to isté. Vstupný text je navzorkovaný podobne ako pri trénovaní a znakom sú priradené indexy. Takto upravený vstupný text je daný modelu na vstup. Model vracia pole pravdepodobností, jednu hodnotu pre každý znak. Každá pravdepodobnosť je upravená hodnotou α, ktorá ovplyvňuje veľkosť pravdepodobností.

$$p' = exp\left(\frac{\ln p}{\alpha}\right) \tag{5}$$

kde p je pravdepodobnosť znaku. S rastúcou hodnotou α sa zväčšujú pravdepodonosti 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. Program generuje básne s rôznou hodnotou α, hodnota sa mení vždy po vygenerovaní básne, užívateľ si môže vybrať.

Pravdepodobnosti po úprave hodnotou α sú exponenciálnou funckiou vrátené do intervalu $[0,1]$ a vyberá sa index s najväčšou pravdepodobnosťou. Index je prevedený na znak a pripísaný k počiatočnému textu. Generovanie ďalšieho znaku je teda vždy ovplyvnené aj predchádzajúcimi vygenerovanými znakmi. Vygenerovaný text:

Keď sa to zabudne národného,
anjelsko, vestu je vernosti.
A zrak tmavá zem túžbu práva,
keď s rozvôľu svoj svätý som pristrieľ,
ja som piesne milostive,
pri oknom dievke spievali.

7. Záver

Cieľom práce bolo 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.

Program bol implementovaný. 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.

V budúcej práci chcem pokračovať v experimentovaní s learging rate, konkrétne použiť meniaci sa learning rate počas trénovania, pokračovať v zbieraní básní a trénovať na väčšom množstve dát a vytvoriť webovú aplikáciu na generovanie básní.

Poďakovanie

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

Literatúra

[1] Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *International Conference on Entertainment Computing*, pages 191–196. Springer, 2009.

[2] Ruli Manurung, Graeme Ritchie, and Henry Thompson. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64, 2012.

[3] Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. i, poet: Automatic chinese poetry composition through a generative

summarization framework under constrained optimization. In *IJCAI*, 2013.

[4] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] *Zlatý fond denníka SME - Najväčšia slovenská elektronická knižnica.* [Online; navštíveno 15.12.2016].

[7] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[8] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.