Векторные представления слов (word embeddings)

курс «Практикум на ЭВМ», весна 2019

Попов Артём Сергеевич

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

25 марта 2018 г.

Задача построения представлений слов (word embeddings)

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
- Интерпретируемые арифметические операции в пространстве $\mathbb{R}^m$
- Качество конечной задачи.

Гипотеза дистрибутивности

Гипотеза дистрибутивности (Harris, 1954):
слова, которые встречаются рядом с одними и теми же словами, имеют схожее значение

встречаются рядом $\Leftrightarrow$ встречаются в окне k:

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

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

Матричные разложения для построения представлений

- $X \in \mathbb{R}^{|W| \times |W|}, \quad X_{uw} = f(u, w, D)$,
  например $X_{uw} = n_{uw}$.
- n_{uw} — сколько раз слова u и w встретились вместе
- $U \in \mathbb{R}^{|W| \times m}$ — матрица представлений
- $V \in \mathbb{R}^{|W| \times m}$ — матрица доп. представлений

Хотим построить матричное разложение X:

$$X = UV^T$$

Способы построения разложения:

Матричные разложения для построения представлений

- $X \in \mathbb{R}^{|W| \times |W|}, \quad X_{uw} = f(u, w, D)$,
  например $X_{uw} = n_{uw}$.
- n_{uw} — сколько раз слова u и w встретились вместе
- $U \in \mathbb{R}^{|W| \times m}$ — матрица представлений
- $V \in \mathbb{R}^{|W| \times m}$ — матрица доп. представлений

Хотим построить матричное разложение X:

$$X = UV^T$$

Способы построения разложения:

- SVD
- Glove
- TopicModel (разложение через KL)

SVD разложение

SVD разложение PPMI:

$$X_{wc} = PPMI(w, c) = \max\left(\log \frac{n_{wc}|D|}{n_w n_c}, 0\right),$$

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

Glove:

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

$$F(n_{wc}) = \begin{cases} (n_{wc}/n_{max})^{\epsilon}, & n_{wc} < n_{max} \\ 1, & \text{иначе} \end{cases}$$

Семейство моделей word2vec

Обозначения далее:

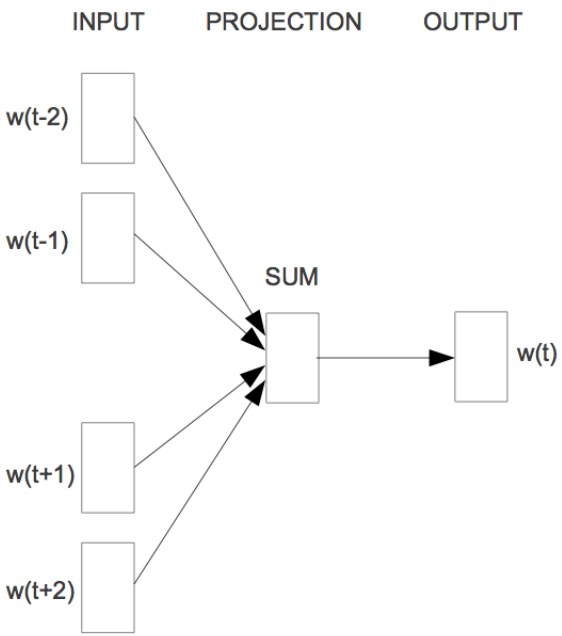
$$\ldots \text{an efficient } \underbrace{\text{method for}}_{\text{контекст, k} = 2} \text{ learning } \underbrace{\text{high quality}}_{\text{контекст, k} = 2} \text{ vector } \ldots$$

- W — множество всех слов, $|W|$ — мощность множества
- Слово w_i — вектор $[0, \ldots, 0, 1, 0, \ldots, 0]$ длины $|W|$
- последовательность $(w_{i+1}, w_{i+2}, \ldots, w_{i+n})$ — w_{i+1}^{i+n}
- $v_w \in \mathbb{R}^m$ — представление слова
- $u_w \in \mathbb{R}^m$ — дополнительное представление слова
- Если $f(w)$ — скалярная функция, то:

