Модели векторных представлений слов

Попов Артём, осень 2022

Natural Language Processing

Напоминание: этапы решения NLP-задачи

1. Выбор верной 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
7. Анализ модели и интерпретация ошибок

Простейшее представление документа – мешок слов (BoW):

$v_d = [n_{wd}]_{w \in W}$, W – словарь коллекции

n_{wd} – частота появления слова w в документе d

Другой подход к представлению документа

1. Каждому слову $w \in W$ сопоставим вектор $v_w \in \mathbb{R}^m$ – **векторное представление слова** (**word embedding**), m – размерность пространства эмбеддинга
2. Преставление документа – агрегация представлений слов (например, сумма или среднее)

В модели BoW представление слова w – one-hot вектор, представление документа – сумма представлений входящих в него слов:

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

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

Свойства one-hot представления слова

Какие основные свойства у one-hot представлений?

Свойства one-hot представления слова

Какие основные свойства у one-hot представлений?

\+ 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

Проблема ортогональности one-hot эмбеддингов

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:

$$\rho(s_1, s_2) = \rho(s_1, s_3) = \sqrt{15}$$ (евклидово)

А есть задачи, где объект – одно слово (поиск синонимов)!

Задача построения представлений слов

Дано: $D = \{w_1, \dots, w_N\}$ – текстовая коллекция (конкатенация всех документов в один вектор)

$w_i \in W$ – одно слово коллекции, W – словарь коллекции

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

Какие представления считать хорошими?

- Близкие по смыслу слова имеют близкие по расстоянию вектора
- Небольшая размерность, $m \ll |W|$
- Интерпретируемые арифметические операции в пространстве $\mathbb{R}^m$
- Качество решения итоговой задачи

Поиграем: угадай слово

- рампетка
- корец
- рында

Поиграем: 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.

Поиграем: 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 1 (Harris, 1954). Слова, совстречающиеся с одними и теми же словами, имеют схожее значение.

Формулировка 2 (Firth, 1957). Слово характеризуется словами, с которыми оно совстречается.

слова совстречаются ⇔ слово находится на расстоянии k слов от другого в документе (в окне/контексте длины k):

. . . an efficient method for learning high quality vector . . .

левый контекст, k=2

правый контекст, k=2

Матрицы совстречаемости слов

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

Каноническая матрица совстречаемости слов (co-occurrence matrix):

$X_{wc} = n_{wc}$ – количество совстречаний w и c по всей коллекции

Матрица PMI (pointwise mutual information):

$$X_{wc} = PMI(w, c) = \log\frac{p(w, c)}{p(w)p(c)} = \log\frac{p(w|c)p(c)}{p(w)p(c)} = \log\frac{n_{wc}\,N}{n_c n_w}$$

Матрица SPPMI (shifted positive PMI):

$$\mathrm{X_{wc}} = \max(PMI(w, c)\ - \log k\,, 0)$$

Матрицы совстречаемости слов

- может не быть симметричной (например, матрица совстречаемостей существительных с глаголами)
- вид матрицы сильно зависит от размера окна k
- строка такой матрицы X_w – представление, решающее проблемы ортогональности

Как получить представление размерностью $\mathrm{m} \ll |W|$?

Матрицы совстречаемости слов

- может не быть симметричной (например, матрица совстречаемостей существительных с глаголами)
- вид матрицы сильно зависит от размера окна k
- строка такой матрицы X_w – представление, решающее проблемы ортогональности

Как получить представление размерностью $\mathrm{m} \ll |W|$?

Воспользоваться методами понижения размерности.

SVD разложение для построения представлений

Хотим построить матричное разложение X:

$$X = UV^T, \qquad U \in \mathbb{R}^{|W| \times m}, \qquad V \in \mathbb{R}^{|W| \times m}$$

Используем Truncated SVD разложение:

$$X = U_m \Sigma_m V_m^T, \qquad U = U_m \sqrt{\Sigma_m}, \qquad V = \sqrt{\Sigma_m} V_m$$

Представления слов — строки матриц U или V.

При определённых условиях такой метод показывает хорошее качество на стандартных бенчмарках.

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

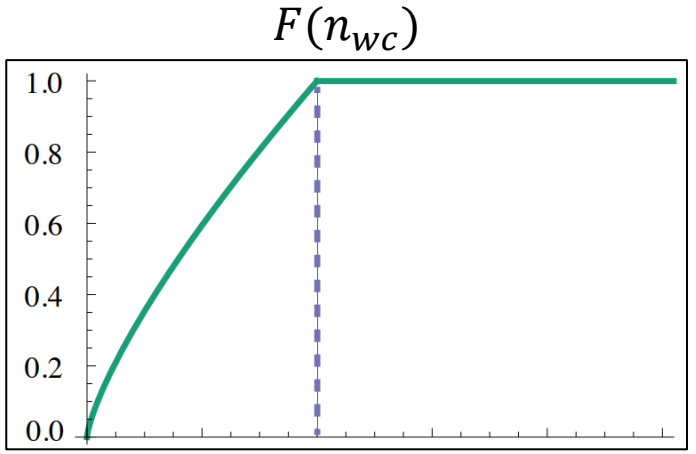
Glove

Методом Adagrad обучается функционал:

$$\mathcal{L}(U,V) = \sum_{w \in W} \sum_{c \in W} F(n_{wc})(\langle v_w, u_c \rangle + b_w + a_c - \log n_{wc})^2 \to \min_{U,V,b,a}$$

Боремся с шумовыми и редкими словами при помощи F:

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



Популярен, но на практике хуже word2vec.

