

Векторные представления слов

Математические методы анализа текстов
осень 2020

Попов Артём Сергеевич

9 сентября 2020

Этапы решения NLP-задачи

1. Выбор метрики качества
2. Сбор обучающих и тестовых данных
3. Предобработка данных
4. **Формирование признакового описания текста**
5. Выбор подхода и класса моделей
6. Обучение моделей и настройка решения

Простейшее признаковое описание текста — мешок слов (bag of words):

- ▶ $v_d = \{n_{wd}\}_{w \in W}$ — признаковое представление документа d
- ▶ W — множество уникальных слов коллекции (словарь)
- ▶ n_{wd} — сколько раз слово w встречается в документе d

Агрегация представлений слов для представлений документа

1. Каждому слову $w \in W$ сопоставим вектор $v_w \in \mathbb{R}^m$ — представление слова (word embedding), m — размерность пространства
2. Представление документа будем вычислять как агрегирующую функцию от векторов слов документа (например, среднее или сумма)

Модель мешка слов (другой взгляд):

1. Каждому слову $w \in W$ соответствует one-hot вектор:

$$v_w = [0, \dots, 0, \underbrace{1}_w, 0, \dots, 0] \in \mathbb{R}^{|W|}$$

2. Представление документа $d = \{w_1, \dots, w_n\}$ вычисляется через сумму:

$$v_d = \sum_{w \in W} n_{wd} v_w = \sum_{w \in d} v_w$$

Свойства one-hot представлений слов

- + Очень легко и быстро построить
- + Неплохое качество решения задач на длинных текстах
- ± Разреженность
 - Большая размерность
 - Ортогональность всех представлений слов
 - Нет механизма обработки незнакомых слов (out of vocabulary, OOV) на тесте

Проблемы возникают на коротких предложениях...

Мы твёрдо верим **в то, что** оправдаем ожидания поклонников оригинальной трилогии *StarWars*.

Мы абсолютно уверены, **что не** разочаруем фанатов классических «Звёздных войн».

Мы пришли **к** выводу, **что** Луна, вероятно, вертится вокруг Земли.

После удаления стоп-слов:

$$\rho(d_1, d_2) = \rho(d_1, d_3) \quad (\text{евклидово, косинусное})$$

А ещё есть задачи, где объект — слово (поиск синонимов).

Задача построения представлений слов

Дано: $D = \{w_1, w_2, \dots, w_N\}$ — текстовая коллекция
 D — конкатенация всех документов
 $w_i \in W$ — слово, W — словарь коллекции

Найти: векторное представление $v_w \in \mathbb{R}^m$ для каждого слова w

Какие представления считать хорошими?

- ▶ Близким по смыслу словам соответствуют близкие по расстоянию вектора.
- ▶ Небольшая размерность — $m \ll |W|$.
- ▶ Интерпретируемые арифметические операции в пространстве \mathbb{R}^m .
- ▶ Качество решения конечной задачи.

Игра «угадай слово»

▶ рампетка

▶ корец

▶ рында

Игра «угадай слово»

► рампетка

Мы вышли на свою охоту за бабочками, каждый с двумя рампетками.

► корец

Петришка бурлыкнул бутылью об лавку и вновь припал к корцу с квасом.

► рында

В рынду бьют каждые полчаса для обозначения времени и для подачи сигналов при тумане.

Игра «угадай слово»

- ▶ **рампетка** — сачок для ловли бабочек.

Мы вышли на свою охоту за бабочками, каждый с двумя рампетками.

- ▶ **корец** — ковш для черпанья воды, кваса.

Петришка бурлыкнул бутылью об лавку и вновь припал к корцу с квасом.

- ▶ **рында** — судовой колокол.

В рынду бьют каждые полчаса для обозначения времени и для подачи сигналов при тумане.

Гипотеза дистрибутивности

Формулировка 1¹: слова, совстречающиеся с одними и теми же словами, имеют схожее значение.

