Векторные представления слов

Математические методы анализа текстов
осень 2019

Попов Артём Сергеевич

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

Алгоритм решения задач машинного обучения:

1. Построение признакового пространства
2. Обучение алгоритма машинного обучения

Как выбрать признаковое пространство в задаче классификации текстов?

Алгоритм решения задач машинного обучения:

1. Построение признакового пространства
2. Обучение алгоритма машинного обучения

Как выбрать признаковое пространство в задаче классификации текстов?

Простая идея:

1. Каждому слову w сопоставим вектор v_w (представление слова / word embedding)
2. Представление документа — функция от векторов слов документа (например, сумма или среднее)

Модель bag-of-words

W — множество всех слов.
Каждому слову $w \in W$ сопоставляем one-hot вектор:

$$v_w = [0, \ldots, 0, \underbrace{1}_{w}, 0, \ldots, 0] \in \mathbb{R}^{|W|}$$

Представление документа $d = \{w_1, \ldots, w_n\}$ задаётся так:

$$v_d = \sum_{w \in d} v_w = \sum_{w \in W} \#\{w \text{ встретилось в } d\} v_w$$

Какие плюсы и минусы у этого представления?

Свойства представлений one-hot

+ Очень легко и быстро построить
+ Неплохое качество решения задач линейными моделями для длинных текстов
± Разреженность
– Большая размерность
– Ортогональность всех представлений слов
– Нет механизма обработки новых слов на тесте

Проблемы возникают на коротких предложениях...

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

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

Мы пришли к выводу, что Земля, вероятно, вертится вокруг Луны.

$$\rho(d_1, d_2) = \rho(d_1, d_3)???$$ (евклидово, косинусное)

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

Задача построения представлений слов

Дано: $D = \{w_1, w_2, \ldots, w_N\}$ — текстовая коллекция
$w_i \in W$ — словарь коллекции

Найти: векторное представление $v_w \in \mathbb{R}^m$ для каждого слова w , где $m \ll |W|$

Какие представления считать хорошими?

- Близким по смыслу словам соответствуют близкие по расстоянию вектора.
- Небольшая размерность.
- 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

Гипотеза дистрибутивности[1]: слова, совстречающиеся с одними и теми же словами, имеют схожее значение.

слово совстречаются $\Leftrightarrow$ встречаются в окне k:

... an efficient $\underbrace{\text{method for}}_{\text{контекст, k = 2}}$ learning $\underbrace{\text{high quality}}_{\text{контекст, k = 2}}$ vector ...

Развитие идеи[2]: слово характеризуется словами, с которыми оно совстречается.

[1] Harris (1954). Distributional structure

[2] Firth (1957). A synopsis of linguistic theory

Матрица совстречаемостей слов (Co-occurrence matrix)

Составим по коллекции матрицу X:

- $X \in \mathbb{R}^{|W| \times |W|}, \quad X_{wc} = f(w, c, D)$

Примеры X_{wc}:

- $X_{wc} = n_{wc}$ — количество совстречаний слов w и c
- $X_{wc} = PMI(w, c) = \log(n_{wc} N) - \log(n_w n_c)$
- $X_{wc} = PPMI(w, c) = max(0, PMI(w, c))$

X_w — представление $\in \mathbb{R}^{|W|}$.
X_w решает проблему ортогональности представлений!

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

Матрица совстречаемостей слов (Co-occurrence matrix)

Составим по коллекции матрицу X:

- $X \in \mathbb{R}^{|W| \times |W|}, \quad X_{wc} = f(w, c, D)$

Примеры X_{wc}:

- $X_{wc} = n_{wc}$ — количество совстречаний слов w и c
- $X_{wc} = PMI(w, c) = \log(n_{wc} N) - \log(n_w n_c)$
- $X_{wc} = PPMI(w, c) = max(0, PMI(w, c))$

X_w — представление $\in \mathbb{R}^{|W|}$.
X_w решает проблему ортогональности представлений!

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:

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

Представления слов — строки матриц U или V.

