

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Bc. Igor Šimko

Generovanie nadpisov kategórií textových dokumentov

Diplomová práca

Študijný program: Informačné systémy
Študijný odbor: 9.2.6 Informačné systémy
Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU, Bratislava
Vedúci práce: Ing. Michal Holub
december 2017

«zadanie»

Anotácia

Generovanie nadpisov kategórií textových dokumentov

Študijný program: Informačné systémy

Autor: Bc. Igor Šimko

Vedúci práce: Ing. Michal Holub

december 2017

Cieľom diplomovej práce je navrhnúť riešenie pre generovanie nadpisov kategórií pre zhluky textových dokumentov s využitím metód hĺbkového učenia. Práca teda rieši generovanie textových sekvenčí zmysluplné popisujúcich určité zhluky dokumentov na základe ich spoločných črt. Navrhnutý model pozostáva z dvoch častí. V prvej časti je nutné vytvoriť jednotnú textovú reprezentáciu dokumentov a v tej druhej vytvoriť nadpisy opisujúce významosť dokumentov s využitím umelých neurónových sietí. Správnosť navrhnutého riešenia je overovaná s využitím štandardných metrič pre výhodnocovanie strojovo vyprodukovaných textov. Výsledný model je demonštrovaný na vybranom voľne dostupnom súbore dát.

ANNOTATION

Generating category titles for text documents

Degree Course: Information systems

Author: Bc. Igor Šimko

Supervisor: Ing. Michal Holub

December 2017

The aim of the work is to propose a solution for generation headings for these clusters (categories) using machine learning. Thus, work addresses generation of text sequences that contains description of certain groups of documents based on their common features. The proposed model consists of two parts. In the first part, it is necessary to create a uniform text representation of documents and in the second part to create heading describing the significance of documents using artificial neural networks. The correctness of the proposed solution is verified using standard methods for evaluating machine-generated text. The resulting model is demonstrated on selected freely available data.

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že na diplomovej práci som pracoval samostatne na základe vlastných teoretických a praktických poznatkov, konzultácií a štúdia odbornej literatúry, ktorej úplný prehľad je uvedený v zozname použitej literatúry.

Bratislava, 13. december 2017

.....

Bc. Igor Šimko

Pod'akovanie

Tento cestou by som sa chcel pod'akovať Ing. Michalovi Holubovi za pomoc, odborné vedenie, trpezlivosť pri konzultáciách, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

Obsah

1	Strojové učenie	1
1.1	Učenie sa s učiteľom	1
1.2	Učenie sa bez učiteľa	2
2	Zhlukovanie	3
2.1	Techniky zhlukovania	3
3	Umelé neurónové siete	5
3.1	Stručná história	5
3.2	Reprezentácia neurónovej siete	5
3.3	Model neurónu	7
3.4	Aktivačné funkcie neurónu	8
3.5	Architektúry neurónových sietí	9
4	Analýza existujúcich riešení	13
4.1	Dostupné dátové súbory	13
4.2	Predspracovanie	14
4.3	Generovanie textovej sekvencie	17
4.4	Sumarizácia textu	18
4.5	Kategorizácia dokumentov	19
4.6	Metriky výhodnocovania	19
4.7	Využívané modely	21
4.8	Zhrnutie analýzy	22
5	Návrh riešenia	24
5.1	Celkový pohľad na model	24
5.2	Predspracovanie dát	25
5.3	Zjednotenie dokumentov	29
5.4	Reprezentácia dokumentu	31
5.5	Architektúra kódér-dekódér	34
6	Overenie riešenia a experimenty	37

6.1	Implementácia modelu	37
6.2	Úvodné pozorovania	40
6.3	Počiatočné výsledky	44
7	Zhrnutie a ďalšia práca	50
7.1	Implementácia jednotnej reprezentácie dokumentov	50
7.2	Aplikovanie mechanizmu pozorovania	50
7.3	Evaluácia	51
A	Príloha - Technická dokumentácia	54
B	Príloha - Nasledujúci plán práce	60
	Použitá literatúra	61

Zoznam obrázkov a tabuľiek

Obrázok 2.1 - Hierarchické zhľukovanie	4
Obrázok 2.2 – K-means zhľukovanie	4
Obrázok 3.1 - Základná vizualizácia umelej neurónovej siete	6
Obrázok 3.2 - Model neurónu	7
Obrázok 3.3 - Architektúra konvolučnej NS	11
Obrázok 3.4 - Ukážka Elmanovej a Jordenovej siete	12
Obrázok 4.1 - Ukážka vektorovej reprezentácie - word2vec	16
Obrázok 4.2 - Ukážka vektorovej reprezentácie slov - GloVe	17
Obrázok 5.1 - Model navrhovaného riešenia	25
Obrázok 5.2 - Prevod textu na indexy	32
Obrázok 5.3 - Vektorová reprezentácia najfrekventovanejších slov	33
Obrázok 5.4 - Diagram aktivít pre spracovanie dokumentov	34
Obrázok 5.5 - Architektúra kóder-dekóder	36
Obrázok 6.1 - Diagram tried pre implementovaný model	38
Obrázok 6.2 - Ukážka webovej aplikácie pre spracovanie článkov	39
Obrázok 6.3 - Histogram početnosti prvých 10 kategórií	41
Obrázok 6.4 - Pomer kategórií ku počtu technických článkov	43
Obrázok 7.1 - Výrez dátového modelu Wikipédie	54
Tabuľka 3.1 - Aktivačné funkcie neurónu	8
Tabuľka 5.1 - Ukážka transformácie textu	28
Tabuľka 5.2 - Zoznam použitých regulárnych výrazov	29
Tabuľka 6.1 - Konfigurácia hyper-parametrov	45
Tabuľka 6.2 - Ukážka generovaných sekvencií	46
Tabuľka 6.3 - Konfigurácia hyper-parametrov	47
Tabuľka 6.4 – Generované názvy kategórií pre model bez mechanizmu pozorovania	47
Tabuľka 6.5 - Konfigurácia hyper-parametrov	48
Tabuľka 6.6 - Ukážka generovaných sekvencií	48

Tabuľka 6.7 - Konfigurácia hyper-parametrov	49
Tabuľka 6.8 - Ukážka generovaných sekvenčí	49

Zoznam použitých skratiek

TF	Term Frequency
IDF	Inverse Term Frequency
LDA	Latentná Dirichletová alokácia
LSA	Latentná sémantická analýza
SVM	Support vector machine
VSM	Vector Space Model
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NS	Neurónová sieť
RNS	Rekurentná neurónová sieť
CNS	Konvolučná neurónová sieť

Úvod

Vďaka narastajúcej sile výpočtovej techniky a veľkosti dnešných dátových súborov sa téma umelých neurónových sietí stáva čoraz zaujímavejšia. S narastajúcim počtom dokumentov v digitálnej podobe je nevyhnutné, pre jednoduchšiu a efektívnejšiu prácu, kategorizovať tieto dokumenty do tém a pod-tém. Manuálna kategorizácia textových dokumentov môže byť veľmi zdĺhavá. Z týchto dôvodov sa mnohí snažia tento proces zautomatizovať.

Navrhnutie, resp. tvorba počítačového programu, ktorý bude generovať súvislý, zmysluplný text alebo len vetu je náročný problém. V našom prípade ide konkrétnie o vytváranie nadpisov kategórií. Tieto nadpisy musia nie len popisovať jednotlivé kategórie textových dokumentov, ale aj slová v týchto nadpisoch musia byť logicky prepojené a tvoriť zmysluplný opis skupiny (kategórie) dokumentov. Techniky strojového učenia sa ukázali byť vhodné pre generovanie strojovo-vytvoreného textu ale aj pre samotné spracovanie prirodzeného jazyka.

Cieľom práce je navrhnúť model neurónovej siete, ktorý pre vstupný zhluk textových dokumentov vygeneruje názov kategórie jednotne opisujúci všetky vstupné dokumenty. Navrhnutý model nerieši priamo zhlukovanie dokumentov ako takých. Predpokladáme teda, že dokumenty, ktoré budú vstupovať do neurónovej siete už budú zoskupené na základe spoločných črt. Okrem samotného generovania výstupnej textovej sekvencie je potrebné navrhnúť metódu pre jednotnú reprezentáciu zhlukov (kategórií) dokumentov.

1 Strojové učenie

Pod pojmom strojové učenie môžeme rozumieť schopnosť počítača, resp. počítačového programu učiť sa bez toho, aby bol tento program tomu priamo naprogramovaný. Tento pojem sa v dnešnej dobe spája s umelou inteligenciou, a metódy a algoritmy strojového učenia sa využívajú v rozličných odvetviach ako je napríklad chémia, biológia, alebo rôzne technologické odvetvia.

Strojové učenie je využívané aj pri problémoch späťich s utriedovaním textových dokumentov do kategórií podľa ich obsahov v spojení s generovaním logických označení týchto dokumentov. Aj keď je výstup takéhoto utriedenia len pomenovanie určitej kategórie, poprípade krátky textový popis, pri strojovom učení sú k tomuto určeniu využívané podobnosti a atribúty naprieč celým dokumentom.

Bežné rozdelenie úloh strojového učenia sa delí do dvoch základných skupín:

- Učenie sa s učiteľom
- Učenie sa bez učiteľa

1.1 Učenie sa s učiteľom

Kontrolované učenie (z angl. „supervised learning“), resp. učenie s učiteľom je jednou z úloh strojového učenia pri ktorej program využíva sadu trénovacích dát. Tieto dátá pozostávajú z radu trénovacích príkladov, ktoré sa delia na vstupné a výstupné dátá. Tieto dátá sa zväčša reprezentujú ako vektory, kde pre každú vstupnú hodnotu prislúcha požadovaná výstupná hodnota. Požadovaný výsledok je taký, že algoritmus vie priradiť k vstupnej hodnote správnu výstupnú hodnotu a určiť tak označenie danej triedy.

Pre kontrolované učenie existuje viacero algoritmov a prístupov, ktoré sa využívajú práve na tento typ problému. Spomenúť môžeme najmä umelé neurónové

siete, ktoré sú bližšie opísane v časti Umelé neurónové siete, rozhodovacie stromy, SVM, lineárna regresia alebo náhodný les (z angl. „random forest“). Klasifikácia či regresia sú prístupy, ktoré môžeme zaradiť práve do skupiny úloh kontrolovaného učenia.

1.2 Učenie sa bez učiteľa

Nekontrolované učenie (z angl. „unsupervised learning“) je typ úlohy strojového učenia, pre ktorú platí fakt, že program, resp. algoritmus nedostáva sadu príkladov ako to bolo v prípade učenia sa s učiteľom. Nemá teda žiadnu informáciu, podľa ktorej by vedel vykonať správnu akciu. Inými slovami, cieľom nekontrolovaného učenia je namodelovať štruktúru alebo distribúciu dát a lepšie tak pochopiť samotné dátá. Nekontrolované učenie môžeme podobne rozdeliť do dvoch skupín a to: zhlukovanie a asociačné pravidlá. Pri tomto type strojového učenia sa využívajú algoritmy ako algoritmus najbližších K-susedov, K-means algoritmus, taktiež neurónové siete alebo apriórny algoritmus.

2 Zhlukovanie

Niekedy sa zhlukovanie (z angl. „clustering“) zvykne označovať ako zhluková analýza. Hlavným cieľom, ktorý sa snažíme pri zhlukovaní dosiahnuť, je určité zoskupenie objektov do spoločných skupín, na základe špecifických vzťahov medzi týmito objektami.

Zhlukovanie patrí do kategórie strojového učenia bez učiteľa. Na rozdiel od klasifikácie, pri zhlukovaní nie sú vopred známe triedy pre objekty a preto sa musí algoritmus naučiť aké triedy zhluk obsahuje.

2.1 Techniky zhlukovania

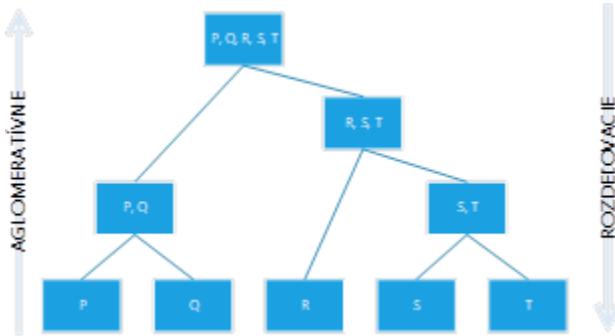
Medzi jedny z techník zhlukovania patria hierarchické a K-means metódy.

2.1.1 Hierarchické zhlukovanie

Taktiež zhlukovanie na základe spojenia, techniky sú založené na základnom princípe, že objekty ktoré majú k sebe bližší vzťah (sú si viac podobné) budú mať od seba menšiu vzdialenosť ako tie, ktoré majú menej spoločných vlastností. Výsledok hierarchického zhlukovania môže byť strom, ktorý sa nazýva dendrogram. Tento strom zobrazuje spájanie jednotlivých zhlukov. Existujú dva prístupy pre generovanie hierarchických zhlukov (Obrázok 2.1):

Agglomeratívne: Začína s bodmi, ktoré reprezentujú jednotlivé zhluky a zlučuje najpodobnejšie alebo najbližšie páry zhlukov. Proces trvá až kým sa nevytvorí len jeden zhluk. Tento prístup si vyžaduje definovanie podobnosti alebo vzdialenosť zhlukov.

Rozdeľovacie: Môžeme povedať, že ide o opak agglomeratívneho prístupu. Proces začína jediným zhlukom a postupne sa rozdeľuje, až pokial nie je každý objekt v nezávislom zhluku.

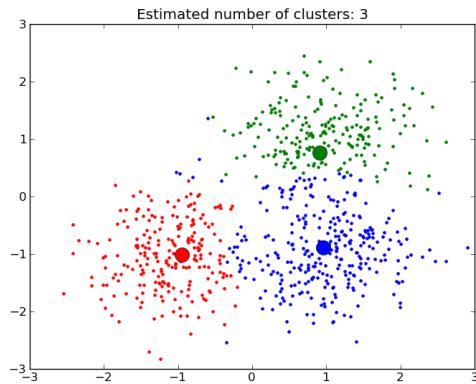


Obrázok 2.1 - Hierarchické zhlukovanie

2.1.2 K-means zhlukovanie

Existuje viacero patriciových techník avšak budeme opisovať len algoritmus K-means, ktorý je najpoužívanejší pre zhlukovanie textových dokumentov. K-means je založený na myšlienke, že stredový bod môže reprezentovať zhluk. Pre K-means používame najmä pojem centroid, ktorý je mediánom skupiny bodov. „Zaujímavosťou je fakt že centroid takmer nikdy nezodpovedá skutočnému dátovému bodu.“ [13]

Nasledujúci Obrázok 2.2 znázorňuje rozdelenie objektov do troch skupín na základe ich spoločných vlastností, resp. rozdielností k ostatným zhlukom spolu s centroidom v každom zhluku.



Obrázok 2.2 – K-means zhlukovanie

[zdroj: scikit-learn.sourceforge.net/]

3 Umelé neurónové siete

Podkladom pre neurónové siete sú neurofyziologické poznatky, resp. spracovanie informácií v nervových bunkách. V nasledujúcich častiach sú opísané základné pojmy z oblasti umelých neurónových sietí, zahŕňajúc základnú reprezentáciu ako aj rôzne architektúry neurónových sietí.

3.1 Stručná história

V roku 1943 na univerzite Chicagu sa skupina amerických vedcov pod vedením McCullocha (psychiater a neuroanatóm) a Pittsa (matematik) začala zaujímať o spojenie biologickej nervovej sústavy s matematickou logikou. Vo svojej práci ukázali, že model, ktorý má dostatočný počet neurónov a správnych spojení (synáps), dokáže vypočítať v podstate ľubovoľnú (vypočítateľnú) funkciu. Tieto neuróny mali na výstupe 0 alebo 1 v závislosti na tom, či vážená suma vstupných signálov prekročila prahovú hranicu.

Táto práca slúžila ako podklad pre ďalšie výskumy v oblasti neurónových sietí a umelej inteligencie. V roku 1958 matematik Frank Rosenblatt popísal doprednú neurónovú sieť – perceptrón (viac v časti 3.4). Táto sieť mala však svoje obmedzenie. Za jej pomoci bolo možné riešiť len lineárne separovateľné problémy (napr. logický AND alebo OR).