слово совстречаются \Leftrightarrow встречаются в окне k :

...an efficient method for **learning** high quality vector ...
 контекст, k = 2 контекст, k = 2

Формулировка 2²: слово характеризуется словами, с которыми оно встречается.

¹Harris (1954). Distributional structure

²Firth (1957). A synopsis of linguistic theory

Матрица совстречаемостей слов (Co-occurrence matrix)

Составим по коллекции матрицу $X \in \mathbb{R}^{|W| \times |W|}$, $X_{wc} = f(w, c, D)$

Примеры:

- ▶ $X_{wc} = n_{wc}$ — количество совстречаний слов w и c
- ▶ $X_{wc} = PMI(w, c)$ — pointwise mutual information

$$PMI(w, c) = \log \frac{p(w, c)}{p(w)p(c)} = \log \frac{n_{wc}}{n_c n_w} + Const$$

n_w — число появлений слова w в коллекции

- ▶ $X_{wc} = PPMI(w, c) = \max(0, PMI(w, c))$

X_w — представление $\in \mathbb{R}^{|W|}$, решающее проблему ортогональности.

Как получить представление $\in \mathbb{R}^m$, $m \ll |W|$?

Матрица совстречаемостей слов (Co-occurrence matrix)

Составим по коллекции матрицу $X \in \mathbb{R}^{|W| \times |W|}$, $X_{wc} = f(w, c, D)$

Примеры:

- ▶ $X_{wc} = n_{wc}$ — количество совстречаний слов w и c
- ▶ $X_{wc} = PMI(w, c)$ — pointwise mutual information

$$PMI(w, c) = \log \frac{p(w, c)}{p(w)p(c)} = \log \frac{n_{wc}}{n_c n_w} + Const$$

n_w — число появлений слова w в коллекции

- ▶ $X_{wc} = PPMI(w, c) = \max(0, PMI(w, c))$

X_w — представление $\in \mathbb{R}^{|W|}$, решающее проблему ортогональности.

Как получить представление $\in \mathbb{R}^m$, $m \ll |W|$? Воспользоваться методами уменьшения размерности.

SVD для построения представлений

Хотим построить матричное разложение X :

$$X = UV^T$$

Используем SVD разложение:

$$X = \hat{U}_d \Sigma_d \hat{V}_d^T, \quad U = \hat{U}_d \sqrt{\Sigma_d}, \quad V = \hat{V}_d \sqrt{\Sigma_d}.$$

Представления слов — строки матриц U или V .

В ¹ показано, что такой метод при определённых условиях показывает хорошее качество на стандартных бенчмарках.

¹Levy et al (ACL 2015), Improving Distributional Similarity with Lessons Learned from Word Embeddings.

Glove

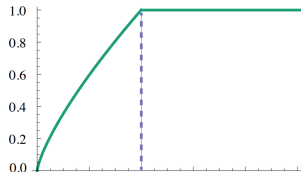
Glove — ещё одно матричное разложение.

Методом Adagrad обучается функционал:

$$\mathcal{L} = \sum_{w \in W} \sum_{c \in W} F(n_{wc}) (\langle u_w, v_c \rangle + b_w + \hat{b}_c - \log n_{wc})^2 \longrightarrow \min_{U, V, b, \hat{b}}$$

Боремся с шумовыми редкими словами с помощью F :

$$F(n_{wc}) = \begin{cases} \left(\frac{n_{wc}}{n_{max}} \right)^{3/4}, & n_{wc} < n_{max} \\ 1, & \text{иначе} \end{cases}$$



Популярен, но на практике обычно хуже word2vec...

Резюме по count-based подходам

- + Неплохое качество в некоторых задачах (но нужно уметь настраивать)
- + Маленькая размерность
- + Близким словам соответствуют близкие вектора
- Нет хорошего механизма обработки новых слов на тесте
- **Основной минус:** необходимо собирать огромную (но разреженную!) матрицу встречаемостей для обучения

Мотивация prediction-based подхода

Хотим обновлять параметры модели «на ходу», не составляя матрицу совстречаемостей.

