Нейронные сети в NLP

word2vec

Лектор: Алтухов Никита Александрович
Аналитик данных Сбербанк

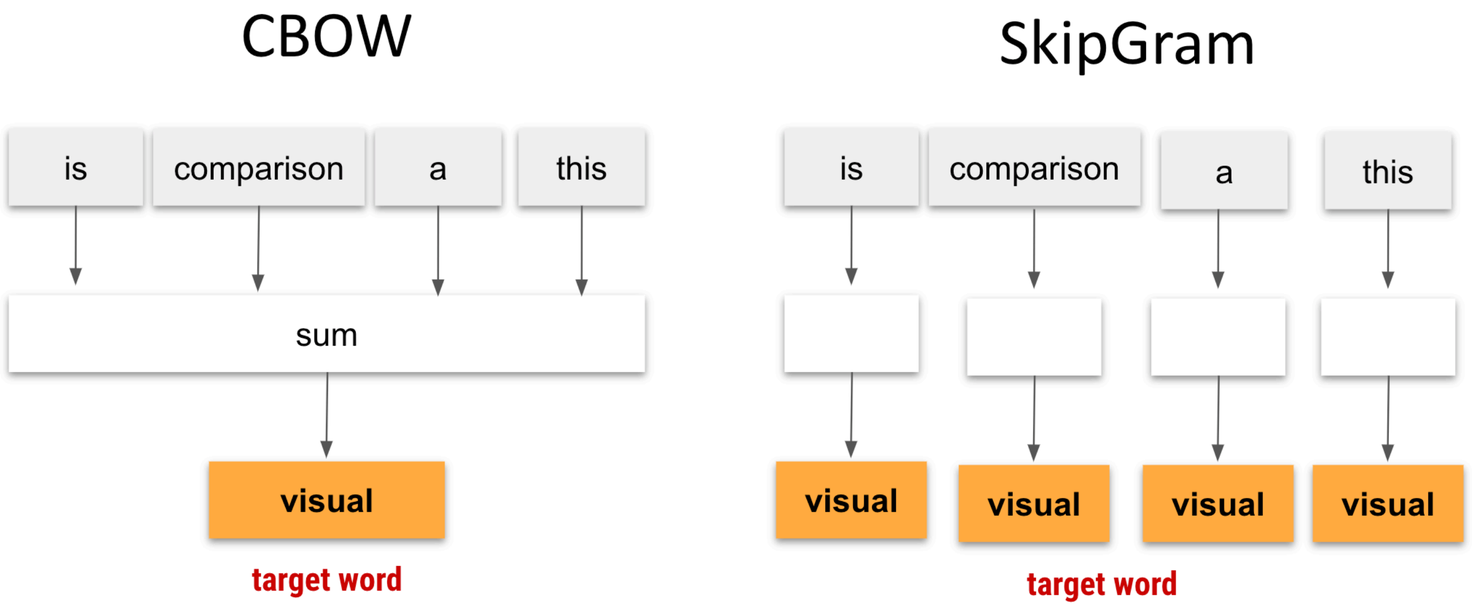
Embeddings

- сжатые векторные представления слов

- Лучше, чем One Hot Encoding:
  - Размерность вектора не зависит от размера словаря, берется порядка 100
  - Векторные представления для семантически близких слов находятся близко в векторном пространстве
- Классический подход: $X_{|T|\times|T|} = USV$, где T - размер словаря (±LSA)
  - Потеря информации
  - Нет процесса обучения

Word2Vec (2013)