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

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

Glove

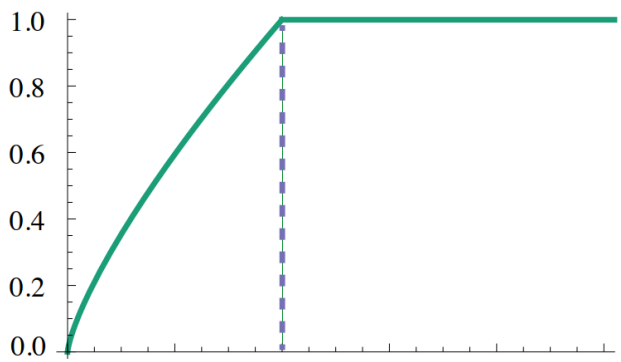
Glove — ещё одно матричное разложение.

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...

Pennington et al (EMNLP 2014), GloVe: Global Vectors for Word Representation

Резюме по count-based подходам

- \+ Относительно неплохое качество в некоторых задачах
- \+ Маленькая размерность
- \+ Близким словам соответствуют близкие вектора
- $\mp$ Плохой механизм обработки новых слов на тесте
- – **Основной минус:** необходимо собирать огромную (но разреженную!) матрицу совстречаемостей для обучения

Мотивация prediction-based подхода

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

Идея: попробуем обучить модель «воспроизводить» 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:

- W — множество всех слов
- последовательность $(w_{i+1}, w_{i+2}, \ldots, w_{i+n})$ — w_{i+1}^{i+n}
- Если $f(w)$ — скалярная функция, то:

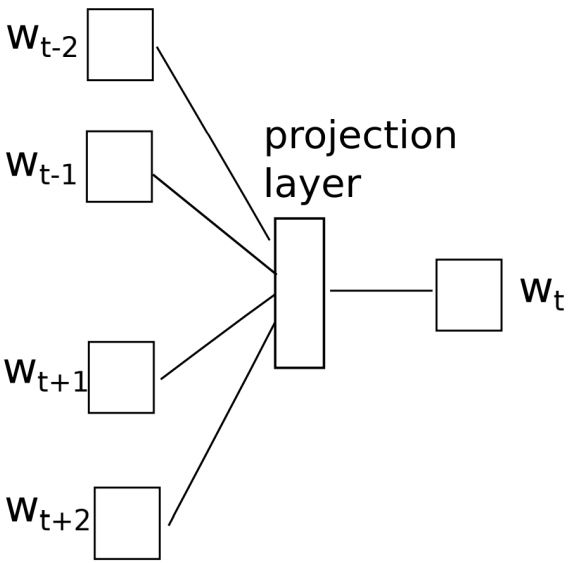
$$\operatorname*{softmax}_{w \in W} f(w) = \frac{\exp(f(w))}{\sum_{w' \in W} \exp(f(w'))}$$

Везде далее мы будем учить две матрицы представлений:

- $v_w \in \mathbb{R}^m$ — представление слова
- $u_w \in \mathbb{R}^m$ — дополнительное представление слова

Модель CBOW

Идея: по словам контекста предсказать центральное слово



$$\sum_{i=1}^{N} \log p\left(w_i \middle| w_{i-k}^{i-1}, w_{i+1}^{i+k}\right) \to \max_{V,U}$$

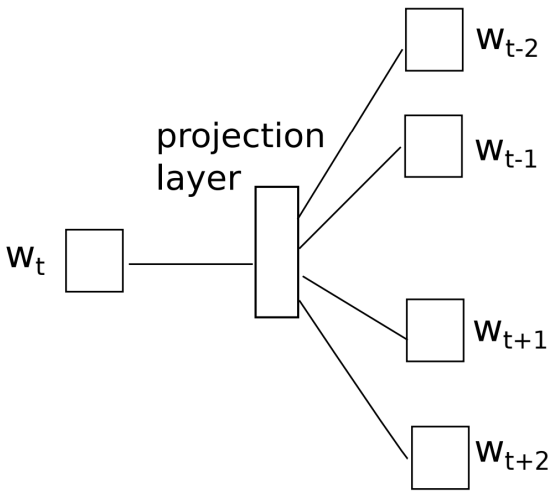
$$u^{-i} = \sum_{\substack{j=-k, \\ j\neq 0}}^{k} u_{w_{i+j}}$$

$$p\left(w \middle| w_{i-k}^{i-1}, w_{i+1}^{i+k}\right) = \underset{w\in W}{\operatorname{softmax}} \left\langle v_w, u^{-i} \right\rangle$$

