ОБРАЗОВАТЕЛЬНЫЙ ИНТЕНСИВ

Остров 10-22

10-22 ИЮЛЯ 2019

Векторные представления слов. Классификация текстов

Баймурзина Диляра

13.07.2019

Векторные представления слов

Постановка задачи векторизации текста

- Обработка естественного языка (Natural language processing, NLP) предполагает представление текста в некотором машиночитаемом формате.
- Слово традиционно представляется в качестве основной единицы языка.
- Как можно представить слово в численном виде?

One-hot вектора слова

- Составим словарь возможных слов V. Для i-го слова w_i в словаре вектор слова будет иметь вид $v_i = \{0, \ldots, 0_{i-1}, 1_i, 0_{i+1}, \ldots, 0\}$.
- Размерность вектора равна размеру словаря (например, 200 000).
- Для каждого слова единственная компонента вектора является единицей, все остальные элементы вектора являются нулями

Проблемы:

- Вектора большой размерности замедляют обработку текстов.
- Расстояние между любыми двумя словами одинаковое, не зависит от семантики[1] слов.

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

[1]Сема́нтика - раздел лингвистики, изучающий смысловое значение единиц языка.

TF-IDF коэффициенты для векторного представления документов

- Дан набор документов D
- Term frequency — насколько слово w важно для документа d:

$$\mathrm{TF}(\mathrm{w}, \mathrm{d}) = |\{w \in d\}|$$

- Inverse document frequency — насколько слово w специфично для документа d:

$$\mathrm{IDF}(\mathrm{w}, \mathrm{D}) = -\log \mathrm{P}(w \mid D) = \log \frac{|D|}{|\{d \in D : w \in d\}|}$$

- TF-IDF для слова w в документе d в наборе текстов D:

$$\mathrm{TF{-}IDF}(\mathrm{w}, \mathrm{d}, \mathrm{D}) = \mathrm{TF} \cdot \mathrm{IDF}$$

TF-IDF коэффициенты для векторного представления документов

Таким образом получаем вектор документа d в наборе документов D:

$$vec(d) = \sum_{w \in V} \text{TF}-\text{IDF}(w, d, D) \cdot \text{onehot}(w)$$

$$= [\text{TF}-\text{IDF}(\text{w}_1, \text{d}, \text{D}), \ldots, \text{TF}-\text{IDF}(\text{w}_\text{n}, \text{d}, \text{D})]$$

Что нужно на самом деле?

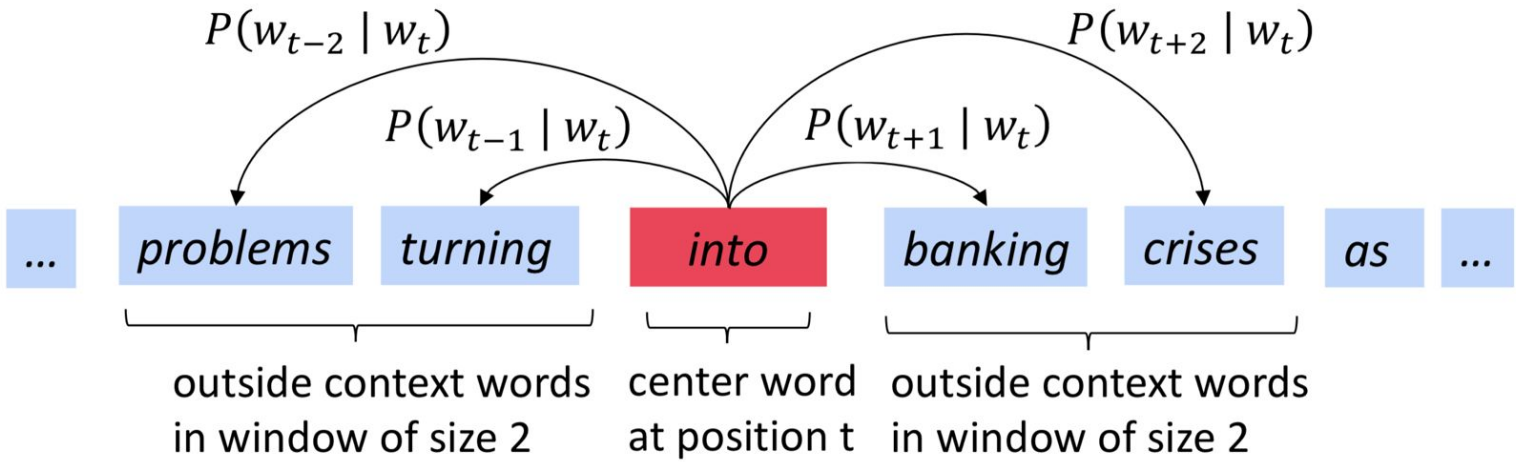
- Плотные вектора для каждого слова.
- Семантически похожие слова должны иметь похожие вектора.
- Для получения семантически близких векторных представлений можно использовать большие наборы текстов (например, статьи Википедии)