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

Резюме по count-based подходам

+ Неплохое качество в некоторых задачах (но нужно уметь настраивать)

+ Маленькая размерность

± Плотные – нет разреженности

+ Близким словам соответствуют близкие вектора

– Нет хорошего механизма обработки новых слов на тесте

– **Основной минус:** необходимо собирать огромную (но разреженную!) матрицу совстречаемостей для обучения

Prediction-based подход: word2vec

Хотим обновлять параметры модели “на ходу” в процессе просматривания коллекции.

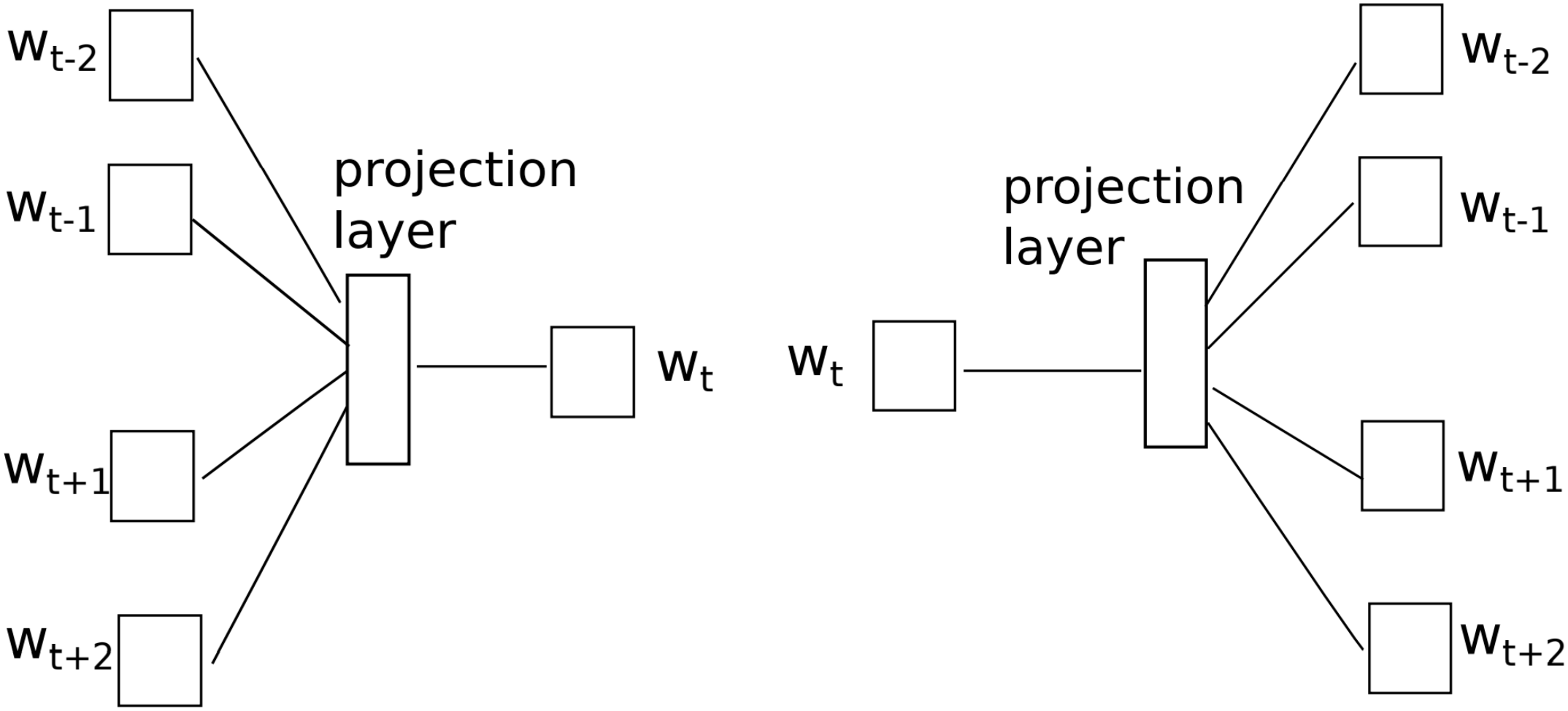
Обучаем модель локально «воспроизводить» гипотезу дистрибутивности:

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

Нам не важно качество решения задачи, нас волнуют параметры, которые получатся в процессе её решения.