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

Модель CBOW

Идея: по словам контекста предсказать центральное слово



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

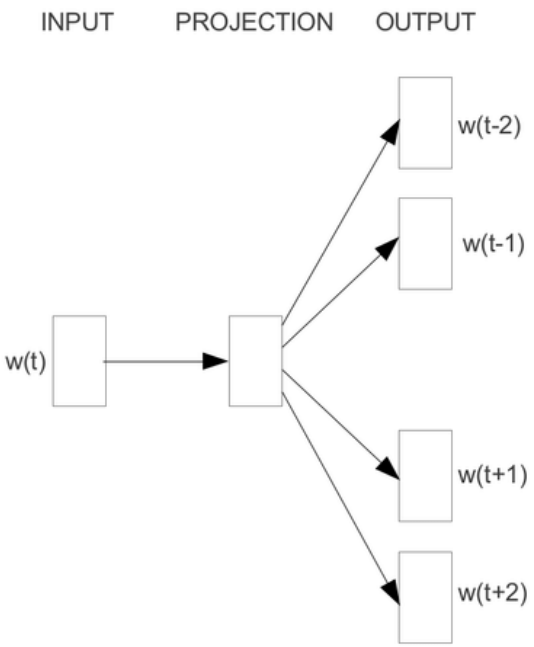
$$u^{-i} = \sum_{\substack{j=-k,\\ j\neq 0}}^{k} u_{w_{i+j}}$$

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

[1] T. Mikov, K. Chen, G. Corrado, J. Dean; Efficient Estimation of Word Representations in Vector Space; 2013

Модель Skip-gram

Идея: по центральному слову предсказать слова из контекста



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

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

- Лучше CBOW для моделирования редких слов
- На практике намного медленнее CBOW
- Сложность итерации SGD для модели — $O(|W|m)$

Skip-gram как count-based метод

Skip-gram можно записать как count-based метод:

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

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

Добавление константы не меняет задачи оптимизации:

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

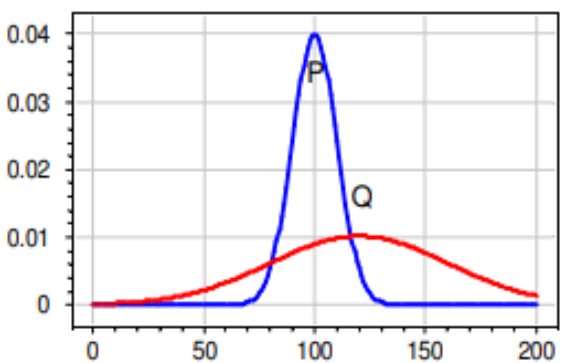
$$= - \sum_{w \in W} n_w \sum_{c \in W} \hat{p}(c|w) \log \frac{\hat{p}(c|w)}{p(c|w)} \to \max_{U,V}$$

KL-дивергенция и её свойства

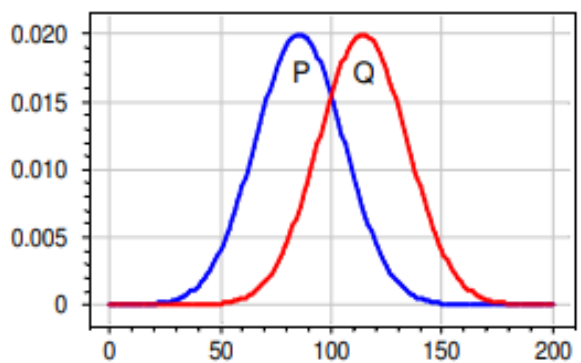
Мера расстояния между распределениями $P = \{p_i\}_{i=1}^s$ и $Q = \{q_i\}_{i=1}^s$.

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

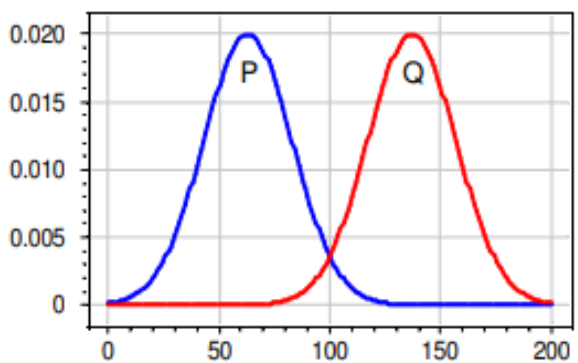
1. $KL(P\|Q) \geq 0$
2. $KL(P\|Q) = 0 \Leftrightarrow P = Q$
3. $KL(P\|Q)$ — мера вложенности P в Q



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



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



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

Skip-gram как count-based метод

Функционал можно записать как минимизацию суммы KL:

$$-\sum_{w\in W} n_w \sum_{c\in W} \hat{p}(c|w)\log\frac{\hat{p}(c|w)}{p(c|w)} =$$

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

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

Таким образом, в модели skip-gram тоже строится матричное разложение.