Идея: обучаем модель «воспроизводить» локально гипотезу Фёрса:

- ▶ Модель CBOW — по словам контекста необходимо предсказать центральное слово
- ▶ Модель Skip-gram — по центральному слову, необходимо предсказать каждое из слов контекста

Обратите внимание! Идея очень схожа с языковой моделью, но контекст — не только слова перед словом.

Напоминания и обозначения

Обозначения для softmax:

- ▶ Если $f(w)$ — скалярная функция, то:

$$\text{softmax}_{w \in W} f(w) = \frac{\exp(f(w))}{\sum_{w' \in W} \exp(f(w'))}$$

Везде далее мы будем учить две матрицы представлений:

- ▶ $v_w \in \mathbb{R}^m$ из матрицы V
- ▶ $u_w \in \mathbb{R}^m$ из матрицы U

Почему две матрицы:

- ▶ Не накладываем дополнительного ограничения (симметричность) на входные данные
- ▶ Проще считать градиенты — быстрее обучаемся

Модель CBOW

Функционал обучения — предсказываем центральное слово по контексту:

$$\sum_{i=1}^N \log p(w_i | C(i)) \rightarrow \max_{V, U}$$

$C(i) = \{w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k}\}$ — локальный контекст w_i

1 этап — вычисление среднего входных векторов:

$$v^{-i} = \frac{1}{2k} \sum_{w \in C(i)} v_w = \frac{1}{2k} \sum_{\substack{j=-k, \\ j \neq 0}}^k v_{w_{i+j}}$$

2 этап — применение линейного слоя с softmax активацией:

$$p(w | (w_i)) = \operatorname{softmax}_{w \in W} U v^{-i} = \operatorname{softmax}_{w \in W} \langle u_w, v^{-i} \rangle$$

Модель Skip-gram

Функционал обучения — предсказываем слова контекста по центральному:

$$\sum_{i=1}^N \sum_{w \in C(i)} \log p(w|w_i) = \sum_{i=1}^N \sum_{\substack{j=-k \\ j \neq 0}}^k \log p(w_{i+j}|w_i) \rightarrow \max_{V,U}$$

$$p(w|w_i) = \operatorname{softmax}_{w \in W} Uv_{w_i} = \operatorname{softmax}_{w \in W} \langle u_w, v_{w_i} \rangle$$

- ▶ CBOW и Skip-gram обучаются с помощью SGD.
- ▶ Skig-gram лучше моделирует редкие слова коллекции.

Какая сложность итерации обучения CBOW и Skip-gram?

Модель Skip-gram

Функционал обучения — предсказываем слова контекста по центральному:

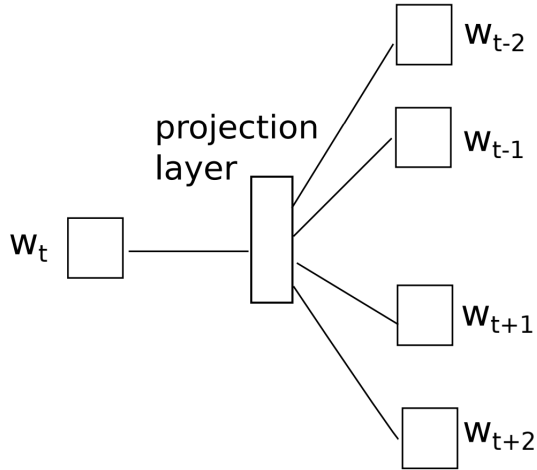
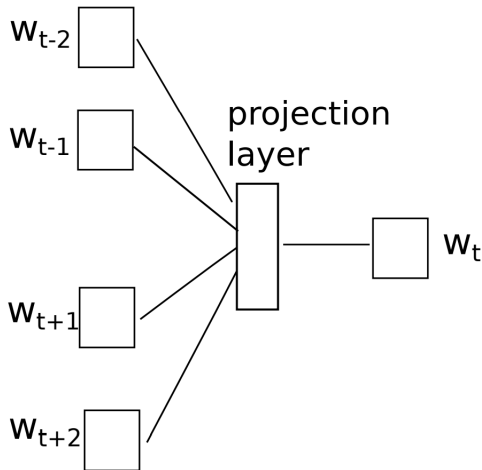
$$\sum_{i=1}^N \sum_{w \in C(i)} \log p(w|w_i) = \sum_{i=1}^N \sum_{\substack{j=-k \\ j \neq 0}}^k \log p(w_{i+j}|w_i) \rightarrow \max_{V,U}$$