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

CBOW vs Skip-gram



Модель CBOW

Обучение — предсказываем центральное слово по контексту:

$$\mathcal{L} = \sum_{i=1}^{N} \log p(w_i | C(i)) \to \max_{U,V} \quad , \qquad U, V \in \mathbb{R}^{|W| \times m}$$

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

Чтобы оценить вероятность, вычисляем вектора контекста и применяем к нему линейный слой с softmax активацией:

$$v^{-i} = \frac{1}{2k} \sum_{w \in C(i)} v_w$$

$$p(w | C(w_i)) = softmax_w\,(Uv^{-i})$$

Rong (2016); word2vec Parameter Learning Explained

Модель Skip-gram

Обучение — предсказываем слова контекста по центральному:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{c \in C(i)} \log p(c|w_i) \to \max_{U,V} \quad , \qquad U, V \in \mathbb{R}^{|W| \times m}$$

$$p(c|w) = softmax_c(Uv_w)$$

CBOW и Skip-gram обучаются с помощью SGD.

Считается, что Skip-gram лучше моделирует редкие слова.

Для представлений обычно используют V, а U забывают.

Какая сложность одной итерации обучения?

Анализ сложности итерации обучения skip-gram

За одну итерацию SGD мы обновляем всю матрицу U:

$$\log p(c|w) = \log softmax_c(Uv_w) = \langle u_c, v_w \rangle - \log \sum_{s \in W} \exp\langle u_s, v_w \rangle$$

$$\frac{d \log p(c|w)}{du_c} = v_w - \frac{v_w \exp\langle u_c, v_w \rangle}{\sum_s \exp\langle u_s, v_w \rangle} = v_w(1 - softmax_c(Uv_w))$$

$$\frac{d \log p(c|w)}{du_t} = -\frac{v_w \exp\langle u_t, v_w \rangle}{\sum_s \exp\langle u_s, v_w \rangle} = v_w(-softmax_t(Uv_w))$$

$$\frac{d \log p(c|w)}{dv_w} = u_c - \frac{\sum_s u_s \exp\langle u_s, v_w \rangle}{\sum_s \exp\langle u_s, v_w \rangle}$$

Итоговая сложность: $O(|W|m)$

Способы ускорения модели

1. Замена softmax на другую функцию, задающую вероятность:
   - **Hierarchical softmax**
   - Differentiated softmax
   - Adaptive softmax
   - Batched softmax
   - …

2. Изменение функционала модели:
   - Noise contrastive estimation
   - **Negative sampling**
   - Importance sampling
   - Self-normalization
   - …

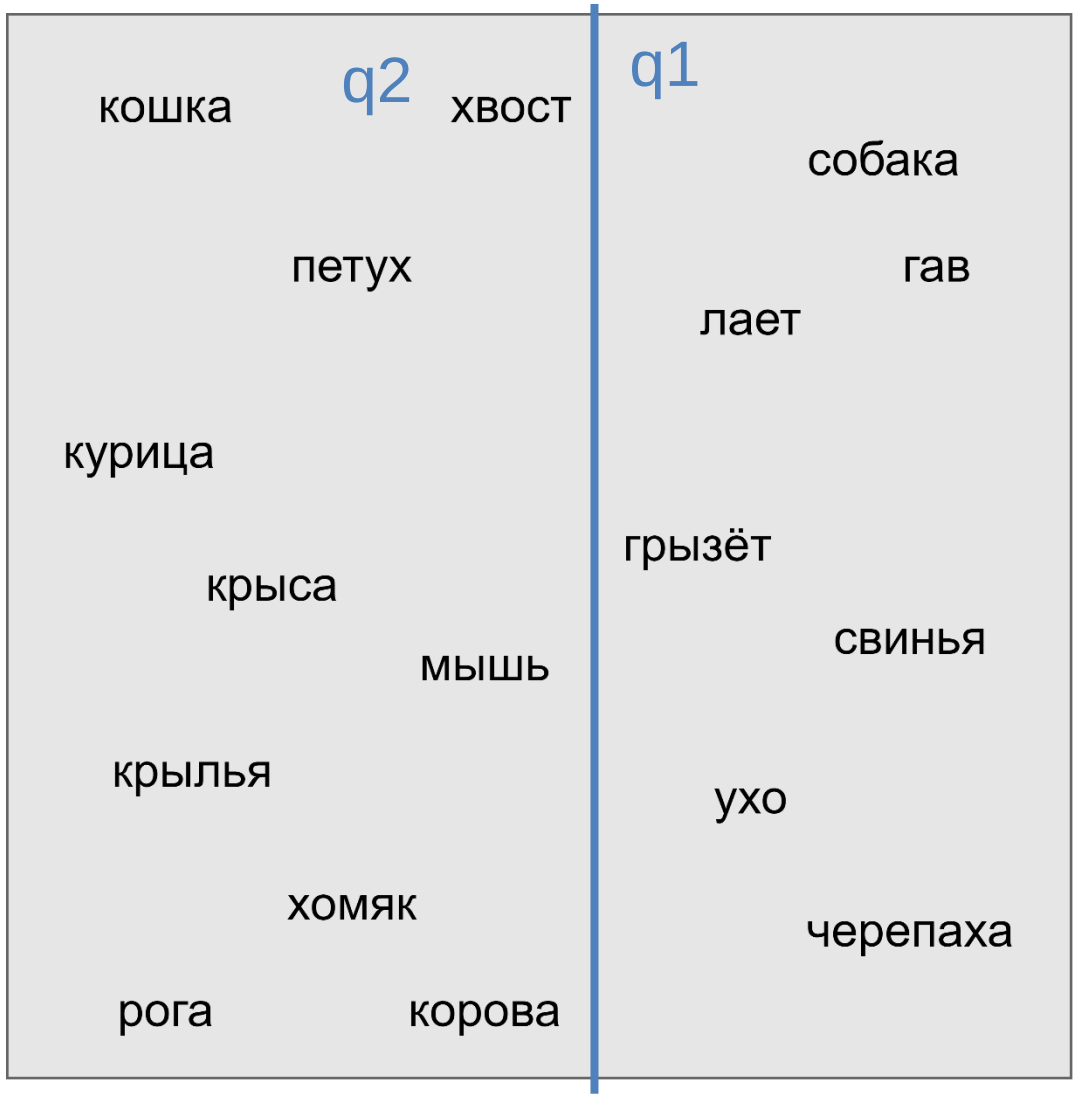
Mikolov (NIPS 2013), Distributed representations of words and phrases and their compositionality

