Основы глубинного обучения

Лекция 9

Работа с последовательностями

Евгений Соколов

esokolov@hse.ru

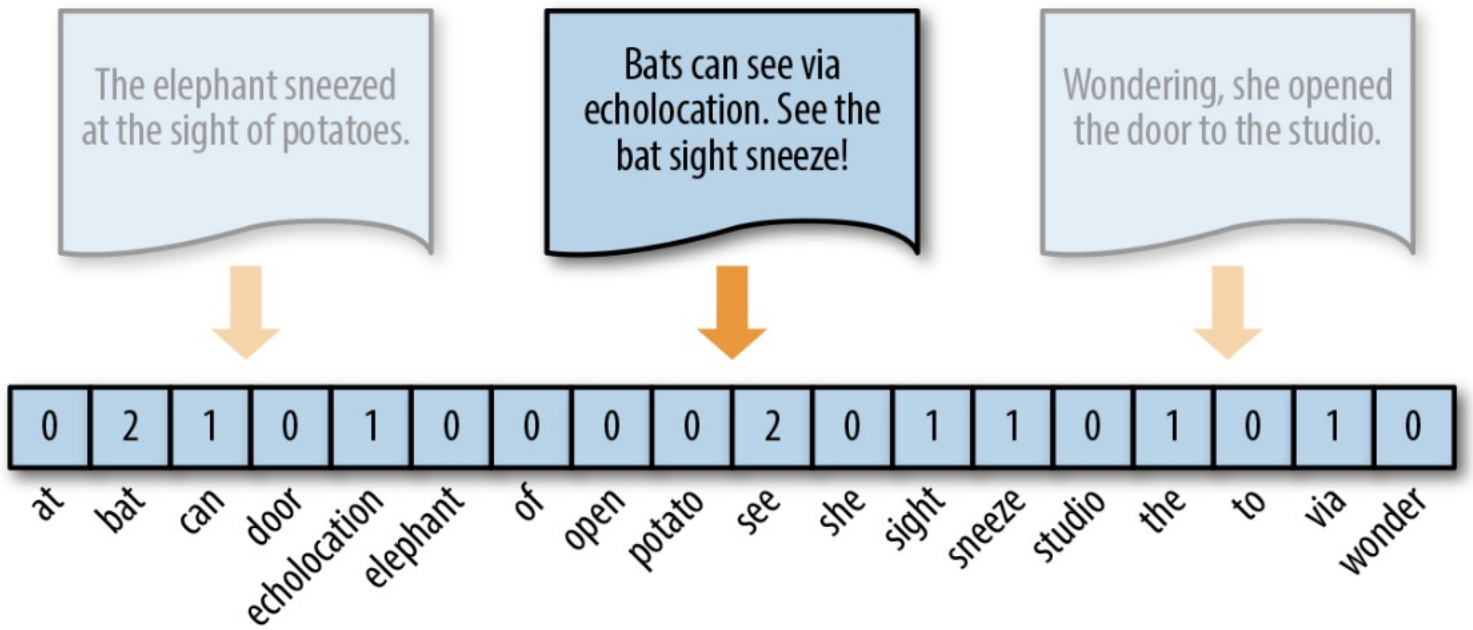
НИУ ВШЭ, 2022

Представления для текстов

Bag of words

- Заводим словарь, состоящий из всех слов в выборке
- Делаем признак-индикатор для каждого слова из словаря
- Можно добавлять n-граммы

Bag of words



https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models-a-new-age-in-nlp-part-1-7ed0c7f3dfc5