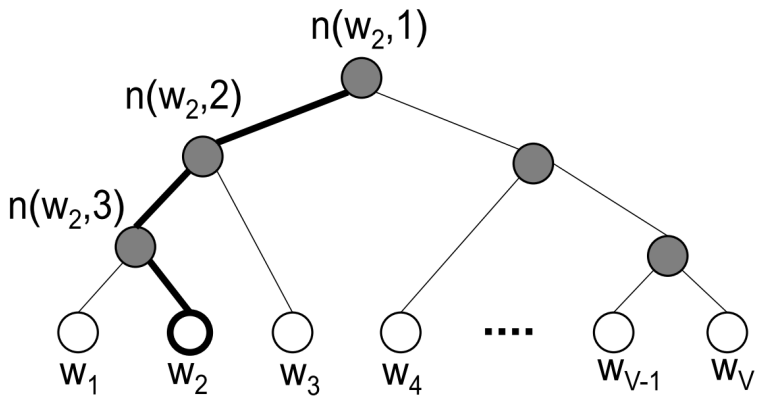
Способы ускорения модели

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

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

Hierarchical softmax (Иерархический мягкий максимум)

- Софтмакс вычисляется как путь в дереве Хафммана.
- Для каждой вершины задано представления
- В листах дерева находятся представления для слов



Пусть $n(w)$ задаёт путь до слова w, $n(w) = [n_1, n_2, n_3, \ldots]$

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

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

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:

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

[1] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean; Distributed Representations of Words and Phrases and their Compositionality; 2013

Infrequent Normalization

Заменим сумму по всем объектам на сумму по сэмплам:

$$\underset{w \in W}{\operatorname{softmax}} f(w) \approx \frac{\exp(f(w))}{\exp(f(w)) + \sum_{w' \in W'} \exp(f(w'))}$$

W' сэмплируется на каждой итерации

Простой способ выбрать W' — взять все слова в батче

[1] J. Andreas, D. Klein; When and why are log-linear models self-normalizing?; 2015

[2] W. Chen, D. Grangier, M. Auli; Strategies for Training Large Vocabulary Neural Language Models; 2016

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 размер контекста на каждой итерации
- Комбинация итоговых векторов — использовать в качестве представления $\alpha v_w + (1 - \alpha u_w)$

Практические рекомендации:

- На небольших датасетах нужно использовать больше негативных сэмплов
- Если документы специфичные, лучше учить модель с нуля, а не использовать предобученные

Агрегация векторов для представления документа

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

Очень хороший бейзлайн в любой задаче!

Модель представлений FastText

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

FastText — поиск представлений для буквенных нграмм.

`where = whe + her + ere`

В Skip-gram меняется только подсчёт вектора v_w:

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

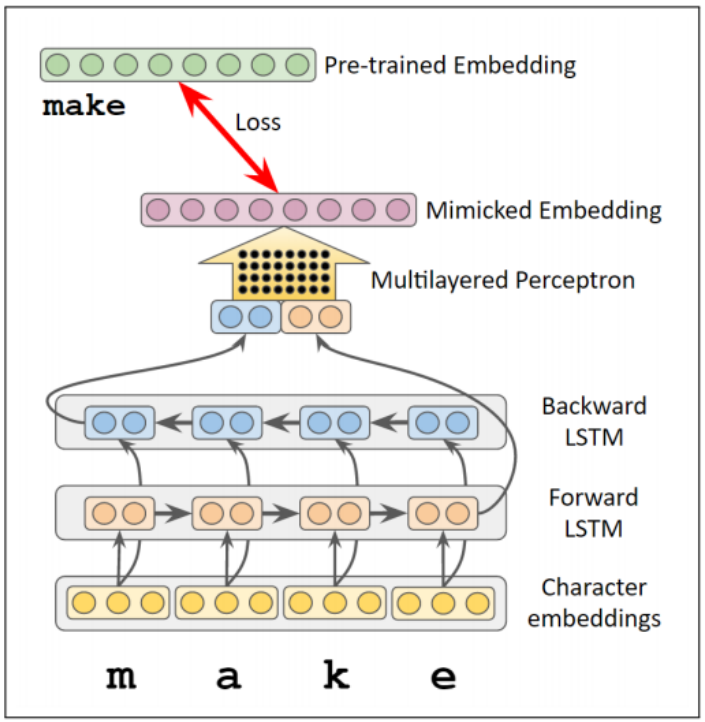
$$p(w|w_i) = \underset{w \in W}{\text{softmax}} \langle v_w, u_{w_i} \rangle$$

