OTUS

Векторные представления слов и работа с предобученными эмбедингами

Сизов Александр

Векторные представления

- **Векторное представление** — сопоставление произвольному объекту некоторого числового вектора в пространстве фиксированной размерности
- Наиболее известный вид – *векторные представления слов (word embedding)*
- Векторы могут обладать разнообразными полезными свойствами, отражать близость объектов в разных смыслах
- Для слов это может быть семантическая близость

Зачем нужны векторные представления

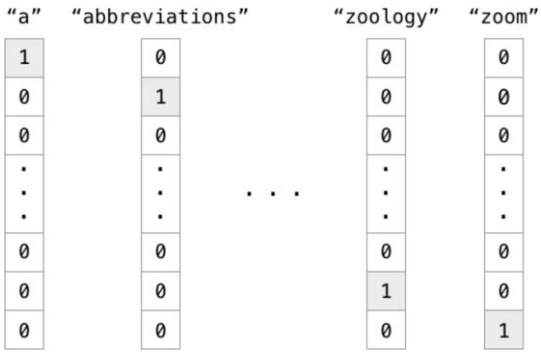
В современных подходах эмбеддинги используются в качестве признаков для решения почти любых задач машинного обучения

В текстовой аналитике это:

1. выделение именованных сущностей (NER)
2. выделение частей речи (POS-tagging)
3. машинный перевод
4. кластеризация документов
5. классификация документов, анализа тональности (sentiment)
6. ранжирование документов
7. генерация текста

One-hot encoding

Самый простой способ кодирования категориальных признаков:



SVD для получения эмбеддингов слов

По корпусу текстов D со словарём T строим **матрицу со-встречаемостей :**

$$X_{|T|\times|T|}$$

Элемент x_{ij} отражает со-встречаемость слов i и j в корпусе текстов.

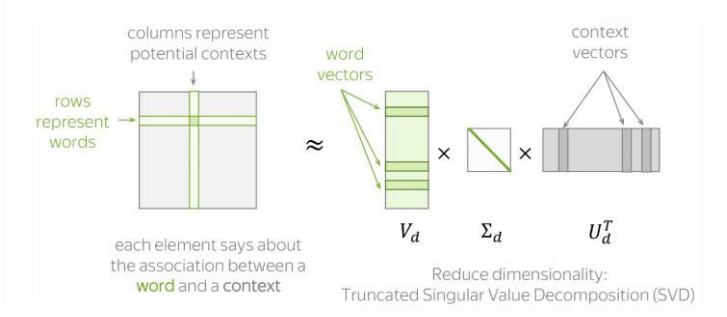
Возможны различные варианты учёта со-встречаемости слов:

- сумма по всей коллекции числа попаданий пары слов в окно фиксированного размера
- количество документов, хоть раз содержащих пару слов
- количество документов, хоть раз содержащих пару слов в окне

SVD для получения эмбеддингов слов

Понижаем размерность через SVD-разложение: $X = USV^T$

Из столбцов матрицы U выбираем первые K компонент



https://lena-voita.github.io/nlp_course/word_embeddings.html

Недостатки SVD

- Относительно низкое качество получаемых представлений
- Сложность работы с очень большой и разреженной матрицей
- Сложность добавления новых слов/документов (решается инкрементальными методами построения)

word2vec

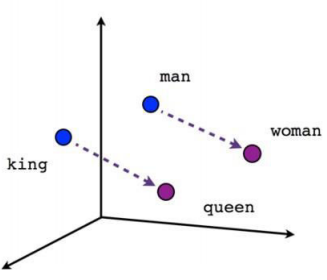
Mikolov, K. Chen, G. Corrado, J. Dean. Efficient Estimation of Word Representations in Vector Space (2013)

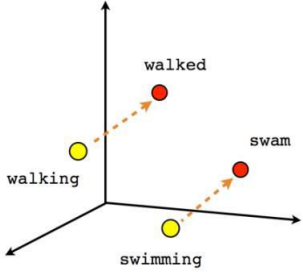
T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean. Distributed Representations of Words and Phrases and their Compositionality (2013).

https://arxiv.org/abs/1301.3781

word2vec

- **word2vec** — группа алгоритмов, предназначенных для получения векторных представлений слов
- **Идея:** «Слова со схожими значениями разделяют схожий контекст»
- Как правило, в векторном представлении семантически близкие слова оказываются рядом