Bag of words

- Слишком много признаков
- Не учитываем смыслы слов
- Семантически похожие тексты могут иметь очень разные представления

word2vec

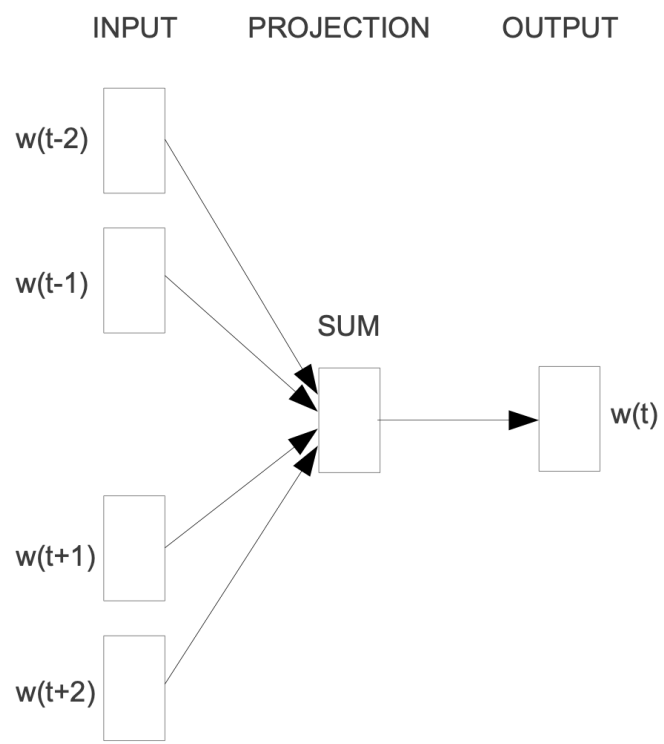
- Попробуем обучить вектор-представление для каждого слова
- Что потребовать от такого представления?

word2vec

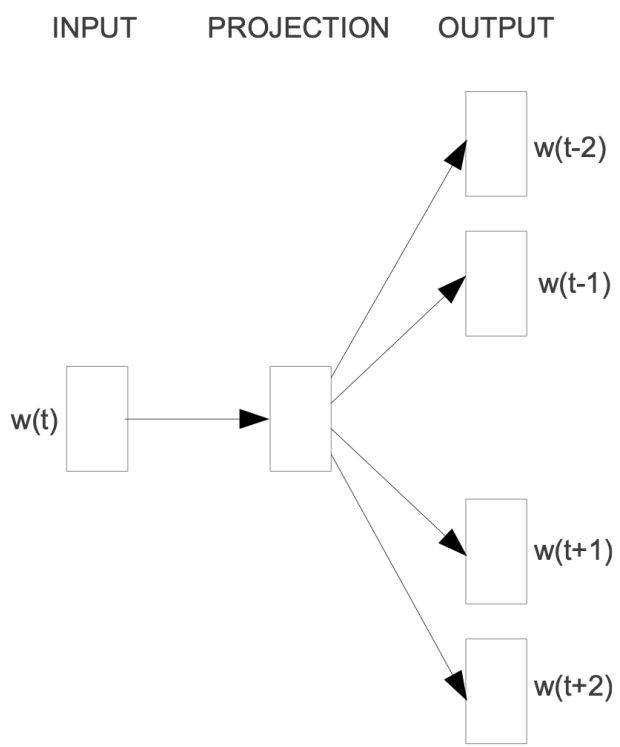
- Попробуем обучить вектор-представление для каждого слова
- Что потребовать от такого представления?

- Важная идея: если выкинуть слово, то оно должно хорошо восстанавливаться по представлениям соседних слов
- Может применять и при работе с изображениями