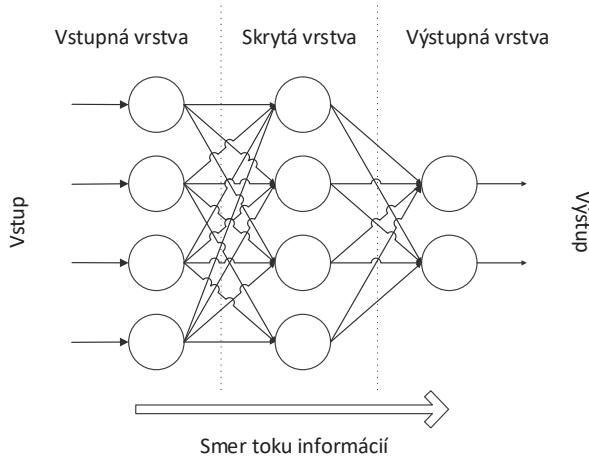
3.2 Reprezentácia neurónovej siete

Formálne je možné opísť neurónovú sieť pomocou orientovaného grafu, ktorý má presne dané pravidlá. Pomocou orientovaného grafu vieme znázorniť tok informácií a aká je základná architektúra danej siete. Jednotlivé neuróny môžeme rozdeliť na:

- **vstupné** – informácia prichádza z externého prostredia a výstup väčšinou smeruje k ďalším neurónom

- **skryté** - informácia prichádza z iných neurónov ale aj vonkajšieho prostredia a pokračuje do iných neurónov
- **výstupné** – podobné skrytým neurónom s rozdielom, že výstup smeruje do externého prostredia

Z týchto neurónov sú tvorené vrstvy neurónových sietí, ktoré sa rovnako rozdeľujú na vstupné skryté a výstupné. Na obrázku Obrázok 3.1 sú znázornené spomínané vrstvy NS spolu s ich neurónmi. Vstupná informácia sa z externého prostredia dostáva do vstupnej vrstvy a prechádza cez skrytú až do výstupnej vrstvy a následne do externého prostredia. Takéto šírenie informácie resp. dát sa nazýva *dopredné šírenie* a využíva sa pri dopredných neurónových sieťach.

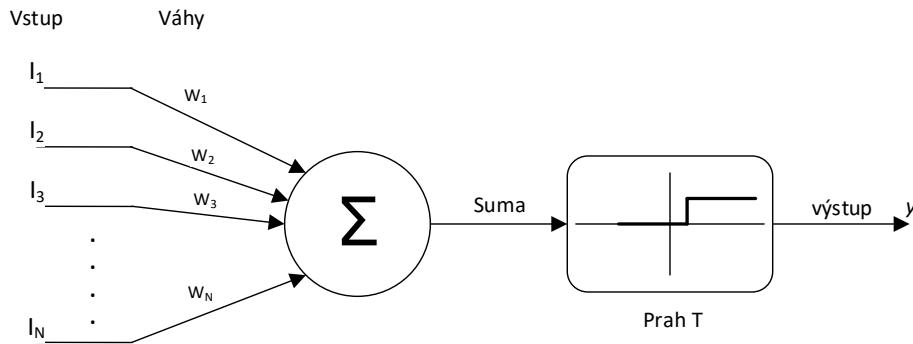


Obrázok 3.1 - Základná vizualizácia umelej neurónovej siete

Na základe týchto informácií môžeme usúdiť že „*NS určená ako orientovaný graf $G = (V, E)$, kde výrazy $V = \{v_1, v_2, \dots, v_N\}$ a $E = \{e_1, e_2, \dots, e_M\}$ označujú neprázdnú vrcholovú množinu resp. hranovú množinu grafu G obsahujúceho N vrcholov (neurónov) a M hrán (spojov). Každý spoj $e \in E$ sa interpretuje ako usporiadaná dvojica dvoch neurónov z množiny V , $e = (v, v')$. Hovoríme, že spoj e začína v neuróne v a končí v neuróne v' .*“ [10]

3.3 Model neurónu

Existuje veľké množstvo neurónov. Tie najjednoduchšie používajú nespojité prenosové funkcie. Zložitejšie opisujú všetky detaile správania sa neurónu živého organizmu. Na obrázku Obrázok 3.2 je vizualizácia základného modelu neurónu, ktorú definovali McCulloch a Pitts. Ide o najpoužívanejší model umelého neurónu.



Obrázok 3.2 - Model neurónu

/zdroj: wwwold.ece.utep.edu.org/

Ide o neurón, ktorý pozostáva zo sady N vstupov I_1, I_2, \dots, I_N a jediného výstupu y . Výstup y nadobúda binárne hodnoty. Funkciu môžeme matematicky vyjadriť ako:

$$y = f \left(\sum_{i=1}^N I_i W_i \right) \quad (3.1)$$

kde, W_1, W_2, \dots, W_N sú normalizované ováhované hodnoty z intervalu $(0,1)$ alebo $(-1, 1)$. Suma je váhovaná suma a T je prahová konštantă, resp. prah (z angl. „threshold“). Funkcia f je kroková funkcia (viď. binárny skok v Aktivačné funkcie neurónu) s prahom $T = 0$

3.4 Aktivačné funkcie neurónu

Pri NS sa častokrát využívajú rôzne aktivačné funkcie. Najpoužívanejšou je funkcia sigmoid. Avšak táto funkcia môže byť nahradená rôznymi inými v závislosti na type riešeného problému. Nasledujúca tabuľka zobrazuje prehľad vybraných aktivačných funkcií, spolu s ich vizualizáciou a funkciami.

Tabuľka 3.1 - Aktivačné funkcie neurónu
 [zdroj:https://en.wikipedia.org/wiki/Activation_function]

Binárny krok		$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$
Sigmoida		$f(x) = \frac{1}{1 + e^{-x}}$
ReLU funkcia		$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$
Gausová funkcia		$f(x) = e^{-x^2}$

Softmax

Ide o zovšeobecnenie logistickej funkcie, ktorá je vhodná pre klasifikačné úlohy. Nech $\sigma(z)_j$ označuje funkciu softmax, kde z je vstup a j označuje špecifickú triedu, pre ktorú má byť vypočítaná hodnota softmax. Potom platí vzťah:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (3.2)$$

Pri klasifikácii textových dokumentov, j bude reprezentovať počet slov zo slovníka. V prípade použití veľmi veľkej slovnej sady bude komplexita narastať kvôli vysokému počtu parametrov v sieti.

3.5 Architektúry neurónových sietí

Neurónové siete, ktoré boli opísané v časti Umelé neurónové siete majú mnoho typov architektúr. V nasledujúcich častiach sú opísané základne najpoužívanejšie modely, ktoré sa využívajú aj pri riešení problémov späťých so spracovaním prirodzeného jazyka (NLP).

3.5.1 Perceptrón

Ide o najjednoduchší model doprednej neurónovej siete s učiteľom. Táto NS obsahuje len jediný neurón, ktorý na vstupe prijíma vektor $\bar{x} = (x_1, x_2, \dots, x_{n+1})$ prostredníctvom synaptických váh, ktoré tvoria tzv. váhový vektor $\bar{w} = (w_1, w_2, \dots, w_{n+1})$ a doplnkový bias neurón¹. Tento vstupný vektor nazývame vzor (z angl. „pattern“). Jednotlivé zložky vzoru nadobúdajú reálne alebo binárne hodnoty.

„Výstup perceptronu o je daný vzťahom:

$$o = f(\text{net}) = f(\bar{w} \cdot \bar{x}) = f\left(\sum_{j=1}^{n+1} w_j x_j\right) = f\left(\sum_{j=1}^n w_j x_j - \theta\right) \quad (3.3)$$

kde premenná **net** označuje váhovanú sumu vstupov, t.j. skalárny (zložkový) súčin váhového a vstupného vektora. Funkcia f sa volá **aktivačná funkcia** perceptronu.“ [12]

¹ Doplňkový neurón s hodnotou +1

V tomto prípade ide o funkciu sigmoid a platí, že:

$$f(\text{net}) = \text{sign}(\text{net}) = \begin{cases} +1, & \text{ak } \text{net} \geq 0 \\ -1, & \text{ak } \text{net} < 0 \end{cases} \quad (3.4)$$

„Rovnica

$$\sum_{j=1}^n w_j x_j - \theta = 0 \quad (3.5)$$

je rovnicou nadroviny v n -rozmernom priestore. Napr. v 2-rozmernom priestore je to rovnica priamky: $w_1 x_1 + w_2 x_2 - \theta = 0$. Váhy perceptrónu predstavujú koeficienty deliacej priamky (nadroviny), ktorá rozdeľuje priestor vzorov na dva pod-priestory.“ [12]

Pomocou perceptrónu dokážeme klasifikovať vzory do dvoch tried, teda vieme riešiť len lineárne separovateľné problémy.

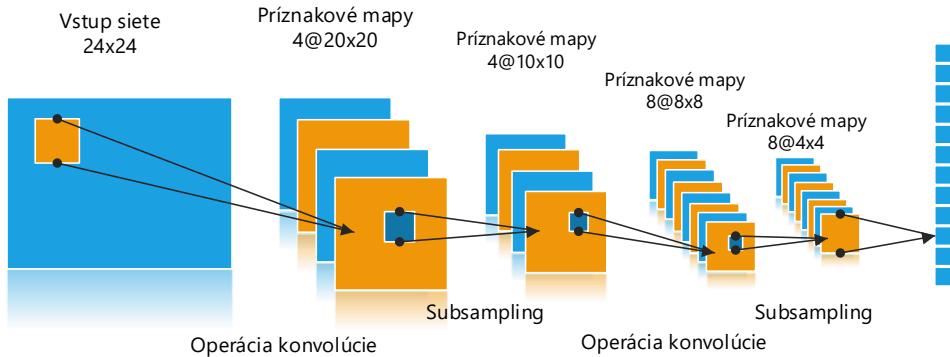
3.5.2 Konvolučné neurónové siete

Špeciálnym druhom viacvrstvových sietí sú konvolučné neurónové siete. Táto sieť našla svoje uplatnenie najmä v oblasti počítačového videnia. Avšak momentálne sa objavujú aj úspešné riešenia, ktoré využívajú konvolučné siete pri spracovaní reči a prirodzeného jazyka.

Základnou vlastnosťou, ktorá je prebratá z vizuálnej mozgovej časti, je lokálne prepojenie. Toto prepojenie umožňuje neurónu v jeho skrytej vrstve napojenie na oblasť neurónu vo vrstve predchádzajúcej. Ďalšou nezanedbateľnou vlastnosťou je fakt, že vrstvy môžu mať n -dimenzionálny charakter, a že neuróny v danej hĺbke majú zdieľané váhy.

Konvolučné siete sa postupne snažia učiť čoraz zložitejšie príznaky. V princípe to funguje tak, že v prvej vrstve (konvolučná vrstva) sieť kóduje nízkoúrovňové príznaky akými sú napríklad detekcie hrán či jednoduché prechody farieb. Vo vrstve subsamplingu, NS zmenšuje príznakové mapy. V tých ďalších môžu byť tieto príznaky napríklad rôzne tvary či viacfarebné prechody. Komplexné tvary pre

konkrétnie objekty nachádzame v posledných vrstvách. Na Obrázok 3.3 je znázornená architektúra opísanej konvolučnej NS.



Obrázok 3.3 - Architektúra konvolučnej NS

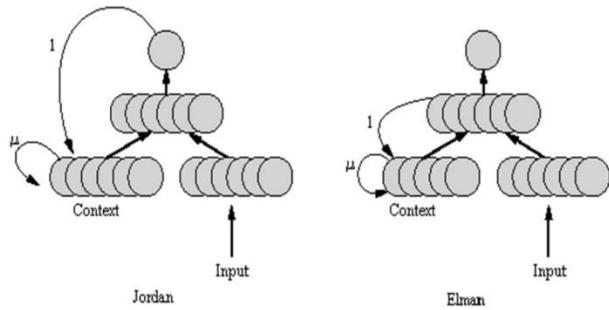
3.5.3 Rekurentné neurónové siete

V prípade rekurentných neurónových sietí RNS, na ich výstupe záleží nie len momentálny vstup siete ale aj história doterajších predkladaných vzorov. K jednému vstupu teda môže a nemusí prislúchať viacero výstupov v závislosti na časovom kontexte. Plusom tohto typu neurónových sietí je fakt, že dĺžka sekvencie sa v rámci trénovacej množiny môže lísiť. To znamená, že vstup nie je obmedzený, resp. nemusí byť fixný. Pozostáva teda zo skrytého stavu a voliteľného vstupu o variabilnej dĺžke sekvencie. [7]

RNS sa často využíva pri spracovaní prirodzeného jazyka nakoľko model ráta s variabilnou dĺžkou viet. Medzi najznámejšie RNS patria Elmanové a Jordanové siete:

- **Elmanová sieť** je trojvrstvová NS, ktorá využíva tzv. kontextovú vrstvu. Táto vrstva je prepojená so skrytou s fixnými váhami. Takto môže NS udržiavať určitý stav, ktorý jej umožní riešiť aj úlohu akou je napríklad predikcia sekvencie. [17]

- **Jordanová siet** je podobná ako Elmanová siet, avšak v tomto prípade je kontextová vrstva napojená na výstupnú. Kontextová vrstva je pri Jordanovej vrstve označovaná tiež ako stavová vrstva, ktorá má rekurentné prepojenie bez nadbytočných uzlov v tomto spojení. [17]



Obrázok 3.4 - Ukážka Elmanovej a Jordenovej siete

[zdroj: <https://www.intechopen.com/>]

LSTM

Medzi rekurentné neurónové siete patrí špeciálna architektúra, ktorá má schopnosť pamätať si určité hodnoty počas dlhej doby, a iné okamžite zahadzovať. LSTM obsahujú informácie mimo normálneho toku rekurentnej siete a to v uzavorenenej bunke. Tieto informácie môžu byť uložené alebo prečítané, podobne ako pamäť v počítači.

Bunka má funkciu rozhodovať o tom čo má ukladať a kedy umožniť čítanie, zapisovanie a vymazávanie. Túto funkciu vykonáva pomocou brán, ktoré sa zatvárajú a otvárajú. Brány sú analógové a implementované pomocou sigmoid.

4 Analýza existujúcich riešení

Existujúce riešenia využívajú rôzne metódy akými sú NS alebo summarizačné modely. V nasledujúcich častiach sú opísané jednotlivé procesy jednako pre spracovanie textu a prirodzeného jazyka ale aj pre tvorbu NS, resp. aké architektúry NS sú využívané pri riešení problému s generovaním logických slovných spojení pre kategórie textových dokumentov.

4.1 Dostupné dátové súbory

Kvalita súboru dát častokrát môže výrazne ovplyvniť výsledky implementovaných riešení. Je nutné vybrať špecifické dáta pre konkrétné typy problémov. Je dôležité si uvedomiť ako sú dáta rozdelené a ako sú vyvážene vzhlľadom na počet tried. Pri textových dokumentov sa využívajú nasledujúce voľne dostupné zdroje:

- **20 newsgroup:** Ide o jeden z najpoužívanejších zdrojov pre riešenie problémov s klasifikáciou dokumentov. Obsahuje približne 20 000 dokumentov, ktoré sú rozdelené do približne 20 kategórií.
- **Reuters:** Taktiež často používaný zdroj dát pre klasifikáciu textových dokumentov. Obsahuje 21 578 dokumentov, ktoré sa nachádzajú v 135 kategóriách. Dáta sú rozdelené na trénovacie (13 625) a testovacie (6 188). Súbor dát je veľmi nevyvážený a obsahuje veľký počet rôznych variácií kategórií.
- **Gigaword Fifth Edition:** Obsahuje vyše 10 miliónov článkov z novín so širokým korpusom slov a k nim korešpondujúcim nadpisom.
- **Wikipédia:** Verzia databázy z dňa 12.6.2017 obsahuje takmer 1 400 000 článkov a vyše 4 milióny revízií. Články sú písané vo formáte wiki-markup (značkovací jazyk) a patria do viacerých kategórií

4.2 Predspracovanie

Väčšina riešení, ktoré generujú či už nadpisy konkrétnych článkov alebo klasifikujú resp. zhľukujú dokumenty, v sebe zahŕňajú aj proces predspracovania týchto textov. V zásade sa jedná takmer vždy o podobný proces.