Mikov et al (2013), Efficient Estimation of Word Representations in Vector Space

Модель Skip-gram

Идея: по центральному слову предсказать слова контекста



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

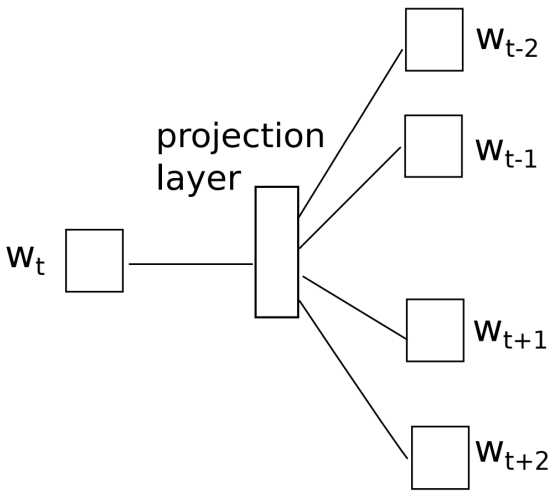
$$p(w | w_i) = \underset{w \in W}{\mathrm{softmax}} \langle v_w, u_{w_i} \rangle$$

CBOW и Skip-gram обучаются с помощью SGD.

Какая сложность одной итерации обучения CBOW и Skip-gram?

Модель Skip-gram

Идея: по центральному слову предсказать слова контекста



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

$$p(w | w_i) = \underset{w \in W}{\mathrm{softmax}} \langle v_w, u_{w_i} \rangle$$

CBOW и Skip-gram обучаются с помощью SGD.

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

Сложность одной итерации skip-gram

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

Посчитаем градиенты по v_t и v_k, $k \neq t$ и $k \neq t$:

$$L_{ts} = \log p(t|s) = \log \underset{w \in W}{\operatorname{softmax}} \langle v_w, u_s \rangle \mid_{w=t}$$

$$\frac{dL_{ts}}{dv_t} = \frac{d \log \underset{t \in W}{\operatorname{softmax}} \langle v_t, u_s \rangle}{dv_t} =$$

$$= u_s - \frac{d \log \sum_{w \in W} \exp(\langle v_w, u_s \rangle)}{dv_t} = u_s(1 - \underset{t \in W}{\operatorname{softmax}} \langle v_t, u_s \rangle)$$

$$\frac{dL_{ts}}{dv_k} = \frac{d \log \underset{t \in W}{\operatorname{softmax}} \langle v_t, u_s \rangle}{dv_k} = -u_s \underset{k \in W}{\operatorname{softmax}} \langle v_k, u_s \rangle$$

Способы ускорения модели

1. Явная аппроксимация софтмакса
   - Hierarchical softmax[1]
   - Differentiated softmax
   - ...
2. Методы, основанные на сэмплировании
   - Noise contrastive estimation
   - Negative sampling[1]
   - Importance sampling
   - Self-normalization
   - Infrequent Normalization
   - ...

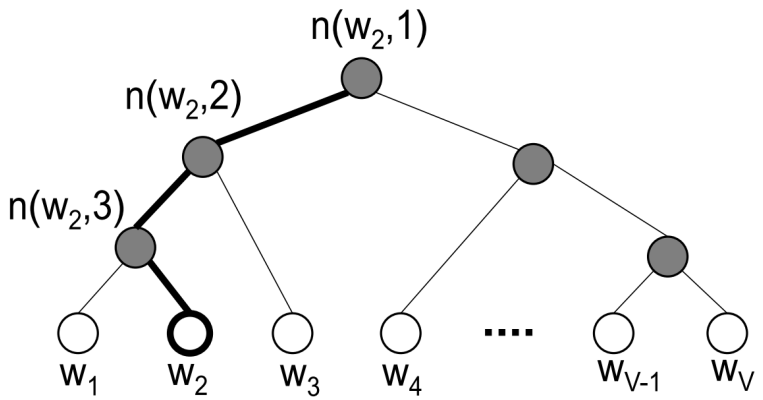
---

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

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

Hierarchical softmax (Иерархический мягкий максимум)

Софтмакс вычисляется через путь в дереве Хаффмана.
В листьях дерева — слова, дерево фиксировано.
Для каждой вершины обучается представление.



Пусть $n(w) = [n_1(w), n_2(w), \ldots]$ задаёт путь до слова w.