word2vec



CBOW



Skip-gram

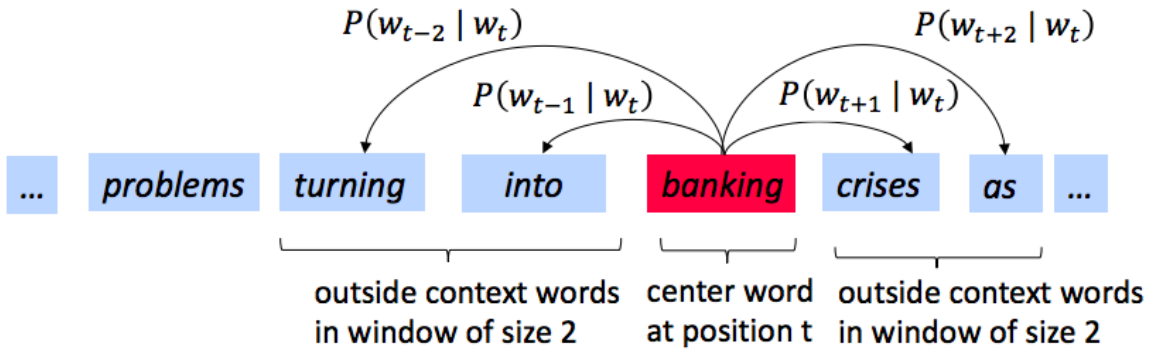
https://arxiv.org/abs/1301.3781

Skip-gram model

- Вероятность встретить слово w_O рядом со словом w_I:

$$p(w_O|w_I) = \frac{\exp\left(\left\langle v'_{w_O}, v_{w_I}\right\rangle\right)}{\sum_{w\in W}\exp\left(\left\langle v'_{w}, v_{w_I}\right\rangle\right)}$$

- W — словарь
- v_w — «центральное» представление слова
- v'_w — «контекстное» представление слова



https://trailhead.salesforce.com/

Skip-gram model

- Вероятность встретить слово w_O рядом со словом w_I:

$$p(w_O|w_I) = \frac{\exp\left(\left\langle v'_{w_O}, v_{w_I}\right\rangle\right)}{\sum_{w\in W}\exp\left(\left\langle v'_{w}, v_{w_I}\right\rangle\right)}$$

- Функционал для текста $T = (w_1 w_2 \dots w_n)$:

$$\sum_{i=1}^{n}\sum_{\substack{-c\le j\le c\\ j\neq 0}} \log p\left(w_{i+j}|w_i\right) \to \max$$

Skip-gram model

- Вероятность встретить слово w_O рядом со словом w_I:

$$p(w_O|w_I) = \frac{\exp\left(\left\langle v'_{w_O}, v_{w_I} \right\rangle\right)}{\sum_{w \in W} \exp\left(\left\langle v'_w, v_{w_I} \right\rangle\right)}$$

- Считать знаменатель ОЧЕНЬ затратно
- Значит, и производные считать тоже долго

Negative sampling

$$p(w_O|w_I) = \log \sigma\left(\left\langle v'_{w_O}, v_{w_I} \right\rangle\right) + \sum_{i=1}^{k} \log \sigma\left(-\left\langle v'_{w_i}, v_{w_I} \right\rangle\right)$$

- w_i — случайно выбранные слова
- Слово w генерируется с вероятностью $P(w)$ — шумовое распределение
- $P(w) = \frac{U(w)^{\frac{3}{4}}}{\sum_{v \in W} U(v)^{\frac{3}{4}}}$, $U(v)$ — частота слова v в корпусе текстов

word2vec: особенности обучения

- Положительные примеры — слова, стоящие рядом
- Отрицательные примеры: подбираем к слову «шум», то есть другое слово, которое не находится рядом
- Важно семплировать в SGD слова с учётом их популярности — иначе будем обучаться только на самые частые слова

