Fakulta elektrotechniky a informatiky

Fakulta Elektrotechniky a Informatiky

Katedra kybernetiky a umelej inteligencie

Predmet : **Základy hlbokého učenia**
kurz 2018 / 2019

Zadanie číslo # 1 (esej) :

Hlboké učenie pre spracovanie prirodzeného jazyka

kľúčové slova : hlboké učenie, konvolučná neurónová sieť, spracovanie prirodzeného jazyka

---

Spracovali :
Karina Cimborová, Martin Durkáč

KKUI
Katedra kybernetiky a umelej inteligencie

Obsah

1 Úvod

Hlboké učenie sa dostalo do povedomia ľudí približne od roku 2006, kedy v práci [4] bola popísaná myšlienka neurónovej siete s početným množstvom skrytých vrstiev, ktorú bolo možné natrénovať, ak váhy boli pri inicializácii vhodne zvolené. Hoci metódy použité pri tomto výskume boli medzičasom prekonané, v tom čase to bol mýľnik [5]. V dnešnej dobe sa ukazuje ako prosperujúca oblasť výskumu strojového učenia. Vo všeobecnosti, cieľom je napodobniť funkciu ľudského mozgu, ktorý je schopný spracovávať a zároveň učiť sa z komplexných vstupných dát a dobre riešiť rôzne druhy zložitých úloh. Úspešne sa aplikuje do niekoľkých oblastí, medzi ktoré patrí napríklad spracovanie obrazov, zvukov, textu [7]. Oblasť aplikácie metód hlbokého učenia, ktorou sa bude zaoberať táto esej je spracovanie prirodzeného jazyka.

2 Špecifikácia oblasti aplikácie a popis problému

Hlavným cieľom spracovania prirodzeného jazyka je vytvoriť programy schopné spracovať a porozumieť prirodzeným jazykom. Prirodzený jazyk sa vzťahuje na jazyk, ktorým hovoria alebo píšu ľudské bytosti, na rozdiel od umelých jazykov, napríklad počítačov, matematických alebo logických jazykov. V skutočnosti sa spracovanie jazyka netýka priamo jazyka, zahŕňa však jazykovo kódované texty v konkrétnom jazyku. Pod týmto všeobecným názvom môžeme rozumieť dialógy, alebo menšie písomné alebo ústne celky, pod ktoré patria odseky alebo vety [8].
Môžeme uvažovať o dvoch druhoch spracovania [1]:

- spracovanie (nazývané aj analýza prirodzeného jazyka), ktoré prebieha na lingvistických dátach (napríklad text) na opravu textu alebo preklad, táto transformácia vyžaduje vo väčšine prípadov medzikrok, ktorého cieľom je extrahovať textové reprezentácie. Vstupom je v tomto prípade text a výstupom je nový text alebo textová reprezentácia.

  text1 ->reprezentácia textu 1 ->text 2

  Pod reprezentáciou sa myslia procesy, ako napríklad text napísaný v inom jazyku ako je prirodzený jazyk a explicitne vyjadruje implicitné informácie o texte (súbor kľúčových slov, syntaxový strom, ...).

- spracovanie prirodzeného jazyka môže vykonávať aj opačnú úloha a to generovanie prirodzeného jazyka tým, že na vstup je potrebná textová reprezentácia na vytvorenie prirodzeného textu. Ak však neexistuje reprezentácia textu a sú k dispozícii iba surové dáta, najprv tieto dáta musia byť preložené do textovej reprezentácie.