Ruder; On word embeddings - Part 2: Approximating the Softmax;

Иерархический софтмакс: идея

$$p(\text{собака}|w) = ?$$

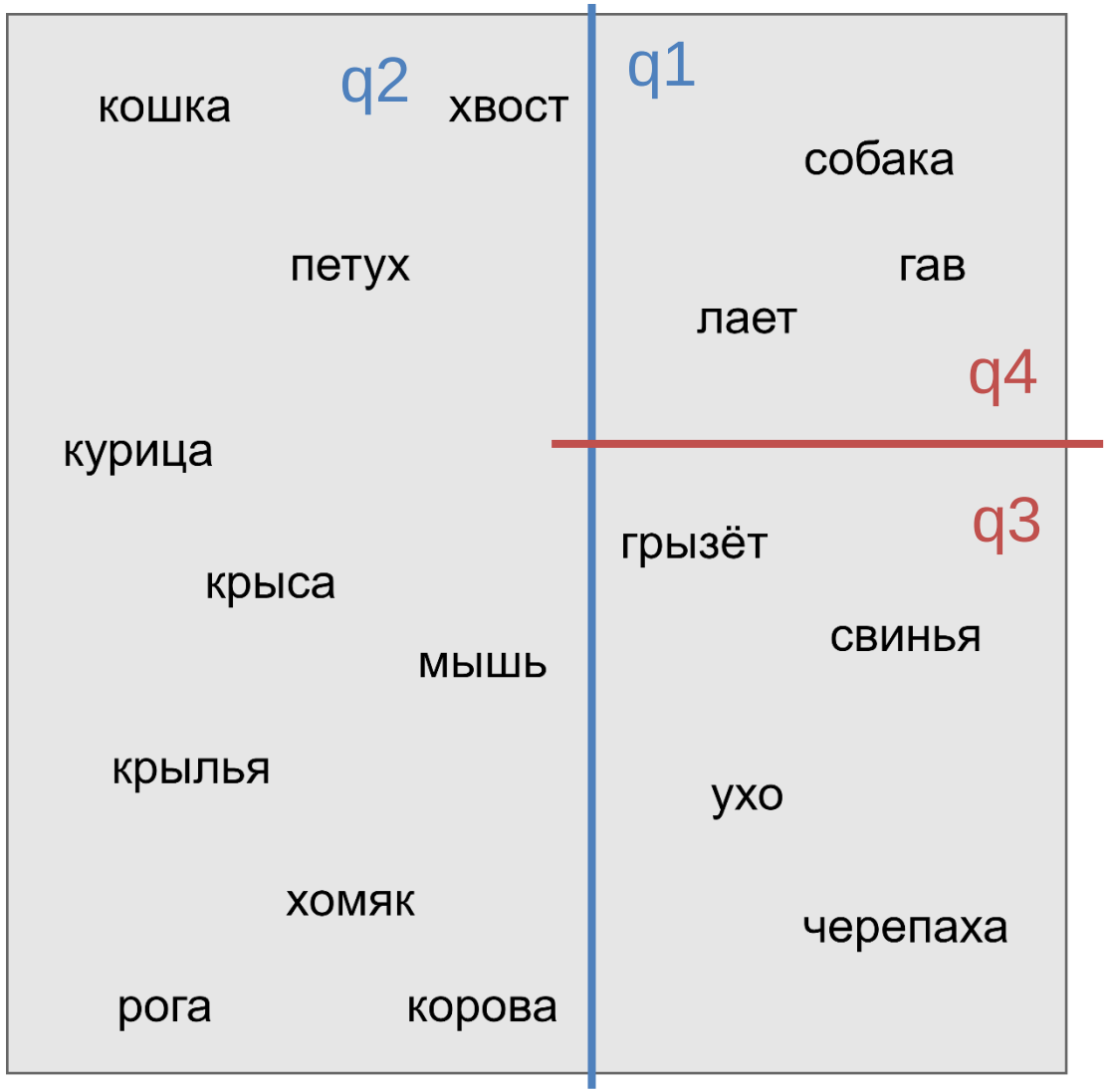
Иерархический софтмакс: идея



$p(\text{собака}|w) = ?$

$p(q_1|w)$ – вероятность, что слово контекста из q_1

Иерархический софтмакс: идея

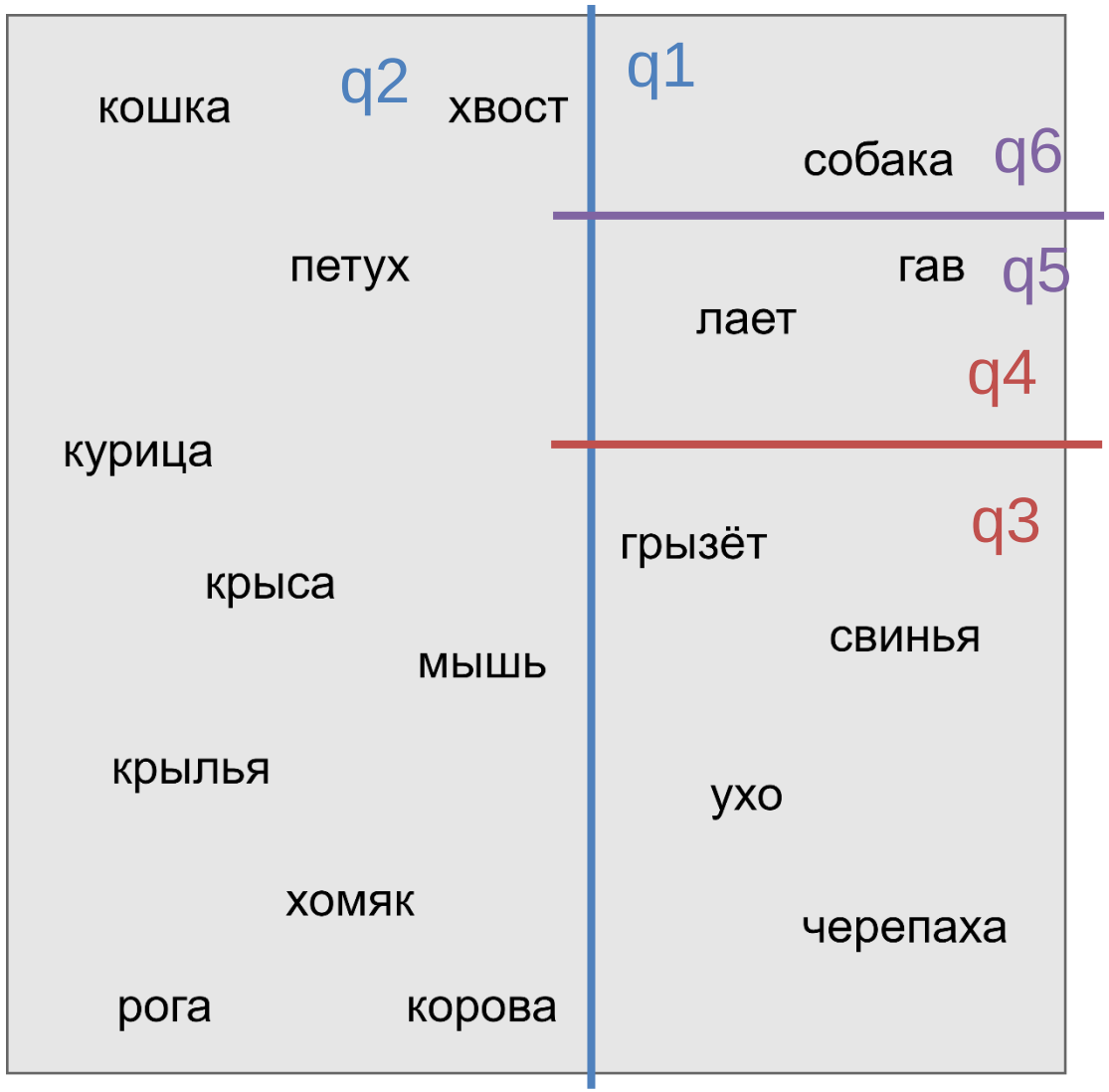


$p(\text{собака}|w) = ?$

$p(q_1|w)$ – вероятность, что слово контекста из q_1

$p(q_4|w, q_1)$ – вероятность что слово контекста из q_4

Иерархический софтмакс: идея



$p(\text{собака}|w) = ?$

$p(q_1|w)$ – вероятность, что слово контекста из q_1

$p(q_4|w, q_1)$ – вероятность что слово контекста из q_4

$p(q_6|w, q_4)$ – вероятность что слово контекста из q_6

$$p(\text{собака}|w) = p(q_1|w) \times p(q_4|w, q_1) \times p(q_6|w, q_4)$$

На что это похоже?