Počas tohto procesu predspracovania sa napríklad konvertujú všetky slová na malé, tzv. lower-casing a odstraňujú sa nadbytočné číslice, ktoré sa zvyknú nahradzovať nulou. Tento krok musíme vykonávať ešte pred samotnou tokenizáciou slov, ak chceme znížiť počet unikátnych slov. Takýto postup môže vidieť v práci [14]. Predspracovanie dát však záleží na konkrétnych dátových súboroch.

4.2.1 TF-IDF

Po úvodnom spracovaní textu sa určuje tzv. početnosť slov v dokumente (z angl. „term-frequency“) a inverzná početnosť slov v dokumente (z angl. „inverse-term-frequency“). Pomocou tejto metódy vieme z textu vymazať málo významné slová, a teda vieme určiť dôležitosť slov. Použitím tejto metódy vieme vytvoriť základný vektor, pomocou ktorého môžeme reprezentovať textový dokument. TF-IDF sa počíta ako súčin jednotlivých častí (TF a IDF) a platí, že:

$$tf(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)} \quad (1.1)$$

kde $f_d(t)$ je výskyt výrazu v dokumente a $\max_{w \in d} f_d(w)$ je maximálny počet výskytu slova w v tom istom dokumente. Pre IDF ďalej platí:

$$idf(t, D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right) \quad (1.2)$$

kde D je korpus všetkých dokumentov a čím sa slovo vyskytuje v dokumentoch častejšie tak je menej dôležité. Pre výsledný TF-IDF platí teda:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (1.3)$$

Pomocou týchto vzťahov vieme vypočítať *TF-IDF* hodnoty pre všetky slová v dokumente. Čím je hodno zredukovať počet črt (z angl. „feature“) v dokumente.

4.2.2 Tokenizácia a lemmantizácia

Princípom tokenizácie je rozdeliť text do slov, fráz alebo symbolov, ktoré sa označujú pojmom *token*. Za token sa v automatickej segmentácii textu považuje akýkoľvek reťazec znakov medzi dvoma bielymi znakmi (z angl. „whitespace“), aj jednotlivé znaky interpunkcie, ktoré nemusia byť oddelené medzerou od predchádzajúceho alebo nasledujúceho tokenu. Text sa teda z formálneho hľadiska skladá z tokenov a medzier (whitespace) [15].

Pojmom lemantizácia označuje proces, ktorým prevedieme slovo do základného tvaru. Napríklad slovo „textových“ prevedieme na slovo „text“.

4.2.3 Reprezentácia slov

Pri spracúvaní prirodzeného jazyka sú využívané neurónové siete ku konštrukcií modelu jazyka. Pri neurónových sietach určených k tomuto typu problému sú podmienené pravdepodobnosť určené z množiny trénovacích dát. V modeli jazyka neurónových sietí je diskrétny vektor dokumentu, ktorý je definovaný na základe výskytu termínov a fráz, pretransformovaný do plynulého vektoru dokumentu s využitím natrénovanej neurónovej siete. Po tomto je tento vektor použitý ako výstup klasifikátora či už pre klasifikáciu textu, získania alebo filtrovania informácií z textu.

[6]

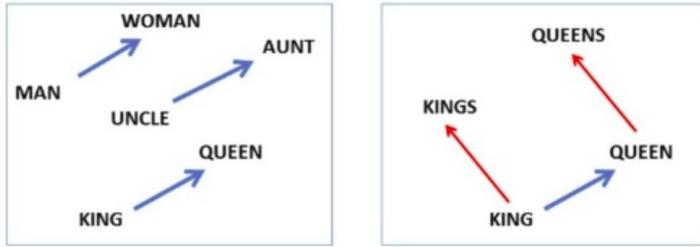
Word2Vec

V nedávnej dobe zamestnanec spoločnosti Google Mikolov, spolu s ďalšími vytvoril voľne dostupnú knižnicu s názvom Word2Vec určenú práve pre hĺbkové učenie. Ide

o jednoduchú RNS, ktorá dokáže mapovať slová na reálne vektory na základe určitej podobnosti slov v texte. Najväčšou výhodou tohto modelu je jeho vylepšenie v presnosti s nižším výpočtovým výkonom [3]. Nakoľko sú slová reprezentované vektormi, môžeme nad nimi robiť vektorové operácie. Príklad takýchto operácií je nasledujúci:

$$vec(\text{„man“}) - vec(\text{„king“}) + vec(\text{„woman“}) = vec(\text{„queen“}) \quad (4.4)$$

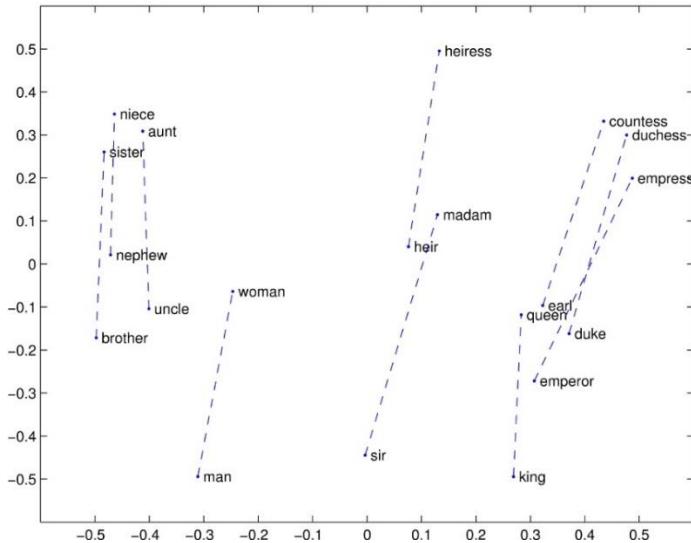
Na obrázku Obrázok 4.1 je graficky znázornená predchádzajúca operácia s vektormi *man – king* a *woman – queen*.



Obrázok 4.1 - Ukážka vektorovej reprezentácie - word2vec

Glove

Ide o algoritmus bez učiteľa, pomocou ktorého je možné vytvoriť vektorovú reprezentáciu slov. Trénovanie sa uskutočňuje na agregovaných globálnych štatistických vektoroch, z určitého korpusu. Okrem využitia priamo trénovacieho algoritmu, je možné použiť predtrénované algoritmy o veľkostiach dimensií 50, 100, 200 a 300. Veľkosť dimenzie je pre každé slovo rovnaká. Na nasledujúcom obrázku Obrázok 4.2 je zobrazená ukážka reprezentácie vybraných slov. Rovnako ako pri word2vec, tak aj v tomto prípade môžeme pozorovať príklad vzdialenosí vektorov slov *man – woman* a *queen – king*.



Obrázok 4.2 - Ukážka vektorovej reprezentácie slov - GloVe

4.3 Generovanie textovej sekvencie

Z historického hľadiska môžeme o generovaní nadpisov rozmýšľať ako o viac tradičnej metóde a to sumarizácii dokumentov, nakoľko generovanie nadpisov môžeme chápať ako veľmi krátku sumarizáciu.

Z oblasti generovania nadpisov, resp. súvislých textov popisujúcich dokument bolo preskúmaných viacerých riešení. Práca [11] popisuje navrhnutú neurónovú sieť, pozostávajúcu s kódovača (vstupný dokument na vektory) a dekódovača (generovanie nadpisu na základe vektorov), pomocou ktorej je v obmedzenom rozsahu možné generovať nadpisy textových dokumentov v Anglickom a Čínskom jazyku. Pre optimalizáciu výsledkov bola navrhnutá stratégia trénovalia tzv. minimálneho rizika. Priamo optimalizovala parametre modelu na úrovni viet vzhľadom na výhodnocovacie metriky.

Niektoří generujú popisky ku predmetom na obrázku pomocou umelých neurónových sietí [1]. V tomto prípade bola aplikovaná CNS pre mapovanie regiónov jednotlivých objektov na obrázkoch s obojsmernou RNS pre generovanie popisov. Rekurentné neurónové siete sa osvedčili byť vhodným kandidátom pre spracovanie

prirodzeného jazyka. Pre generovanie textových popisov, resp. viet sú využívané taktiež LSTM rekurentné NS.

V posledných rokoch sme mohli zaznamenať narastajúci počet tzv. *sequence-to-sequence* (seq2seq) neurónových modelov [21]. Tieto modely sa zväčša učia distribuované reprezentácie vstupnej sekvencie (viď. Reprezentácia slov) a podľa toho generujú výstupnú sekvenciu. Výhoda modelov spočíva v tom, že tieto modely sa učia sémantické mapovanie priamo podľa dvojíc sekvencií dokument-nadpis bez ručne vytvorených funkcií. Neurónové modely ukázali veľkú silu pre summarizáciu textu a generovanie nadpisov nakoľko môžu flexibilne modelovať sémantiku dokumentu z interných slovných sekvencií v dokumente. Mnohé externé informácie o dokumente však môžu zohrávať taktiež dôležitú úlohu pri summarizácii textu. Napríklad dokumenty sú zvyčajne zoskupené do rôznych tém (kategórií) a v rámci určitej témy môžeme v dokumentoch nájsť isté spoločné summarizačné vzory. Napríklad dokumenty o ekonomike zvyknú obsahovať ceny alebo rôzne štatistické údaje a dokumenty o sociálnych udalostiach alebo nehodách obsahujú čas, polohu a miesto udalostí. [22].

4.4 Sumarizácia textu

Automatická summarizácia textu má dva prístupy: extraktívny a súhrnný. Metóda extrakcia je metóda, ktorá vyberá dôležité slová alebo dôležité vety z textov podľa ich ohodnotenia. Extraktívny prístup má však aj určité obmedzenia. Tým prvým je, že vygenerovaný abstrakt obsahuje pôvodné vety a vety nie sú skrátené aj v prípade, že obsahujú redundantné informácie. Druhé obmedzenie je že, ak extraktívny prístup vyberá slová z textov, je ľažké skonštruovať prirodzenú vetu z extrahovaných slov. Na druhej strane, súhrnný prístup je metóda, ktorá generuje abstrakty z originálnych textov, ako je ľudská činnosť. Prístup môže vytvoriť vysokokvalitný abstrakt, ale vyžaduje viacero techník spracovania prirodzeného jazyka. [16]

Ako prvé je aplikovaná metóda TF-IDF, ktorou sa určí dôležitosť slova. Na následné ohodnocovanie dôležitosti viet môžu vplývať faktory, ktoré sú zachytené aj

v [16], kde sú vety, ktoré sú napríklad na začiatku odstavcov ohodnotené väčšími váhami. Záleží teda aj na pozícii viet v samotnom texte.

Niekteré výskumy, ako aj napríklad [2], využívajú tradičné metódy akými sú napríklad metóda TF-IDF, známa VSM, LSA či LDA avšak s rozšírením neurónových sietí stúpa vyžitie práve týchto modelov. Aplikácia NS pri summarizácii textových dokumentov je inšpirovaná strojovým prekladom prirodzeného jazyka .

4.5 Kategorizácia dokumentov

Presné zhlukovanie si vyžaduje presnú definíciu blízkosti medzi dvojicou objektov, a to buď v podobe párovanej podobnosti alebo vzdialenosťi. Boli navrhnuté široko používané merania vzdialenosťi ako napríklad kosínusová podobnosť a Jaccardov korelačný koeficient. Merania ako Euklidová vzdialenosť a relatívna entropia boli použité pri zhlukovaní pri výpočte vzdialostných dvojíc. [23]

Iní klasifikujú dokumenty do spoločných kategórií a označujú tak dokumenty spoločnými označeniami [8]. Cieľom práce bola klasifikácia viacerými označeniami (z angl. „multi-label classification“) s využitím dvoch modelov NS. Prvý model bola štandardná dopredná NS (viacvrstvový perceptrón) a druhý bola CNS.

4.6 Metriky vyhodnocovania

Pre rozličné prístupy sú v oblasti kategorizácie textových dokumentov a generovania nadpisov využívané viaceré metriky v závislosti od konkrétnych riešení.

4.6.1 Kvalita textu

Pri hodnotení vygenerovaného textu nemôžeme vždy automaticky hodnotiť každý atribút. Pri summarizácii textu boli využívané aj metriky tzv. kvalitatívneho hodnotenia textu z lingvistického hľadiska, ktoré boli vyhodnocované subjektívne účastníkmi výskumov, a to [20]:

- **gramatická správnosť** – či text neobsahuje nie textové prvky (rôzne chybné znaky) alebo slová s chybami
- **redundancia** – text by nemal obsahovať opakujúce sa informácie
- **zrozumiteľnosť** – podstatné mená a zámená by mali zreteľne vysvetľovať podstatu

4.6.2 Metriky využívané v existujúcich modeloch

Metrika alebo algoritmus BLEU vyhodnocuje kvalitu textu, ktorý bol strojovo preložený z jedného prirodzeného jazyka do druhého. Táto metrika je podľa [1] vhodná pre vyhodnocovanie kvality strojovo vygenerovaného textu. Pri [11] bola zvolená vyhodnocovacia metrika pre vygenerované titulky novín tzv. ROUGE metrika, ktorá pozostáva zo súboru viacero metrík a využíva sa pri vyhodnocovaní automatizovaných summarizácií textov a strojových prekladov NLP. Podobne ako BLEU porovnáva preklady voči referenčným vetám. Okrem týchto metrík sa taktiež osvedčili metriky ako METEOR alebo F-Meranie.

BLEU

Princíp tejto metriky je, že čím je preklad vygenerovaný strojom bližšie k ľudskému prekladu, tým je daný strojový preklad lepší. Bleu skóre je v zásade rozdiel medzi ľudským a strojovým prekladom textu. Problémom tejto metriky je, že penalizácia za malú zmenu textu je veľmi malá avšak aj takáto malá zmena môže výrazne zmeniť zmysel celého prekladu. Pomocou tejto metriky teda nedokážeme zachytiť vyhodnotiť preklady, ktoré súce neobsahujú rovnaké slová ale majú rovnaký význam. Skóre sa počíta ako pomer počtu slov z textu m , ktorý označujeme ako kandidát (strojovo vygenerovaný text), ktorý sa nachádza v referenčnom texte k celkovému počtu slov w v kandidátovi. Platí teda vzťah:

$$P = \frac{m}{w_t} \quad (4.5)$$

ROUGE

Táto metrika sa ako aj predchádzajúca sa využíva pri evaluácii automatických sumarizácií a strojových prekladov. V rámci tejto metriky je možné vyhodnotovať 5 pod-metrik: Rouge-N, 1, 2, L, W, S, SU. Rouge-N je ku príkladu porovnanie prekrývania n-gramov medzi strojovo vygenerovaným prekladom a ľudským prekladom. V prípade že ide o Rouge-1 alebo Rouge-2, pôjde o 1-gram alebo bigram.

4.7 Využívané modely

Rôzne výskumné práce ukázali [16][14][11][1], že v prípade ak sa jedná o generovanie logických pomenovaní textových dokumentov je možné použiť rozličné modely a techniky. Nie vždy sú používané len NS, najmä pri summarizácii textu.

Pri rozoznávaní vzorov sú aplikované architektúry neurónových sietí na rozpoznávanie objektov. Tieto neurónové siete môžu extrahovať sľubné charakteristické črty z neoznačených obrázkov, ktorých je obrovský počet. Aplikácia týchto neurónových sieti bola použitá pri tzv. *SuperVision*. Ide o konvolučnú neurónovú sieť, ktorá dokáže klasifikovať obrázkové dátá na základe ich komponentov.

4.7.1 Výsledky využívaných modelov

Model predstavený v [11] s použitím MLE dosahoval porovnatelné výsledky s existujúcimi systémami pre generovanie nadpisov. Systémy využívajúce kombinácie CNS v spojení s Elmanovou RNS dosahovali o niečo lepšie výsledky ako CNS v spojení LSTM. V porovnaní s inými architektúrami a optimalizáciou parametrov pomocou MLE dosahovala architektúra s optimalizáciou MRT viditeľne lepšie výsledky na súbore dát Gigaword.