  surové dáta ->textová reprezentácia ->prirodzený text

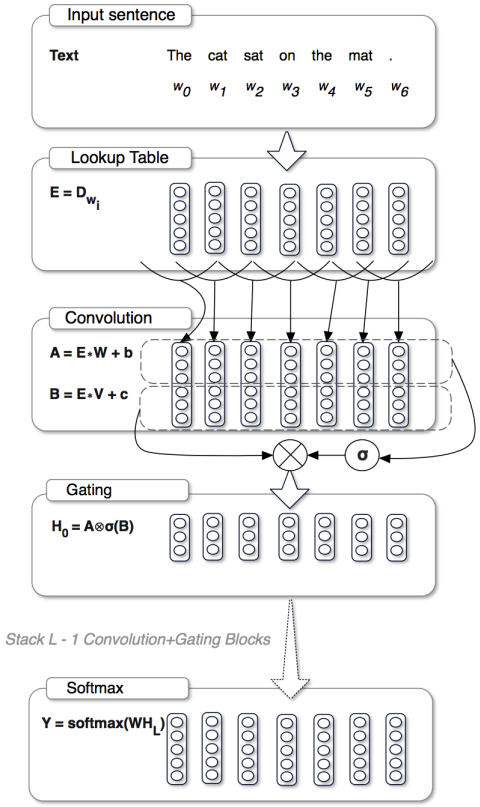
Spracovanie vyžaduje rôzne nástroje, ktoré je možné rozdeliť do kategórií [1]:

- lingvistika - popisuje rôzne formálne znalosti jazyka
- matematika - ponúka výpočtové základy potrebné na konverziu lingvistických techník do výpočtového procesu

- informatika - ponúka prostriedky a spôsoby, ako dokončiť všetky projekty, ktoré vyžadujú spracovanie prirodzeného jazyka

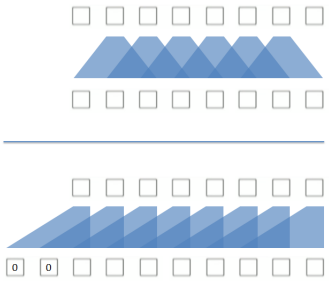
3 Topológia a popis neurónovej siete

V tejto eseji sme sa rozhodli opísať výskum Yann N. Dauphina a kolektívu, ktorý v článku [3], opísali nový prístup ku spracovaniu prirodzeného jazyka prostredníctvom gated konvolučných neurónových sietí (CGNN). Gating sa ukazuje veľmi podstatné pre rekurentné neurónové siete, aby boli možné dosiahnuť čo najlepší výkon. Rekurentné spojenia sú nahrádzané prechodnými gated konvolúciami. Gated lineárne jednotky, prezentované pri tejto neurónovej sieti, redukujú problém miznúceho gradientu pri hlbokých architektúrach tým, že poskytujú lineárnu cestu pre gradienty pri zachovaní nelineárnych schopností [3].



Obr. 1: Architektúra gated konvolučnej siete [3]

Ako je možné vidieť na Obr. 1, na vstupe sa nachádza sekvencia slov, ktoré je potrebné pretransformovať na vektor pomocou look-up tabuľky. Nasleduje niekoľko vrstiev gated konvolúcií, ktorých výsledkom je výsledný stav skrytých vrstiev. Každá vrstva je zložená z konvolučného bloku, ktorý produkuje dva samostatné konvolučné výstupy a gating bloku, ktorý využíva výstup z jedného konvolučného výstupu na kontrolu informácie z druhého konvolučného výstupu, ktorá je potrebná pre nasledujúce vrstvy. Konvolúcia vo všeobecnosti má okno šírky



Obr. 2: Rozdiel medzi konvolúciami [2]

k pre daný časový okamih, vďaka čomu má informácie o predchádzajúcich alebo nasledujúcich časových okamihoch. Konvolúcia použitá v tomto prípade (viď Obr. 2 dole) dokáže získať informácie len o prechádzajúcich stavoch a súčasnom stave [2].
Výstup z konvolúcií a gating mechanizmu je zabalený do reziduálneho bloku, ktorý spája vstup s výstupom. Nakoniec je použitá funkcia adaptívneho softmaxu, ktorá vyberá nasledujúce slovo z obrovského zoznamu slov, ktorý daný model obsahuje [3].

4 Hyperparametre a parametre siete

Hodnoty hyperparametrov boli zvolené krížovou validáciou s náhodným vyhľadávaním na validačnej množine. Pre architektúru modelu boli vybrané hyperparametre ako počet reziduálnych blokov, ktorý sa pohybuje v intervale 1, ..., 10, veľkosť embeddings medzi 128, ..., 256, počet jednotiek od 128 do 2048 a šírka kernelu medzi 3, ..., 5.
Pokiaľ ide o optimalizáciu, koeficient učenia bol navzorkovaný jednotne v intervale [1, ... 2] , moment rovný 0.99 a orezávanie nastavené na 0.1. Tieto optimálne hodnoty sa pri striedaní datasetov nemenili [3].