$$p(w|w_i) = \operatorname{softmax}_{w \in W} Uv_{w_i} = \operatorname{softmax}_{w \in W} \langle u_w, v_{w_i} \rangle$$

- ▶ CBOW и Skip-gram обучаются с помощью SGD.
- ▶ Skig-gram лучше моделирует редкие слова коллекции.

Какая сложность итерации обучения CBOW и Skip-gram? $O(|W|)$

Модели CBOW и Skip-gram



Сложность одной итерации skip-gram

Пусть w_i это s -ое слово словаря, w_{i+j} — t -ое.

Посчитаем градиенты по u_t и u_k , $k \neq t$ и $k \neq t$:

$$L_{ts} = \log p(t|s) = \log \operatorname{softmax}_{w \in W} \langle u_w, v_s \rangle \big|_{w=t}$$

$$\begin{aligned} \frac{dL_{ts}}{du_t} &= \frac{d \log \operatorname{softmax}_{t \in W} \langle u_t, v_s \rangle}{du_t} = \\ &= v_s - \frac{d \log \sum_{w \in W} \exp(\langle u_w, v_s \rangle)}{du_t} = v_s (1 - \operatorname{softmax}_{w \in W} \langle u_w, v_s \rangle) \big|_{w=t} \end{aligned}$$

$$\frac{dL_{ts}}{du_k} = \frac{d \log \operatorname{softmax}_{t \in W} \langle u_t, v_s \rangle}{du_k} = -v_s \operatorname{softmax}_{w \in W} \langle u_w, v_s \rangle \big|_{w=k}$$

Способы ускорения модели

1. Замена softmax на другую функцию, задающую распределение:
 - ▶ Hierarchical softmax¹
 - ▶ Differentiated softmax
 - ▶ ...
2. Замена функционала модели на более простой:
 - ▶ Noise contrastive estimation
 - ▶ Negative sampling¹
 - ▶ Importance sampling
 - ▶ Self-normalization
 - ▶ Infrequent Normalization
 - ▶ ...

¹Mikolov (NIPS 2013), Distributed representations of words and phrases and their compositionality

²Ruder; On word embeddings - Part 2: Approximating the Softmax; <http://ruder.io/word-embeddings-softmax/>

Hierarchical softmax. Структура дерева.

Идея: заменить softmax на другую функцию, оптимизация которой будет иметь сложность $O(\log|W|)$.

Предварительный этап:

- ▶ Перед обучением модели по множеству пар слов и их частот строится бинарное дерево Хаффмана.
- ▶ Каждой вершине дерева соответствует обучаемое представление.
- ▶ Листья дерева соответствуют словам. Представления в листьях — искомые представления для слов.
- ▶ Представления внутренних вершин дерева используются для вычисления вероятности $p(\text{right}|w, n)$ — вероятность, что слово w лежит в правом поддереве вершины n .

$$p(\text{right}|n, w) = \sigma(\langle u_n, v_w \rangle) = 1 - p(\text{left}|n, w)$$

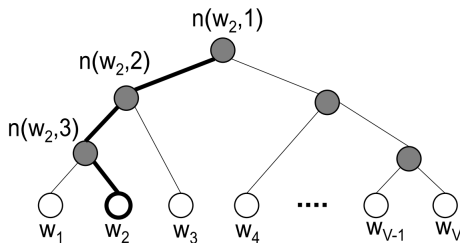
Hierarchical softmax. Обучение модели.

