Proiect DMDW

Misspelling Oblivious Word Embeddings (MOE)

SPELL CHECKER deployed as a REST API using Flask using

MOE (fasttext)

Student: Digori Gheorghe
Master: MTI

Cuprins

1. Importanta si domenii de aplicari in practica pentru algoritmul ales.

Încorporarea de cuvinte constituie un element de bază pentru multe aplicații practice din PNL și discipline conexe. Tehnici precum Word2Vec și GloVe au fost utilizate pe scară largă în practică. Unul dintre dezavantajele lor este însă faptul că nu pot furniza încorporari pentru cuvinte care nu au fost observate la timpul de formare, adică cuvinte în *afara vocabularului* (OOV). În sarcinile din lumea reală, textul de intrare este adesea generat de oameni și scrieri greșite, o sursă comună de cuvinte OOV, sunt frecvente apar până la 15% din interogările de căutare web. În consecință, calitatea aplicațiilor din aval de încorporarea de cuvinte în scenarii reale se diminuează.

Permiterea simplă a includerii unor greșeli de ortografie în corpuri și vocabulare în metodologiile existente nu poate oferi rezultate satisfăcătoare. Diminuarea literelor greșite ar împiedica, cel mai probabil, încorporarea lor să demonstreze proprietăți interesante. Încercarea de a echilibra reprezentarea greșelilor de ortografie cu reprezentarea variantelor ortografiate corect în datele de instruire, prin introducerea artificială a variantelor cu scrieri greșite pentru fiecare cuvânt în corpus ar provoca, pe de altă parte, o creștere exponențială a dimensiunii datelor de instruire, făcând instruire la modelele nefezibile.

Pentru a remedia această deficiență, sa propus un nou model care combină algoritmul FastText cu o sarcină supravegheată care încorporează variante greșite apropiate de variantele corecte ale acestora.

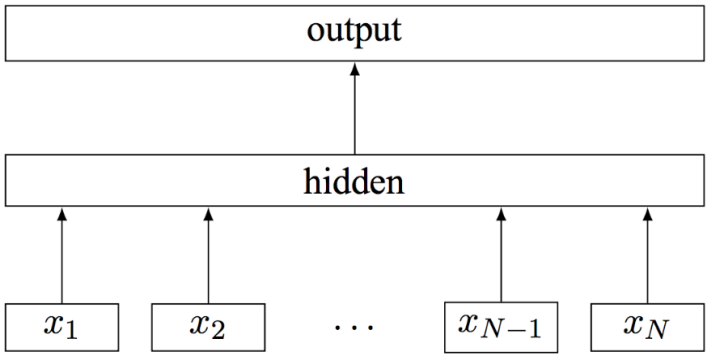
Încorporarea de cuvinte tradiționale este bună pentru a rezolva o mulțime de probleme în aval de procesare a limbajului natural (NLP), cum ar fi clasificarea documentației și recunoașterea entității numite (NER). Cu toate acestea, unul dintre dezavantaje este lipsa capacității de a manipula vocabularul în afara vocabularului (OOV).

Facebook introduce încorporarea obligatorie (cuvânt) încorporată (MOE), care depășește această limitare. MOE extinde arhitectura fastText pentru a o realiza. Prin urmare, această poveste trece prin metoda și arhitectura de formare fastText.

2. Prezentare generala algoritm.

2.1 Fasttext

FastText este o bibliotecă dezvoltată de Facebook pentru clasificarea textului, dar poate fi folosită și pentru a învăța încorporarea de cuvinte. De când a devenit open-source în 2016, a fost adoptat pe scară largă datorită vitezei sale de pregătire, precum și performanței ridicate.



(0) Architecture of fastText (Joulin et al., 2016)

Conform lucrării principale, modelul este o rețea neuronală simplă, cu un singur strat. Reprezentarea bag-of-words a textului este introdusă pentru prima dată într-un strat de căutare, unde încorporarea este preluată pentru fiecare cuvânt. Apoi, acele încorporari de cuvinte sunt mediate, astfel încât să se obțină o singură încorporare medie pentru întregul text.

În stratul ascuns terminăm cu un n_words x dimnumăr de parametri, unde dimeste dimensiunea încorporărilor și n_wordsdimensiunea vocabularului. După medie, avem doar un singur vector care este apoi alimentat unui clasificator liniar: aplicăm softmax peste o transformare liniară a ieșirii stratului de intrare.

Transformarea liniară este o matrice cu dim x n_output, unde n_outputsunt clasele de ieșire cu număr. În lucrarea originală, probabilitatea finală de jurnal este:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n))$$

Unde:

- x_n este reprezentarea originală cu un cod hot la un cuvânt (sau caracteristică n-gram),
- **A** este matricea look_up care recuperează încorporarea cuvintelor,
- **B** este transformarea liniară a ieșirii,
- *f* este funcția softmax

2.2 MOE

MOE deține proprietățile fundamentale ale FastText și Word2Vec, oferind în același timp o importanță explicită cuvintelor greșite.

Funcția pierdere a **MOE** este o sumă ponderată a două funcții de pierdere: LFT și LSC . L F T este funcția de pierdere a FastText care surprinde relații semantice între cuvinte. L S C sau pierderea corecției ortografice urmărește să coreleze încorporarea cuvintelor greșite cu scriere greșită apropiate de încorporarea variantelor lor corect corelate în spațiul vectorial. Definim L S C astfel:

$$L_{SC} := \sum_{(w_m, w_e) \in M} [\ell(\hat{s}(w_m, w_e)) + \sum_{w_n \in N_{m,e}} \ell(-\hat{s}(w_m, w_n))]$$

(1)

unde m este un set de perechi de cuvinte (w m , w e) astfel încât w e ∈ V este cuvântul așteptat (corect scris) și w m este scrierea greșită a acestuia. N m , e este un set de eșantioane negative aleatorii din V \ { w m , w e } . L S C folosește funcția logistică ℓ (x) = l o g (1+ e - x) introduse în secțiunea. Funcția de notare ^ s este definită după cum urmează:

$$\hat{s}\ (w_m, w_e) = \sum_{\mathbf{v}_g,\, g \in \hat{\mathcal{G}}_{w_m}} \mathbf{v}_g{}^T \mathbf{v}_e$$

(2)

unde ^ G w m : = G w m \ { w m } . Prin urmare, funcția de notare este definit ca produsul scalar dintre suma vectorilor de intrare ale subwords de w m și vectorul de intrare w e . Formal, termenul ℓ (^ s (w m , w e)) impune predictibilitatea w e dat w m . Intuitiv, optimizând L S Cîmpinge reprezentarea unei greșeli de scriere w m mai aproape de reprezentarea cuvântului așteptat we . De asemenea, este demn de menționat faptul că embeddings pentru wm și we cota de aceiași parametri setați. Funcția completă de pierdere a **MOE** , LMOE , este definită după cum urmează:

L M O E : = (1-α) L F T +α | T | | M |L S C (3)

Optimizarea simultană a funcțiilor de pierdere L F T și L S C nu este o sarcină simplă. Acest lucru se datorează faptului că cele două funcții de pierdere diferite itera peste două seturi de date diferite: corpusul de text T , iar ortografiile greșite date CCD M . Procesul de optimizare ar trebui să fie agnostic la dimensiunile T și M pentru a preveni ca rezultatele să fie grav afectate de aceste mărimi. Prin urmare, scalăm LS C cu coeficientul | T | / | M |.

În acest fel importanța unei actualizări unice stocastice Gradient Descent (SGD) pentru L F T devine echivalentă cu o actualizare unică SGD pentru LSC . Mai mult, α este hiperparametrul care stabilește importanța pierderii corecției ortografice L S C în raport cu LFT , făcând astfel **MOE** o generalizare a FastText.

3. Rezultate publicate, aspecte semnalate in literature.

În prezentul scenariu, una dintre cele mai utilizate forme de încorporare de cuvinte este Word2Vec, care este utilizat pentru a analiza răspunsurile la sondaj și pentru a obține informații din recenzii ale clienților, printre altele. Dar, odată cu apariția MOE, va exista o diferență uriașă în modul în care încorporările de cuvinte folosite pentru a lucra pentru sarcinile NLP. Noul model va îmbunătăți capacitatea și capacitatea de a aplica încorporarea de cuvinte în cazuri reale.

- MOE își propune să rezolve limitările de abordare a cuvintelor malformate în aplicațiile din lumea reală prin generarea de încorporari de înaltă calitate și semantic valabile pentru scrieri greșite
- Acest model depășește linia de bază fastText pentru sarcina de asemănare a cuvântului atunci când sunt implicate scrieri greșite
- MOE depășește atât întrebări semantice, cât și sintactice. Păstrează calitatea analogiilor semantice, îmbunătățind în același timp analogiile sintactice

Cercetătorii de la Facebook optimizează și îmbunătățesc sarcinile de prelucrare a limbajului natural (NLP) prin diferite abordări, cum ar fi utilizarea reprezentanților codificatorului bidirecțional al Google de la Transformers (BERT) și alte abordări conexe pentru a împinge cercetarea de ultimă oră în AI conversațională, îmbunătățirea sistemelor de înțelegere a conținutului și mult mai mult.

Din rezultatele testelor se observă că, în medie, sunt mai multe șanse să fie indetificata corecția folosind **MOE** decât cu FastText.

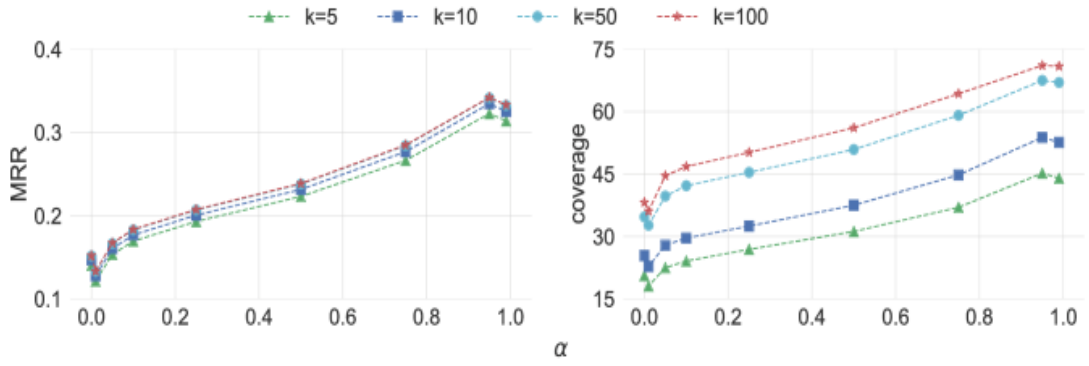


Figura 1: Rezultate experimentale pentru sarcina Validitate de vecinătate.

Valorile α = 0 reprezintă linia de bază, FastText. În partea stângă prezentăm scorurile MRR rezultate.

În lumea reală, în timp ce caută ceva pe web sau discută cu cineva, etc., oamenii adesea introduc text care conține scrieri greșite. Această nouă metodă va ajuta la îmbunătățirea capacității de a aplica încorporarea de cuvinte la aceste scenarii din lumea reală. Acum, MOE poate fi utilizat în diverse domenii precum întreprinderile centrate pe clienți pentru a obține informații acționabile din recenziile și feedback-urile clienților, chatbots, muzică și sisteme de recomandare video, printre altele

4. Prezentare solutie SPELL CHECKER

Testearea algoritmului MOE a fost efectuata folosind o aplicatie scrisa in Python folosind libraria FastText si Flask. Acest proiect utilizează o abordare bazată pe învățare profundă pentru a sugera până la trei recomandări corecte de ortografie pentru orice cuvânt dat.

Arhitectura de rețea (antrenata) folosește arhitectura **FastText** , dezvoltată de Facebook. Are avantajul unor vectori tradiționali de cuvinte, cum ar fi **Glove** , **word2vec** , etc, în sensul că poate produce vectori de cuvinte, chiar și pentru cuvinte din eșantion.

Fiecare cuvânt este reprezentat într-un spațiu de 100 de dimensiuni, iar modelul este instruit pe date cu window_size=5 și min_count=5 cu ajutorul bibliotecii “genesim”. Modelul instruit conține reprezentarea 100-D a tuturor caracterelor unice posibile ale setului de date de instruire.

5. Seturi de date folosite in testare

5.1 Seturi de date MOE

Încorporările MOE sunt instruite pe un nou set de date despre scriere greșită care este o colecție de cuvinte ortografiate corect, împreună cu scrierea greșită a cuvintelor respective. Mărimea totală a setului de date despre scrieri greșite conține mai mult de 20 de milioane de perechi de instanțe și este utilizată pentru a măsura pierderea corecției ortografice.

Setul de date este diponibil la adresa:
https://github.com/facebookresearch/moe

Acest model este diferit de alte metode cunoscute de încorporare a cuvintelor, cum ar fi word2vec și GloVe. Metodele actuale sunt lipsite de furnizarea încorporațiilor pentru cuvintele care nu au fost observate în timpul antrenamentului - sau din cuvintele din vocabular (OOV). Acest lucru duce la un rezultat nesatisfăcător, deoarece permite tratarea textului care conține argou, scrieri greșite etc.

5.2 Setul de date Spelling checker

Aplicatia Spelling checker prezentata in aceasta lucrarea a fost antreata pe un set de date Web Inventory of Transcribed and Translated Talks aceasta fiind o versiune gata de utilizare în scopuri de cercetare a transcrierilor multilingve ale discuțiilor TED. Acest set fiind disponibil la adresa:

https://wit3.fbk.eu/get.php?path=XML_releases/xml/ted_en-20160408.zip&filename=ted_en-20160408.zip

6. Evaluarea rezultatelor

6.1 Evaluare rezultate MOE

Rezultatele experimentale pentru versiunea canonică a sarcinii de analogie a cuvântului, prezentate în *figura 1*, arată că **MOE** realizează mai rău decât FastText în sarcina de analogie semantică. Pe de altă parte, **MOE** are performanțe mai bune decât linia de bază în sarcina analogiilor sintactice. Rezultatele pentru varianta scrisă greșit a sarcinii arată că, performanța generală atât a liniei de bază, cât și a **MOE** este mai slabă decât a variantei canonice. Pentru valori mici de $\alpha \in \{ 0{,}01 , 0{,}05 \}$, **MOE** depășește valoarea de bază în sarcina semantică, obținând un scor cu peste 67% mai bun decât FastText pentru $\alpha = 0{,}01$.

MOE depășește valoarea de bază în sarcina sintactică pentru toate valorile testate de α , îmbunătățindu-se cu peste 80% pentru α = 0,75 . Pentru α = 0,01 , care a atins cel mai bun rezultat semantic, îmbunătățirea pe sarcina sintactic este de peste 33%.

Tendințele pe care le observăm atât în varianta canonică, cât și scrisă greșit a sarcinii analogii cuvântului par să valideze alegerea funcției noastre de pierdere pentru **MOE**. Este clar că componenta FastText a pierderii este indispensabilă pentru a învăța relațiile semantice dintre cuvinte. De fapt, este singura componentă a funcției de pierdere care încearcă să învețe aceste relații. Prin urmare, reducerea importanței sale (prin creșterea valorii α) se reflectă printr-o decădere a scorului analogiilor semantice.

Componenta de corecție ortografică a funcției pierderi, pe de altă parte, încurajează relația dintre cuvintele ortografiate corect și greșelile lor. Ca efect secundar, acesta adaugă și informații suplimentare sub-cuvântului în model. Acest lucru explică performanțele noastre bune în sarcina analogiilor sintactice. După cum arată rezultatele noastre privind varianta scrisă cu scris greșit a sarcinii, am îmbunătățit peste linia de referință în înțelegerea analogiilor cu cuvintele greșite, care a fost unul dintre principiile de proiectare pentru **MOE** .

Test Data	**100% Misspelled**			**Original**		
Training Data	**Original**	**100% Miss.**	**10% Miss.**	**Original**	**100% Miss.**	**10% Miss.**
FastText, $\alpha = 0.0$	30.47	79.71	65.70	94.33	57.16	94.14
MOE, $\alpha = 0.01$	29.04	80.66	67.94	**94.55**	59.11	94.21
MOE, $\alpha = 0.05$	28.52	**81.17**	**68.92**	94.25	58.95	93.92
MOE, $\alpha = 0.1$	30.94	80.97	67.30.	94.45	58.88	**94.29**
MOE, $\alpha = 0.25$	29.00	80.13	67.63	94.37	58.67	94.01
MOE, $\alpha = 0.5$	29.19	80.43	66.76	94.27	57.29	93.94
MOE, $\alpha = 0.75$	30.94	78.65	64.53	94.18	57.67	93.81
MOE, $\alpha = 0.95$	32.40	75.28	62.29	93.09	60.21	92.52
MOE, $\alpha = 0.99$	**32.57**	73.36	61.36	90.91	**60.62**	90.53

Table 2: Performance on POS tagging task for UPOS tags using CRF. The models were trained on 100 epochs with an early stop (small difference on validation error) mechanism enabled. Considering F1 score, we evaluate on 2 variants of test data: Original (correctly spelled) on the right hand side of the table and 100% misspelled on the left hand side.

6.2 Evaluare rezultate aplicatie demo Spelling checker

Aplicatia Spelling checker realizata in limbajul Python (versiunea 3.6) a obtinut rezultate bune in urma testarii, astfel rata de predictie a cuvantului correct este aproximativ 90%.

Aplicatia prezinta 2 componente:

- componenta de antrenare model pe baza algoritmului Fasttext, fisierul “model.py”
- componenta de testare si interactiune model antrenat “app.py”

Interactiune aplicatie:

1) Antreneaza modelul: python3 model.py

2) Rulare aplicatie Flask: python3 app.py

   Output:

```
^Cgdigori@GDIGORI-RO:~/dmdw_proj/FastText-Spell-Checker$ python3 app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 303-806-661
127.0.0.1 - - [14/May/2020 22:20:08] "GET /spellCorrect HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2020 22:20:48] "GET /spellCorrect HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2020 22:21:04] "GET /spellCorrect HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2020 22:21:19] "GET /spellCorrect HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2020 22:21:37] "GET /spellCorrect HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2020 22:21:58] "GET /spellCorrect HTTP/1.1" 200 -
127.0.0.1 - - [14/May/2020 22:22:10] "GET /spellCorrect HTTP/1.1" 200 -
```

3) Pentru testare, se va deschide un alt terminal si se va trimite un GET la URL-ul API-ului portit

   Exemplu: curl -X GET http://127.0.0.1:5000/spellCorrect -d query='sellection'

   Output:

```
gdigori@GDIGORI-RO:~$ curl -X GET http://127.0.0.1:5000/spellCorrect -d query='sellection'
{
    "prediction": [
        "selection"
    ]
}
gdigori@GDIGORI-RO:~$
```

7. Referinte

- Bengio et al. (2003) Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.
- Bojanowski et al. (2017) Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Collobert and Weston (2008) Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- Cucerzan and Brill (2004) Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP 2004*, pages 293–300.
- Deerwester et al. (1990) Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.
- Devlin et al. (2018) Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Finkelstein et al. (2001) Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.
- Hinton (1986) Geoffrey E Hinton. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.
- Kim et al. (2016) Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.
- Levenshtein (1966) Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710.
- Levy and Goldberg (2014) Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.
- Luong et al. (2013) Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.