5 Výber L funkcie a jej matematický popis

Trénovanie gated konvolučných sietí prebiehalo prostredníctvom Nesterovho momenta [6]. Nesterov momentum prístup je mierna modifikácie gradientného zostupu, ktorý urýchľuje natrénovanie a výrazne zlepšuje konvergenciu. Pre definovanie samotného Nesterovho momenta je potrebné zacať definovaním klasickej metódy gradientného zostupu:

$$\theta_{t+1} = \theta_t - \eta \nabla l(\theta), \tag{1}$$

kde ∇ je gradient stratovej funkcie l, ktorá hovorí akým smerom je potrebné sa pohybovať v priestore parametrov, aby sme zvýšili stratu. Odpočítaním tohto násobku stratu znižujeme o vzdialenosť, ktorá je závislá od rýchlosti učenia η.

Ďalším zlepšením metódy gradientného zostupu je gradientný zostup s momentom. Momentum urýchľuje pohyb pozdĺž smeru silného zlepšenia (zníženie strát) a tiež pomáha sieti vyhnúť

sa lokálnym minimám. Intuitívne súvisí s pojmom hybnosti vo fyzike. Momentový gradientný zostup je popísany ako:

$$\begin{aligned} v_{t+1} &= \mu v_t - \eta \nabla l(\theta), \\ \theta_{t+1} &= \theta_t + v_{t+1}, \end{aligned} \tag{2}$$

kde v je rýchlosť a μ je parameter hybnosti, ktorý riadi, ako rýchlo sa môže rýchlosť meniť a ako veľmi lokálny gradient ovplyvňuje dlhodobý pohyb. V každom časovom kroku sa rýchlosť aktualizuje podľa lokálneho gradientu a potom sa aplikuje na parametre.

Poslednou iteráciou je samotná metóda Nesterovho momenta, ktorý je malou zmenou oproti gradientnému zostupu s momentom. V tomto prípade ale gradientový bod nie je vypočítany z aktuálnej pozície θ_t v priestore parametrov, ale z pozície $\theta_{intermediate} = \theta_t + \mu v_t$. To pomáha, pretože zatiaľ čo gradientový bod vždy smeruje správnym smerom, hybnosť nemusí. Ak hybnosť ukazuje nesprávnym smerom, gradient môže stále „ísť späť“ a opraviť ho v rovnakom kroku aktualizácie.

$$\begin{aligned} v_{t+1} &= \mu v_t - \eta \nabla l(\theta + \mu v_t), \\ \theta_{t+1} &= \theta_t + v_{t+1}, \end{aligned} \tag{3}$$

Rovnica 3 popisuje výsledný vzťah pre metódu Nesterovho momenta.

6 Matematický popis jednotlivých častí dopredného prechodu siete

Vstupom do modelu opísaného v článku [3] je sekvencia slov reprezentovaných tenzorom X o veľkosti [seq_length, emb_sz], kde seq_length je dĺžka sekvencie a emb_sz je dimenzionalita vložených dát. Ďalej v neurónovej sieti je použitých viacero vrstviev gated konvolúcií na vytvorenie konečného skrytého stavu H o veľkosti [seq_length, c_out].

Ako už bolo spomenuté v kapitole 3, konvolučný blok bude produkovať dva samostatné konvolučné výstupy A a B. Prvý konvolučný výstup sa vypočíta ako:

$$A = X * W + b, \tag{4}$$

kde X je tenzor o veľkosti [seq_length, emb_sz], W je konvolučný filter o veľkosti [k, in_c, out_c], b je vektor biasu dĺžky *out_c* . Samotný konvolučný filter bude mať pre prvú vrstvu *in_c* rovnú *emb_sz* a *out_c* je počet výstupných kanálov. Po aplikovaní konvolúcie bude mať vektor A veľkosť [seq_length, out_c]. Obdobne sa vypočíta aj vektor B pričom na neho je aplikovaný iný filter a vektor biasov c:

