ТЕХНОСФЕРА

Лекция 3
Текстовая релевантность

Владимир Гулин

28 февраля 2018 г.

План лекции

Ранжированный поиск

- Поиск, который мы видели до этого был булевым
  - Документ либо подходит, либо нет
- Хорошо подходит для продвинутых экспертов, которые хорошо понимают что им нужно и что они могут найти в коллекции документов
  - Также хорошо подходит для приложений (анализ тысяч результатов)
- Плохо для большинства пользователей
  - Обычный пользователь никогда не будет писать булев запрос
  - Большинство пользователей не хотят просматривать тысячи результатов поиска

Проблемы булева поиска

- Булевы запросы часто возвращают либо слишком мало (=0) либо слишком много результатов (тысячи)
- Запрос 1: “скачать бесплатно” (4 млн. результатов)
- Запрос 2: “скачать бесплатно без регистрации без смс без ключа без хурмы без кидалова без торрента” (0 результатов)
- От пользователя это требудет соответсвующего навыка, чтобы составить запрос, который вернул бы приемлимое количество результатов
  - AND приводит к малому числу результатов
  - OR к слишком большому

Модели ранжированного поиска

- Ранжированный поиск возвращает упорядоченный список документов из коллекции по запросу
- Вместо языка запросов и операторов, используются просто слова из человеческого языка

Вопрос:

- Применяется ли в современных поисковых системах булев поиск?

Схема ранжирования в поиске

Этапы ранжирования

Релевантность

Прежде чем начать...

Вопрос:

- Что такое релеватный документ?

Релевантность

Сложное комплексное понятие, учитывающее множество факторов и зачастую крайне субьективное

В информационном поиске релеватность рассматривается с нескольких сторон

- тематическая релеватность
- пользовательская релеватность
- текстовая релеватность
- ...

Ранжированный поиск

Когда поисковая система возвращает ранжированный список результатов, большой объем не является проблемой

- Количество найденных результатов не является проблемой для пользователя
- Мы показываем только top $k(\approx 10)$ результатов
- Таким образом, мы не огорчаем пользователя

Предположение:

- Алгоритм ранжирования работает хорошо :)

Ранжированный поиск

Оценка релеватности документа

- Хотим вернуть документы, в порядке наиболее полезных для пользователя
- Каким образом мы можем составить такой порядок в соотвествии с запросом?
- Назначим каждому документу оценку (score) для каждого документа по запросу (например из $[0, 1]$)
- Эта оценка должна отражать на сколько хорошо документ подходит запросу

Вычисление веса

- Нужен способ назначения веса паре запрос-документ
- Начнем с запроса из одного термина
- Если термина нет в документе, то вес равен 0
- Чем чаще встречается термин в документе, тем выше вес

Модель мешка слов

- Не учитывается порядок слов в документе
- John is quicker than Mary и Mary is quicker than John
- Вася быстрее Маши и Маша быстрее Васи
- НО Мать любит дочь и Дочь любит мать (не ясно кто кого любит)
- Такая модель называется моделью мешка слов
- Это шаг назад, так как координатный индекс может различить такие докуенты
- Вернемся к использованию координатной информации позже

Частота термина

- Частота $tf_{t,d}$ термина t в документе d определяется как количество раз, сколько t встречается в d
- Хотим использовать tf при расчете весов. Но как?
- Просто частота не торт!
  - Документ с 10 вхождениями релевантнее документа с 1 вхождением (в 10 раз!!!).
- Релеватность не увеличивается пропорционально частоте

Логарифмическое взвешивание

- Логарифмическая частота термина t в d:

$$w_{t,d} = \begin{cases} 1 + log\ tf_{t,d}, & \text{if } tf_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- Вес для пары запрос-документ: сумма по всем терминам t, входящим в q и d:
- Вес

$$= \sum_{t \in q \cap d} (1 + log\ tf_{t,d})$$

- Вес равен 0, если в документе нет ни одного термина из запроса.

Документная частота

- Редкие термины информативнее частотных (стоп-слова)
- Рассмотрим термин запроса, который редко встречается в корпусе (например *“серобуромалиновый”*)
- Любой документ, содержащий в себе этот термин, скорее всего будет релевантен запросу *“серобуромалиновый”*
- То есть, имеет смысл давать больший вес редким терминам

Обратная документная частота

IDF

- df_t - документная частота термина t (количество документов, содержащих t)
- df_t - обратная мера инфортивности t
- $df_t \leq N$
- Опредилим idf (inverse document frequency) термина как

$$idf_t = log \frac{N}{df_t}$$

Пример idf

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

$$idf_t = log \frac{N}{df_t}$$

- $N = 10^6$
- Для каждого термина в корпусе только одно значение idf

IDF в ранжировании

Значим ли IDF для запросов для одного термина

- IPhone

IDF не влияет на однословные запросы

- Idf влияет на ранжирование запросов из двух и более слов
- Для запросов типа “серобуромалиновые штаны”, взвешивание по IDF приводит к большому вкладу термина “серобуромалиновый”, чем термин “штаны’

Взвешивание TF-IDF

- Вес термина $tf - idf$ это произведение его весов tf и idf:

$$w_{t,d} = (1 + log\ tf_{t,d}) \cdot log \frac{N}{df_t}$$

- Самая известная модель взвешивания в информационом поиске
- Растет с ростом числа вхождений слова в документ
- Растет со степенью редкости термина

Ранжирование по TF-IDF

$$Score(q, d) = \sum_{t \in q \cap d} tf.idf_{t,d}$$

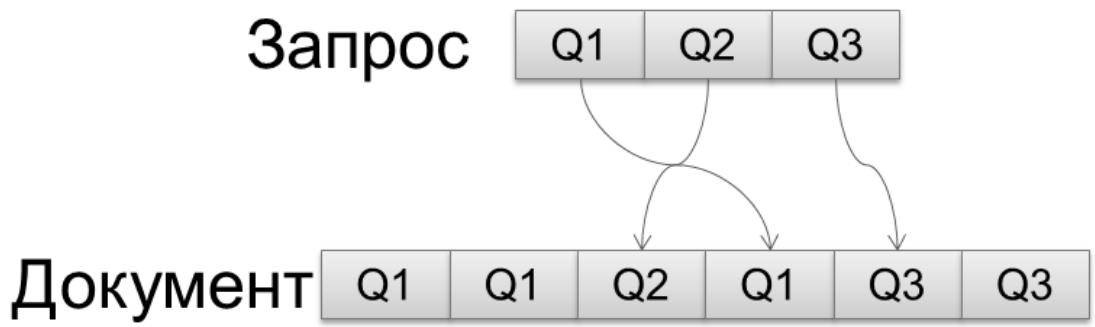
N-gramm TF-IDF

- Ничего не мешает нам аналогично с вхождениями слов рассматривать вхождения n-грамм
- А еще мы их можем взять с разным весом

$$Score(q, d) = \sum_{n} \alpha_n \sum_{t_n \in q \cap d} tf.idf_{t_n, d}$$

Пассажный алгоритм

Пассаж - фрагмент документа, размера, не превышающего заданный, в котором встречаются все термы запроса, либо значительная часть термов запроса, суммарный IDF которых превышает заданное ограничение.