Пусть $n(w) = [n_1(w), n_2(w), \dots]$ задаёт путь от корня до слова w и пусть:

$$p(w|w_i) = p(n(w)|w_i) = \prod_{j=1}^{|n(w)|-1} \underbrace{p(n_j(w) \rightarrow n_{j+1}(w) | n_j, w_i)}_{\text{right or left}}$$

Пример на картинке:

$$p(w_2|w_x) = p(\text{left}|w_x, 1)p(\text{left}|w_x, 2)p(\text{right}|w_x, 3)$$



Negative sampling (сэмплирование негативных примеров)

Исходный метод: вероятность встретить w в контексте c в коллекции $|W|$ вероятностных распределений, каждое с $|W|$ исходами

Negative sampling: вероятность встретить пару (w, c) в коллекции $|W| \times |W|$ вероятностных распределений, каждое с 2 исходами

$$p(1|c, w) = \sigma(\langle v_c, u_w \rangle) = 1 - p(0|c, w)$$

В чём проблема следующей модели?

$$\sum_{i=1}^N \sum_{w \in C(i)} \log p(1|w, w_i) \rightarrow \max_{V, U}$$

Negative sampling (сэмплирование негативных примеров)

Исходный метод: вероятность встретить w в контексте c в коллекции $|W|$ вероятностных распределений, каждое с $|W|$ исходами

Negative sampling: вероятность встретить пару (w, c) в коллекции $|W| \times |W|$ вероятностных распределений, каждое с 2 исходами

$$p(1|c, w) = \sigma(\langle v_c, u_w \rangle) = 1 - p(0|c, w)$$

В чём проблема следующей модели?

$$\sum_{i=1}^N \sum_{w \in C(i)} \log p(1|w, w_i) \rightarrow \max_{V, U}$$

Переобучение. Только один класс в модели.

Negative sampling (сэмплирование негативных примеров)

Чтобы не переобучаться, будем на каждой итерации сэмплировать n случайных негативных примеров:

$$\sum_{i=1}^N \left(\sum_{w \in C(i)} \log p(1|w_{i+j}, w_i) + \sum_{w'_k \sim p(w)^{3/4}} \log p(0|w_i, w'_k) \right) \rightarrow \max_{V,U}$$

Часто функционал записывают так:

$$\sum_{i=1}^N \left(\sum_{w \in C(i)} \log p(1|w_{i+j}, w_i) + K \mathbb{E}_{w \sim p(w)^{3/4}} \log p(0|w_i, w) \right) \rightarrow \max_{V,U}$$

Важно. Приём популярен не только при обучении skip-gram, но и в любой ситуации, когда у вас в выборке только позитивные пары.

Дополнительно

Трюки для модели:

- ▶ Subsampling — случайное удаление частых слов
С вероятностью $1 - t/n_w$ удаляем слово из обучения
 t — выбранный порог, n_w — частота слова
- ▶ Dynamic window — случайный выбор размера контекста на каждой итерации
- ▶ Комбинация итоговых векторов — использовать в качестве представления $\alpha v_w + (1 - \alpha) u_w$

Общепопулярные практические рекомендации:

- ▶ Размер представлений от 100 до 400
- ▶ Если документы специфичные, лучше учить модель на этом специфичном домене

Резюме по word2vec

- + Хорошее качество в самых разных прикладных задачах.
- + Маленькая размерность.
- + Близким словам соответствуют близкие вектора.
- Плохой механизм обработки новых слов на тесте.
- ≠ Требуют большего корпуса чем count-based модели.

OOV слова

Проблема OOV слов (Out of vocabulary): отсутствие векторов для слов, которых не было в коллекции.

Простые способы решения проблемы (word2vec и count-based):

- ▶ использование специального UNK токена для всех редких и новых слов
- ▶ восстановление нового слова по его контексту

Продвинутые способы решения проблемы

- ▶ исходная модель должна работать не со словами, а с символами или символьными n-граммами
- ▶ обучению по обученным векторам функции, восстанавливающей вектор по символам или символьным n-граммам.

Модель представлений FastText

FastText¹ — построение представлений слов как суммы представлений для буквенных n -грамм слова.