$$B = X * V + c, \tag{5}$$

na tieto dva výstupy je použitý mechanizmus definovaný v [3] ako "gated linear unit"(GLU). Tento mechanizmus obsahuje násobenie po prvku definovane následovne:

$$A \otimes sigmoid(B),$$

respektíve (6)

$$(X * W + b) \otimes sigmoid(X * V + c),$$

kde B obsahuje informácie o tom aké informácie sa odovzdajú z vektora A pre ďalšiu vrstvu v hierarchi. Koncepčne je mechanizmus gated dôležitý, pretože umožňuje výber slov alebo funkcií, ktoré sú dôležité pre predpovedanie ďalšieho slova a poskytuje mechanizmus na učenie a odovzdávanie len relevantných informácií.

Dauphin a kol. tiež pridali zvyškové preskočenie spojenia do každej vrstvy, podobne ako architektúra ResNet. Zvyškové spojenia minimalizujú problém miznúceho gradientu, čo umožňuje vybudovať oveľa hlbšie siete. Potom vstup do vrstvy (X) je pripočítaný do konvolučného vstupu A vynosobeným po prvku s vstupom B:

$$X + (A \otimes sigmoid(B)),$$

respektíve (7)

$$X + ((X * W + b) \otimes sigmoid(X * V + c))$$

Posledným elementom, ktorý Dauphin s kol. využili bola štruktúra úzkeho miesta v rámci vrstvy, taktiež ako v architektúre ResNet. Účelom tejto štruktúry je znížiť výpočtové náklady na konvolučnú operáciu tým, že sa najprv zníži rozmer údajov.

V publikácií [3] bolo 5 takýchto vrstviev naskladaných na seba, aby sa vytvoril konečný výstup skrytého stavu H. Na konci je aplikovaný adaptívny $softmax$ na výber slova z obrovského zoznamu slov.

7 Vyhodnotenie popísaného experimentu

Popísaný expertiment bol porovnávaný v [3] s RNN a LSTM (long-short-term-memory). Opisovaný experiment bol nastavený na zhodné parametre, konkrétne 1 GPU a adaptívny $softmax$ ako výstupnú funkciu. Na testovacom datasete *Google billion words* GCNN prekonala konkurenciu s dosiahnutou perplexitou 38.1 zatiaľ čo obdobný LSTM mal perplexitu 39.8. Treba podotknúť, že GCNN mala silnejší výkon a väčšiu výpočtovú efektivitu.

Model	Test PPL	Hardware
Sigmoid-RNN-2048 (Ji et al., 2015)	68.3	1 CPU
Interpolated KN 5-Gram (Chelba et al., 2013)	67.6	100 CPUs
Sparse Non-Negative Matrix LM (Shazeer et al., 2014)	52.9	-
RNN-1024 + MaxEnt 9 Gram Features (Chelba et al., 2013)	51.3	24 GPUs
LSTM-2048-512 (Jozefowicz et al., 2016)	43.7	32 GPUs
2-layer LSTM-8192-1024 (Jozefowicz et al., 2016)	30.6	32 GPUs
BIG GLSTM-G4 (Kuchaiev & Ginsburg, 2017)	23.3*	8 GPUs
LSTM-2048 (Grave et al., 2016a)	43.9	1 GPU
2-layer LSTM-2048 (Grave et al., 2016a)	39.8	1 GPU
GCNN-13	38.1	1 GPU
GCNN-14 Bottleneck	31.9	8 GPUs

Obr. 3: Výsledky pre trénovanie na *Google Billion words* datasete [3]

Nakoľko vyššie spomenutý dataset obsahuje najmä pomerne krátke vety obsahujúce priemerne 20 slov, Dauphin a kol. sa rozhodli otestovať svoju modifikovanú GCNN aj na datasete obsahujúcom dlhšie sekvencie. Zvolili si dataset *WikiText-103*, ktorý obsahuje namiesto jednej vety celý článok z Wikipédie, čím sa priemerná dĺžka zvýšila na 4000 slov.