Пассажный алгоритм

Оценка пассажа

- TF-IDF
- Полнота
- Порядок слов
- Правильность словоформ
- Кучность
- Близость к началу
- Особенность зоны документы

Параметрические зоны и индексы

- До сих пор документ представлялся последовательностью терминов
- Обычно документы состоят из нескольких частей, с определённой семантикой
  - Автор
  - Заголовок
  - Дата публикации
  - Язык
  - Формат
  - И т.п.
- Это все метаданные

Компактность вхождений

- Текстовые запросы: набор терминов, введённых в поисковую строчку
- Пользователи предпочитают документы, где термины запроса находятся на небольшом расстоянии друг относительно друга
- Пусть w будет наименьшим окном в документе, содержащим все термины запроса
- Например для запроса [strained mercy] такое окно в документе The quality of mercy is not strained равно 4 (в словах)
- Как учесть это в итоговом scrore?

Как объединить все вместе?

$$Score(q, d) = \sum_n \alpha_n \sum_{t_n \in q \cap d} tf.idf_{t_n,d} + \sum_p score_p(q, d)$$

- Линейная модель
- Первое слагаемое n-граммный TF-IDF
- Второе слагаемое взвешенная сумма пассажных ранков

$$score_p(q, d) = score(pos, proximity, tf.idf, zone \ldots)$$

Векторная модель ранжирования

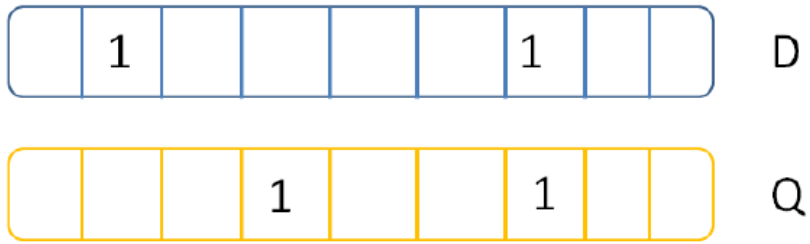
Документы и запросы - это вектора в T мерном пространтсве, где T - общее количетсво термов (словоформ, основ, фраз и т.д.)

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{iT}) \quad Q = (q_1, q_2, \ldots, q_T)$$

Коллекция документов предствляется матрицей.

	$Term_1$	$Term_2$	$\ldots$	$Term_t$
Doc_1	d_{11}	d_{12}	$\ldots$	d_{1t}
Doc_2	d_{21}	d_{22}	$\ldots$	d_{2t}
$\vdots$	$\vdots$			
Doc_n	d_{n1}	d_{n2}	$\ldots$	d_{nt}

Векторная модель ранжирования



Документы ранжируются в соответсвии с схожестью вектора соответствующего запросу и вектора соотвествующего документу

$$cosine(Q, D) = \frac{Q^T D}{\|Q\|\|D\|}$$

Вопрос:

- А почему именно косинус?

Векторная модель

- ✔ простая модель ранжирования
- ✔ можно использовать любую меру схожести векторов
- ✔ можно использовать любую схему взвешивания термов
- ✖ Модель работает в предположении о независимости термов
- ✖ Невозможно определить способ оптимального ранжирования

Вероятностная модель ранжирования

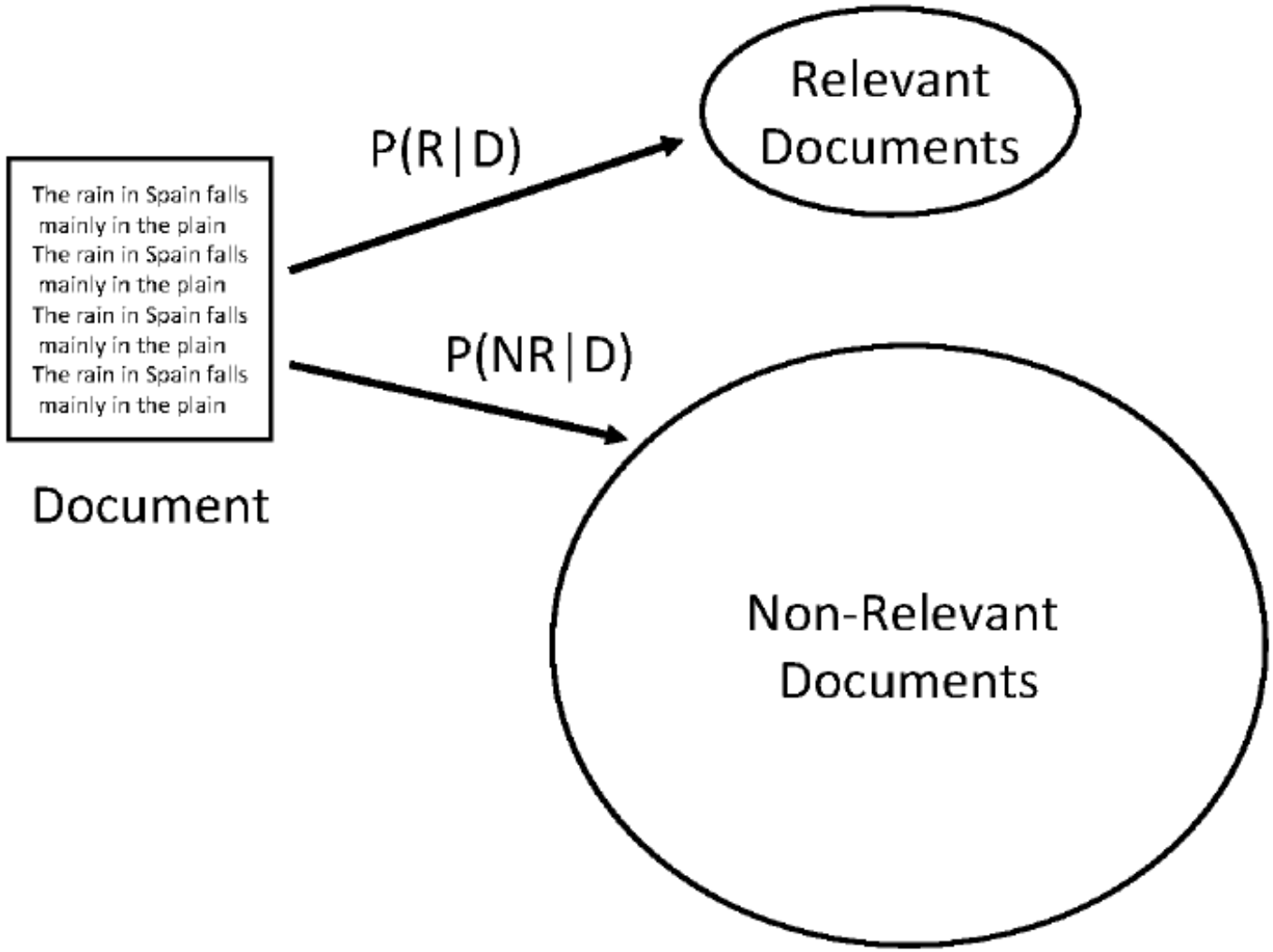
Принцип вероятностного ранжирования

- Если поисковая система в ответ на запрос пользователя отдает документы в порядке уменьшения вероятности их релеватности запросу, то качество такой поисковой системы будет максимальным (Robertson, 1977)

Ключевой вопрос:

- Какова вероятность того, что пользователь оценит данный документ как релеватный для этого запроса?