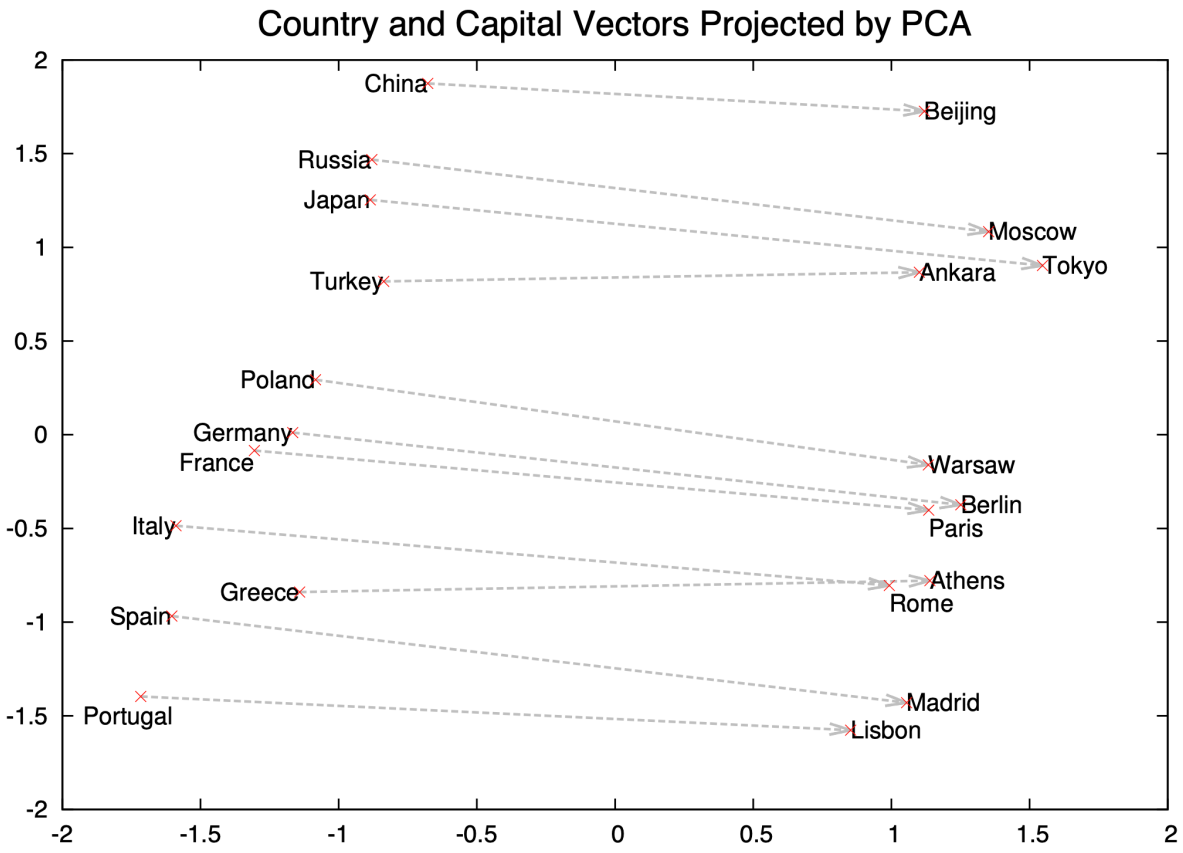
Как это использовать?

- Можно искать похожие слова
- Можно менять формы слов
- Можно искать определённые отношения
- Можно использовать как признаки для моделей

word2vec

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

word2vec


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Moscow
Japan
Tokyo
Turkey
Ankara
Poland
Warsaw
Germany
Berlin
France
Paris
Italy
Athens
Greece
Rome
Spain
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

word2vec

- Яркий пример self-supervision
- Сейчас находит применения для изображения и даже для табличных данных
- Оказывается, в данных очень много информации даже без разметки

Проблемы word2vec

- Не учитываем структуру слов
- Не закладываем никакой априорной информации о разных формах одного слова
- Не умеем обрабатывать опечатки