V [1] vyhodnocovali správnosť určenia regiónu na obrázku pre jednotlivé predmety ale aj vygenerované popisy k týmto objektom. V prípade, že používali model, ktorý sa zameriaval viac na obrázok ako taký a nie na konkrétné regióny boli BLEU výsledky nižšie. Vygenerované popisky, resp. vety boli taktiež dlhšie a teda negatívne to vplývalo na metriku BLEU. S porovnaním s dvoma modelmi využívajúcich LSTM siete boli výsledky lepšie avšak v porovnávaných prácach využívali menej výkonné funkcie knižnice AlexNet.

V porovnaní so seq2seq a CopyNet dosiahol model v [22], kde bola použitá latentná Dirichletova alokácia (LDA) o 4% lepšie výsledky pri vyhodnocovaní metrikou ROUGE-F. Tréning RNS si však vyžadoval až 500 tisíc iterácií pre každú tému dokumentov. Témy boli preto rozdelené na základných 5 skupín

V dnešnej dobe je v prípade neurónových sietí využívaný model seq2seq ako architektúra aj pre generovanie názvov článkov. Jedno z najvýznamnejších aplikácií tohto modelu môžeme vidieť pri projekte *textsum* od spoločnosti Google. Počas tohto projektu pracovali s databázou Gigaword. Model dokáže vygenerovať nadpisy článkov na základe krátkeho (120 slov) abstraktu.

4.8 Zhrnutie analýzy

Nami preskúmanými dostupnými zdrojmi sme dospeli k záveru, že doposiaľ žiadna práca neriešila náš konkrétny typ problému, a teda generovanie nadpisov kategórií pre zhluky textových dokumentov. Existuje viacero prístupov a prác, ktoré riešia podobné problémy. Ide najmä o práce, ktoré sa venujú generovaniu sekvencií, resp. textových spojení určujúcich názvy dokumentov. Každá práca využíva rozličné metódy z oblasti strojového učenia.

Generovanie nadpisov kategórií textových dokumentov by sme mohli teda rozdeliť na dve na sebe navzájom závislé časti. Prvá časť tohto problému je samotné zhlukovanie dokumentov a druhá je generovanie resp. krátká summarizácia týchto dokumentov.

V prípade zhlukovania boli preskúmané viaceré štandardné metódy, ktoré poskytujú možnosti ako zoskupiť dokumenty do zhlukov (kategórií).

Rôzne práce riešia problémy so summarizáciou a generovaním nadpisov textových dokumentov. Pri riešeniach sú využívané viaceré modely aj z oblasti hĺbkového učenia. NS ktorých opis je uvedený v časti Umelé neurónové siete, poskytujú riešenia pre generovanie a spracovanie prirodzeného jazyka.

Správnosť riešení v uvedených prácach je overovaná viacerými metrikami (vid. Metriky vyhodnocovania). Vždy sú uvádzané viaceré, najmä kvôli ich porovnaniu aj s inými prácami.

5 Návrh riešenia

V tejto kapitole sa postupne zameriame na predspracovanie dátového súboru, zjednotenie dokumentov a vytvorenie nového dokumentu a nakoniec na reprezentovanie dokumentu a opis základnej architektúry riešenia.

5.1 Celkový pohľad na model

Nami navrhnuté riešenie rieši generovanie textovej sekvencie na základe vstupného dokumentu. Model nerieši problém zhľukovanie dokumentov. Predpoklad pre metódu je, že dokumenty sú už v samostatných kategóriách. Generovanie sekvencie je na úrovni slov. Vstupné dokumenty zjednocujeme na základe spoločných črt do jedného spoločného dokumentu, ktorý je vstupom pre model NS. Výstupom modelu je reťazec slov opisujúci zhľuk dokumentov. Samotné riešenie sa skladá z troch základných častí:

❖ Predspracovanie

- Ako prvé je nutné dokumenty očistiť od nadbytočných elementov a pretransformovať tak dokumenty len na čistý text.
- V rámci dátového súboru Wikipédie je nutné odstrániť značkovacie elementy (wiki-markup) a jednotlivé sekcie článkov (napr. odkazy na iné dokumenty, obrázky, a pod.).
- Aplikovanie regulárnych výrazov pre finálne očistenie dokumentu.

❖ Tvorba nového dokumentu

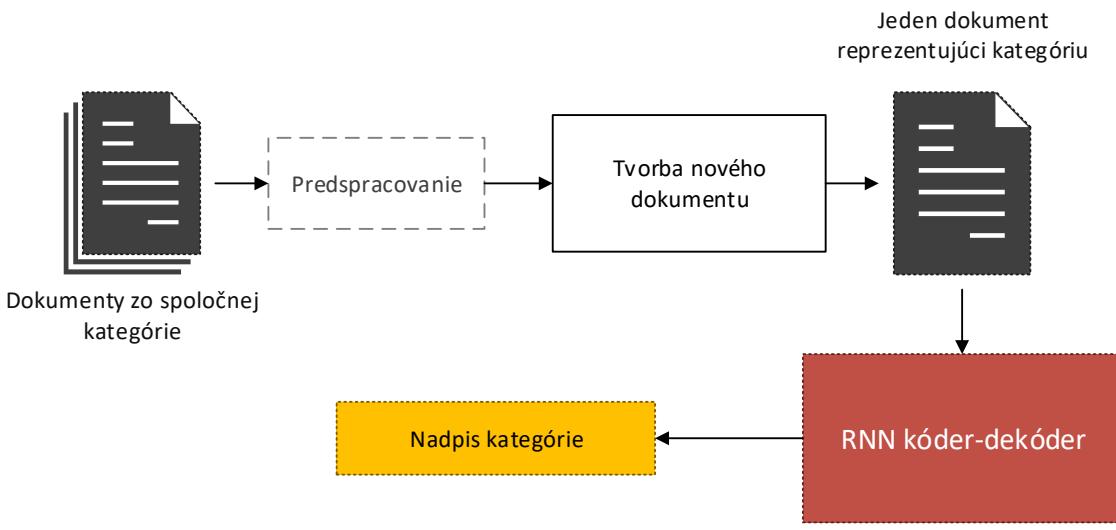
- Po úvodnom predspracovaní je pre každý dokument vytvorená matica najfrekventovanejších výrazov (bez stop slov). Predpokladáme, že najfrekventovanejšie slová v dokumentoch budú určovať ich podobnosť.

Z viet, v ktorých sa nachádzajú tieto slová vytvoríme nový dokument, ktorý bude teda obsahovať vety zo všetkých dokumentov.

❖ Generovanie nadpisu kategórie

- Samotné generovanie nadpisu sa skladá z dvoch častí:
 - V prvej časti vytvoríme vektorovú reprezentáciu dokumentu
 - V druhej časti vygenerujeme textovú sekvenciu na základe vstupného vektora

Na nasledujúcim obrázku Obrázok 5.1 je abstraktne znázornená základná architektúra navrhovaného riešenia.



Obrázok 5.1 - Model navrhovaného riešenia

5.2 Predspracovanie dát

V prípade predspracovania, ide o podstatnú časť riešenia. Ak by sme napríklad vynechali, resp. odstránili niektoré z kľúčových slov, novovytvorený dokument by nemusel dostatočne zahŕňať spoločné črty dokumentov.

Na druhej strane, ak by sme v dokumentoch nechali niektoré značkovacie elementy, malo by to za následok skreslenie frekvencie výskytu slov v dokumente a predpokladali by sme, že slová s nízkou výpovednou hodnotou (napr. id, odkazy,

elementy štýlu textu, ...) sú spoločnými črtami dokumentov. Takéto uvažovanie by však bolo chybné, nakoľko tieto elementy sú istým typom stop slov a môžu sa nachádzať v každom dokumente v závislosti na zdroji.

5.2.1 Zvolený dátový súbor

Po analýze zdrojov Reuter, 20 newsgroup, Gigaword Fifth Edition a Wikipédia sme dospeli k záveru, že pre generovanie nadpisov kategórií, bude najvhodnejší zdroj Wikipédia. Pre tvorbu a testovanie nášho riešenia je totiž dôležité už mať rozdelené dokumenty do kategórií. Názvy týchto kategórií by nemali byť len generické a veľmi všeobecné (napr. šport alebo kultúra), ale mali by byť konkrétnejšie ako napríklad *Cicavce z Afriky*. Iné zdroje nám neposkytujú takúto možnosť a teda nemajú takto štruktúrované dátá. Wikipédia je preto najvhodnejšia.

Pre naše účely potrebujeme z databázy časti (tabuľky) - *categorylinks* a *pages*. V týchto dátach sa nachádzajú konkrétnie články a k nim korešpondujúce kategorie. Každý článok však spadá pod viacero kategórií. V súčasnom riešení nefiltrujeme kategorie, ktoré majú spoločné články. Môže sa teda stať, že jeden článok prejde NS viackrát avšak s inou prislúchajúcou kategóriou. Tento problém je možné však vyriešiť buď odstránením článku z trénovanej množiny alebo ho ponecháme v množine a budeme predpokladať, že pri tvorbe nového dokumentu bude pre NS predstavovať nový text.

5.2.2 Všeobecné predspracovanie

Naše riešenie je možné aplikovať v zásade na akékoľvek dokumenty, resp. na dokumenty z akéhokoľvek zdroja. Tu je však dôležité aby výsledný text bol očistený od niektorých elementov.

V princípe je nutné vykonať na dokumentoch nasledujúce úpravy:

- Odstránenie značkovacích elementov. Napr. v prípade html dokumenty musíme odstrániť xml značky, napríklad z html dokumentu

```
<div class="container">  
    <h2>Náš nový text</h2>  
</div>
```

po spracovaní ostane len text: *Náš nový text*

- Odstrániť nadbytočné údaje o dokumente, ktoré by mohli skresľovať tvorbu matice TF – čísla strán, odkazy, obrázky, ...
- Vybrať rôznorodé nadpisy kategórií. Zamedziť tomu aby bolo v názvoch často len jedno slovo, prípade rovnaké časti názvu viackrát.

5.2.3 Predspracovanie dát z Wikipédie

Ako sme už načrtli v časti 5.2.1, predspracovanie dát z Wikipédie si vyžaduje vykonať viacero úprav.

Vybrať rôznorodé nadpisy kategórií

Ešte pred tým ako budeme môcť predspracovať samotné texty, musíme odfiltrovať tie články, ktoré majú špecifické názvy kategórií alebo obsahujú v text určité slová. Tieto články zväčša patria medzi technické kategórie alebo sú príliš všeobecného charakteru.

Odstránenie značkovacích elementov

Okrem samotného odfiltrovania niektorých článkov z databázy, musíme taktiež prekonvertovať značkovací jazyk Wikipédie na čistý text, v ktorom sa tieto značky nebudú nachádzať, resp. budú len veľmi zriedka. V rámci tejto transformácie sme

použili voľne dostupnú knižnicu² *txtwiki.js*, ktorá poskytuje pomerne jednoduchým spôsobom možnosť transformácie značkovacieho jazyka na čistý text.

Pre tento účel sme vytvorili jednoduchú webovú stránku, kde po nahraní súboru vo formáte json je možné pretransformovať dátu na čistý text. V nasledujúcej tabuľke je ukážka výsledku transformácie textu.

Tabuľka 5.1 - Ukážka transformácie textu

Text pred transformáciou
<pre> {{Unreferenced date=July 2009}}\n{{Decadebox BC 101}}\n==Events and trends==\n\n* [[1012 BC]]—Acastus, [[Archons of Athens Archon of Athens]], dies after a reign of 36 years and is succeeded by his son Archippus. [[Solar Eclipse]] seen in [[Ugarit]] from 6:09 PM to 6:39 PM, May 9.\n* [[1010 BC]]—[[Uzzah]], King of [[Judah]] dies, believed to have been smitten by God for violating divine law by touching the [[Ark of the Covenant]].</pre>
Text po transformácii
<pre> {{unsourced date=February 2009}}Shalmaneser II was King of Assyria from 1031 BC to 1019 BC. He succeeded his father, Ashurnasirpal I and was succeeded by his son, Ashur-nirari IV, but beyond this little is known of his reign.</pre>

Môžeme si všimnúť, že po transformácii zmizla väčšina elementov avšak nie úplne všetky. Tento problém ďalej riešime v nasledujúcom kroku úprav.

Odstrániť nadbytočné údaje o dokumente

Po úvodnej transformácii je nutné na text aplikovať rôzne regulárne výrazy aby sme čo najviac vyčistili text od nadbytočných a zavádzajúcich informácií. Nakoľko sme transformáciou nedocielili úplne odstránenie značkovacích elementov, musíme tak urobiť teraz. V Tabuľka 5.2 sa nachádza prehľad aplikovaných regulárnych výrazov v poradí v akom boli aplikované na text.

²Dostupné na: <https://github.com/joaomsa/txtwiki.js>

Tabuľka 5.2 - Zoznam použitých regulárnych výrazov

Regulárny výraz	Popis
{(. *?)}	Všetky reťazce medzi znakmi {}
/(.*?)=(.*?)	Vzory kľúč=hodnota oddelené zvislou čiarou
}*	Zvyšné prázdne značky
{*	
/ n +	Po sebe nasledujúce riadky
== <i>External links</i>	Všetky reťazce nachádzajúce sa po časti <i>External links</i>
== s*(/^ n r)*)	

Okrem týchto úprav je dôležité aby sme odstránili z článkov zoznamy kategórií do ktorých spadajú. Každý článok z Wikipédie má na konci textu aj zoznam kategórií.

5.3 Zjednotenie dokumentov

Existujúce riešenia poskytujú viacero spôsobov ako spracovať viacero dokumentov a následne vytvárať pre nich sumarizáciu. V našom riešení sme sa rozhodli reprezentovať dokumenty v kategórii jediným novo vytvoreným článkom. Je to najmä z dôvodu, že v prípade rozsiahlych dokumentov je spracovanie a trénovanie modelu veľmi náročné a taktiež chceme priniesť nový pohľad, resp. preskúmať nové riešenia tejto problematiky. Vytvorením novej reprezentácie kategórie z najfrekventovanejších výrazov sa je možné zamerať len na najdôležitejšie vety z dokumentov a spojiť tak spoločné črty dokumentov.

5.3.1 Najfrekventovanejšie výrazy

Ako prvé musíme určitým spôsobom nájsť spoločné črty dokumentov a taktiež nájsť kľúčové slová, ktoré sú špecifické pre dané zhľuky. Kľúčové slová je možné vyextrahovať použitím metódy TF-IDF. V našom prípade však túto metódu nemôžeme aplikovať. Dôvodom je fakt, že dokumenty, ktoré sú zo spoločnej kategórie

obsahujú taktiež veľa spoločných slov. Metódou TF-IDF by sme teda predpokladali, že tieto slová nie sú pre články kľúčové keďže sa nachádzajú častokrát vo všetkých článkoch.

Rozhodli sme sa z metódy využiť len frekventovanosť výrazov. Tu však musíme najskôr eliminovať stop slová. Spoločné slová medzi dokumentami sice nemajú taký výskyt ako slová napríklad *alebo*, *a*, *možno*, *to*, avšak môžu byť taktiež označené ako stop slová. To je pre náš model kontra indikatívne. Frekventované výrazy budú ďalej slúžiť ako podklad pre extrahovanie konkrétnych viet z dokumentov.

5.3.2 Tvorba reprezentácie kategórie

Sekvenciu po sebe idúcich slov môžeme nazývať taktiež n-gramy. V dokument musíme ako prvé nájsť najčastejšie sa vyskytujúce n-gramy, teda sekvencie, ktoré sú pre dokumenty spoločné. Predpokladáme, že tieto n-gramy by mali taktiež obsahovať najfrekventovanejšie slová v rámci kategórie. Povedzme, že existujú dokumenty:

Dokument A – rýchla hnedá líška

Dokument B – rýchla hnedá líška a rýchla hnedá mačka a rýchly hnedý pes.