Решаем задачу бинарной классификации



Байесовский классификатор

Оптимальное решающее правило

- Документ D релевантен запросу Q, если $P(R = 1|D, Q) > P(R = 0|D, Q)$

Оценка вероятностей

- Воспользуемся формулой Байеса

$$p(R|D) = \frac{P(D|R)P(R)}{P(D)}$$

Модель бинарной независимости

- Документ представляет собой бинарный вектор термов
- Запрос представляет собой бинарный вектор термов
- Предположение о независимости появления термов в документе

$$P(D|R) = \prod_{i=1}^{t} P(d_i|R)$$

Модель бинарной независимости

p_i - вероятность встретить i-ый термин в релевантных документах

s_i - вероятность встретить i-ый термин в нерелевантных документах

$$\frac{P(D|R)}{P(D|NR)} = \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i} =$$

$$= \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \left(\prod_{i:d_i=1} \frac{1-p_i}{1-s_i} \cdot \prod_{i:d_i=1} \frac{1-s_i}{1-p_i} \right) \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i} =$$

$$= \prod_{i:d_i=1} \frac{p_i(1-s_i)}{s_i(1-p_i)} \cdot \prod_{i} \frac{1-p_i}{1-s_i}$$

Модель бинарной независимости

- Оценка отношения правдоподобия

$$\sum_{i:d_i=1} log \frac{p_i(1-s_i)}{s_i(1-p_i)}$$

- Запрос несет информацию о релеватных документах
- Если нет никакой дополнительной информации, то пологаем, что p_i постоянна, а s_i можно оценить по коллекции

$$log \frac{0.5(1-\frac{n_i}{N})}{\frac{n_i}{N}(1-0.5)} = log \frac{N-n_i}{n_i}$$

Модель бинарной независимости

Честная оценка

	Relevant	Non-relevant	Total
$d_i = 1$	r_i	$n_i - r_i$	n_i
$d_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - r_i$
Total	R	$N - R$	N

$$p_i = (r_i + 0.5)/(R + 1)$$

$$s_i = (n_i - r_i + 0.5)/(N - R + 1)$$

- Функция оценки:

$$\sum_{i:d_i=q_i=1} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)}$$

BM25

Классический алгоритм ранжирования, основанный на модели бинарной независимости

$$\sum_{i \in Q} log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1)f_i}{K + f_i} \cdot \frac{(k_2 + 1)qf_i}{k_2 + qf_i}$$

- значения k_1, k_2, K подбираются эмпирически

$$K = k_1((1 - b) + b \cdot \frac{dl}{avgdl})$$

- На TREC $k_1 = 1.2$, $k_2 \in [0, 1000]$, $b = 0.75$

BM25

Пример:

- Запрос: “president lincoln” (qf = 1)
- $r = R = 0$ (нет информации о релевантных документах)
- Размер коллекции $N = 500000$ документов
- Термин “president” содержится в 40000 документов ($n_1 = 40000$)
- Термин “lincoln” содержится в 300 документов ($n_2 = 300$)
- Термин “president” встречается 15 раз в документе ($f_1 = 15$)
- Термин “lincoln” встречается 25 раз в документе ($f_2 = 25$)
- $\frac{dl}{avdl} = 0.9$
- $k_1 = 1.2$, $b = 0.75$, $k_2 = 100$

BM25

Пример:

$$BM25(Q, D) =$$

$$= log \frac{(0 + 0.5)/(0 - 0 + 0.5)}{(40000 - 0 + 0.5)/(500000 - 40000 - 0 + 0 + 0.5)} \times \frac{15(1.2 + 1)}{1.11 + 15} \times \frac{1(100 + 1)}{100 + 1} +$$

$$+ log \frac{(0 + 0.5)/(0 - 0 + 0.5)}{(300 - 0 + 0.5)/(500000 - 300 - 0 + 0 + 0.5)} \times \frac{25(1.2 + 1)}{1.11 + 25} \times \frac{1(100 + 1)}{100 + 1} =$$

$$= log \left[\frac{460000.5}{40000.5} \cdot \frac{33}{16.11} \cdot \frac{101}{101}\right] + log \left[\frac{499700.5}{300.5} \cdot \frac{55}{26.11} \cdot \frac{101}{101}\right] = 20.66$$

BM25F

BM25

$$score(Q, D) = \sum_{i=1}^{n} Idf(q_i) \frac{f(q_i, D)(k_1 + 1)}{f(q_i, D) + k_1(1 - b + b\frac{dl}{avgdl})}$$

BM25F

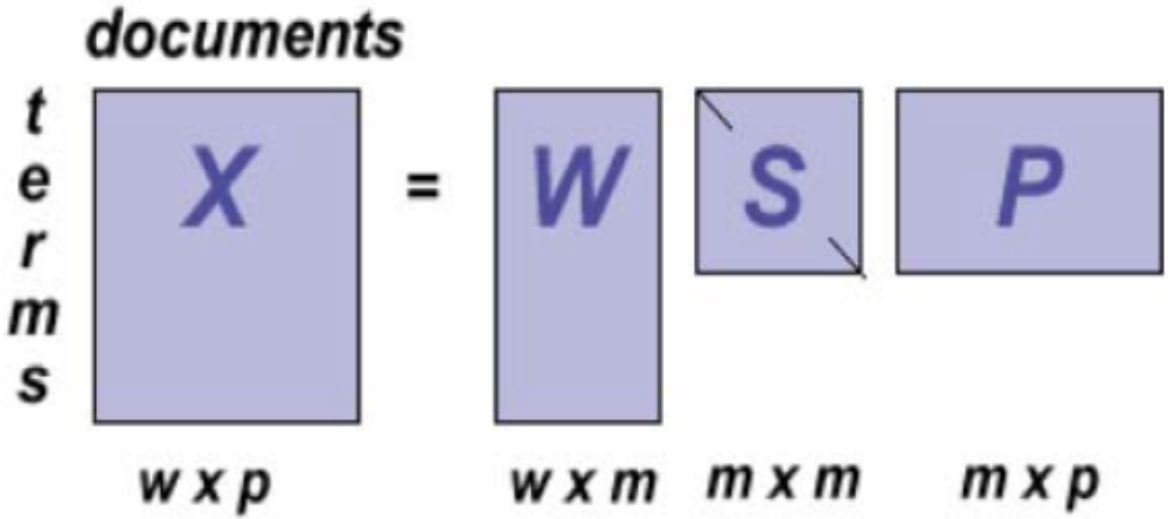
$$score(Q, D) = \sum_{i=1}^{n} Idf(q_i) \frac{\sum_E rank(E)(k_1 + 1)}{\sum_E rank(E) + k_1(1 - b + b\frac{dl}{avgdl})}$$

$rank(E)$ - функция взвешивания вхождения слова в документ (зависит от зоны, позиции и т.д.)