[1] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov; Enriching Word Vectors with Subword Information; 2016

Модель MIMICK для OOV слов

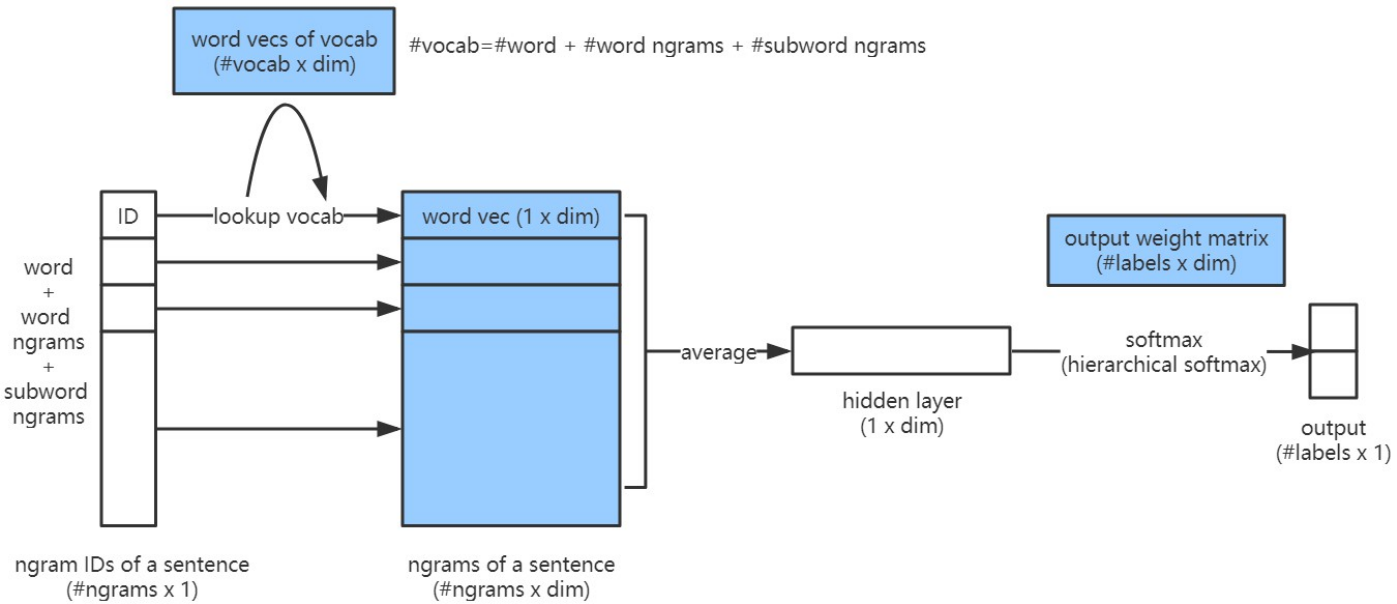
- $f_\theta(w)$ строит векторное представление для w посимвольно
- MSE для предобученных векторов и выходов f_θ:

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



Модель классификации FastText

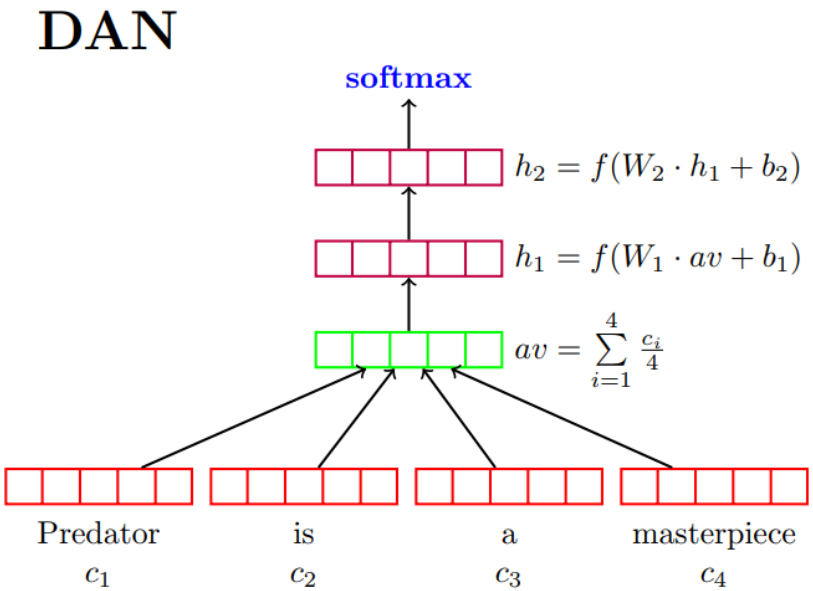
Идея: end-to-end линейная классификация + эмбединги