FastText

- Заменим каждое слово на «мешок»
- «руслан» -> (<руслан>, <ру, рус, усл, сла, лан, ан>)
- Слово w заменяется на набор токенов $t_1, \dots, t_n$
- Мы обучаем векторы токенов: $v_{t_1}, \dots, v_{t_n}$ (на самом деле есть «центральные» и «контекстные» версии всех векторов)
- $z_w = \sum_{i=1}^{n} v_{t_i}$ — вектор слова
- Все остальные детали — как в word2vec

https://arxiv.org/abs/1607.04606

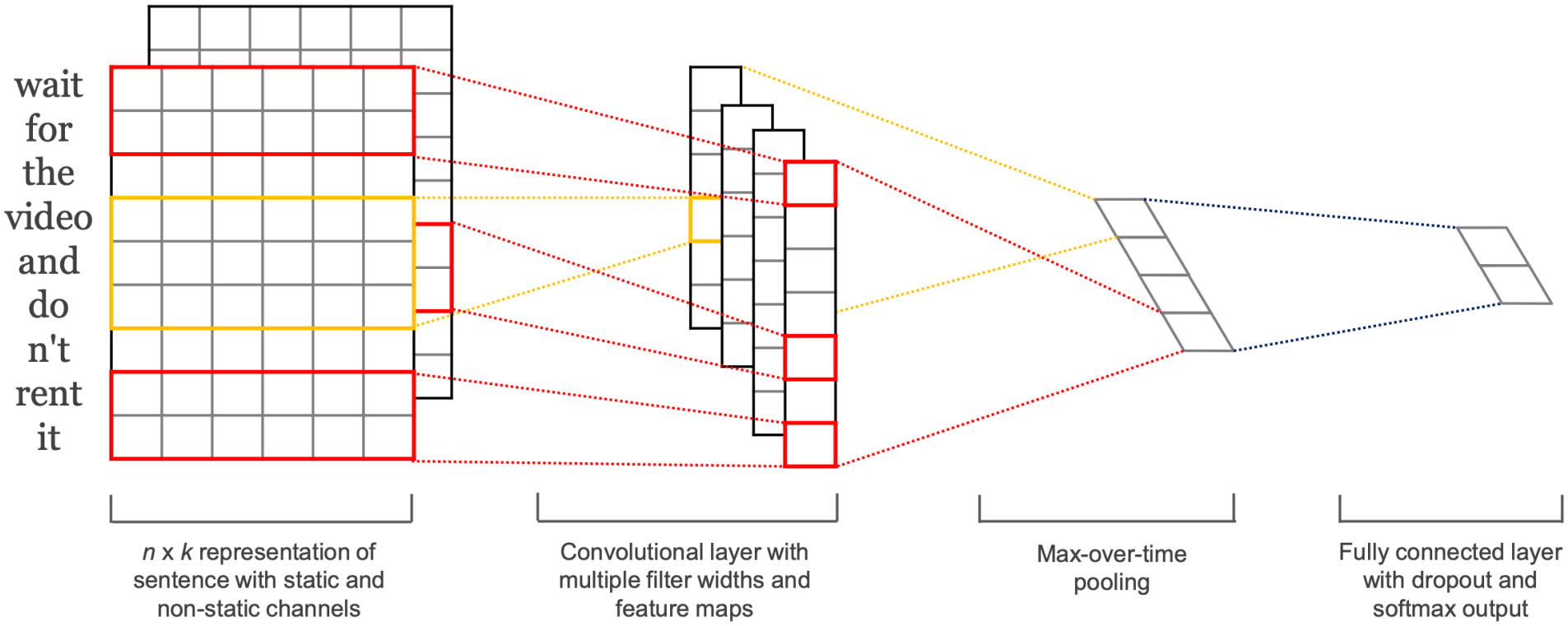
Что бывает ещё?

- GloVe
- ELMo/BERT (в следующих лекциях)

Работа с текстом

- Векторные представления строятся для слов
- Можно просто усреднить по всем словам — получим признаки для текста
- Можно усреднять с весами
- Можно ли умнее?

CNN для последовательностей



https://arxiv.org/abs/1408.5882

CNN для последовательностей

- Можно обучать представления слов с нуля
- А можно инициализировать с помощью w2v — это сильно лучше!

https://arxiv.org/abs/1408.5882

CNN для последовательностей

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

https://arxiv.org/abs/1408.5882

Минусы

- Ищем выразительные «локальные» комбинации
- Не пытаемся понять смысл текста в целом