Недостатки рассмотренных моделей

Какие недостатки есть в тех моделях, которые рассмотрели?

Latent semantic analysis (1988)



$$\hat{D} = A^T D, \quad \hat{Q} = A^T Q,$$

$$A = W_k S^{-1}$$

$$sim(Q, D) = \frac{\hat{Q}\hat{D}}{\|Q\|\|D\|}$$

Latent semantic analysis (1988)

Фактически латентно-семантический анализ - это применение SVD разложения к матрице “термин-документ”

- ✔ Оценка близости документов
- ✔ Оценка близости терминов
- ✔ Кластеризация документов
- ✔ Взвешивание пары запрос-документ
- ✖ Низкая скорость для больших коллекций

Тематическое моделирование

Что такое тема?

- тема - семантический кластер текстов
- тема - набор терминов предметной области
- тема - условное распределение на множестве слов

$$p(w|t) - \text{вероятность слова } w \text{ в теме } t$$

- тема - тематический профиль документа

$$p(t|d) - \text{вероятность темы } t \text{ в документе } d$$

Цель тематической модели:

Найти латентные темы документов коллекции по наблюдаемым распределениям слов $p(w|d)$ в документах.

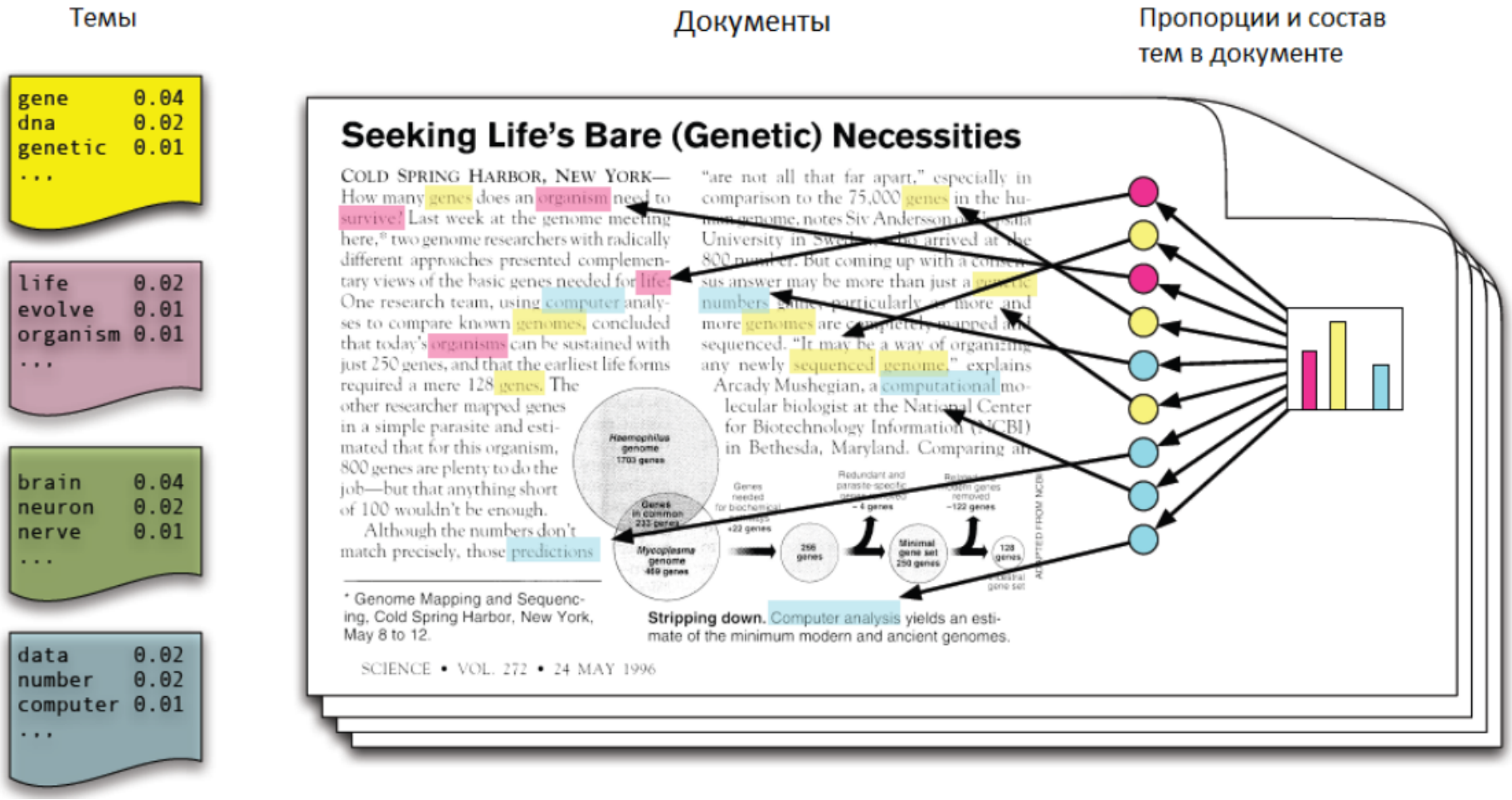
Тематическое моделирование

Основные положения:

- Модель мешка слов для документов (порядок не важен)
- Модель мешка документов для коллекции (порядок не важен)
- Коллекция - это i.i.d. выборка $(d_i, w_i, t_i)_{i=1}^n \sim p(d, w, t)$
- d_i, w_i - наблюдаемые переменные, t_i - скрытые
- Гипотеза условной независимости: $p(w|d, t) = p(w|t)$
- Считаем, что тексты предобработаны (стемминг, лемматизация, удаление стоп-слов и т.д.)