- Как оценить схожесть двух слов при наличии корпуса текстов?
- Похожие слова чаще встречаются в похожих контекстах.

- “You shall know a word by the company it keeps” (c) John Rupert Firth

Алгоритм word2vec

- Выбрать большой набор текстов.
- Определить и зафиксировать словарь слова.
- Случайно инициализировать вектора слов из словаря.
- Для каждой позиции t в тексте с центральным словом c и контекстом o посчитать вероятность встретить слово c в контексте o (или наоборот).
- Обновить вектора для максимизации данной вероятности.
- Повторять, пока вектора не сойдутся.



Как вычислить вероятность при заданных векторах слов?

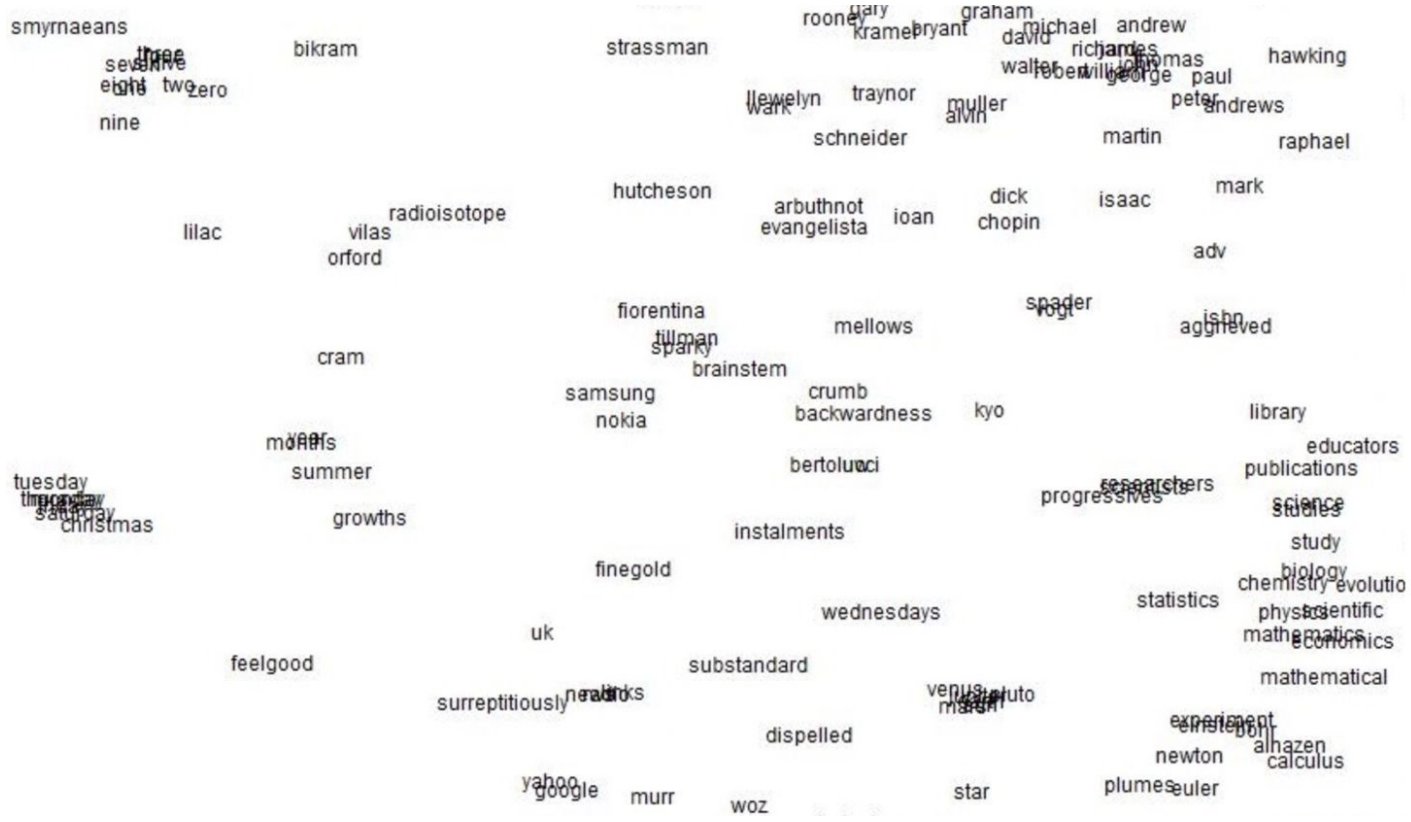
$$P(w_c \mid w_o) = \frac{e^{w_c . w_o}}{\sum_{w_i \in vocab} e^{w_i . w_o}}$$