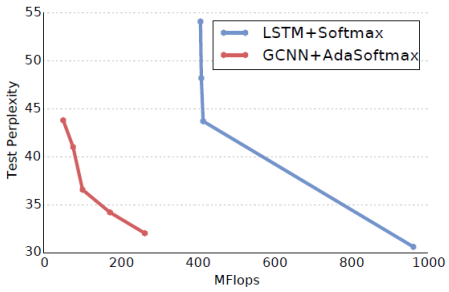
Model	Test PPL	Hardware
LSTM-1024 (Grave et al., 2016b)	48.7	1 GPU
GCNN-8	44.9	1 GPU
GCNN-14	37.2	4 GPUs

Obr. 4: Výsledky pre trénovanie na *WikiText-103* datasete [3]

Na obrázku 4 je možné vidieť, že aj v tomto prípade GCNN prekonalo LSTM, pričom model GCNN-8 má 8 vrstiev, každá po 800 jednotiek a LSTM má 1024 jednotiek. Tieto výsledky ukazujú, že GCNN siete môžu obsahovať aj dlhší kontext, pričom si zachovávajú kvalitné výsledky.

8 Skúsenosti a čas spracovania

Podľa dosiahnutých výsledkov opísaných v prechádzajúcej kapitole, Dauphin a kolektív zhodnotili, že trénovanie s použitím Nestorovho momentu významne zvýšilo rýchlosť konvergencie s minimálnym dodatočným výpočtom a uchovaním ďalších parametrov v porovnaní so štandardným stochastickým gradientom zostupu. Rýchlosť konvergencie sa takisto zvyšovala s gradientovým orezaním a normalizácia váh zabraňuje akumulovaniu gradientu, ktoré je charakteristické pre rekurentné neurónové siete [3].



Obr. 5: Porovnanie GCNN s adaptívnym softmaxom a LSTM so štandardným softmaxom [3]

Ako je možné vidieť na Obr. 5, GCNN s adaptívnym softmaxom vyžaduje len zlomok operácií, na to aby dosiahla rovnaké hodnoty perplexity ako LSTM so štandardným softmaxom. Pre porovnanie, GCNN dosiahla perplexitu 31.9 pri dvoj-týždňovom trénovaní na 8 GPU narozdiel od LSTM, ktorej výsledná perplexita bola 30.6 a vyžadovala 3 týždne trénovania na 32 GPU.

8.1 Výpočtová efektívnosť

Pre jazykové modely je veľmi dôležitá vypočtová efektívnosť. Počas popísaneho experimentu [3] využívali viacero metrík pre vyhodnotenie efektívnosti daného prístupu. Začali meraním *priepustnosti* modelu ako počet tokenov, ktoré je možné spracovať za sekundu. Priepustnosť môže byť maximalizovaná spracovaním paralelne s využitím sekvenčných operácií. V kontraste, *citlivosť* je rýchlosť spracovania vstupov postupne, jeden token v čase. Priepustnosť je dôležitá, nakoľko označuje požadovaný čas na spracovanie korpusu textu a citlivosť je indikátorom času na dokončenie spracovania vety. Prostredníctvom vyhodnocovania hromady viet súčasne (dávkovania) má model nízku citlivosť ale naopak vysokú priepustnosť.V popísanom príklade je takýto model pomalý pri spracovaní jednotlivých viet, ale môže spracovať veľa viet za rozumný čas.

	Throughput		Responsiveness
	(CPU)	(GPU)	(GPU)
LSTM-2048	169	45,622	2,282
GCNN-9	121	29,116	29,116
GCNN-8 Bottleneck	**179**	**45,878**	**45,878**

Obr. 6: Porovnanie GCNN a LSTM sieti v rýchlosti spracovania tokenov/s [3]

Ako je možné vidieť na Obr. 6 GCNN s využitím mechanizmu úzkeho miesta popísaného v kapitole 6 je dôležité pre zachovanie vysokej výpočtovej efektívnosti.