Тематическое моделирование

$$p(w|d) = \sum_{t \in T} p(w|t) \cdot p(t|d)$$



Тематическое моделирование

Дано:

- W - словарь слов
- D - коллекция документов
- $d = \{w_1 \dots w_{n_d}\}$ - документ
- n_{dw} - число раз, когда слово w встретилось в документе d
- n_d - длина документа d

Найти:

Параметры модели $\frac{n_{dw}}{n_d} \approx p(w|d) = \sum_{t \in T} \psi_{wt} \theta_{td}$

$\psi_{wt} = p(w|t)$ - вероятности слов w в каждой теме t

$\theta_{td} = p(t|d)$ - вероятности тем w в каждой документе d

Тематическое моделирование

- $X = (d_i, w_i)_{i=1}^n$ - исходные данные
- $T = (t_i)_{i=1}^n$ - скрытые переменные, темы
- $\Omega = (\Psi, \Theta)$ - параметры

Нужно по X найти Ω

Максимизируем неполное правдоподобие

$$ln\, p(X|\Omega) = ln \sum_T p(X, T|\Omega) \to \max_\Omega$$

EM алгоритм:

$$\text{E-step:} \quad q(T) = p(T|X, \Omega)$$

$$\text{M-step:} \quad \sum_T q(T) ln\, p(X, T|\Omega) \to \max_\Omega$$

Тематическое моделирование

$p(\Omega)$ - априорное распределение параметров модели

Принцип максимума правдоподобия

$$p(X, \Omega) = p(X|\Omega)p(\Omega) \to \max_{\Omega}$$

$$ln\, p(X, \Omega) = ln\, p(X|\Omega) + \ln\, p(\Omega) \to \max_{\Omega}$$

Обозначим $R(\Omega) = \ln\, p(\Omega)$

PLSA [Hofmann, 1999]: $R(\Omega) = 0$

LDA [Blei, 2003]: $R(\Omega) = ln \prod_{t \in T} Dir(\psi_t|\beta) \prod_{d \in D} Dir(\theta_d|\alpha)$

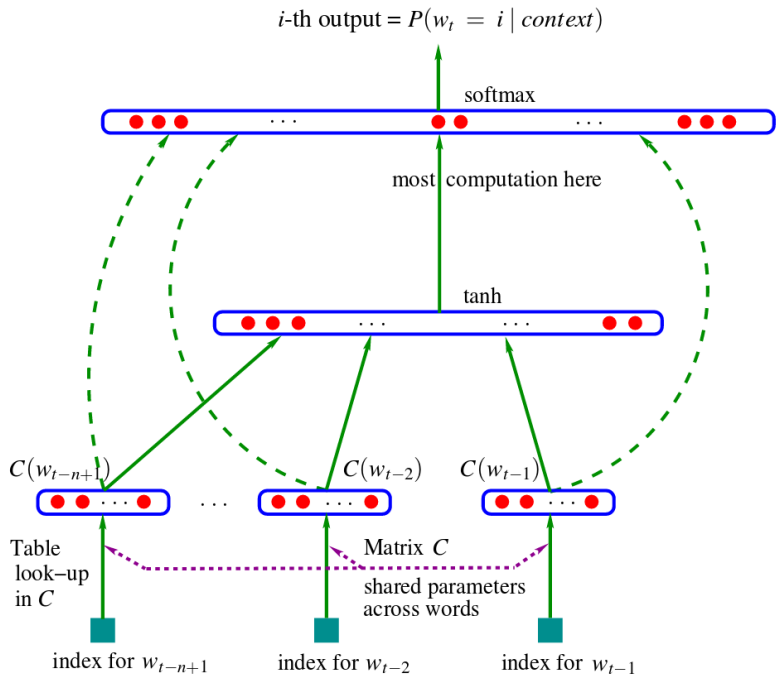
EM алгоритм:

$$\text{E-step:} \quad q(T) = p(T|X, \Omega)$$

$$\text{M-step:} \quad \sum_{T} q(T) ln\, p(X, T|\Omega) + R(\Omega) \to \max_{\Omega}$$

A Neural Probabilistic Language Model (Y. Bengio 2003)

Пытаемся с помощью нейросетей оценить вероятность следующего слова по набору из предыдущих слов (сеть может быть как последовательной так и рекурентной)



✖ Долго и сложно обучать

Дистрибутивная гипотеза

Гипотеза:

Лингвистические единицы, встречающиеся в схожих контекстах, имеют близкие значения.

Вывод:

Значит, векторы слов, можно построить с помощью контекстов этих слов.

Представление слов контекстами

Дано:

- V - словарь слов
- C - множество контекстов

Можем построить матрицу S размера $|V| \times |C|$, элементы которой будут описывать связь слова w_i с контекстом c_j.

Например, можно взять положительную поточечную взаимную информацию (PPMI):

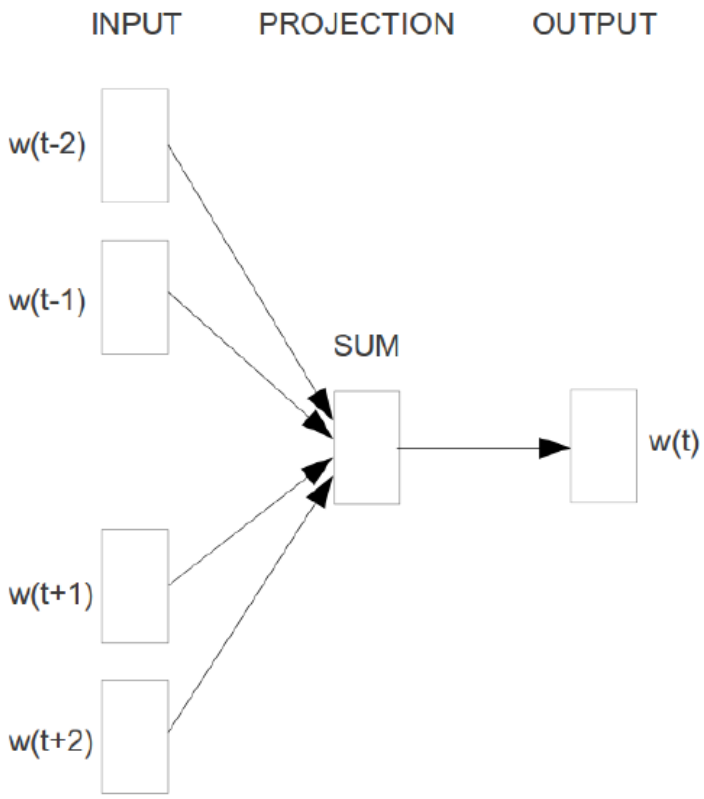
$$S_{i,j} = max(PMI(w_i, c_j), 0),$$

$$PMI(w, c) = log \frac{p(w, c)}{p(w)p(c)} = log \frac{freq(w, c)|V|}{freq(w)freq(c)}$$