$$w_o = \sum_{w_j \in context} w_j$$

$$L = - \sum_{w_i, w_o \in data} \log P(w_c \mid w_o)$$

- w_c - вектор центрального слова c
- w_o - вектор контекста o
- *vocab* - выбранный словарь слов
- *context* - central word window

Свойства word2vec

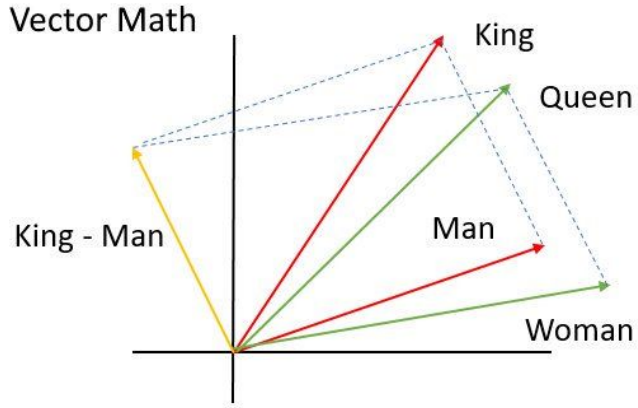


Свойства word2vec

- King - Man + Woman ≈ Queen

- London - England + Russia ≈ Moscow(general domain)
- London - England + Russia ≈ Kremlin (news)

- Surgeon - Man + Woman ≈ Nurse
- Cosmetics - Woman + Man ≈ Pharmaceuticals
- Feminism - Woman + Man ≈ Conservatism



GloVe

- word2vec использует только локальную информацию (контекст в несколько слов).
- word2vec не учитывает информацию об общей взаимной встречаемости слов.
- Алгоритм можно улучшить, добавив глобальную статистику по всем документам

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

- X_{ij} - количество появлений слова j в контексте i,
- b_i и b_j - коэффициенты смещения для симметризации,
- f - взвешивающая функция для балансирования влияния часто и редко встречающихся слов.