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

$$p(right|n, w) = \sigma(\langle v_n, v_w \rangle) = 1 - p(left|n, w)$$

Negative sampling (сэмплирование негативных примеров)

Skip-gram: вероятность встретить пару (w, c) в коллекции

Skip-gram negative sampling: вероятность того, что пара (w, c) может встретиться в коллекции:

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

В чём проблема следующей модели?

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

Negative sampling (сэмплирование негативных примеров)

Skip-gram: вероятность встретить пару (w, c) в коллекции

Skip-gram negative sampling: вероятность того, что пара (w, c) может встретиться в коллекции:

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

В чём проблема следующей модели?

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

Переобучение. Только один класс в модели.

Negative sampling (сэмплирование негативных примеров)

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

$$\sum_{i=1}^{N}\Big(\sum_{\substack{j=-k\\ j\neq 0}}^{k}\log p(1|w_{i+j},w_i)+\sum_{w'_k\sim p(w)^{3/4}}\log p(0|w_i,w'_k)\Big)\to\max_{V,U}$$

Часто функционал записывают так:

$$\sum_{i=1}^{N}\Big(\sum_{\substack{j=-k\\ j\neq 0}}^{k}\log p(1|w_{i+j},w_i)+K\mathbb{E}_{w\sim p(w)^{3/4}}\log p(0|w_i,w)\Big)$$

Дополнительно

Трюки для модели:

- Subsampling — случайное удаление частых слов
  С вероятностью $1 - t/n_w$ удаляем слово из обучения
  t — выбранные 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

Skip-gram как count-based метод

Skip-gram можно записать как count-based метод:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{\substack{j=-k \\ j\neq 0}}^{k} \log p(w_{i+j}|w_i) = \sum_{w\in W} \sum_{c\in W} n_{wc} \log p(c|w) =$$

$$= \sum_{w\in W} n_w \sum_{c\in W} \frac{n_{wc}}{n_w} \log p(c|w) \to \max_{U,V} \quad (1)$$

Добавление константы не меняет задачи оптимизации:

$$(1) \Leftrightarrow \sum_{w\in W} n_w \sum_{c\in W} \frac{n_{wc}}{n_w} \left(\log p(c|w) - \log \frac{n_{wc}}{n_w} \right) =$$

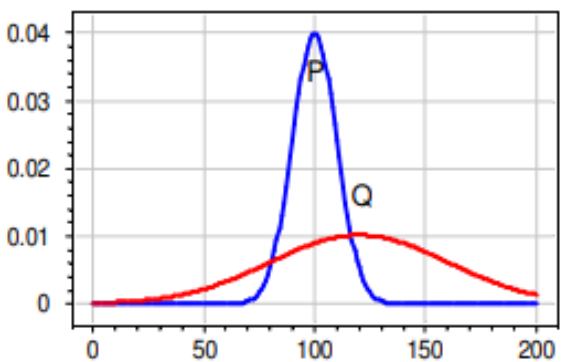
$$= -\sum_{w\in W} n_w \sum_{c\in W} \hat{p}(c|w) \log \frac{\hat{p}(c|w)}{p(c|w)} \to \max_{U,V} \quad (2)$$

KL-дивергенция и её свойства

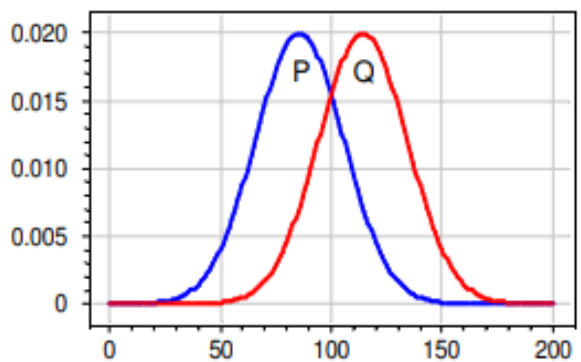
Мера расстояния между распределениями $P = \{p_i\}_{i=1}^s$ и $Q = \{q_i\}_{i=1}^s$.