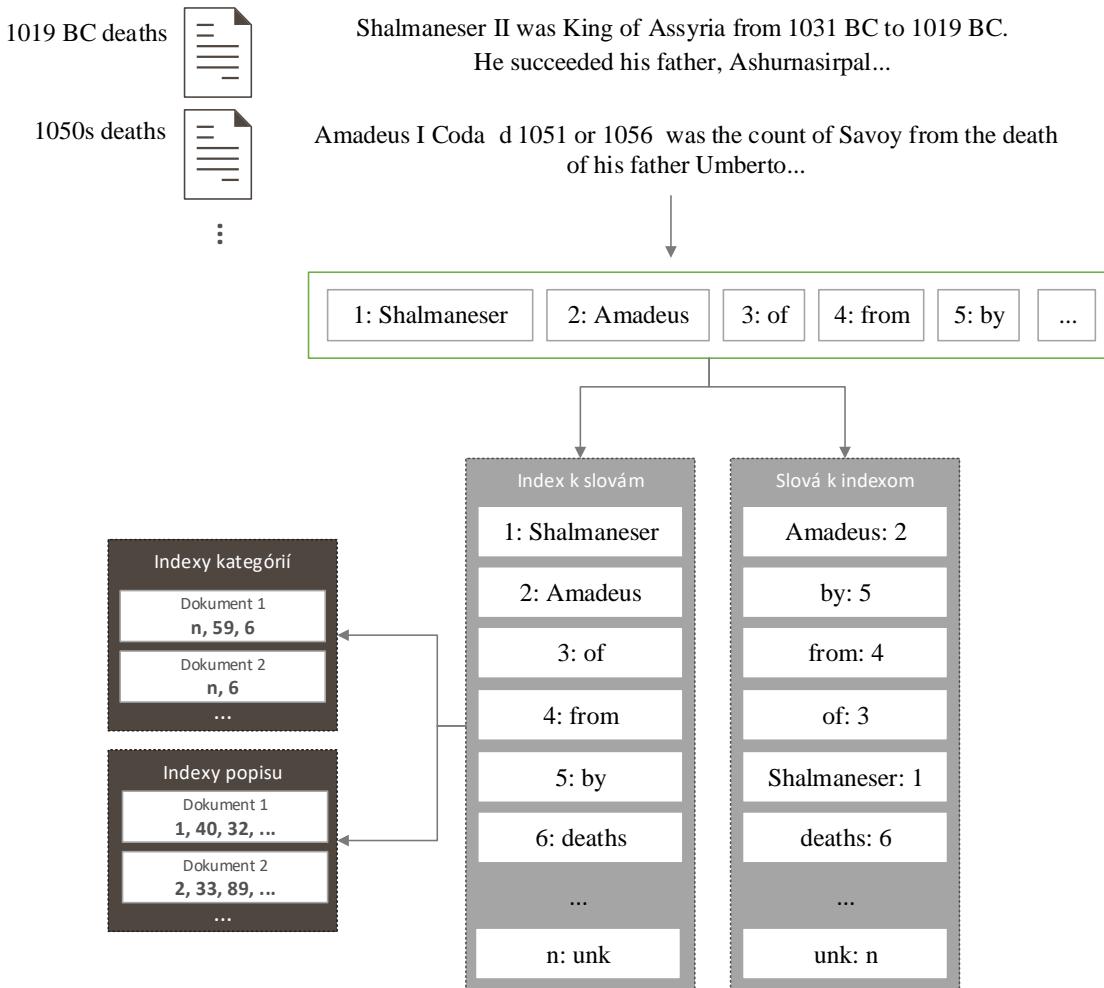
V prípade tri-gramov ($n=3$) by boli podobné [„rýchla“, „hnedá“, „líška“]. Pre bi-gramy ($n=2$) budú spoločné n-gramy [„rýchla“, „hnedá“]. Po nájdení spoločných n-gramov, môžeme vyextrahovať z dokumentu buď celé vety, v ktorých sa tieto frázy nachádzajú alebo ponechať n-gramy aby samotné reprezentovali spoločné črty dokumentov. V takomto prípade by sme nový dokument vytvorili na základe najfrekventovanejších n-gramov v dokumentoch. Takáto reprezentácia dokumentu môže byť však nedostatočná nakoľko niektoré dokumenty nemusia mať veľa spoločných n-gramov no pritom môžu patriť do rovnakej kategórie. Po nájdení najfrekventovanejších n-gramov vytvoríme novú reprezentáciu kategórie, zlúčením najčastejšie sa vyskytujúcich n-gramov.

5.4 Reprezentácia dokumentu

Vstupom pre NS nemôže byť slovo resp. sekvencia ako taká. Dokument musíme reprezentovať v určitej forme. Pre náš model je takáto reprezentácia vektorová. Každý dokument musíme najskôr rozdeliť na slová, ktoré budeme reprezentovať vektormi, aby sme mohli vytvoriť výslednú maticu reprezentujúcu celkový dokument.

5.4.1 Tokenizácia a tvorba indexov

Prvou úlohou je vytvorenie tzv. tokenov. V našom prípade sú tokeny konkrétnie slová z textu. Výstupom tohto kroku je slovník na základe vstupného textu. Do slovníka pridávame okrem existujúcich slov aj špecifické slovo *unk*. Týmto slovom totiž nahradzujeme neznáme slová, ktoré sa nachádzajú mimo slovníka. Po vytvorení slovníka vytvárame taktiež matice – slová k indexom a opačne. Vstupom tejto metódy sú všetky dokumenty, na ktorých ideme trénovať NS. Obrázok 5.2 znázorňuje prevod popisov a k nim prislúchajúcim kategóriám na matice indexov.



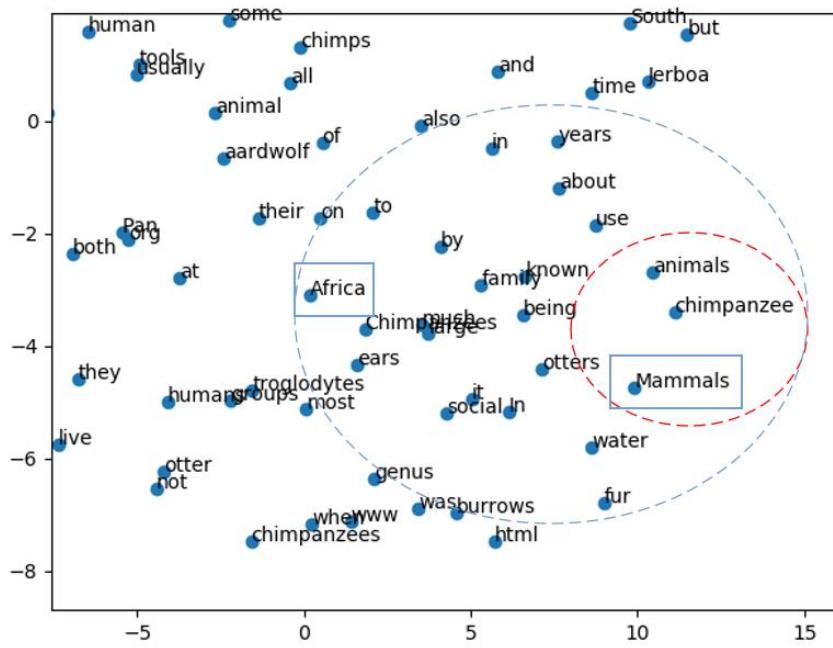
Obrázok 5.2 - Prevod textu na indexy

5.4.2 Matica vektorov

Pre vytvorenie vektorovej reprezentácie sme sa rozhodli využiť korpus ~ 400 tisíc slov z knižnice GloVe. Okrem samotného korpusu slov, je v tejto knižnici vypočítaná vektorová reprezentácia každého slova z korpusu. Knižnica GloVe, je ako už bolo spomenuté v časti Reprezentácia slov, poskytuje o niečo efektívnejšiu prácu s vektormi v porovnaní s napríklad word2vec. Pri GloVe, môžu byť slová reprezentované vektormi o rozmeroch dimenzie (50, 100, 200, 300)

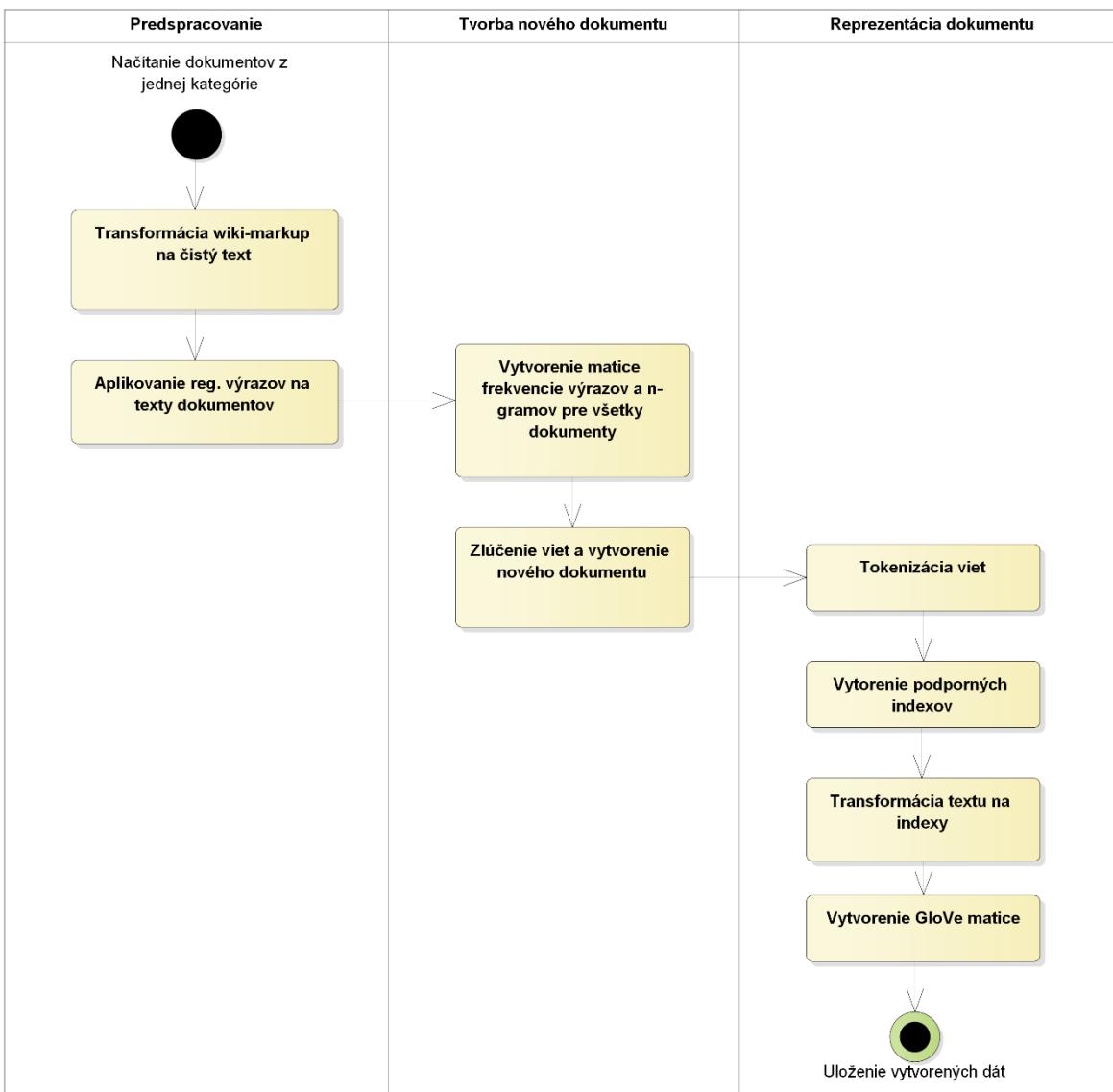
Použitím pretrénovaných vektorov GloVe, môžeme reprezentovať slová z článkov podobne ako to je na obrázku Obrázok 5.3. Na obrázku je znázornená reprezentácia najčastejšie sa vyskytujúcich slov z článkov kategórie *Cicavce z Afriky*

(z angl. „Mammals of Africa“). Z grafu môžeme zistíť, že slová ako *animals*, *chimpanzee* alebo *Mammals* majú medzi sebou krátke vzdialenosť. Predpokladáme, že ak sa v dokumentoch vyskytujú často frekventované slová, ktoré taktiež opisujú spoločné témy, budú vzdialenosť medzi týmito vektormi podobne malé. Graf zobrazuje priblížené znázornenie reprezentácie slov.



Obrázok 5.3 - Vektorová reprezentácia najfrekventovanejších slov

Celkový pohľad na úvodné spracovanie a vytvorenie vektorovej reprezentácie novovytvoreného dokumentu sa nachádza na diagrame aktivít na obrázku Obrázok 5.4. Výstupom tejto časti navrhnutej metódy je reprezentácia názovov kategórií a k nim prislúchajúcim dokumentom, matica GloVe a indexovaný slovník. Tieto dátia slúžia následne ako podklad pre NS.



Obrázok 5.4 - Diagram aktivít pre spracovanie dokumentov

5.5 Architektúra kóder-dekóder

Po preskúmaní existujúcich riešení sme zistili, že najlepší prístup bude zvoliť architektúru kóder-dekóder. Ako bolo spomenuté, táto architektúra vykazuje pozoruhodné výsledky, a je už použitá v riešení pri summarizáciách textu a teda je možné porovnávanie nášho modelu s inými príbuznými modelmi. Tento model NS bol zo začiatku využívaný pri prekladoch prirodzeného jazyka, kde vstupom bola

v jednom jazyku a výstupom bola rovnaká veta avšak v inom jazyku. Tento princíp sme aplikovali taktiež v modeli postupnosti (sekvencia-na-sekvenciu).

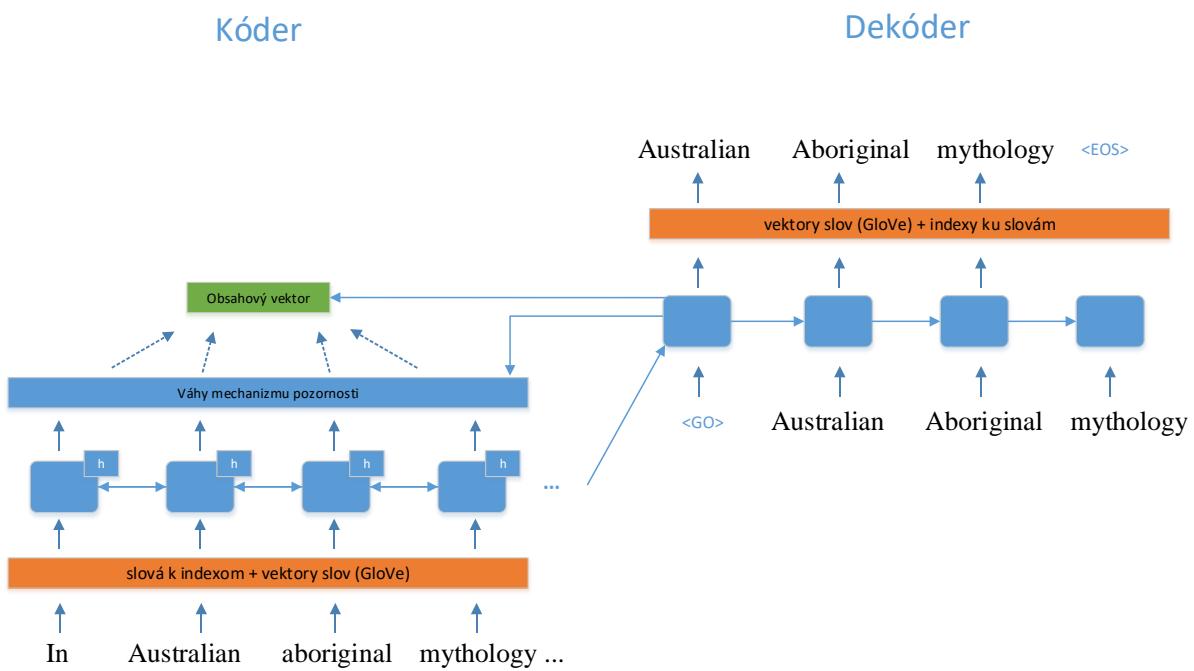
Architektúra kóder-dekóder sa skladá z dvoch rekurentných NS, kde tou prvou je kóder a druhou je dekóder. Kóder ma za úlohu pre vstupnú sekvenciu vytvoriť vektor, ktorý bude istým spôsobom reprezentovať túto sekvenciu. Druhá NS prijíma na vstupe vektor vytvorený kóderom a „snaží sa ho dekódovať“ do výstupnej sekvencie na základe pravdepodobnosti výskytu nasledujúceho prvku v sekvencii. Táto architektúra je primárne navrhnutá pre tzv. problémy sekvencia-na-sekvenciu (z angl. „sequence-to-sequence“), kde pre vstupnú sekvenciu (znaky, čísla, slová, ...) je generovaná výstupná sekvencia.