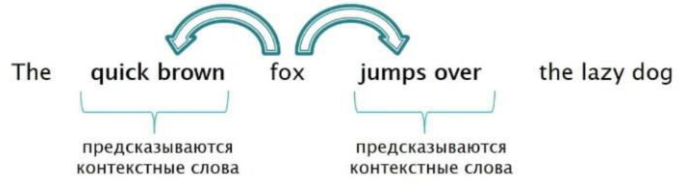




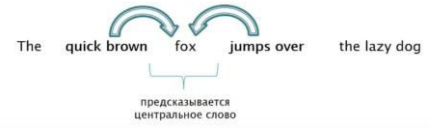
Две модели

Skip-gram: по текущему слову предсказываем контекст

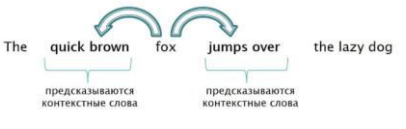
Continuous BOW: по контексту предсказываем текущее слово



Word2Vec



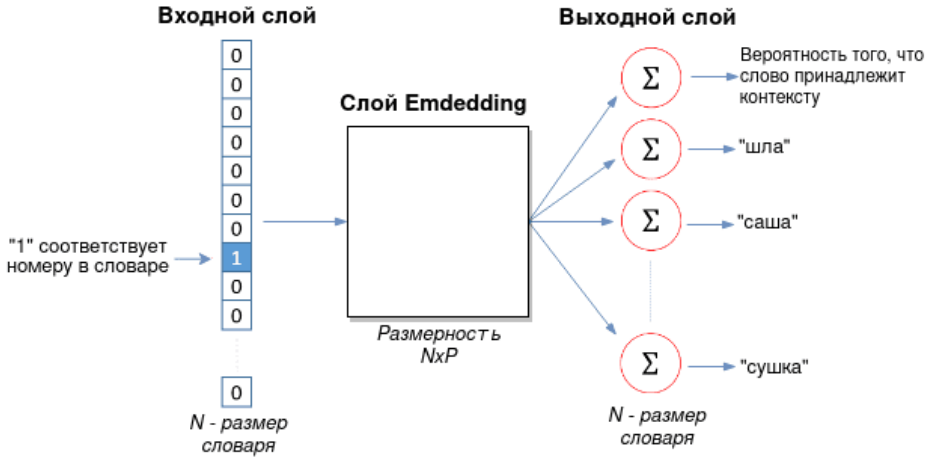
CBOW



Skip-gram

https://arxiv.org/abs/1301.3781

Архитектура w2v

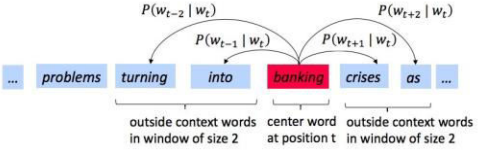


Skip-gram model

- 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:

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

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

Negative Sampling

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

- 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

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

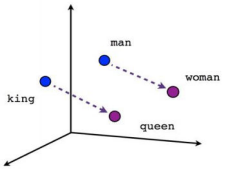
- w_i — случайно выбранные слова

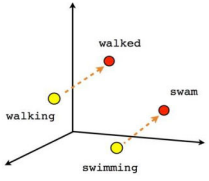
- Положительные примеры — слова, стоящие рядом
- Отрицательные примеры: подбираем к слову «шум», то есть другое слово, которое не находится рядом
- Важно семплировать в SGD слова с учётом их популярности — иначе будем обучаться только на самые частые слова

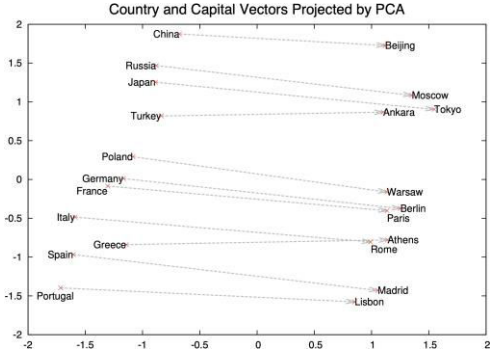
Как это использовать?

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

word2vec


man
woman
king
queen


walked
swam
walking
swimming


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Ankara
Turkey
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Athens
Rome
Greece
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

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
- Трансформерные модели (BERT и т. п.)

Работа с текстом

- Векторные представления строятся для слов
- Можно просто усреднить по всем словам — получим признаки для текста
- Можно усреднять с весами
- Популярный вариант: *Word2Vec/FastText с весами tf-idf*
- Можно ли умнее?

Практика