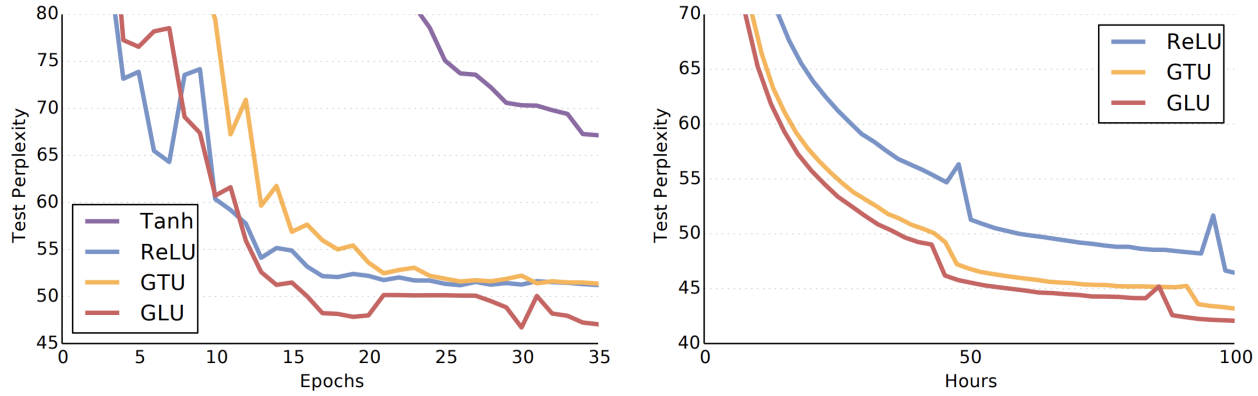
8.2 Gating mechanizmus

Dôležitý míľnik v [3] bol dosiahnutý práve použitím gating mechanizmu, preto v tejto časti opíšeme porovnanie gated lineárnych jednotiek (GLU) s modelmi bez tohto prístupu. Pre po-

rovnanie sa okrem GLU využívali klasické siete využívajúce *ReLU*, *Tanh* aktivácie spolu s LSTM gating mechanizmom popísanym takto:

$$tanh(X * W + b) \otimes \sigma(X * V + c) \tag{8}$$

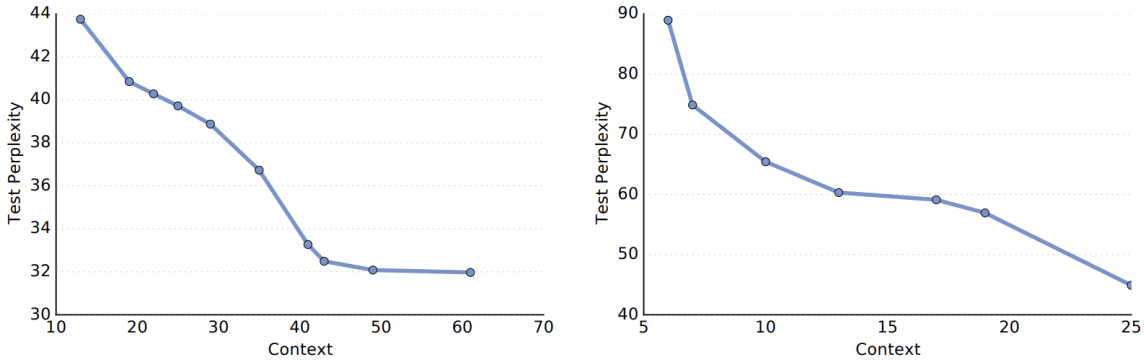
Obrázok 7 ukazuje, že GLU konverguje na *WikiText-103* datasete viac oproti ostatným metódam. Podobne ako GLU, ReLU má lineárnu dráhu, ktorá umožňuje jednoduchý prechod gradientov aktívnych jednotiek. To znamená oveľa rýchlejšiu konvergenciu pre ReLU aj GLU. Na druhej strane pre Tanh a GTU platí, že nemajú lineárnu dráhu a teda trpia problémom miznúceho gradientu.
Na obrázku 7 bol opakovaný rovnaký experiment na väčšom datasete *Google Billion Words*. Pre všetky metódy bol stanovený pevný čas výpočtov po dobu 100 hodín z dôvodu značného času potrebného na trénovanie tohto problému. Podobne ako pri *WikiText-103* datasete, GLU dosahujú najlepšie výsledky v tomto probléme.