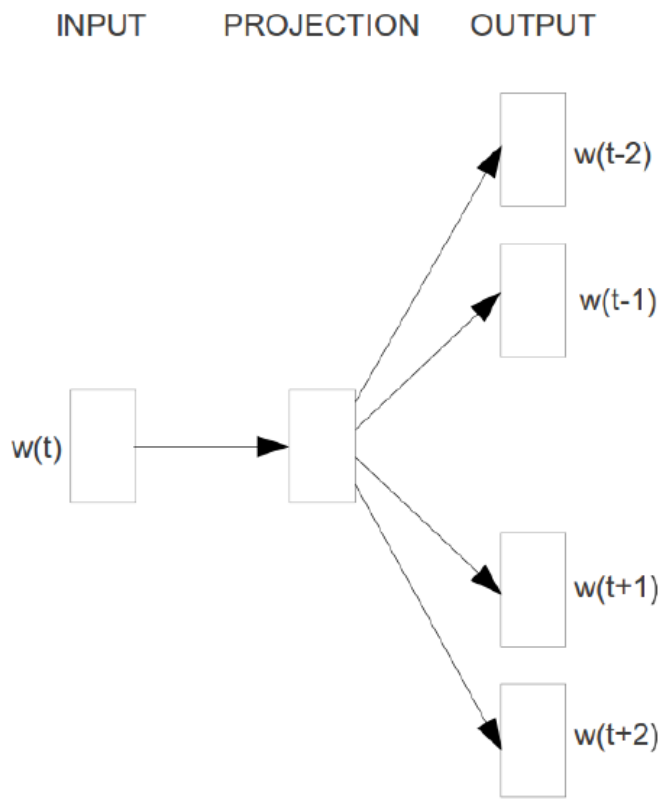
✖ Очень большая размерность матрицы

Word2Vec (2013)