5.5.1 Model postupnosti (seq2seq)

V našom riešení sme použili architektúru kóder-dekóder, pričom sme aplikovali princíp prekladu prirodzeného jazyka. V článkoch sa stretнемe s faktom, že samotný článok je oveľa dlhší ako názov kategórie. Teda vstupná sekvencia je omnoho dlhšia ako výstupná sekvencia. Tento problém vieme vyriešiť architektúrou kóder-dekóder tak, že si stanovíme fixnú veľkosť vstupného vektora, a v prípade, že je výstup kratší tak použijeme tzv. odsadenie (z angl. „padding“) a teda vo vektore, ktorý reprezentuje dané slovo, doplníme na zvyšné miesta nulu. To môžeme urobiť až priamo v spomínanom vektor alebo vytvoríme nový token podobne ako *unk*, ktorý bude reprezentovať odsadenie (napr. <PAD>).

Vstupom do kódera je matica slov k indexom, resp. vektory slov GloVe. Výstupom kódera je v našom prípade len vektorová reprezentácia vstupnej sekvencie. Kóder obsahuje n neurónov LSTM. Tento typ neurónu nie je jediný, ktorý je možné využiť. Taktiež je možné použiť obmenu GRU neurónu. Počas výpočtu skrytého stavu sa taktiež vypočítava skóre pre mechanizmus pozornosti. V princípe ide o hodnoty, kde čím vyššie je skóre tým viac pozornosti sa kladie na dané slovo v dekóderi.

Dekóder, generuje výstup slovo po slove tým, že počíta pravdepodobnosť nasledujúceho slova v sekvencii. Vstupom do dekódera je skrytý stav teda vektor reprezentujúci sekvenciu z kódera. Rovnako ako pri predošej RNS, aj v tejto sieti sú použité LSTM neuróny. Dekóder začiatocným špeciálnym tokenom $<GO>$ vypočíta pravdepodobnosť nasledujúceho, resp. prvého slova v sekvencii. Tu však do modelu vstupujú taktiež váhy mechanizmu pozornosti. K týmto váham model pristupuje pri každom generovaní nasledujúceho slova. Na základe tohto mechanizmu nestrácame pri dlhých textoch obsah celej vstupnej sekvencie, keďže ako vstup do dekódera je len výstup (skrytý stav) z posledného neurónu kódera. Na obrázku Obrázok 5.5 je vizualizovaná architektúra modelu seq2seq.



Obrázok 5.5 - Architektúra kódér-dekóder

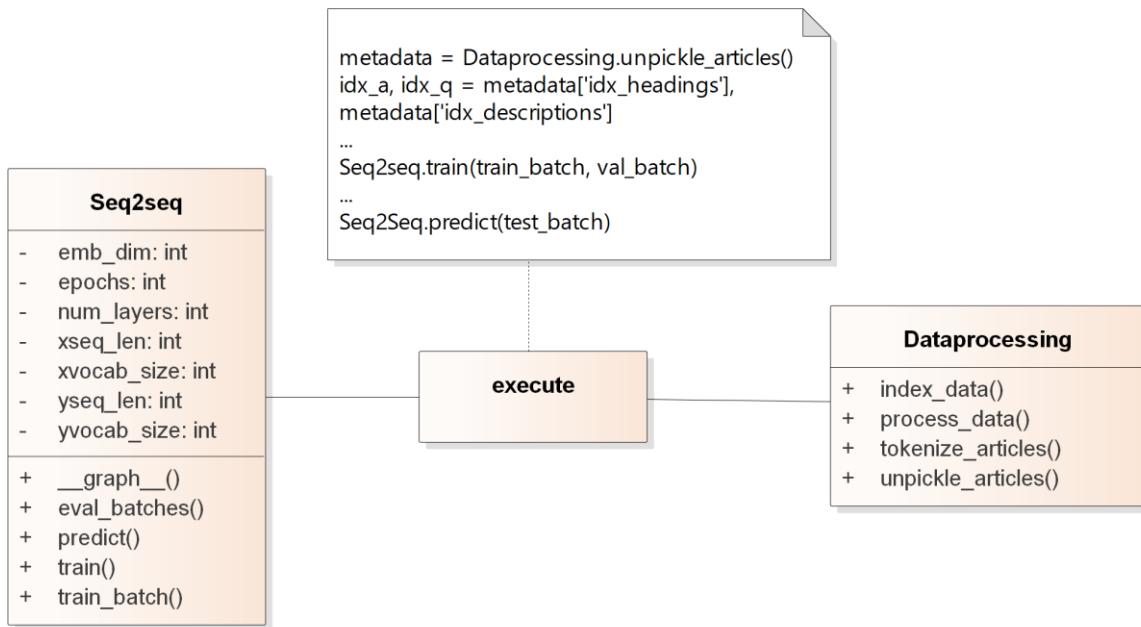
6 Overenie riešenia a experimenty

V nasledujúcich častiach sa zameriame na konkrétny výsledky a úvodné pozorovania navrhovaného modelu pre generovanie nadpisov kategórií.

6.1 Implementácia modelu

Model je implementovaný v jazyku python. Navrhnuté riešenie sme sa rozhodli implementovať s použitím existujúcej knižnice pre podporu práce s neurónovými sieťami. Konkrétnie ide o knižnicu Tensorflow. Pre spracovanie prirodzeného jazyka sme použili knižnicu NLTK (Natural language tool-kit). Knižnica poskytuje metódy pre tokenizáciu a indexovanie dokumentov a pod. Riešenie je rozdelené do troch častí. Celý proces začína pri spracovaní dát v *execute.py*. Inicializáciou triedy Dataprocessing sa volá metóda *proces_data*, pomocou ktorej sú vytvorené indexy a slovník, ktoré sú následne uložené v tzv. *pickle* súbore. Volaním metódy *unpickle_articles* sa uložené dátá (indexy pre nadpisy a články) znova načítajú.

Nasleduje vytvorenie grafu v rámci triedy Seq2seq. V tomto kroku je graf vyskladaný na základe vstupných hyper-parametrov (viď. 6.1.2). Vytvorený model NS je možné trénovať na vstupnej množine trénovacích dát (*train_batch*). Po ukončení trénovania je model uložený a následne je možné generovať názvy kategórií pre vstupné testovacie dátá (*test_batch*). Diagram tried implementovaného modelu je zobrazený na obrázku Obrázok 6.1.



Obrázok 6.1 - Diagram tried pre implementovaný model

6.1.1 Spracovanie dát

Prvou časťou vytvoreného modelu je predspracovanie a následná tvorba indexov zo vstupných dát Wikipédie. Celá logika tohto procesu sa nachádza v triede Dataprocessing. Spracovanie dát vychádza z diagramu aktivít na obrázku Obrázok 5.4.

Úvodné spracovanie dát

Ešte pred tým ako môžeme spracovať dátá v samotnom modeli, musíme odstrániť spomínané značkovacie elementy Wikipédie. Pre tento problém sme vytvorili jednoduchú webovú aplikáciu, v ktorej je toto spracovanie možné s pomocou knižnice txtwiki.js. Vstupný súbor vo formáte json musí obsahovať nasledujúce kľúče:

- „txt“ – neupravený text článku z databázy Wikipédie
- „cat“ – prislúchajúcu kategóriu článku

Select local file:

No file chosen

[download](#)

Obrázok 6.2 - Ukážka webovej aplikácie pre spracovanie článkov

Po spracovaní súboru sú dátá uložené taktiež vo formáte json v nasledujúcej podobe:

- „parsed_text“ – text skrátený o väčšinu značkovacích elementov
- „category“ „ – rovnaký ako vstupný text

Tokenizácia vytvorenie indexov

Po úvodnom spracovaní dát, môžeme dátá ďalej predspracovať použitím časti nášho modelu pre spracovanie dát (*Dataprocessing.py*). Ako prvé sú na texty aplikované regulárne výrazy z časti Predspracovanie dát z Wikipédie . Okrem aplikovania regulárnych výrazov sme odstránili z článkov všetky reťazce, ktoré sa nachádzajú za slovom *Category*. Nechceme totiž aby v článkoch boli cieľové názvy kategórií. Všetky články sú orezané na n znakov v závislosti od nastavenia parametra *max_descriptions*.

Nasleduje tokenizácia viet v rámci jednotlivých článkoch a vytvorenie tabuľiek - indexov k slovám a slová k indexom. Tokenizácia prebieha v rámci metódy *tokenize_articles*, ktorej výstupom sú články a názvy kategórií rozdelené po slovách (tokenoch).

Filtrovanie technických článkov

Články sme filtrovali na základe nasledujúcich špecifických slov jednak v kategóriách ale aj priamo v textoch (* označuje akýkoľvek text): *REDIRECT*, *Redirects**, *All_pages**, **#All_articles**, *Disambiguation_pages*, *Articles*lacking**, *Articles*needing*cleanup*, *Articles*from**, *Wiki**, *All*articles**, **stubs*

6.1.2 Trénovanie/testovanie

Samostatne stojaca trieda Seq2seq poskytuje nadstavbu nad existujúcou implementáciou architektúry kóder-enkóder v knižnici Tensorflow. Model je však nutné ako prvé inicializovať, resp. vytvoriť graf s využitím prednastavených hyperparametrov:

- Veľkosť dimenzie
- Max. veľkosť vstupnej sekvencie
- Počet epoch
- Veľkosť slovníka
- Počet RNN vrstiev
- Veľkosť dávky

Po vytvorení modelu, rozdelíme vstupné dátá na tri časti – Testovacia, trénovacia a validačná. Delenie je v pomere 0.8 : 0.1 : 0.1. Model momentálne trénujeme na vybranej množine 1000 článkov zo 100 kategórií. Po natrénovaní modelu môžeme vygenerovať názvy kategórií pomocou metódy *predict*, ktorá vygeneruje predpovede vo formáte indexov k slovám, ktoré môžeme jednoducho preložiť na základe vopred pripravených matíc.

6.2 Úvodné pozorovania

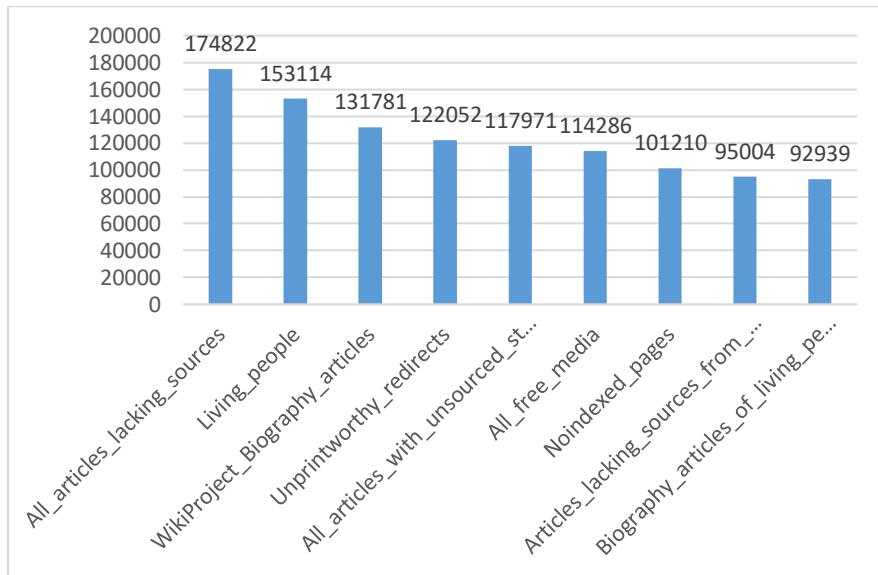
Počas úvodných etáp riešenia projektu sme zistili niekoľko faktov, ktoré negatívne vplývajú na výsledný model. Okrem negatívnych faktorov sme preskúmali aj možnosti aplikovania nových mechanizmov, pomocou ktorých je možné zlepšiť budúce výsledky modelu.

6.2.1 Analýza dát

V našom riešení sme použili súbor voľne dostupnú databázu Wikipédie zo stránky dumps.wikimedia.org. Aktuálna verzia dát (september 2017) má veľkosť ~60 GB vo

formáte zip a po extrahovaní dát je táto veľkosť viac ako 150 GB. Nakoľko je práca s takto mohutnými údajmi náročná, rozhodli sme sa zvoliť menší súbor dát a nepoužiť tak najaktuálnejšie články a taktiež použiť len pre nás najdôležitejšie časti a teda články a k nim prislúchajúce kategórie. Zvolená verzia databázy (marec 2010) má \sim 30 GB a obsahuje 73 000 článkov. Kategórie sú štruktúrované v tabuľke *categorylinks*. V tejto tabuľke je taktiež cudzí klúč na tabuľku *page*, na ktorú vieme pripojiť tabuľku *revision* a následne *text*. Týmto spôsobom vieme zistiť všetky údaje, ktoré budeme potrebovať (text a kategória).

Celkový počet kategórií je 409 063. Z histogramu na obrázku Obrázok 6.3 môžeme vyčítať, že najpočetnejšie kategórie (obsahujúce najviac článkov) v dátach sú technické kategórie Wikipédie. Tieto kategórie obsahujú veľké počty dokumentov v porovnaní z inými.



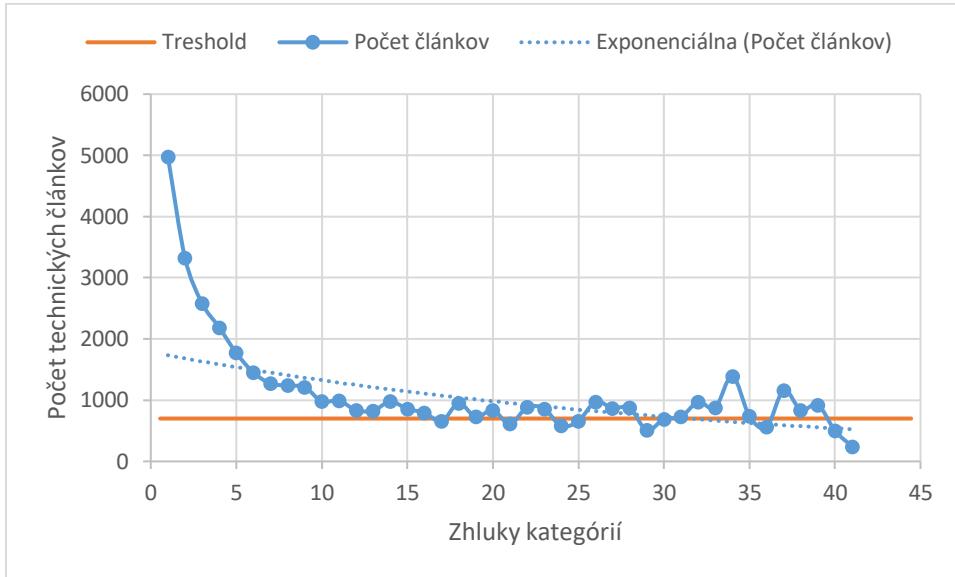
Obrázok 6.3 - Histogram početnosti prvých 10 kategórií

Po postupnej analýze dát sme zistili, že kategórie ktoré majú v priemere viac ako 5 a menej ako 100 dokumentov obsahujú takmer nulový počet technických kategórií. V prípade kategórií kde početnosť je väčšia ako 100 dokumentov, sa v údajoch sice nenachádzajú technické kategórie avšak je v nich častí výskyt názvov obsahujúcich

slovo článok (z angl. „articles“). Ide napríklad o kategórie ako *Psychologické články*, *Wiki články s vysokou prioritou*, *Články zasahujúce do iných článkov* a pod. Problém pri týchto názvoch je že stále opisujú dokumenty na vysokej úrovni abstrakcii alebo sú špecifické pre daný dátový súbor Wikipédie. V prípade kategórií, s nižšou početnosťou (menej ako 5), sa jedná o kategórie zväčša na presmerovanie na iné kategórie. Tieto prípady sa vyskytujú taktiež z dôvodu špecifických wiki údajov. Celkový počet týchto kategórií je 164 020.