В Skip-gram меняется только подсчёт вектора u_w :

$$u_w = \sum_{g \in G(w)} u_g, \quad G(w) \text{ — } n\text{-граммы слова } w$$

Пример: $G(\text{where}) = \text{_wh} + \text{whe} + \text{her} + \text{ere} + \text{re_}$

¹Bojanowski et al (ACL 2017); Enriching Word Vectors with Subword Information; 2016

Методы генерализации обученных представлений

- ▶ Исходные данные — матрица представлений V для слов из W
- ▶ $f_\theta(w)$ строит представление для w по символьной информации, например:

$$f_\theta(w) = \sum_{g \in G(w)} \theta_g$$

$$f_\theta(w) = LSTM_\theta(S(w)), \quad S(w) — \text{символы } w$$

- ▶ Обучение f_θ :

$$\sum_{w \in W} \|f_\theta(w) - v_w\|^2 \rightarrow \min_{\theta}$$

¹Pinter et al (EMNLP 2016); Mimicking Word Embeddings using Subword RNNs

²Zhao et al (EMNLP 2018); Generalizing Word Embeddings using Bag of Subwords

Вспомним начало лекции...

Какие представления считать хорошими?

1. Близким по смыслу словам соответствуют близкие по расстоянию вектора.
2. Небольшая размерность.
3. Интерпретируемые арифметические операции в пространстве \mathbb{R}^m .
4. Качество решения конечной задачи.

Эксперимент

Рассмотрим модели, обученные по двум датасетам:¹

- ▶ Статьи Википедии + Национальный корпус русского языка
- ▶ Статьи сайта Lurkmore (3.5K статей)

Для Википедии используем модель с сайта RusVectors².

Для Lurkmore обучим модель с нуля с помощью пакета Gensim.

¹идея позаимствована из лекции Мурата Апишева для курса «Анализ Неструктурированных данных» ФКН ВШЭ

²ruwikiscorpora-func_upos_skipgram_300_5_2019

Детали предобработки

Коллекция Луркморье:

- ▶ Все символы кроме букв были удалены
- ▶ Все слова лемматизированы (pymorphy2)
- ▶ Один документ — один абзац (важно при учёте контекста)
- ▶ Абзацы меньше двух слов были удалены

Коллекция Википедии:

- ▶ Все слова лемматизированы (UDPipe)
- ▶ Каждое слово преобразовано в слово_{часть речи}

Детали обучения на коллекции Луркморье

```
from gensim.models import Word2Vec
from gensim.models.word2vec import LineSentence

data_loader = LineSentence("lurkmore_all.txt")

model_lurk = Word2Vec(
    data_loader, # данные
    size=100, # размер представлений
    sg=0, hs=0, # тип алгоритма
    window=5, # размер окна
    min_count=5, # минимальная частота
    workers=4, iter=20,
)
```

Детали загрузки модели по Википедии

```
from gensim.models import KeyedVectors

model_wiki = KeyedVectors.load_word2vec_format(
    # путь к бинарнику модели
    "nkr1_w2v/model.bin",
    binary=True,
)
```

Операции с векторами в gensim

Получить вектор из модели:

```
word_embedding = model_lurk.wv['вектор']  
word_embedding.shape  
# (100, )
```

Поиск похожих слов к арифметической комбинации:

```
similar_token_info = model_lurk.most_similar(  
    positive=['мужчина', 'король'],  
    negative=['женщина'],  
    topn=10  
)
```

Похожие слова¹

Википедия

most_similar(россия_PROPN)

страна 0.695

европа 0.679

российский 0.604

франция 0.582

германия 0.574

most_similar(полковник_NOUN)

подполковник 0.904

майор 0.875

генерал 0.805

генерал-майор 0.799

ротмистр 0.770

Луркморье

most_similar(россия)

ссср 0.759

сша 0.754

германия 0.741

рашка 0.730

грузия 0.719

most_similar(полковник)

генерал 0.648

подполковник 0.647

майор 0.599

генералмайор 0.573

адмирал 0.557

¹при выводе для википедии pos-теги удалялись при отсутствии повторений

Похожие слова

Википедия

most_similar(тролль_NOUN)