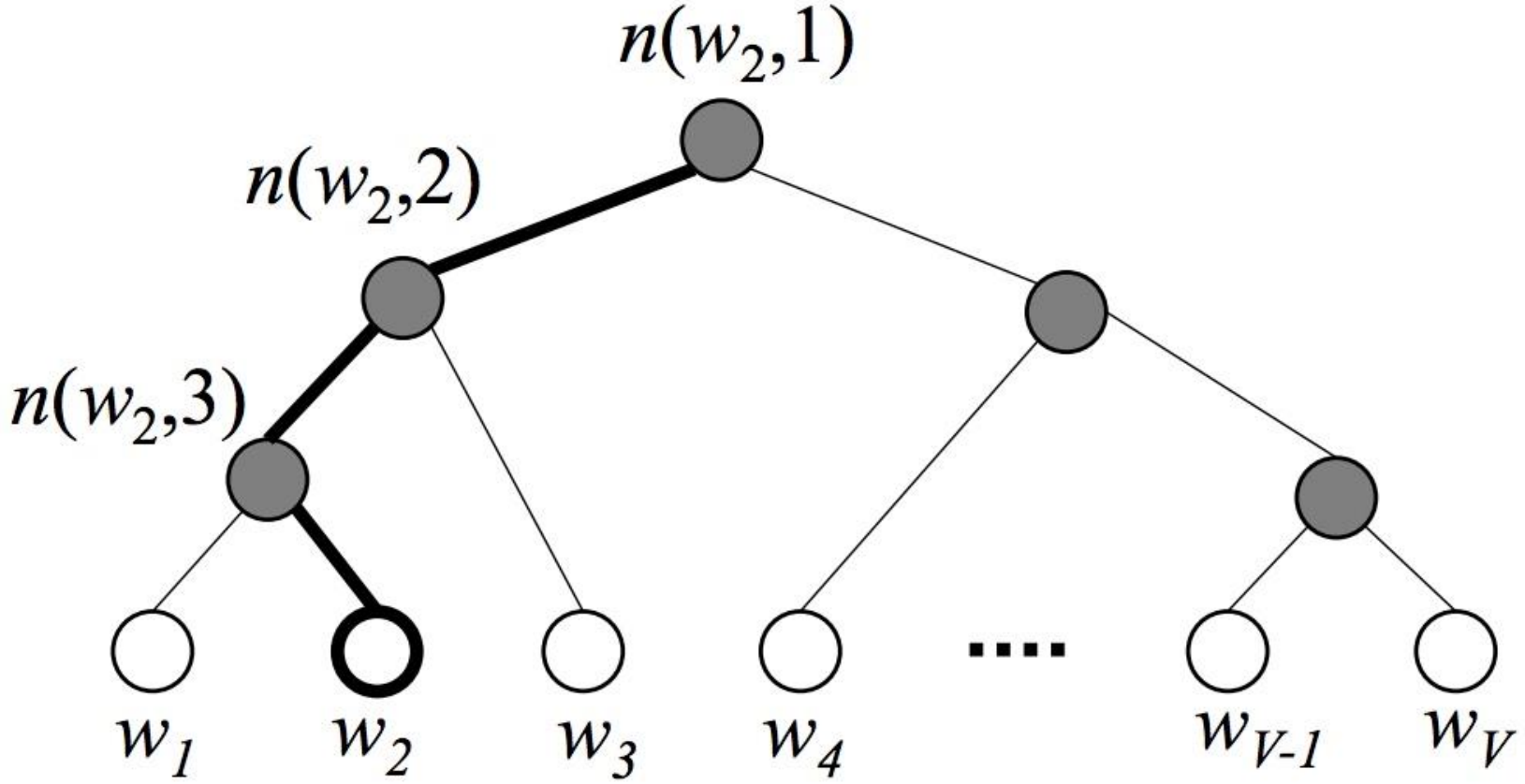
Иерархический софтмакс (hierarchical softmax)

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 листах – искомые представления для слов.
- Чем частотнее слово, тем ближе оно к корню дерева

Пример дерева в HS



Вычисление вероятности в вершинах

Представления внутренних вершин дерева используются для вычисления $p(right|w,n)$ – вероятность того, что слово из контекста w лежит в правом поддереве вершины n.