6.2.2 Nevyvážený súbor dát

Jedným z prvých problémov, ktorý sme riešili bola konzistentnosť dát, resp. jeho transformácia na predlohu pre náš model. Ako sme spomínali v časti 6.2.1, najpočetnejšie kategórie sú práve tie technické. Okrem technických kategórií obsahuje Wikipédia taktiež veľký počet kategórií veľmi abstraktných alebo veľmi rozsiahlych. Ide o kategórie ako napríklad *Ludia narodený v 20. storočí*, *Úmrtia v 90. rokoch* a pod. Kategórie súce majú výpovednú hodnotu a pomenovávajú určité spoločné charakteristiky dokumentov, avšak obsahujú mnoho článkov a tieto články častokrát opisujú niečo dôležitejšie a primárna kategória týchto článkov je úplne niečo odlišné. Pre ďalšiu prácu s dátami sme sa rozhodli tieto kategórie odfiltrovať. Ako prvé však bolo nutné vytvoriť histogram početnosti jednotlivých kategórií a zistiť tak približné početnosti kategórií, pri ktorých klesá početnosť týchto všeobecných pomenovaní. Pre našu metódu sme museli vytvoriť vlastný súbor s použitím dát Wikipédie, z ktorého sme na základe pozorovaní a analýzy vyselektovali len niektoré kategórie a k nim prisľúchajúce články.



Obrázok 6.4 - Pomer kategórií ku počtu technických článkov

Graf zobrazuje na Y osi počet technických článkov v skupine 10 000 kategórií (os X), ktoré sú zoradené od najpočetnejších po najmenej početné. Vidíme, že počet technických článkov začína výraznejšie klesať pri prvých 10 000 kategóriách. Z týchto dôvodov filtrovame kategórie o maximálny počet článkov z technickej kategórie na základe premennej *threshold*. Vyberáme len tie kategórie kde počet technických článkov je menší ako 700 na 10 000 kategórií.

6.2.3 Korpus anglického jazyka

V rámci vytvárania matice vektorov slov GloVe sme zistili, že v prípade trénovacieho súboru bolo až 46 % označených tokenom určujúcim neznáme slovo – *unk*. Dôvodom je, že vo veľkom počte článkov z Wikipédie sa nachádzajú slová, ktoré nie sú v korpuze slov GloVe. Ide najmä o podstatne mená a špecifické pomenovania v cudzích jazykoch. V prípade tak vysokého počtu neznámych slov je sieť „pretrénovaná“ na tokeny *unk* a výsledkom tak sú nežiadane výstupy. Problém sme momentálne vyriešili odstránením kategórií z trénovacieho súboru dát, ktoré obsahujú aspoň jeden takýto token.

Maticu vektorov GloVe používame s využitím predtrénovaných vektorov. Túto maticu je však možné vytvoriť aj natrénovaním nového modelu na vlastnej množine článkov. Týmto spôsobom môžeme teda minimalizovať počet neznámych slov v článkoch.

6.2.4 Pamäťové obmedzenia

Pri vytváraní grafu, resp. modelu NS sme narazili na momentálne pamäťové obmedzenia. Pri vytváraní modelu s dimenziou vektora pre slová v matici GloVe sme pozorovali, že v prípade dimenzie väčšej ako 300 je minimálna veľkosť pamäte 3GB. Na túto veľkosť taktiež vplýva veľkosť slovníka, ktorý je vytvorený z trénovacích dát. Momentálne teda pracujeme v závislosti od týchto obmedzení. Nakoľko nevieme, či je nutné tieto hyper-parametre zväčšovať, rozhodli sme sa ponechať tvorbu modelu v týchto podmienkach.

6.3 Počiatočné výsledky

Navrhované riešenie, resp. model NS momentálne pracuje na architektúre kóderekóder. Model využíva len základnú architektúru a je limitovaný vo viaceru smeroch. Ako vstupom sú úvodné slová vybraných článkov. Vstupná sekvencia je tak ohraničená na 30 znakov. Toto ohraničenie je najmä z dôvodu, že takýto jednoduchý model bez mechanizmu pozorovania pri vstupných sekvenciach dlhších ako 30 znakov výrazne stráca na stratovej funkcií a taktiež výrazne klesá BLEU skóre [24]. Model je trénovaný len postupne na všetkých článkoch z trénovacej množiny. Nevytvárame teda jednotnú reprezentáciu dokumentov. Dokumenty z jednej kategórie sú trénované nezávisle na sebe. Pre tieto dokumenty sme však dokázali vo väčšine prípadov vygenerovať zhodné názvy kategórii. To však nie je až tak smerodajné nakoľko vstupné sekvencie sú v porovnaní s celkovou dĺžkou dokumentov veľmi krátke a nie je zaručené, že v článkoch sa v úvodných častiach nachádza najväčšia výpovedná hodnota.

6.3.1 Model seq2seq bez mechanizmu pozorovania

Doposiaľ implementovaný model architektúry kóder-dekóder, neobsahuje spomínaný mechanizmus pozorovania. Naše doterajšie výsledky a taktiež iné štúdie však ukazujú, že v prípade dlhých vstupných sekvencií je nutné tento mechanizmus zahrnúť do modelu NS. Pomocou tohto mechanizmu je možné model upravovať už pre dlhšie sekvencie ako ~ 40 znakov.

Doterajšie konfigurácie hyper-parametrov sme upravovali v závislosti na predchádzajúcich výsledkoch modelu len na základe vizuálnej kontroly vygenerovaných názvov kategórií a taktiež na základe hodnoty loss funkcie.

6.3.2 Konfigurácia hyper-parametrov

Trénovanie s obmenou nasledujúcich hyper-parametrov bolo prevádzané na vybranom trénovacom súbore (1000 článkov, 100 kategórií). V nasledujúcich tabuľkách sú uvedené konkrétnie hodnoty jednako NS ale aj vstupného textu (matice vektorov slov) a loss funkciu v závislosti od času (epochy).

6.3.2.1 Dlhý vstup bez filtrovania neznámych slov

V rámci prvej konfigurácie, sme veľkosť vstupného textu obmedzili len mierne (na 300 znakov).

Tabuľka 6.1 - Konfigurácia hyper-parametrov

Hyper-parameter	Hodnota
Veľkosť dimenzie	256
Max. veľkosť vstupnej sekvencie	300
Počet epoch	940
Veľkosť slovníka	1000
Počet RNS vrstiev	3
Veľkosť dávky	16

The chart displays the loss function over 910 epochs. The y-axis represents the loss value, ranging from 0.00 to 7.00. The x-axis represents the epoch number, ranging from 10 to 910. The loss starts at approximately 6.0 and drops sharply to around 2.5 by epoch 70. It then fluctuates slightly between 2.2 and 2.6 for the remaining epochs.

Epoch	Loss
10	6.0
20	5.5
30	4.5
40	3.8
50	3.2
60	2.8
70	2.5
80	2.4
90	2.3
100	2.4
150	2.3
200	2.4
250	2.3
300	2.4
350	2.3
400	2.4
450	2.3
500	2.4
550	2.3
600	2.4
650	2.3
700	2.4
750	2.3
800	2.4
850	2.3
900	2.4
910	2.3

Vo vygenerovaných názvoch kategórií môžeme vidieť, že konfigurácia nebola správna a všetky vygenerované názvy obsahovali neznáme tokeny *unk*. Je to najmä kvôli nefiltrovaniu článkov, ktoré spadajú práve do tých kategórií, ktoré obsahujú neznáme slová.

Tabuľka 6.2 - Ukážka generovaných sekvencií

Vstupný text	Originálny názov kategórie
	Generovaný názov kategórie
Narrative traffic is data communications consisting of unk or encrypted messages written in a natural language and transmitted in accordance with standard unk and procedures Examples of narrative traffic include Messages that are placed on paper tape and transmitted via a teletypewriter TTY	Data transmission unk unk unk
In Australian aboriginal mythology specifically Gurra and Bandicoot Karora is a creator god He was born in a lake and after fathering many children he returned there to slumberKarora is also A town in Eritrea A playgroup in Harlow Essex An Australian open source software group focusing	Australian Aboriginal mythology unk unk unk unk
In Maya mythology Xumucane was one of the thirteen creator gods who helped construct humanity	Maya gods unk unk

6.3.2.2 Dlhý vstup s filtrovaním neznámych slov

Po prvom výsledku generovania názvu kategórie sme zistili, že v článkoch sa nachádza veľa slov, ktoré sa nenachádzajú v slovníku ako to je opísane v časti 6.2.3.

Tabuľka 6.3 - Konfigurácia hyper-parametrov

Hyper-parameter	Hodnota
Veľkosť dimenzie	300
Max. veľkosť vstupnej sekvencie	100
Počet epoch	2000
Veľkosť slovníka	4000
Počet RNN vrstiev	3
Veľkosť dávky	32

The chart displays the training progress of the model. The Y-axis represents the Loss function, ranging from 0.00 to 7.00. The X-axis represents the Epochs, ranging from 10 to 910. The blue line shows a rapid initial drop in loss, followed by a slow, steady decrease that plateaus after approximately 200 epochs.

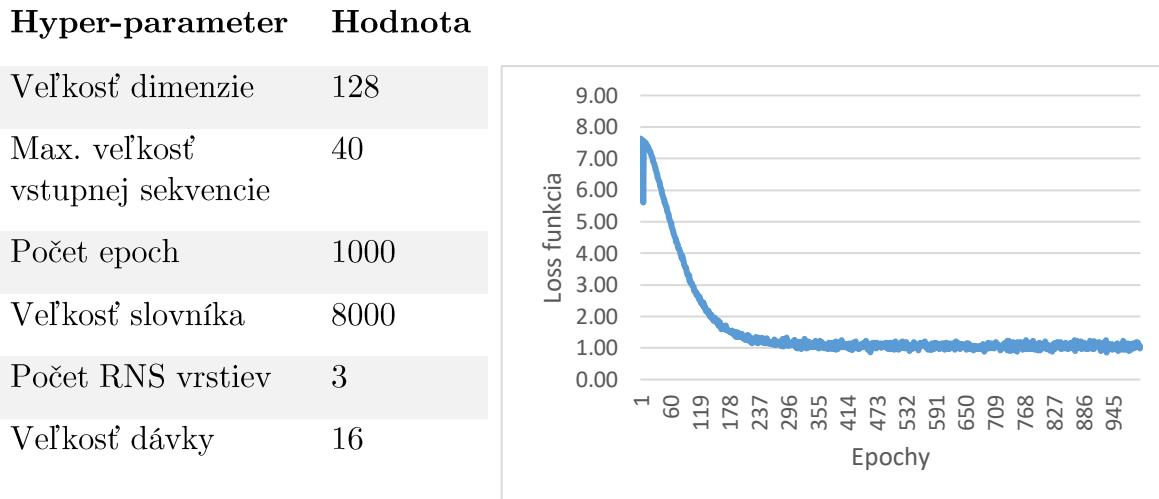
Tabuľka 6.4 – Generované názvy kategórií pre model bez mechanizmu pozorovania

Vstupný text	Originálny názov kategórie
	Generovaný názov kategórie
Narrative traffic is data communications consisting of unk or encrypted messages written in a natural language and transmitted in accordance with standard unk and procedures Examples of narrative traffic include Messages that are placed on paper tape and transmitted via a teletypewriter TTY	Data transmission Articles
In Australian aboriginal mythology specifically Gurra and Bandicoot Karora is a creator god He was born in a lake and after fathering many children he returned there to slumberKarora is also A town in Eritrea A playgroup in Harlow Essex An Australian open source software group focusing	Australian Aboriginal mythology Articles
In Maya mythology Xumucane was one of the thirteen creator gods who helped construct humanity	Maya gods Articles

6.3.2.3 Filtrovanie neznámych slov a skrátený vstup

Pred trénovaním modelu, boli z dátovej množiny odstránené všetky články, ktoré sa nachádzali v kategórii, ktorá obsahovala token *unk*. Obmedzená bola taktiež veľkosť vstupnej sekvencie.

Tabuľka 6.5 - Konfigurácia hyper-parametrov



V rámci vygenerovaných názvov kategórií sme pozorovali, že model momentálne negeneruje názvy, v ktorých sa nachádzajú čísla. V porovnaní s predchádzajúcou konfiguráciou, sme pozorovali výrazné zlepšenie po obsahovej stránke vygenerovaných názvov kategórií, viď. tabuľku Tabuľka 6.6.

Tabuľka 6.6 - Ukážka generovaných sekvenčí

Vstupný text	Originálny názov kategórie
	Generovaný názov kategórie
Events and trends c 1674	1670s BC
	BC
thumbIn Norse mythology Hildi	Mythological queens
	Articles containing
Kalamunda National Park is a n	National parks of Western
	Australia
	African

Pri nasledujúcom nastavení parametrov sme doposiaľ dosiahli najlepšie výsledky. Jednak najnižšie hodnoty loss funkcie ale aj viditeľné výsledky pri vygenerovaných kategóriách.

Tabuľka 6.7 - Konfigurácia hyper-parametrov

Hyper-parameter	Hodnota
Veľkosť dimenzie	300
Max. veľkosť vstupnej sekvenčie	40
Počet epoch	1000
Veľkosť slovníka	8000
Počet RNN vrstiev	3
Veľkosť dávky	128

V porovnaní s predchádzajúcou konfiguráciou sme pozmenili najmä veľkosť dimenzie. Zväčšením dimenzie sme dosiahli pomerne obšírnejšie a po obsahovej stránke lepšie výsledky ako to bolo v predchádzajúcej konfigurácii.

Tabuľka 6.8 - Ukážka generovaných sekvencii

Vstupný text	Originálny názov kategórie
	Generovaný názov kategórie
In Australian Aboriginal	Australian Aboriginal mythology
	Aboriginal Aboriginal mythology
In Maya mythology X	Maya gods
	Aztec gods
Kalamunda National Park is a n	National parks of Western Australia
	All in

7 Zhrnutie a ďalšia práca

Navrhli sme a implementovali model neurónovej siete na základe existujúcej architektúry kóder-dekóder. Na základe zdroja Wikipédie, sme vytvorili vlastný dátový súbor vďaka čomu je možné takto modifikované údaje použiť pre trénovanie modelu NS. Momentálne má model najvýraznejšie obmedzenie v počte vstupných slov. Predstavili sme taktiež návrh jednotnej reprezentácie dokumentov v rámci jednej kategórie. Počiatočný model zatiaľ neevaluujeme žiadnu z navrhovaných metrík.

7.1 Implementácia jednotnej reprezentácie dokumentov

Ako ďalším krokom pri implementácii navrhnutého modelu bude zapracovanie tvorby jednotnej reprezentácie dokumentov v kategóriách. Túto reprezentáciu budeme vytvárať na základe spoločných n-gramov a najfrekventovanejších výrazov. Novovytvorený dokument sa bude teda skladať buď z viet, v ktorých sa nachádzajú najfrekventovanejšie slová v rámci kategórie alebo zo najfrekventovanejších n-gramov. V nasledujúcom vývoji práce chceme porovnať obe metódy a zistiť, ktoré dokumenty budú lepšie opisovať spoločné črty v rámci jednej kategórie.

7.2 Aplikovanie mechanizmu pozorovania

Jednou z prvých možností aplikovania tohto mechanizmu bude zapracovať do LSTM neurónov mechanizmus pozorovania implementovaný priamo v knižnici Tensorflow. Predpokladáme, že aplikovaním tohto mechanizmu budeme schopní pracovať aj so sekvenciami dlhšími ako 40 znakov. Existujúci model je nutné teda len upraviť a zapracovať tak tzv. *AttentionWrapper*, implementovaný priamo v knižnici Tensorflow.