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

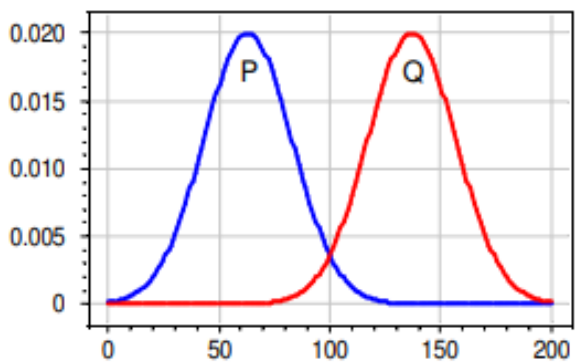
1. $KL(P\|Q) \geq 0$
2. $KL(P\|Q) = 0 \Leftrightarrow P = Q$
3. $KL(P\|Q)$ — мера вложенности P в Q



$KL(P\|Q) = 0.44$
$KL(Q\|P) = 2.97$



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



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

Skip-gram как count-based метод

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

$$(2) - \sum_{w \in W} n_w KL(\hat{p}(c|w) \| p(c|w)) \to \max_{U,V} \quad \Leftrightarrow$$

$$\Leftrightarrow \sum_{w \in W} n_w KL(\hat{p}(c|w) \| p(c|w)) \to \min_{U,V}$$

Обратите внимание! Skip-gram схожа с тематической моделью PLSA, обученной по документам, составленным по совстречаемостям слов[1].

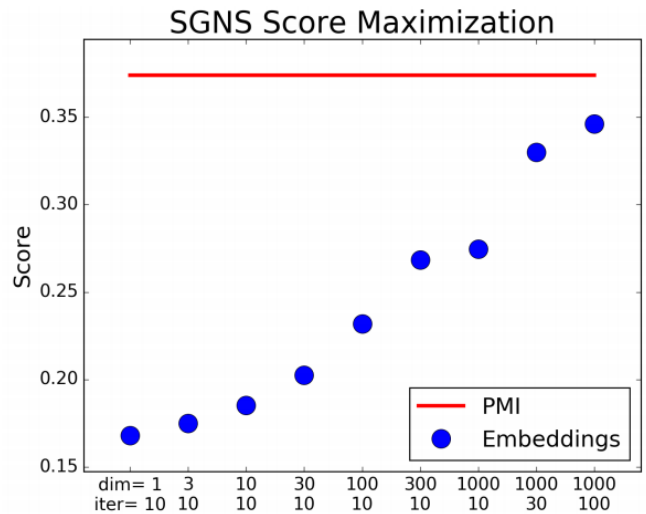
[1] Potapenko et al (2017). Interpretable probabilistic embeddings: bridging the gap between topic models and neural networks

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]

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

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

Резюме по word2vec

+ Относительно неплохое 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.

Два возможных решения проблемы:

- С самого начала учим модель, работающую с символами или символьными n-граммами.
- По обученным векторам учим функцию, восстанавливающую вектор по символьной информации.

Модель представлений FastText

FastText[1] — поиск представлений для буквенных нграмм.

`where = _wh + whe + her + ere + re_`

В Skip-gram меняется только подсчёт вектора v_w:

$$v_w = \sum_{g \in G_w} g_w, \quad G_w \text{ — нграммы для w}$$

$$p(w|w_i) = \underset{w \in W}{\text{softmax}} \langle v_w, u_{w_i} \rangle$$

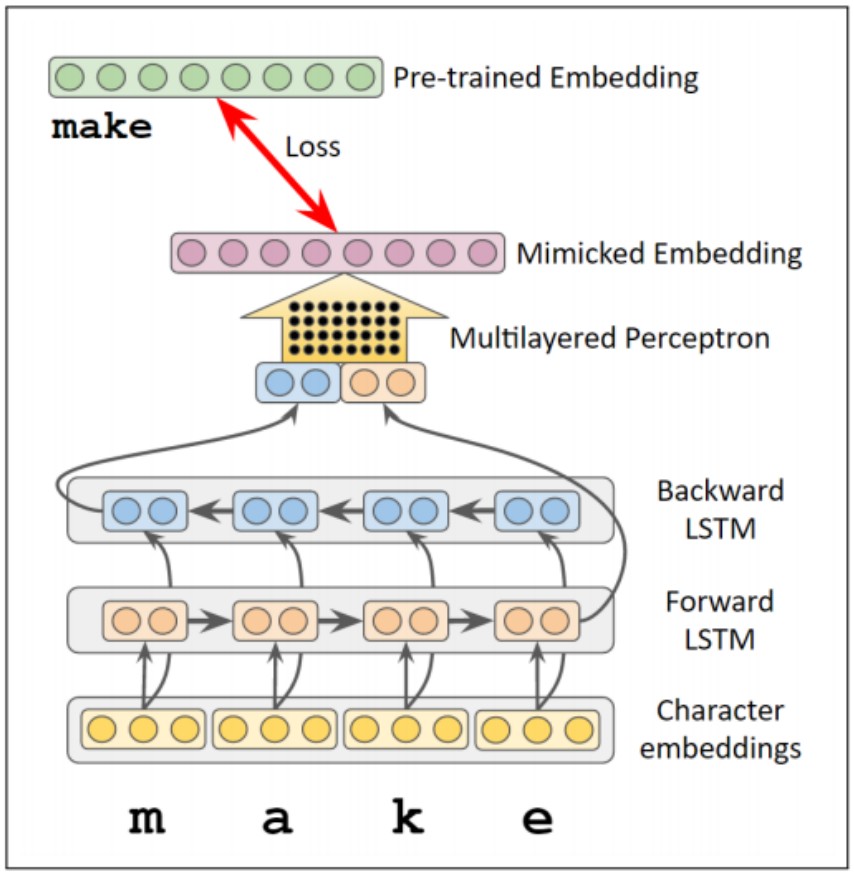
v_w может вычисляться через любую функцию от символов

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

Модель MIMICK для OOV слов

- Исходные данные — матрица эмбеддингов V для слов из W
- $f_\theta(w)$ строит эмбеддинг для w посимвольно
- Обучение f_θ:

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



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

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 RusVectores[2].

Для Lurkmore обучим модель с нуля с помощью пакета Gensim.

---

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

[2] ruwikiruscorpora-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
    "nkrl_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 слова[1]

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

[1] при 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
яндексяча 0.441

Луркморье

гугл 0.593
google 0.508
гуголь 0.504
rm 0.502
кэш 0.497

король - мужчина + женщина:

королева_NOUN 0.754
королева_ADV 0.672
принц 0.627
королева_ADJ 0.625
король 0.623

император 0.583
королевский 0.555
фараон 0.548
халиф 0.523
герцог 0.523

Instrinic 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

Задача близости

При правильной обработке коллекции count-based не уступают word2vec в задаче близости:

win	Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk
2	PPMI	.732	**.699**	.744	.654
	SVD	.772	.671	**.777**	.647
	SGNS	**.789**	.675	.773	**.661**
	GloVe	.720	.605	.728	.606
5	PPMI	.732	**.706**	.738	**.668**
	SVD	.764	.679	**.776**	.639
	SGNS	**.772**	.690	.772	.663
	GloVe	.745	.617	.746	.631

[1] Levy et. al. Improving distributional similarity with lessons learned from word embeddings, 2015

Задача аналогий

Задачу аналогий word2vec решает существенно лучше:

win	Method	Google Add / Mul	MSR Add / Mul
2	PPMI	.552 / .677	.306 / .535
	SVD	.554 / .591	.408 / .468
	SGNS	.676 / **.689**	.617 / **.644**
	GloVe	.649 / .666	.540 / .591
5	PPMI	.518 / .649	.277 / .467
	SVD	.532 / .569	.369 / .424
	SGNS	.692 / **.714**	.605 / **.645**
	GloVe	.700 / .712	.541 / .599

Есть разные мнения о задаче аналогий, см. статьи.

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

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

Как можно использовать word embeddings?

1. Решать задачи поиска близких слов, синонимов и т.п.
2. Получить представление документа, которое будет использоваться в других задачах машинного обучения
3. Использовать 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 instrinic задачам!

Агрегация векторов для представления документа

- Сумма векторов
- Среднее векторов
- Сумма с tf-idf или idf весами
- Координатный max-pool
- Координатный hierarchical-pool (усреднение соседних по окну слов, затем max-pool)

Очень хороший бейзлайн в любой задаче!

Полезные ссылки

- Gensim — пакет, позволяющий легко работать с различными моделями эмбеддингов
- fasttext — библиотека fasttext для обучения эмбеддингов fasttext с нуля
- Wikipedia2Vec — эмбеддинги для разных языков
- RusVectores — сайт с эмебеддингами на русском языке
- StarSpace — ещё одна модель эмбеддингов, позволяющая учить их под конечную задачу