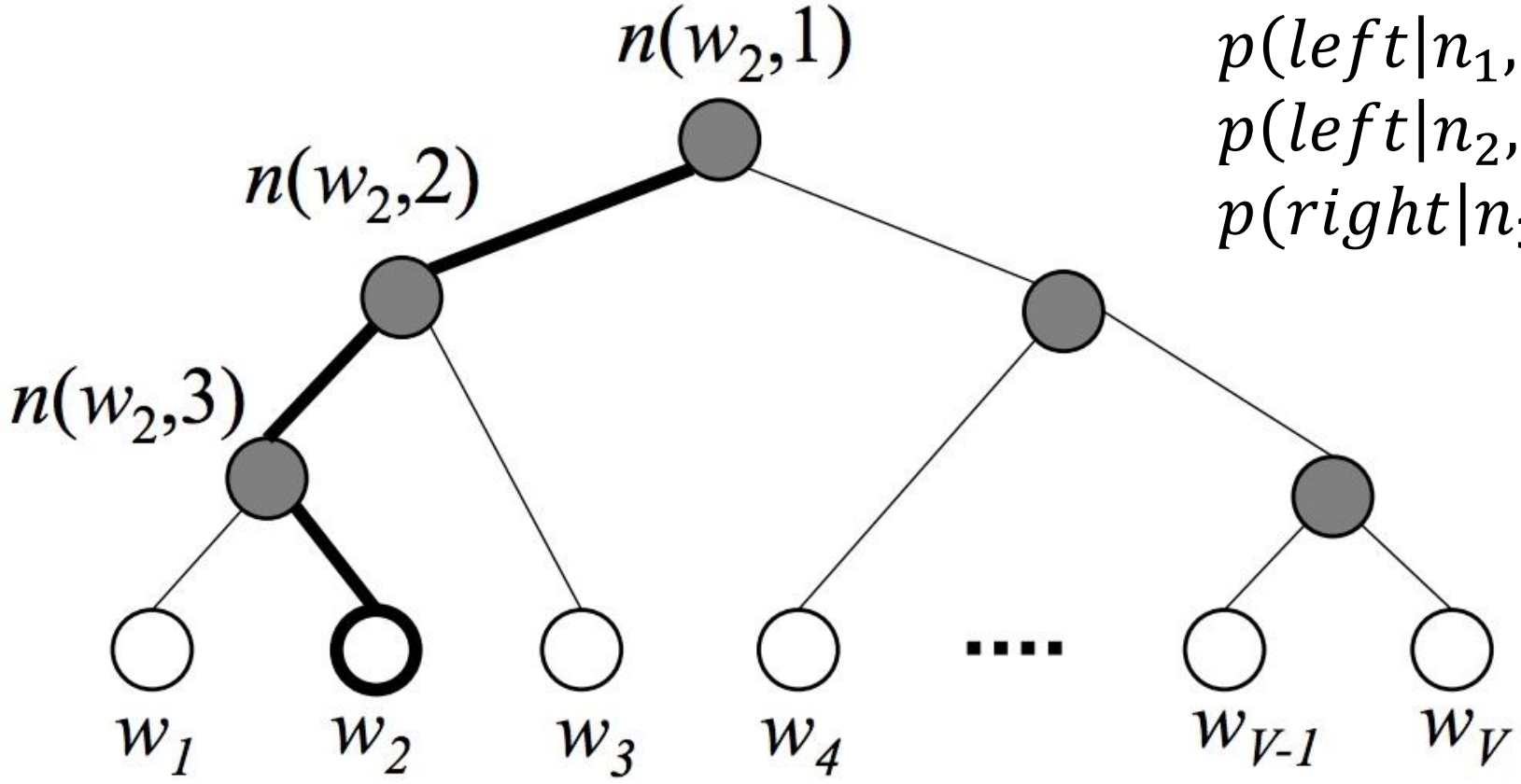
$$p(right|w,n) = \sigma(\langle v_w, u_n \rangle) = 1 - p(left|w,n)$$

Пусть $n(\mathrm{c}) = [n_1(\mathrm{c}); n_2(\mathrm{c}); \dots]$ задаёт путь от корня до слова c.

Будем вычислять вероятность $p(c|w)$ следующим образом:

$$p(c|w) = p(n(c)|w) = \prod_{j \in n(c)} p(\underbrace{n_j(c) \to n_{j+1}(c)}_{\text{right или left}}|w, n_j(c))$$

Пример вычисления вероятности в HS



$$p(w_2|w) = p(left|n_1,w) \times p(left|n_2,w) \times p(right|n_3,w)$$

Негативное сэмплирование (negative sampling)

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

$$p(c|w) = softmax_c(Uv_w)$$

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

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

В чём проблема такой модели?

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

Негативное сэмплирование (negative sampling)

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

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

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 – с вероятностью $1 - t/n_w$ удаляем слово из коллекции
- dynamic window – случайно выбираем размер контекста на каждой итерации обучения
- комбинация представлений – используем 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

± Плотные – нет разреженности

\+ Близким словам соответствуют близкие вектора

– Плохой механизм обработки новых слов на тесте

∓ Требуют большего корпуса чем count-based модели

Проблема OOV

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

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

- игнорирование новых слов
- использование специального UNK токена для редких слов на этапе обучения и для всех новых слов на тесте
- восстановление нового слова по его контексту

Модель представлений FastText

FastText – построение представлений слов как суммы представлений для буквенных n-грамм слова.

В Skip-gram меняется только подсчёт вектора v_w:

$$v_w = \sum_{g \in G(w)} v_g\,, \qquad \text{где } g \text{ – буквенные } n \text{ – граммы}$$

Пример. G(where) = _wh + her + ere + re_

В FastText G(w) – множество хэшей n-грамм для экономии памяти.

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

Генерализация представлений

Дано: матрица V для словаря W

Пусть $f_\theta(w)$ – эмбеддинг слова w по символьной информации:

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

$$f_\theta(w) = LSTM_\theta(w)$$

Для обучения $f_\theta(w)$ не нужна исходная коллекция:

$$\|f_\theta(w) - v_w\|^2 \to \min_\theta$$

Вспомним начало лекции

Какие представления считать хорошими?

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

Эксперимент

Рассмотрим модели, обученные по двум датасетам:

- статьи Википедии + Национальный корпус русского языка
- статьи сайта Lurkmore (3.5К статей)

Для Википедии используем модель с сайта RusVectores.

Для Lurkmore обучим модель с нуля с помощью пакета Gensim.

Идея позаимствована из лекции Мурата Апишева для курса «Анализ Неструктурированных данных» ФКН ВШЭ.

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}

Похожие слова

most_similar(россия_PROPN)

страна 0.695

европа 0.679

российский 0.604

франция 0.582

германия 0.574

most_similar(россия)

ссср 0.759

сша 0.754

германия 0.741

рашка 0.730

грузия 0.719

most_similar(полковник_NOUN)

подполковник 0.904

майор 0.875

генерал 0.805

генерал-майор 0.799

ротмистр 0.770

most_similar(полковник)

генерал 0.648

подполковник 0.647

майор 0.599

генералмайор 0.573

адмирал 0.557

Похожие слова

most_similar(тролль_NOUN)

гном 0.661

троллый 0.656

эльф 0.627

тролли 0.609

гоблин 0.589

most_similar(тролль)

троллинг 0.668

лурко** 0.538

провокатор 0.530

фрик 0.517

быдло 0.516

most_similar(музыка_NOUN)

мелодия 0.702

джаз 0.669

пение 0.649

песня 0.642

танец 0.630

most_similar(музыка)

мелодия 0.668

рэп 0.647

попёс 0.642

песнь 0.641

звук 0.630

Похожие слова

most_similar(мгу_PROPN)

мгу 0.843

лгу 0.773

м::в::ломоносов 0.728

мпгу 0.701

спбгу 0.697

most_similar(мгу)

университет 0.755

вуз 0.665

пту 0.656

мгимо 0.646

аспирант 0.640

most_similar(физтех_PROPN)

физтех_NOUN 0.701

мфти 0.694

мифи 0.632

физтех_DET 0.580

мирэа 0.578

most_similar(физтех)

мехмат 0.537

мифь 0.524

мгимо 0.518

мгу 0.502

филфак 0.496

Арифметические операции (триплеты)

яндекс - россия + сша

гугл 0.518

yahoo 0.467

пентагон 0.464

symantec 0.443

яндексяча 0.441

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

Полезные ссылки

- Gensim — пакет, позволяющий легко работать с различными моделями эмбеддингов (в том числе учить с нуля)
- fasttext — библиотека для обучения fasttext эмбеддингов с нуля
- Wikipedia2Vec — эмбеддинги для разных языков
- RusVectores — сайт с эмебеддингами для русского языка
- StarSpace — любопытная библиотека/модель, позволяющая учить эмбеддинги под конечную задачу

Итоги занятия

- Один из способов строить представление документа – агрегация представлений входящих в него слов
- Count-based подходы позволяют получать плотные представления хорошего качества, но требует подсчёта и хранения матрицы совстречаемостей по коллекции
- Prediction-based подходы позволяют получать плотные представления высокого качества и не требуют подсчёта матрицы совстречаемости
- У классических CBOW и Skip-gram большая вычислительная сложность итерации обучения, нужно использовать HS или NS
- FastText – стандарт для обучения эмбеддингов слов, позволяющий вычислять представления для OOV слов

Бонус!

Если закончили лекцию быстрее…

Skip-gram как count-based метод

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{c \in C(i)} \log p(c|w_i) = \sum_{w \in W} \sum_{c \in W} n_{wc} \log p(c|w) =$$

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

Добавление константы не меняет оптимизационную задачу:

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

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

Skip-gram как count-based метод

Запишем функционал как минимизацию взвешенной KL-дивергенции:

$$\sum_{w \in W} n_w \sum_{c \in W} \frac{n_{wc}}{n_w} \left(\log \frac{\hat{p}(c|w)}{p(c|w)} \right) = \sum_{w \in W} n_w \sum_{c \in W} KL(\hat{p}(c|w) || p(c|w)) \to \min_{V,U}$$

Skip-gram это матричное разложение матрицы $X_{cw} = \hat{p}(c|w)$

Интересный факт. Тематическая модель PLSA имеет схожий функционал при специальном задании коллекции.