гном 0.661

троллый 0.656

эльф 0.627

тролли 0.609

гоблин 0.589

most_similar(музыка_NOUN)

мелодия 0.702

джаз 0.669

пение 0.649

песня 0.642

танец 0.630

Луркморье

most_similar(тролль)

троллинг 0.668

лурко** 0.538

провокатор 0.530

фрик 0.517

быдло 0.516

most_similar(музыка)

мелодия 0.668

рэп 0.647

попёс 0.642

песнь 0.641

звук 0.630

Похожие слова

Википедия

most_similar(мгу_PROPN)

мгу 0.843

лгу 0.773

м::в::ломоносов 0.728

мпгу 0.701

спбгу 0.697

most_similar(физтех_PROPN)

физтех_NOUN 0.701

мфти 0.694

мифи 0.632

физтех_DET 0.580

мирэа 0.578

Луркморье

most_similar(мгу)

университет 0.755

вуз 0.665

пту 0.656

мгимо 0.646

аспирант 0.640

most_similar(физтех)

мехмат 0.537

мифь 0.524

мгимо 0.518

мгу 0.502

филфак 0.496

Арифметические операции в пространстве

яндекс - россия + сша:

Википедия

гугл 0.518

yahoo 0.467

пентагон 0.464

symantec 0.443

яндескяча 0.441

король - мужчина + женщина:

королева_NOUN 0.754

королева_ADV 0.672

принц 0.627

королева_ADJ 0.625

король 0.623

Луркморье

гугл 0.593

google 0.508

гуголь 0.504

rm 0.502

кэш 0.497

император 0.583

королевский 0.555

фараон 0.548

халиф 0.523

герцог 0.523

Intrinsic задачи для оценивания

Задача близости:

Данные: Список троек: w_1, w_2 — слова, x — близость между ними

Модель: Измеряем близости между w_1 и w_2 , например $\cos(u_{w_1}, u_{w_2})$

Мера: Корреляция Спирмена между двумя списками близостей

Задача аналогий:

Данные: Список четвёрок слов w_1, w_2, w_3, w_4

w_1 относится к w_2 так же, как w_3 к w_4

Модель: Находим самое близкое слово к $u_{w_3} - u_{w_1} + u_{w_2}$

Мера: Доля правильно найденных слов

¹Rogers et. al. (*SEM 2017), The (Too Many) Problems of Analogical Reasoning with Word Vectors

²T. Linzen (2016), Issues in evaluating semantic spaces using word analogies

³Levy et. al. Improving distributional similarity with lessons learned from word embeddings, 2015

Как можно использовать word embeddings?

1. Решать задачи поиска близких слов, синонимов и т.п.
2. Получить представление документа/предложения, которое можно использовать для решения задачи машинного обучения
3. Использовать представление слова в качестве фиксированного представления в сложной архитектуре (например, рекуррентной сети)
4. Использовать для инициализации представлений в сложной архитектуре

Измерение качества моделей по конечной задаче всегда лучше чем измерение по intrinsic задачам!

Агрегация векторов для представления документа

- ▶ Сумма векторов
- ▶ Среднее векторов
- ▶ Взвешенная сумма (tf-idf или idf веса)
- ▶ Координатный max-pool
- ▶ Координатный hierarchical-pool — усреднение соседних по окну слов, а затем max-pool

Очень хороший бейзлайн в любой задаче!