1. По входному предложению строится набор нграмм
2. Усредняются представления нграмм
3. Применяется линейный слой с softmax активацией

[1] A. Joulin et al; Bag of Tricks for Efficient Text Classification; 2016

Deep Averaging Network



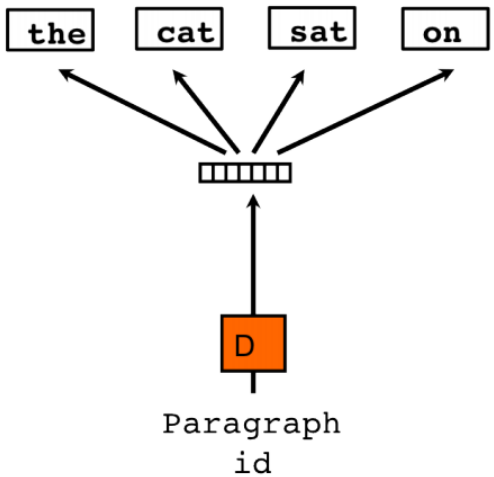
Чем DAN отличается от FastText:

1. Больше одного линейного слоя
2. Аналог dropout при усреднении

[1] M. Iyyer, V. Manjunatha, J. Boyd-Graber, H. Daume III; Deep Unordered Composition Rivals Syntactic Methods for Text Classification; 2015

Модель представлений paragraph2vec (doc2vec)

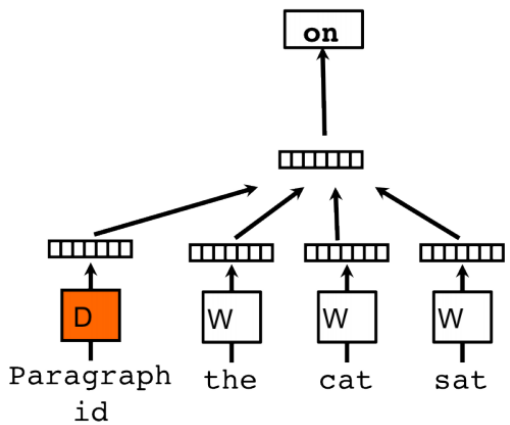
P-DBOW (paragraph distributed bag of words)



$$\sum_{d \in D} \sum_{i=1}^{n_d} \log p(w_i|d) \to \max_{V,U}$$

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

PV-DM (paragraph vectors distributed model)



$$\sum_{d \in D} \sum_{i=1}^{n_d} \log p(w_i|w_{i-k}^{i-1}, w_{i+1}^{i+k}, d) \to \max_{U,V}$$

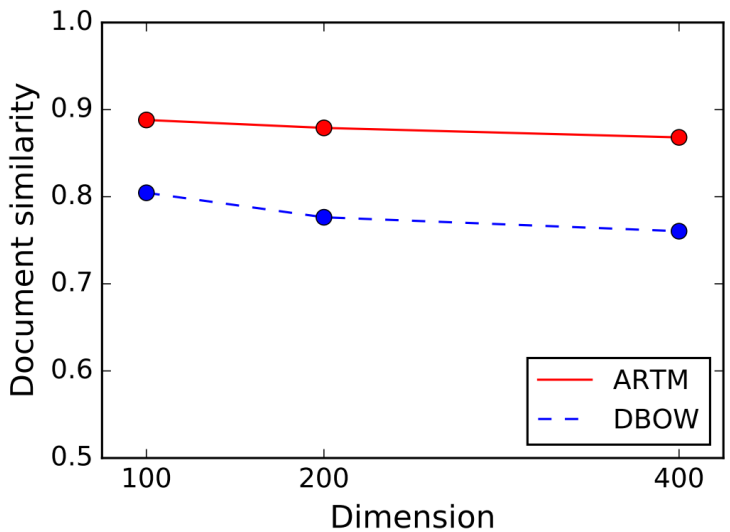
$$p(w|\ldots) = \underset{w \in W}{\text{softmax}} \left\langle v_w, u^{-i} + u_d \right\rangle$$

[1] Distributed Representations of Sentences and Documents; 2014

Задача поиска статей

Задача близости триплетов (датасет стаей arxiv.org)