- группа алгоритмов для получения векторный представлений слов

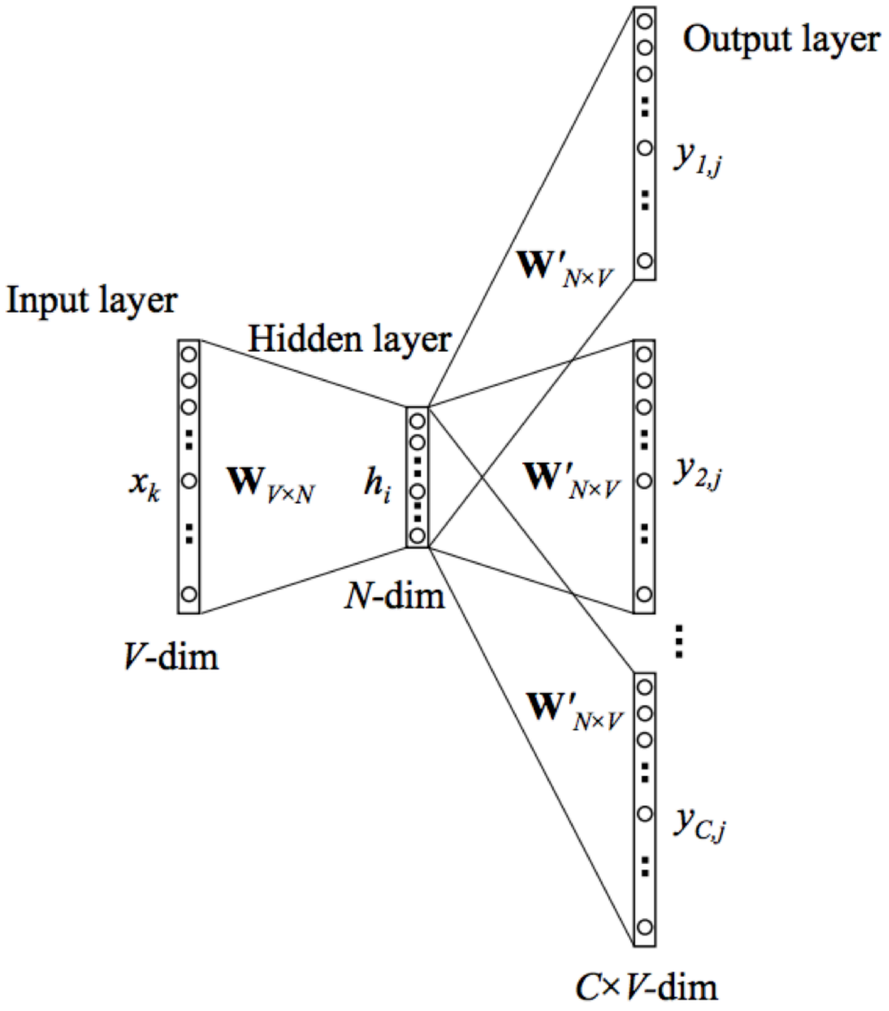


This is a visual comparison

Что больше похоже на T9?