Obr. 7: Učiace krivky na *WikiText-103* datasete (vľavo) a *Google Billion words* datasate (vpravo) [3]

8.3 Veľkosť kontextu

Obrázok 8 znázorňuje vplyv veľkosti kontextu pre gated CNN. V [3] skúšali rôzne kombinácie hĺbky siete a šírky kernelu pre každú veľkosť kontextu a vybrali to najlepšie pre každú veľkosť. Všeobecne platí, že väčšie kontexty zlepšujú presnosť, ale drasticky sa znižujú s oknami väčšími ako 40 slov, dokonca aj pre *WikiText-103*, kde môžeme podmieniť celý článok Wikipédie. Táto skutočnosť znamená, že neobmedzený kontext ponúkaný rekurentnými modelmi nie je nevyhnutne potrebný pre jazykové modelovanie.



Obr. 8: Test zmäzku v závislosti od dĺžky kontextu na *Google Billion words* datasate (vľavo) a *WikiText-103* datasete (vpravo) [3]

Obrázok 8 tiež ukazuje že *WikiText-103* má oveľa väčší úžitok z väčšieho kontextu ako *Google Billion Words*, kde výkon degraduje s menším kontextom. *WikiText-103* poskytuje oveľa viac kontextu ako *Google Billion Words*, kde priemerná veľkosť vety je 20 slov. Napriek tomu že pri *WikiText-103* dasatesete je priemerná dĺžka textu takmer 4000 slov, bolo zistené, že silný výkon možno dosiahnuť aj s veľkosťou kontextu 30 slov.

9 Zhodnotenie a záver

Predstavili sme konvolučnú neurónovú sieť pre modelovanie prirodzeného jazyk s novým gated mechanizmom. V porovnaní s rekurentnými neurónovými sieťami, tento prístup vytvára hierarchickú reprezentáciu vstupných slov, čo uľahčuje zachytiť závislosti na dlhých vzdialenostiach. Rovnaká vlastnosť uľahčuje učenie, pretože funkcie sú odovzdané cez pevný počet vrstiev a nelinearít, na rozdiel od rekurentných sietí, kde je počet krokov líši v závislosti od pozície slova na vstupe. Výsledky ukazujú, že popisovaná gated konvolučná sieť dosahuje na *WikiText-103* absolútnu dominanciu. Na *Google Billion Word* benchmark datasete sa ukázalo, že konkurenčné výsledky možno dosiahnuť s podstatne nižším objemom zdrojov.

Literatúra

[1] ARBI M.Z 2015 - Natural Language Processing

[2] BRESSLER D. 2018 - Building a convolutional neural network for natural language processing. Dostupné na internete: <https://towardsdatascience.com/how-to-build-a-gated-convolutional-neural-network-gcnn-for-natural-language-processing-nlp-5ba3ee730bfb>[5.5.2019]

[3] DAUPHIN Y.N. - FAN A. - AULI M. - GRANGIER D. 2017 - Language Modeling with Gated Convolutional Networks. International Conference on Machine Learning

[4] HINTON G. - OSINDERO S. - TEH Y. 2006 - A Fast Learning Algorithm for Deep Belief Nets. Neural computation. 18. 1527-54. 10.1162/neco.2006.18.7.1527.

[5] KURENKOV A. 2019 - A 'Brief' History of Neural Nets and Deep Learning, Part 4. Dostupné na internete: <https://medium.com/@andreykurenkov/a-brief-history-of-neural-nets-and-deep-learning-part-4-61be90639182>[5.5.2019]

[6] SUTSKEVER I. - MARTENS J. - DAHL G. - HINTON G. - 2013 - On the importance of initialization and momentum in deep learning

[7] TIANCHUAN DU - SHANKER V. K. - Deep Learning for Natural Language Processing

[8] ZADEH L.A. 1999 - From computing with numbers to computing with words. From manipulation of measurements to manipulation of perceptions. — Int. J. Appl. Math. Comput.Sci., Vol. 12, No. 3, pp. 307–324