Полезные ссылки

- ▶ Gensim — пакет, позволяющий легко работать с различными моделями эмбедингов
- ▶ fasttext — библиотека fasttext для обучения эмбедингов fasttext с нуля
- ▶ Wikipedia2Vec — эмбединги для разных языков
- ▶ RusVectores — сайт с эмбедингами на русском языке
- ▶ StarSpace — ещё одна модель эмбедингов, позволяющая учить их под конечную задачу
- ▶ Word Mover's Distance — необычный и эффективный способ вычисления расстояний между предложениями с помощью эмбедингов слов

Skip-gram как count-based метод

Skip-gram можно записать как count-based метод:

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^N \sum_{\substack{j=-k \\ j \neq 0}}^k \log p(w_{i+j}|w_i) = \sum_{w \in W} \sum_{c \in W} n_{wc} \log p(c|w) = \\ &= \sum_{w \in W} n_w \sum_{c \in W} \frac{n_{wc}}{n_w} \log p(c|w) \rightarrow \max_{U,V} \quad (1)\end{aligned}$$

Добавление константы не меняет задачи оптимизации:

$$\begin{aligned}(1) &\Leftrightarrow \sum_{w \in W} n_w \sum_{c \in W} \frac{n_{wc}}{n_w} \left(\log p(c|w) - \log \frac{n_{wc}}{n_w} \right) = \\ &= - \sum_{w \in W} n_w \sum_{c \in W} \hat{p}(c|w) \log \frac{\hat{p}(c|w)}{p(c|w)} \rightarrow \max_{U,V} \quad (2)\end{aligned}$$

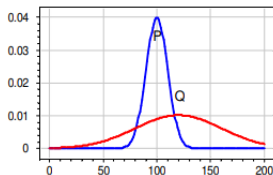
KL-дивергенция и её свойства

Мера расстояния между распределениями

$P = \{p_i\}_{i=1}^s$ и $Q = \{q_i\}_{i=1}^s$.

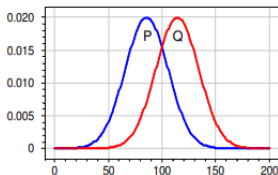
$$KL(P\|Q) = \sum_i p_i \log \frac{p_i}{q_i}$$

1. $KL(P\|Q) \geq 0$
2. $KL(P\|Q) = 0 \Leftrightarrow P = Q$
3. $KL(P\|Q)$ — мера вложенности P в Q



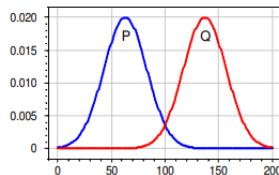
$$KL(P\|Q) = 0.44$$

$$KL(Q\|P) = 2.97$$



$$KL(P\|Q) = 0.44$$

$$KL(Q\|P) = 0.44$$



$$KL(P\|Q) = 2.97$$

$$KL(Q\|P) = 2.97$$

Skip-gram как count-based метод

В модели skip-gram строится матричное разложение матрицы $X_{wc} = \hat{p}(w|c)$:

$$\begin{aligned}
 (2) - \sum_{w \in W} n_w KL(\hat{p}(c|w) \| p(c|w)) &\rightarrow \max_{U, V} \Leftrightarrow \\
 &\Leftrightarrow \sum_{w \in W} n_w KL(\hat{p}(c|w) \| p(c|w)) \rightarrow \min_{U, V}
 \end{aligned}$$

Обратите внимание! Skip-gram схожа с тематической моделью PLSA, обученной по документам, составленным по совстречаемостям слов¹.

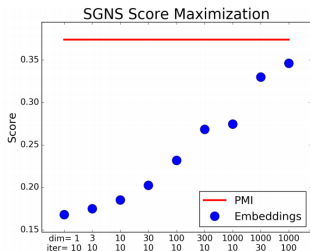
¹Potapenko et al (2017). Interpretable probabilistic embeddings: bridging the gap between topic models and neural networks

Интерпретация skip-gram negative sampling

Утверждение (Леви)¹

Пусть для любых $w, c \in W$ результат $\langle v_w, u_c \rangle$ не зависит от других пар слов. Тогда, в точке максимума SGNS для любых $w, c \in W$ будет выполнено:

$$\langle v_w, u_c \rangle = PMI(w, c) - \log K$$



На практике эффект наблюдается при больших размерах представлений.²

¹O. Levy et al (NIPS 2014), Neural Word Embedding as Implicit Matrix Factorization

²O. Melamud et al (ACL 2017), Information-Theory Interpretation of the Skip-Gram Negative-Sampling Objective Function