Архитектуры CBOW и Skip-gram

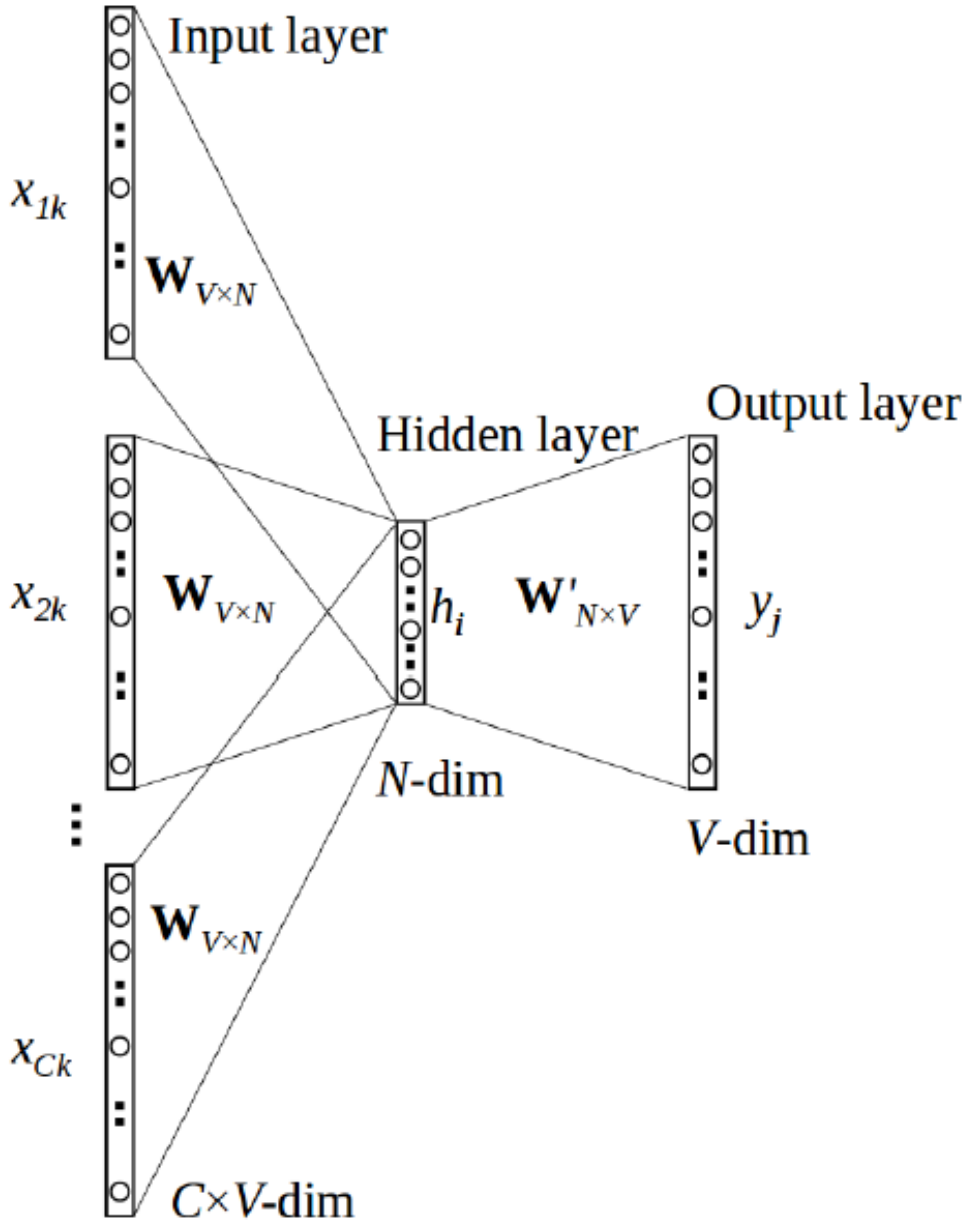


CBOW

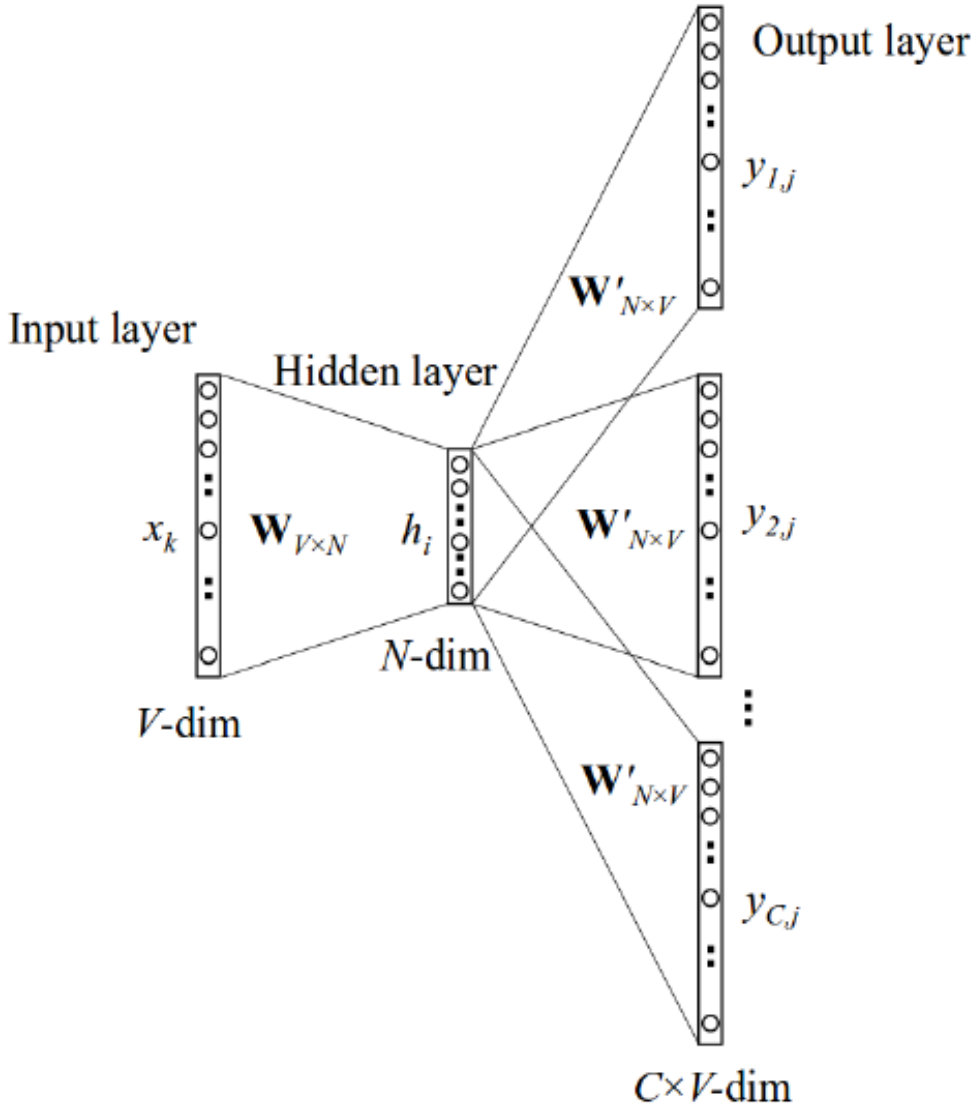


Skip-gram

CBOW (Continious Bag of Words)



Skip-gram


Output layer
$y_{1,j}$
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
$y_{2,j}$
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
$y_{C,j}$
$C\times V$-dim

Skip-gram

Оптимизируем

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} log\ p(w_{t+j}|w_t)$$

Вычисление вероятностей выходных слов

Используется softmax

$$p(w_O|w_I) = \frac{\exp\langle \mathbf{v}'_{w_O}, \mathbf{v}_{w_I} \rangle}{\sum_{w=1}^{|V|} exp(\langle \mathbf{v}'_w, \mathbf{v}_{w_I} \rangle)}$$

На практике эту формулу применять сложно, так как вычисление градиента пропорционально $|V|$.

На практике применяют разные аппроксимации: иерархический softmax или negative sampling.

Negative sampling

Идея:

Не будем рассматривать все слова из словаря, а учтем только рассматриваемое слово + подмешаем еще k отрицательных примеров.
Заменим $log\, p(w_O|w_I)$ на

$$log\, \sigma(\langle \mathbf{v}'_{w_O}, \mathbf{v}_{w_I} \rangle) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} [log\, \sigma(-\langle \mathbf{v}'_{w_i}, \mathbf{v}_{w_I} \rangle)]$$

- $k \approx 5 - 20$ для небольших выборок
- $k \approx 2 - 5$ для больших данных

Смысл Negative Sampling

- Фактически, мы имеем два распределения слов: “положительное” (D) и “отрицательное” (N).
- Мы их смешиваем в пропорции $1 : k$
- Задача модели: угадать, из какого распределения пришло слово

Смысл Negative Sampling

По предположению,

$$p(D|w,c) = \sigma(\langle w,c\rangle)$$

Но по формуле Байесса

$$p(D|w,c) = \frac{p(w,c|D)p(D)}{p(w,c|D)p(D) + p(w,c|N)p(N)}$$

Считаем, что контексты в негативных примерах не зависят от слова:

$$p(w,c|N) = p(w|D)p(c|D)$$

Таким образом

$$p(D|w,c) = \frac{p(w,c|D)\frac{1}{k+1}}{p(w,c|D)\frac{1}{k+1} + p(w,c|N)\frac{k}{k+1}} =$$

$$= \frac{1}{1 + k\frac{p(w|D)p(c|D)}{p(w,c|D)}}$$

Смысл Negative Sampling

Заметим, что выражение стоящее в знаметеле очень похоже на взаимную информацию

$$PMI(w, c) = log \frac{p(w, c|D)}{p(w|D)p(c|D)}$$

Таким образом,

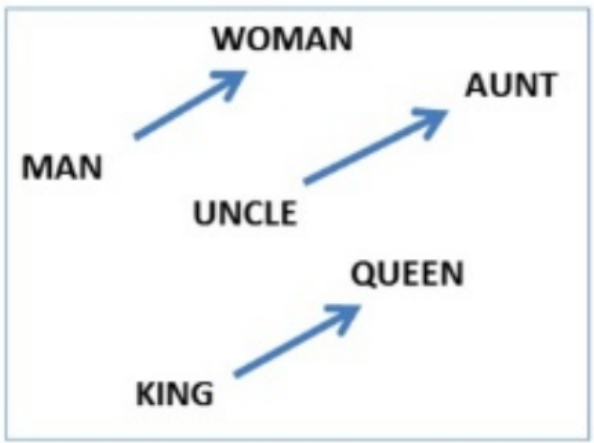
$$p(D|w, c) = \frac{1}{1 + ke^{-PMI(w,c)}}$$

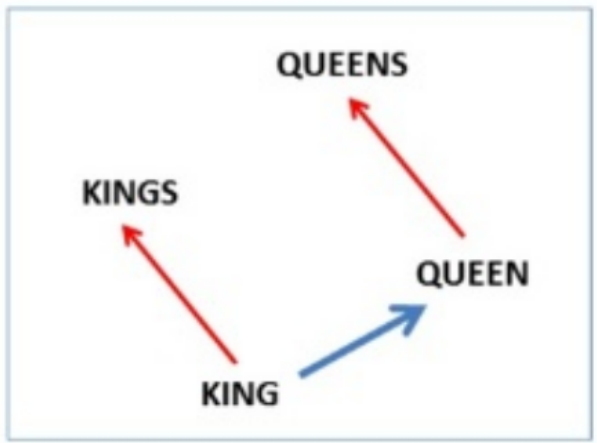
$$\langle w, c \rangle = PMI(w, c) - ln\, k - \text{“сдвинутый” } PMI$$

Skip-gram Negative Sampling эквивалентен факторизации матрицы “сдвинутого” PMI.