Постановка задачи векторизации текста

Ближайшие соседи слова “frog”:

- Frogs
- Toad
- Litoria
- Leptodactylidae
- Rana
- Lizard
- Eleutherodactylus

Image source: nlp.stanford.edu/projects/glove

Проблема с out-of-vocabulary словами

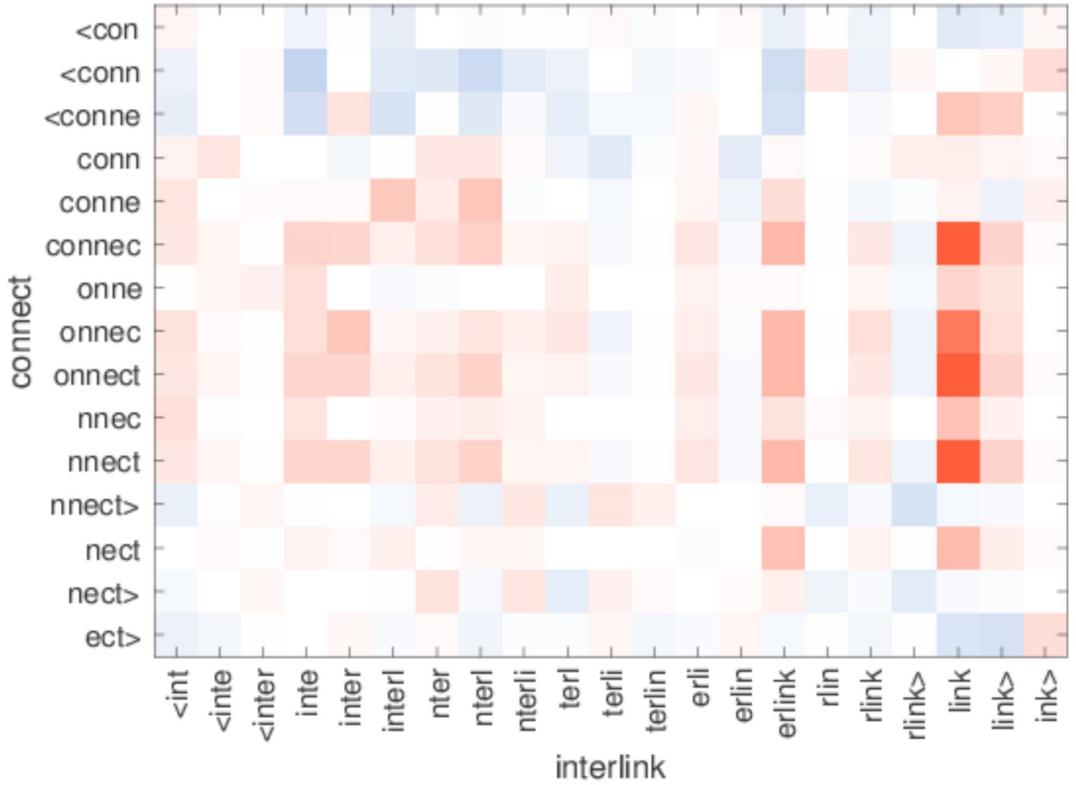
- Как выбрать словарь V?
- Как поступать с out-of-vocabulary словами в тексте?
- Какими именно векторами обозначать out-of-vocabulary слова?

Алгоритм fastText

- Вместо использования словаря слов, используем словарь n-grams.
- n-gram - сочетание *n* символов.
- Инициализировать векторные представления для n-grams.
- Получить векторное представление слова, сложив вектора входящих в него n-grams.
- Обучить векторные представления методом схожим с word2vec

- fastText n-gram (n = 3) представление слова “where”:
- <wh, whe, her, ere, re>, <where>

Матрица схожести n-grams (*n*=6) алгоритма fastText



Enriching Word Vectors with Subword Information, Bojanowski et al., 2017

Классификация текстов

Постановка задачи классификации текстов

Пусть X — множество текстов, Y — множество классов.