7.3 Evaluácia

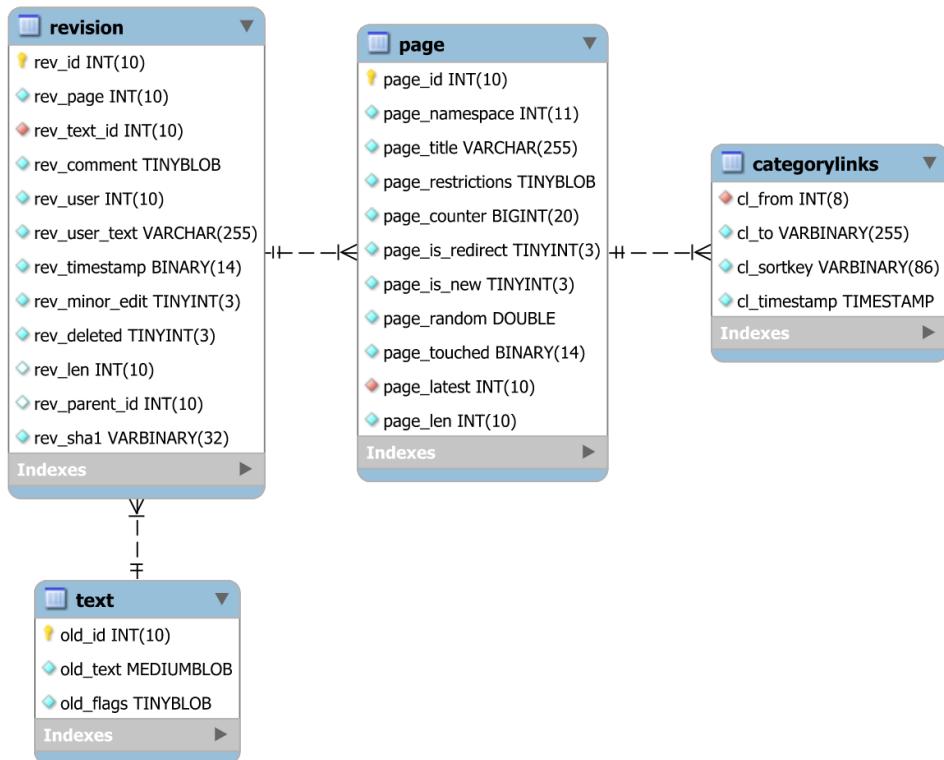
Na základe navrhovaných metrík pre vyhodnocovanie vygenerovaného textu, budeme implementovať metriku BLEU ako aj viacero typov ROUGE metrík. Okrem automatického vyhodnocovania, plánujeme vytvoriť dotazník, pomocou ktorého budeme zisťovať či vygenerované texty respondenti označia za názvy kategórií s väčšou výpovednou hodnotou v porovnaní s pôvodnými.

A Príloha - Technická dokumentácia

A.1 Model databázy Wikipédie

Pred tým ako je možné importovať dátu do databázy, musíme inicializovať, resp. vytvoriť model na základe modelu na obrázku Obrázok 7.1. Pre naše účely postačuje vytvoriť menšiu verziu databázy (len 4 tabuľky). Dátu pozostávajú z dvoch častí³:

- *articles* (enwiki-yyyymmdd-pages-articles.sql)
 - o tabuľky: *revision*, *page*, *text*
- *categorylinks* (enwiki-yyyymmdd-categorylinks.sql)
 - o tabuľky: *categorylinks*



Obrázok 7.1 - Výrez dátového modelu Wikipédie

³ Dostupné na <https://dumps.wikimedia.org/archive/enwiki/20100312/>

A.2 Filtrovanie dátového súboru

Pred samotným spracovaním dát sme museli vyfiltrovať články a kategórie obsahujúce spomínané výrazy v časti 6.1.1. Robili sme tak priamo v MySQL databáze:

```
select
    p.page_id,
    p.page_title,
    convert(t.old_text using utf8) as txt,
    convert(c.cl_to using utf8) as cat
from wiki.page p
    join wiki.revision r on p.page_latest = r.rev_id
    join wiki.text t on t.old_id = r.rev_text_id
    join wiki.categorylinks c on c.cl_from = p.page_id
) as t
where 1=1
    and t.txt not like '%#REDIRECT%'
    and t.cat not like '%#All%pages%'
    and t.cat not like '%#All%articles%'
    and t.cat not like '%Disambiguation_pages%'
    and t.cat not like '%Articles%lacking%%'
    and t.cat not like '%Articles%needing%cleanup%'
    and t.cat not like '%Articles%from%'
    and t.cat not like '%Wiki%'
    and t.cat not like '%Redirects%'
    and t.cat not like '%All%articles%'
    and t.cat not like '%stubs'
```

Text článku a názov kategórie je vo formáte BLOB a musíme ho teda konvertovať do čitateľnej podoby. Výsledok hľadania uložíme do súboru vo formáte JSON, ktorý je pripravený na ďalšie spracovanie pomocou prípravenej webovej aplikácie.

A.3 Spracovanie dát

Pred spracovaním dát za pomoci implementovaného riešenia je nutné text najskôr transformovať s využitím knižnice *txtwiki* (viď. Spracovanie dát). Súbor vo formáte json načítame do pamäte:

```
1. def load_raw_data(filename):
2.     with open(filename, 'r', encoding='utf8') as fp:
3.         raw_data = pd.DataFrame(json.load(fp))
```

Po načítaní súboru môžeme začať dátu spracovávať a aplikovať regulárne výrazy a taktiež limitovať maximálnu dĺžku vstupnej sekvencie (článku).

```
1. def process_data():
2.
3.     filename = path.join(file_path, 'train_1K.json')
4.     raw_data = load_raw_data(filename)
5.     raw_data['category'] = raw_data['category'].apply(lambda x: x.replace('_', " "))
6.     raw_data['parsed_text'] = raw_data['parsed_text'].apply(lambda x:
7.         re.sub('{(.*)}', "", x))
8.     raw_data['parsed_text'] = raw_data['parsed_text'].apply(lambda x:
9.         re.sub('\|(.*)=(.*?)*', "", x))
10.    raw_data['parsed_text'] = raw_data['parsed_text'].apply(lambda x:
11.        re.sub('^{.*}', "", x))
12.    raw_data['parsed_text'] = raw_data['parsed_text'].apply(lambda x:
13.        re.sub('[\n]+', "", x))
14.    raw_data['parsed_text'] = raw_data['parsed_text'].apply(lambda x:
15.        x.split('Category:')[0])
16.    ...
```

Názvy kategórií ako aj texty článkov rozdelíme postupne po vetách na tokeny za pomoci knižnice NLTK a metódy *tokenize*.

```
1. def tokenize_sentence(sentence):
2.     return ' '.join(list(tokenize(sentence)))
```

Najdôležitejšou časťou je vytvorenie indexov k slovám a slová k indexom. Okrem indexov, vytvárame taktiež zoznam početnosti jednotlivých slov.

```
1. def index_data(tokenized_sentences, vocab_size):
2.
3.     freq_dist = nltk.FreqDist(itertools.chain(*tokenized_sentences))
4.     vocab = freq_dist.most_common(vocab_size)
5.     print ('Vocab length: {}'.format(len(vocab)))
6.
7.     idx2word = ['_'] + [UNK] + [x[0] for x in vocab]
8.     word2idx = dict([(w, i) for i, w in enumerate(idx2word)])
9.
10.    return (idx2word, word2idx, freq_dist)
```

Nasleduje odsadenie (padding) sekvencií kratších ako maximálna dĺžka. Do týchto sekvencií doplnujeme číslo 0.

```
1. def pad_seq(seq, lookup, max_length):
2.     indices = []
3.
4.     for word in seq:
5.         if word in lookup:
6.             indices.append(lookup[word])
7.         else:
8.             indices.append(lookup[UNK])
9.
10.    return indices + [0]*(max_length - len(seq))
```

A.4 Kóder-dekóder

Po úvodnom spracovaní dát je model NS vytvorený za pomoci triedy Seq2Seq nasledujúcim spôsobom s využitím viacerých hyperparametrov.

```
11. model = seq2seq_wrapper.Seq2Seq(xseq_len=xseq_len,
12.                                     yseq_len=yseq_len,
13.                                     xvocab_size=xvocab_size,
14.                                     yvocab_size=yyvocab_size,
15.                                     ckpt_path='ckpt/',
16.                                     emb_dim=emb_dim,
17.                                     num_layers=3,
18.                                     epochs=1000
19.                                     )
```

Ďalším krokom je natrénovanie modelu na trénvacej množine (*train_batch_gen*) a jeho validácia na validačnej množine (*val_batch_gen*).

```
1. sess = model.train(train_batch_gen, val_batch_gen)
```

Výstupom počas trénovania sú údaje o aktuálnej iterácii a veľkosti loss funkcie:

```
[2017-11-29 21:23:16.847176] - Building Graph
[2017-11-29 21:23:29.904057] - Training started
[2017-11-29 21:23:35.153324] - Model saved to disk at iteration #1
[2017-11-29 21:23:35.153376] - val loss : 7.195374
[2017-11-29 21:23:37.441201] - Model saved to disk at iteration #2
[2017-11-29 21:23:37.441249] - val loss : 7.098650
```

Po natrénovaní modelu môžeme spustiť generovanie názvov kategórií pomocou funkcie *predict*.

```
1. output = model.predict(sess, input_)
2.
3. for ii, oi in zip(input_.T, output):
4.     q = data_utils.decode(sequence=ii, lookup=metadata['idx2word'], separator=' ')
5.     decoded = data_utils.decode(sequence=oi, lookup=metadata['idx2word'], separator=' ')
6.             '.split(' ')
7.     if decoded.count('unk') == 0:
8.         if decoded not in replies:
9.             print('description : [{0}]; category : [{1}]'.format(q,
10.                         ''.join(decoded)))
```

Pri inicializácii grafu modela NS najskôr nainicializujeme vstupné pole pre kódér a taktiež dekóder. Pri inicializácii poľa pre dekóder pridávame na začiatok kľúčove slovo GO, ktoré predstavuje špeciálny token, ktorým začína sekvencia výstupov.

```
1.     # vstupy kodera
2.     self.enc_ip = [ tf.placeholder(shape=[None,],
3.                                     dtype=tf.int64,
4.                                     name='ei_{0}'.format(t)) for t in range(xseq_len) ]
5. ...
6.     # vstupy dekodera : 'GO' + [ y1, y2, ... yt-1 ]
7.     self.dec_ip = [ tf.zeros_like(self.enc_ip[0], dtype=tf.int64, name='GO') ]
8.                   + self.labels[:-1]
```

Nasleduje vytvorenie LSTM neurónu a teda celkového n-vrstvového modelu NS.

```
1.     # vytvorenie LSTM neuronu
2.     basic_cell = tf.contrib.rnn.DropoutWrapper(
3.         tf.contrib.rnn.BasicLSTMCell(emb_dim, state_is_tuple=True),
4.         output_keep_prob=self.keep_prob)
5.     # n-vrstvovy model
6.     stacked_lstm = tf.contrib.rnn.MultiRNNCell([basic_cell]*num_layers,
7.                                                 state_is_tuple=True)
```

Vytvorený model vstupuje ako argument pre vstavanú knižnicu Tensorflow, konkrétnie pre funkciu *embedding_rnn_seq2_seq*. Okrem modelu NS vstupujú do tejto funkcie aj veľkosť slovníka pre vstupné a výstupné sekvencie, veľkosť dimenzie pre vektory slov, výstupy kódera a dekódiera.

```

1.         with tf.variable_scope('decoder') as scope:
2.             # vytvorenie seq2seq modelu
3.             self.decode_outputs, self.decode_states =
4.                 tf.contrib.legacy_seq2seq.embedding_rnn_seq2seq(self.enc_ip, self.dec_ip, stacked_lstm,
5.                                                               xvocab_size, yvocab_size, emb_dim)
6.             # testovanie modelu, kde vystup z minuleho casoveho kroku (timestep)
7.             # je vstupom aktualneho casoveho kroku
8.             self.decode_outputs_test, self.decode_states_test =
9.                 tf.contrib.legacy_seq2seq.embedding_rnn_seq2seq(
10.                   self.enc_ip, self.dec_ip,
11.                   stacked_lstm, xvocab_size,
12.                   yvocab_size, emb_dim,
13.                   feed_previous=True)

```

Počas každej iterácie, model vypočíta loss funkciu.

```

1.     # loss funkcia
2.     self.loss = tf.contrib.legacy_seq2seq.sequence_loss(self.decode_outputs,
3.                                                       self.labels, loss_weights, yvocab_size)

```

Pre model nastavíme rýchlosť učenia (z angl. “learning rate”) na optimalizačný algoritmus – Adam.

```

1.     # rychlosť učenia - Adam
2.     self.train_op =
3.         tf.train.AdamOptimizer(learning_rate=lr).minimize(self.loss)

```

B Príloha - Nasledujúci plán práce

TÝŽDEŇ	POPIS PRÁCE
1	Implementovanie mechanizmu pozornosti.
2	Ladenie mechanizmu pozornosti – zistenie max. dĺžky textov. Trénovanie modelu na dlhých textoch.
3	Konkretizácia návrhu implementácie pre jednotnú reprezentáciu dokumentov z kategórií. Začiatok implementácie navrhnutého modelu.
4	Pokračovanie v implementácii navrhnutého riešenia pre jednotnú reprezentáciu dokumentov. Analýza novovytvorených dokumentov.
5	Implementácia metrík vyhodnocovania BLEU, ROUGE.
6	Trénovanie modelu spolu s implementovanými metrikami a jednotnej reprezentáciu dokumentov.
7	Vytvorenie rozsiahleho dátového súboru pre trénovanie, testovanie a evaluáciu modelu.
8	Ladenie konfigurácií modelu.
9	Vyhodnocovanie na základe dotazníka. Ladenie konfigurácií modelu a vyhodnocovanie modelu.
10	Sumarizácia pozorovaní. Spísanie poznatkov a pozorovaní do práce.
11	Príprava na obhajobu.

Použitá literatúra

- [1] KARPATHY, Andrej; FEI-FEI, Li. Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. str. 3128-3137.
- [2] JU, Ronghui, et al. An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis. In: Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. IEEE, 2015. str. 2276-2283.
- [3] YASOTHA, R.; CHARLES, E. Y. A. Automated text document categorization. In: Intelligent Computing and Information Systems (ICICIS), 2015 IEEE Seventh International Conference on. IEEE, 2015. str. 522-528.
- [4] MIKOLOV, Tomas, et al. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.
- [5] LE, Quoc V. Building high-level features using large scale unsupervised learning. Proceedings of the International conference on acoustics, speech and signal processing. IEEE, 2013. 8595-8598.
- [6] BENGIO, Yoshua. Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2009, 2.1: 1-127.
- [7] CHO, Kyunghyun, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.
- [8] Lenc, L., & Král, P. Deep Neural Networks for Czech Multi-label Document Classification. Zcu.Cz, 1-12.
- [9] MACHOVÁ, Kristína. Strojové učenie: Princípy a algoritmy. Elfa, 2002.
- [10] KVÁSNIČKA, Vladimír, et al. Úvod do teórie neurónových sietí.
- [11] AYANA, Shiqi Shen; LIU, Zhiyuan; SUN, Maosong. Neural headline generation with minimum risk training. arXiv preprint arXiv:1604.01904, 2016.
- [12] NÁVRAT, P., et al. Umelá inteligencia. Slovenská technická univerzita v Bratislave, Bratislava.
- [13] STEINBACH, Michael, et al. A comparison of document clustering techniques. In: KDD workshop on text mining. 2000. str. 525-526.
- [14] LOPYREV, Konstantin. Generating news headlines with recurrent neural networks. arXiv preprint arXiv:1512.01712, 2015.
- [15] HORÁK, Radovan, et al. Tokenizácia, lematizácia a morfologická anotácia Slovenského národného korpusu. 2004.
- [16] Y. Hayashi, H. Yanagimoto. Title Generation with Recurrent Neural Network. 2016. 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Kumamoto, 2016, str. 250-255. doi: 10.1109/IIAI-AAI.2016.109
- [17] CRUSE, Holk. Neural networks as cybernetic systems. Brain, minds, and media. [online] Dostupné na: <http://www.brains-minds-media.org/archive/289>. [Dátum cit. 10.5.2017]
- [18] LEUNG, K. Ming. Naive bayesian classifier. Polytechnic University Department of Computer Science/Finance and Risk Engineering, 2007.

- [19] DREISEITL, Stephan; OHNO-MACHADO, Lucila. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 2002, 35.5: 352-359.
- [20] STEINBERGER, Josef; JEZEK, Karel. Evaluation measures for text summarization. *Computing and Informatics*, 2009, 28.2: 251-275.
- [21] SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems*. 2014. p. 3104-3112.
- [22] XU, Lei, et al. Topic Sensitive Neural Headline Generation. arXiv preprint arXiv:1608.05777, 2016.
- [23] HUANG, Anna. Similarity measures for text document clustering. In: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand. 2008. p. 49-56.
- [24] LUONG, Minh-Thang; PHAM, Hieu; MANNING, Christopher D. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.