Свойства выученных представлений

vec(“man”) – vec(“king”) + vec(“woman”) = vec(“queen”)





Недостатки

- ✖ Качество сильно зависит от обучающий данных, их количества, размера векторов (на вектора большой размерности нужно много вычислительных затрат)
- ✖ Усреднение векторов слов работает плохо уже на текстах среднего размера (не говоря уже о больших)
- ✖ Из word2vec нельзя получить представление фиксированного размера для текстов переменного размера

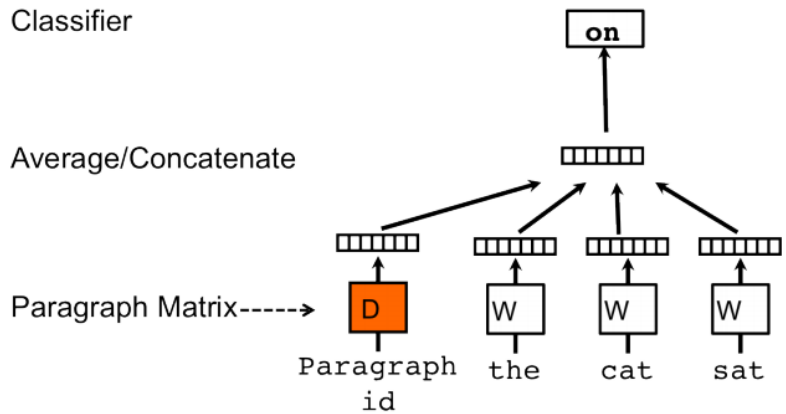
Doc2vec (Mikolov, 2014)

- Обобщение word2vec модели на целые документы (фразы, предложения и т.д.)
- Преобразует текст произвольной длины в вектор фиксированного размера
- Distributed Memory (DM)
- Distributed Bag of Words (DBOW)

Doc2vec

Distributed Memory (DM)

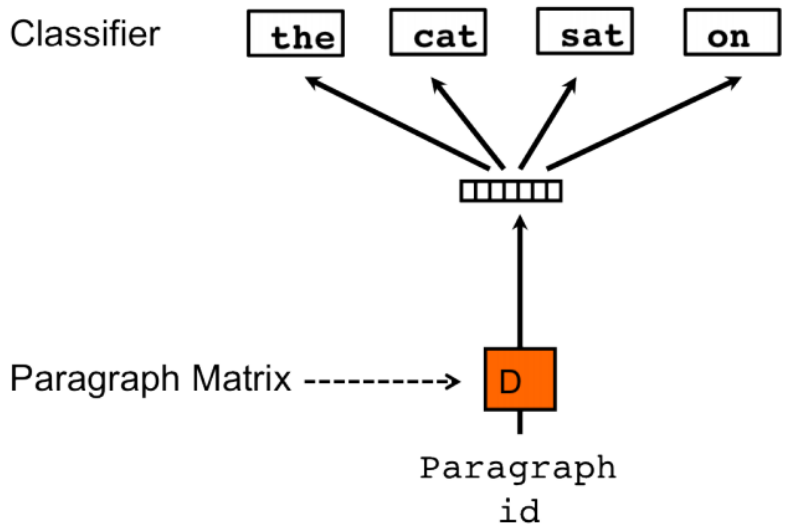
- Назначим и рандомно проинициализируем paragraph vector
- Будем предсказывать слово из текста используя контекст и paragraph vector
- Идем скользящим окном по всему документу, сохраняя при этом paragraph vector фиксированным (поэтому Distributed Memory)
- Обновление прооисходит при помощи SGD и backprop



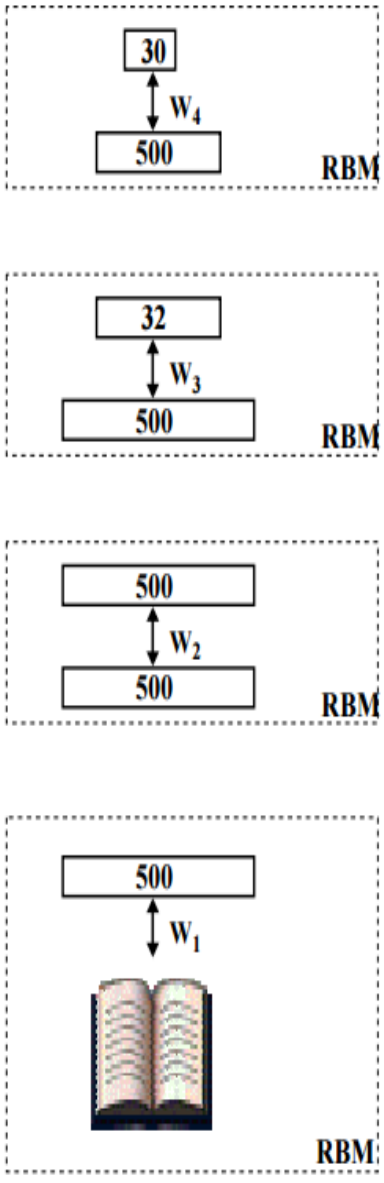
Doc2vec

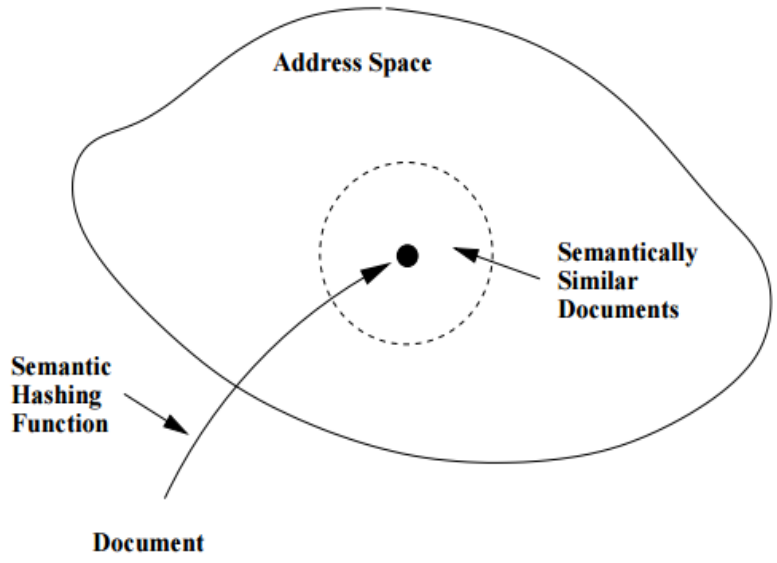
Distributed Bag of Words (DBOW)

- Используем только paragraph vector (вектора слов не используем)
- Берем окно из слов в параграфе и случайно семплируем какое из слов предсказать используя paragraph vector (игнорируем порядок слов)
- Очень просто и требует меньше ресурсов
- Но при этом хуже по качеству, чем DM (однако DM + DBOW работают лучше!)



Semantic hashing (Hinton, Salakhutdinov, 2009)





Document & entity representation learning toolkits

- **Gensim** `https://github.com/RaRe-Technologies/gensim`
- **SERT** `http://www.github.com/cvangysel/SERT`
- **cuNVSM** `http://www.github.com/cvangysel/cuNVSM`
- **HEM** `https://ciir.cs.umass.edu/downloads/HEM`

Задача

Дано:

- Набор вопросных запросов из поиска@mail.ru
- Набор документов, соответствующих этим запросам

Задание:

- Используя модели текстового ранжирования (tf-idf, BM25, BM25F, passage, LSA, LDA, word2vec, doc2vec) необходимо отранжировать данные документы по запросам
- В качестве метрики качества будет выступать NDCG

Формат:

- Kaggle конкурс сроком на 1 месяц

Вопросы