Существует неизвестная целевая зависимость - отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{ (x_1, y_1), \dots, (x_m, y_m) \}$.

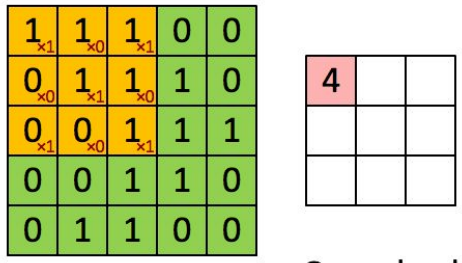
Требуется построить алгоритм $a: X \to Y$, способный классифицировать произвольный объект x из X.

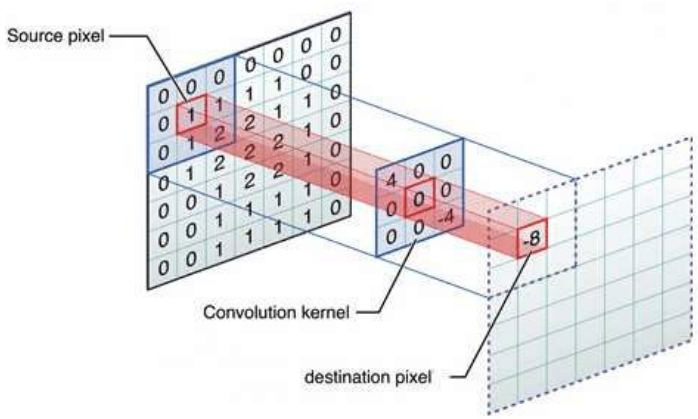
Основные методы решения задачи классификации текстов

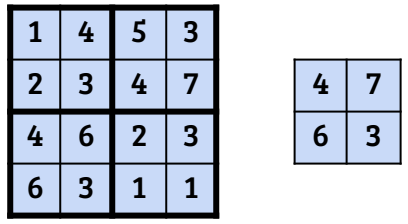
- Подход “на правилах” (rule-based or dictionary-based)
- Алгоритмы машинного обучения (Machine Learning)
    - в основном, на векторных представлениях текстов целиком
- Нейронные модели глубокого обучения (Deep Learning)
    - char-level - векторные представления символов
    - *token-level - векторные представления токенов (слов, пунктуации)*
    - sentence-level - векторные представления текстов целиком

Свёрточные нейронные сети

- Свёрточный слой (convolutional layer) включает в себя определенное количество фильтров, ядра (kernel) свёртки которых преобразуют предыдущий слой по фрагментам, суммируя результаты матричного произведения для каждого фрагмента.
- Слой субдискретизация (pooling layer) необходим для нелинейного уплотнения признаков, для избавления от излишне подробных признаков.







Max pooling with filter 2x2 and stride of size 2

Gif from http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Свёрточные нейронные сети для текстов

- 2D представление текста - потокенное векторное представление размера n x k.
- 1D convolution с f фильтрами возвращает n x f представление.
- Over-time pooling для ухода от количества токенов.