Вход: список троек документов $d_1, d_2, d_3 : \ d_1 \sim d_2, \ d_1 \nsim d_3$
Проверим свойство $sim(d_1, d_2) > sim(d_1, d_3)$ для модели
Выход: Доля правильных ответов



модель	результат
Topic Model	0.88
P-DBOW	0.802
Skip-gram (усред.)	0.867

Некоторые практические рекомендации

- Никогда не использовать paragraph2vec
- Учить свои FastText представления только если есть большая обучающая выборка
- Можно модифицировать модели, подставляя внутрь сложный энкодер

Измерение качества

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

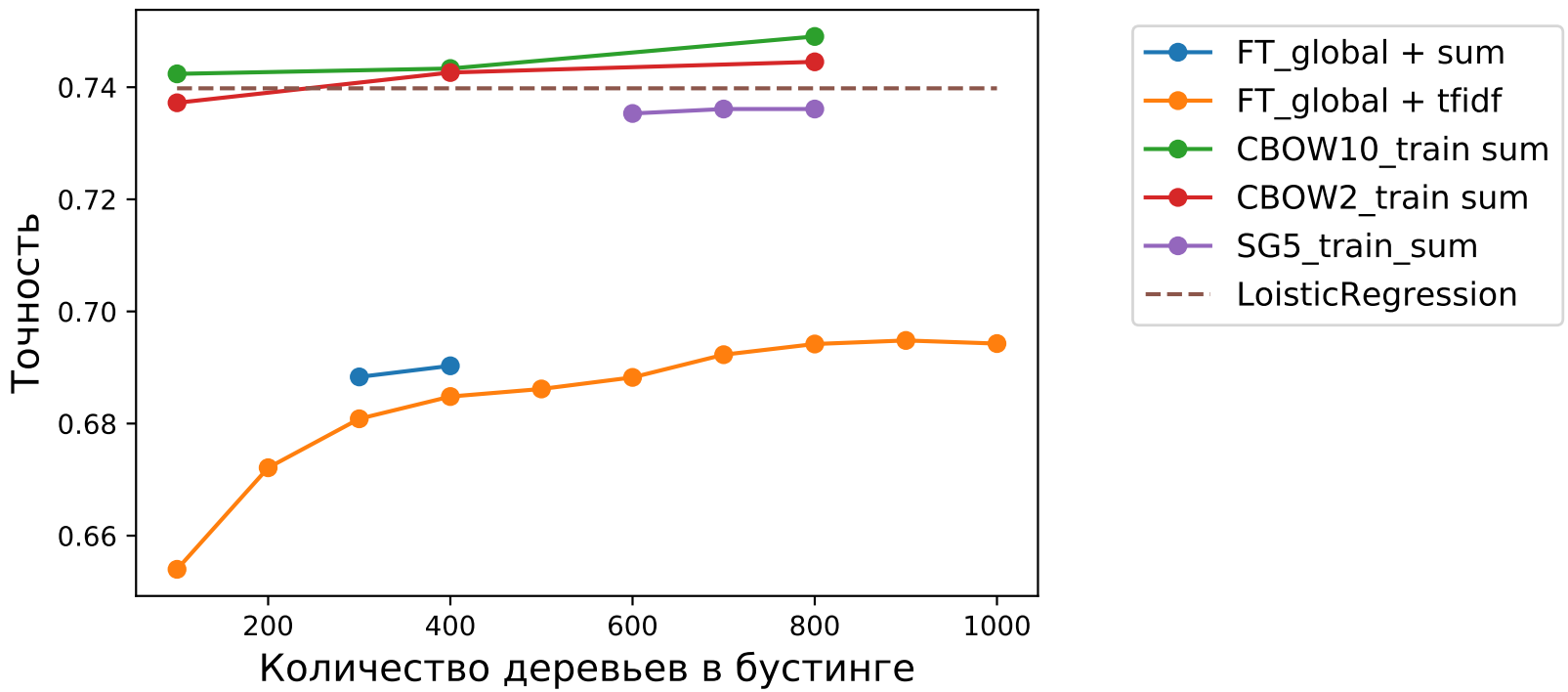
Задачу аналогий word2vec решает существенно лучше:

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

Задача классификации (отзывы клиентов супермаркета)

Вход: коллекция отзывов, 50584 документов, 17 классов

Качество на отложенной выборке:



На специфичных коллекциях предобученные эмбединги могут плохо работать

Зависимость от размерности

CBOW на маленьких коллекциях часто лучше skip-gram: