Inteligență artificială
Laborator 8

26 noiembrie 2019

Reprezentări vectoriale (*word embeddings*): vectori pentru reprezentarea cuvintelor.

One-hot encoding: fiecare cuvânt din corpus este reprezentat printr-un vector de dimensiune V în care la o anumită poziție avem valoarea 1; V numărul de cuvinte unice din corpus.

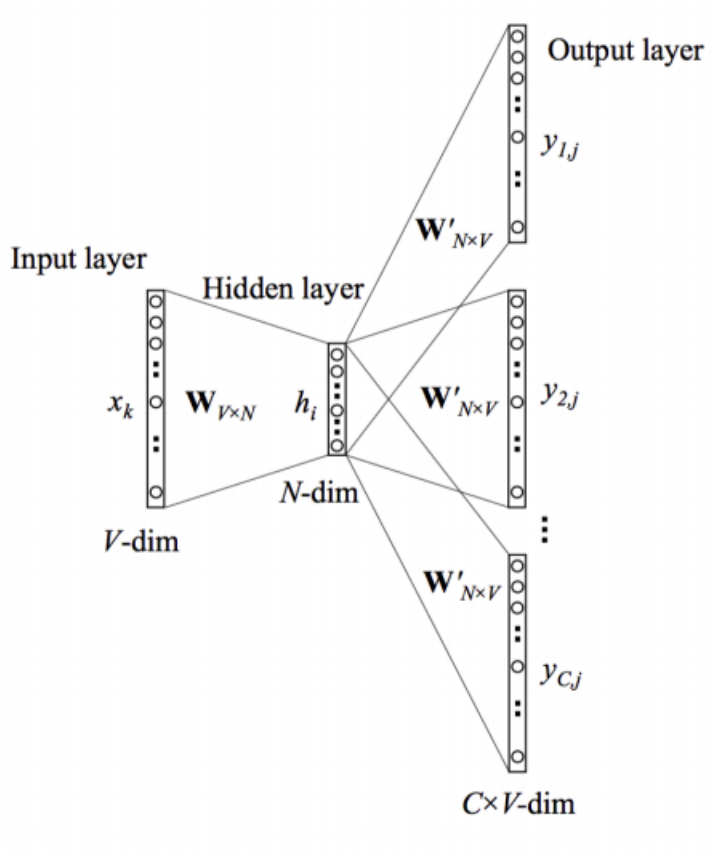
Un cuvânt e reprezentat printr-un vector (un punct într-un spațiu semantic multi-dimensional). Cuvintele similare (semantic) se găsesc în apropiere.

Cuvintele care apar în aceleași contexte au înțelesuri similare.

Word2vec

În loc să numărăm de câte ori apare un cuvânt w în apropierea altui termen, antrenăm un clasificator binar pentru a prezice acest lucru.
Vom considera ponderile învățate ca și reprezentări vectoriale.

Modelul Skip-gram (SG): utilizează cuvântul pentru a prezice contextul.



- x vector codificat *one-hot* care corespunde unui cuvânt dat ca intrare rețelei neuronale; V numărul de cuvinte unice din corpus.
- $y_1, \dots y_C$ vectorii codificați *one-hot* care corespund cuvintelor din ieșire.
- Matrici de ponderi W, W'; N numărul de neuroni din stratul ascuns

Matricea de ponderi W conține codificările vectoriale ale cuvintelor din vocabular (ca și linii).

1. Forward propagation

$$h = x^T W$$

Pentru vectorul x cu $x_k = 1$ și 0 în rest, ieșirea stratului ascuns e linia k din W (nu avem funcție de activare).

Utilizând matricea de ponderi W', calculăm scorul u_j pentru fiecare cuvânt din vocabular:

$$u = W'^T h$$

Aplicăm funcția softmax pentru a calcula ieșirea y

$$y = softmax(u)$$

y_j probabilitatea ca w_j să fie cuvânt din context.

2. Backward propagation: ajustăm matricile de ponderi W și W' utilizând gradienții erorilor.

Funcția de eroare (*loss*) este probabilitatea de a avea cuvintele de ieșire (din context) $w_{O,1}, \ldots, w_{O,C}$, dat cuvântul de intrare w_I:

$$\begin{aligned} E &= -\log p(w_{O,1}, w_{O,2}, \ldots, w_{O,C}|w_I) \\ &= -\log \Pi_{c=1}^{C} \frac{\exp(u_{j_c^*})}{\sum_{j'=1}^{V} \exp(u'_j)} = -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} exp(u_{j'}) \end{aligned} \tag{1}$$

j_c^* este indicele celui de-al c-lea cuvânt de ieșire.

Calculăm gradienții erorilor $\frac{\partial E}{\partial W'}$ și $\frac{\partial E}{\partial W}$.

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w'_{ij}}$$

$$\frac{\partial E}{\partial u_j} = y_j - t_j = e_j$$

t este ieșirea adevărată.

$$\Rightarrow \frac{\partial E}{\partial w'_{ij}} = e_j \cdot h_i$$

Pentru W,

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{ij}} = e_j \cdot w'_{ij} \cdot x_i$$

Temă: construiți reprezentarea vectorială a unui text (de minim 500 de cuvinte).

- (0.1p) citirea textului (din fișier)
- (0.2p) preprocesarea textului: împărțirea în propoziții, tokenizare (păstrează doar cuvinte), convertește la litere mici, elimină cuvintele *stopwords* (o listă de *stopwords* poate fi descarcată de aici)
- (0.1p) generarea datelor de antrenare: vectori *one-hot* (pentru cuvântul țintă și cuvintele din context)
- (0.6p) învățarea reprezentării vectoriale, utilizând modelul Skip-gram
  (0.3p) alternativ, puteti utiliza modelul pre-antrenat din libraria gensim `https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec`

- (0.2p) pentru o listă de cuvinte, afișați cuvintele cele mai similare cu acestea
- (0.2) opțional: utilizează algoritmul t-SNE din libraria *sklearn.manifold* pentru reducerea dimensionalității și pentru vizualizarea vectorilor de cuvinte

Termen limită: săptămâna 12 (17-23 decembrie 2020).

Bibliografie:

- *word2vec Parameter Learning Explained* `https://arxiv.org/pdf/1411.2738.pdf`
- Tutorial `http://alexminnaar.com/2015/04/12/word2vec-tutorial-skipgram.html`