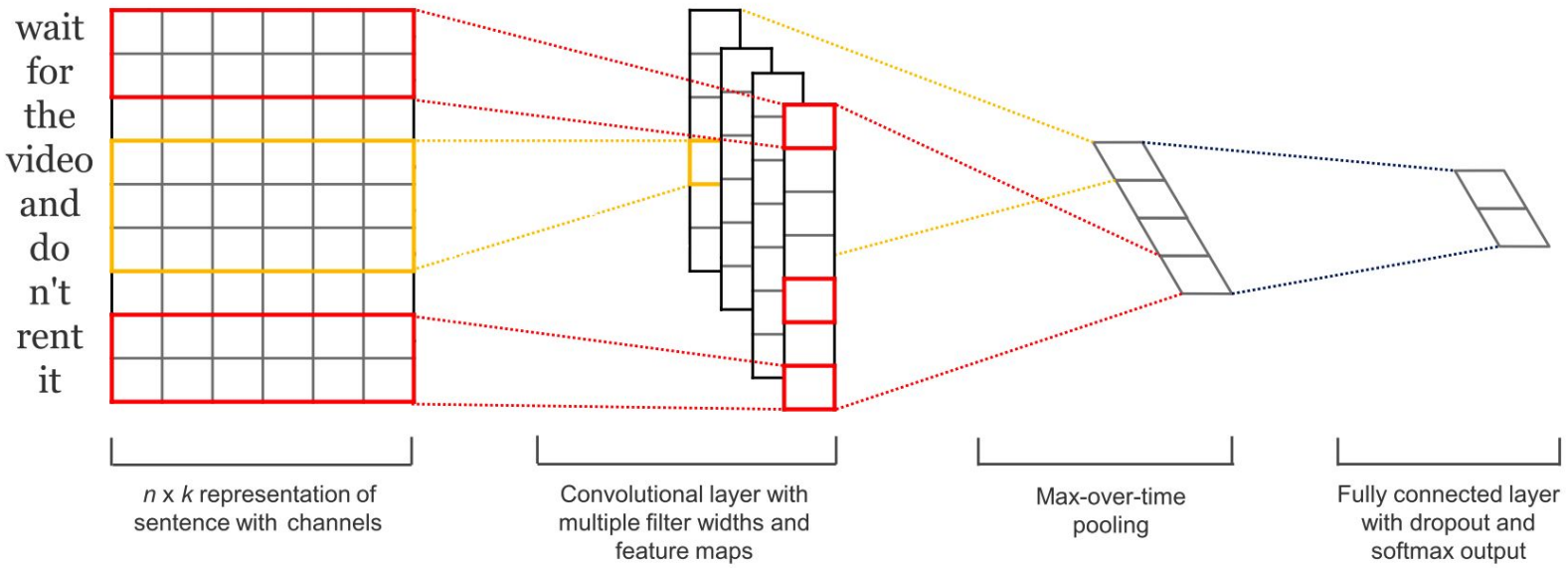
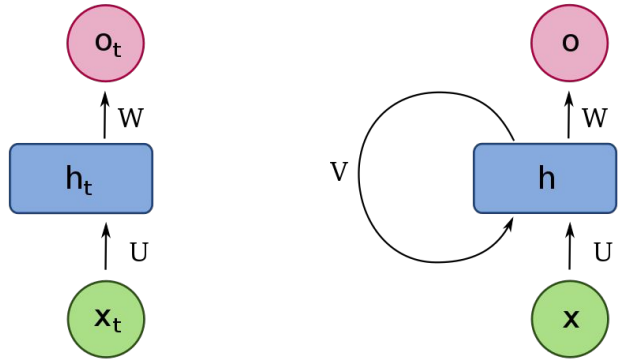


Image from https://www.aclweb.org/anthology/D14-1181

Рекуррентные нейронные сети

- Текст - это последовательность токенов (временных шагов).
- Для работы с последовательностями необходимо использовать память.
- Переиспользование скрытого состояния с предыдущего временного шага (токена) может решить эту проблему.



Заключение

- word2vec, GloVe, fastText для векторизации текста
- word2vec, GloVe с фиксированным словарем
- fastText работает на уровне n-gram

- Свёрточные нейронные сети для классификации коротких текстов
- Рекуррентные нейронные сети зачастую достигают более высокого качества

DeepPavlov
docs.deeppavlov.ai

Business solutions, support & partnerships
iPavlov.ai

Исследования и разработки выполнены при поддержке Фонда поддержки проектов Национальной технологической инициативы и ПАО "Сбербанк". Идентификатор проекта 0000000007417F630002.