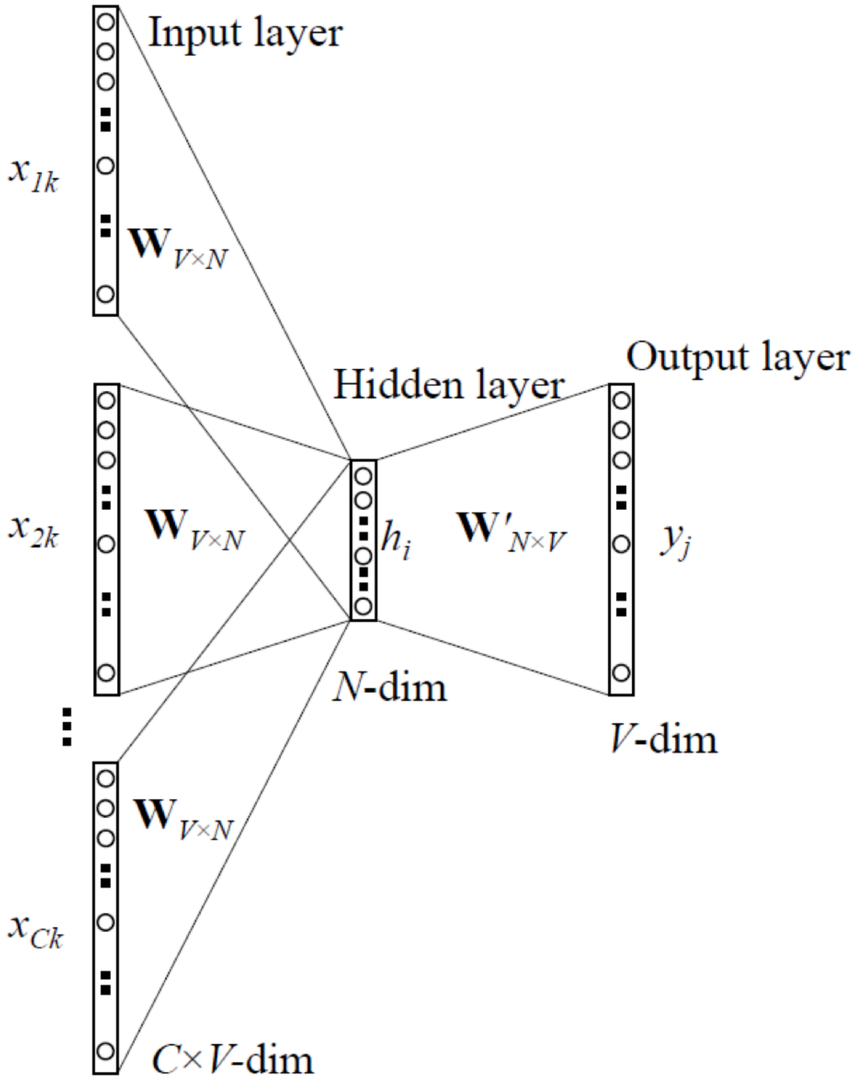
http://rusvectores.org/ru/models/

Skip-Gram



Source Text	Training Samples generated from source text			
I will have orange juice and eggs for breakfast	(will, I)	(will, have)	(will, orange)	
I will have orange juice and eggs for breakfast	(have, I)	(have, will)	(have, orange)	(have, juice)
I will have orange juice and eggs for breakfast	(orange, will)	(orange, have)	(orange, juice)	(orange, and)
I will have orange juice and eggs for breakfast	(juice, have)	(juice, orange)	(juice, and)	(juice, eggs)
I will have orange juice and eggs for breakfast	(and, orange)	(and, juice)	(and, eggs)	(and, for)
I will have orange juice and eggs for breakfast	(eggs, juice)	(eggs, and)	(eggs, for)	(eggs, breakfast)
I will have orange juice and eggs for breakfast	(for, and)	(for, eggs)	(for, breakfast)	

Лучше обучается на маленьком датасете, неплохой результат на редких словах



CBOW

Continuous Bag of Words

Непрерывно распределенный мешок слов.

Непрерывный, так как вектор из пространства $\mathbb{R}^n$

Учится быстрее в несколько раз, чем Skip-Gram, лучше качество на частотных словах

Обучение

SoftMax $$softmax(o^{(i)}) = \frac{exp(o^{(i)})}{\sum_{j=0}^{m} exp(o^{(j)})}$$

Log(SoftMax) $$LogSoftmax(x)_i = x_i - log(\sum_{j=1}^{K} e^{x_j})$$

CrossEntropyLoss $$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Negative Sampling

SoftMax на большом словаре слишком дорого!

- Нужно максимизировать близость целевого слова к контекстному и минимизировать близость других слов к целевому. Считать каждый раз близость для каждого слова в выборке?!
- Давайте выбирать несколько отрицательных примеров (порядка 5) рандомно* и будем считать сигмоиду на каждом, а не softmax на всех. Следовательно, максимизируем близость к контексту и минимизируем близость к пяти рандомным словам.

* - не совсем рандомно, редкие слова будем брать чуть чаще

Word2Vec

- Обучение без учителя
- Хорошие результаты на **большом** датасете
- Что делать со словами, которых нет в словаре?

FastText (2015)

- Промышленное решение от FaceBook
- Строит эмбеддинги не слов, а слогов, слово = среднее арифметическое эмбеддингов

Выводы

- Не учитываем семантику предложения
- Для каждого слова всего один вектор, хотя в разных контекстах слово может иметь разные значения
- Очень хорошо подходит для входных данных в более сложные (языковые) модели

Ссылки, источники

- Ссылка на GitHub с лекцией и материалами: https://github.com/nikitosl/spbu-nlp-2020
- Ссылка на Google Colab: https://colab.research.google.com/drive/1FurnE4VbNwDgQgOtRs7vNKgdNguAFCRQ#scrollTo=P2-5rOAXQYhR
- https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314
- https://dlcourse.ai
- https://towardsdatascience.com/deep-learning-for-nlp-with-pytorch-and-torchtext-4f92d69052f