Preprocessing

1) tokenization - бьем последовательность (предложение/текст) на слова/пары слов/подслова.

2) удаляем стоп-слова (слишком частые/бесполезные слова)

3) нормализация

   а) стемминг - обрубаем слова до корня

   б) лемматизация - приводим слова в единую нормальную форму

Porter stemmer

- Published in 1979
- Base starting option

Snowball stemmer (Porter 2)

- Based on Porter
- More aggressive
- Most popular option now

Lancaster stemmer

- Published in 1990
- The most aggressive
- Easy adding of your own rules

17

заметка:
- пунктуация - отдельные токены
- если имена собственные не важны, то всё приводим к нижнему регистру

Bag of words (BoW)

предпроцессим текст → каждый токен словаря превращаем в бинарный признак → получаем бинарный вектор

векторное представление последовательности

(+)

- очень просто :)

(−)

- не учитывается порядок слов в предложении
- размерность векторов большая и разреженная
- векторы не нормализованы
- одни и те же слова могут иметь разные формы и окончания

заметка: мы можем учитывать локальный порядок слов, используя n-граммы.
проблема: их слишком много
решение:
- убираем слишком частые/редкие
- убираем 2-граммы с артиклями/союзами

идея: добавить устойчивые слово-сочетания как отдельные токены

TF-IDF

вопрос: как определить важность слов для текста в BoW?

решение: посчитать TF-IDF — term frequency – inverse document frequency

$tf(t,d) = \#t \text{ в } d$

$idf(t) \quad \log \frac{N - \#d}{\#d, \text{в которых есть } t}$

- чем больше tf, тем важнее слово для документа
- чем больше док-в, имеющих слово t, тем сильнее дискредитируется важность слова в целом

заметка: если слово встречается во всех док-х, то $idf(t) = \log \frac{N}{N} = \log 1 = 0$
слово будет иметь нулевую важность

идея: можно использовать tf-idf как веса для усреднения, когда есть представления слов, а мы хотим получить представление документа

давайте хотя бы как-то использовать контекст

способ: (контекст = документ/предложение/n-грамма)

PMI → SVD → векторные представления слов

(#слов на #слов) (#слов на k)

- Better solution: Pointwise Mutual Information (PMI)

$$PMI = log\frac{p(u,v)}{p(u)p(v)} = log\frac{n_{uv}n}{n_u n_v}$$

- Much better solution: **Positive PMI (pPMI)**

$$pPMI = \max(0, PMI)$$

Word 2 Vec

идея метода: - давайте по слову предсказывать контекст или слово по контексту

- слова, встречающиеся в похожих контекстах, близки по смыслу и эмбеддингам

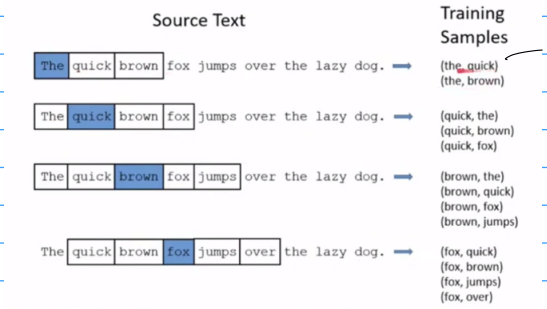
- скорость получения emb. слова в обученном W2V:

O(1), потому что когда подаем 1 слово, то мы подаем строку с одной 1, то есть при умножении на м-цу (мн. слов), мы берём 1 столбец.

Получается словарь

алгоритм:

1) Берём окно размера |X| и итерируемся им по тексту



подход Skip-gram

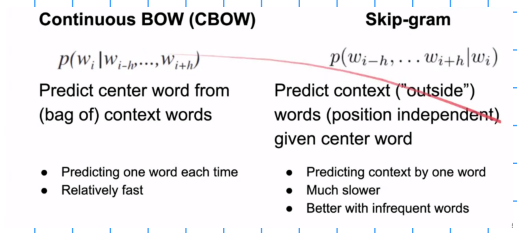
2) составляем пары для обучения в зависимости от подхода (CBoW или Skip-gram)

а) CBoW - continious bag of words

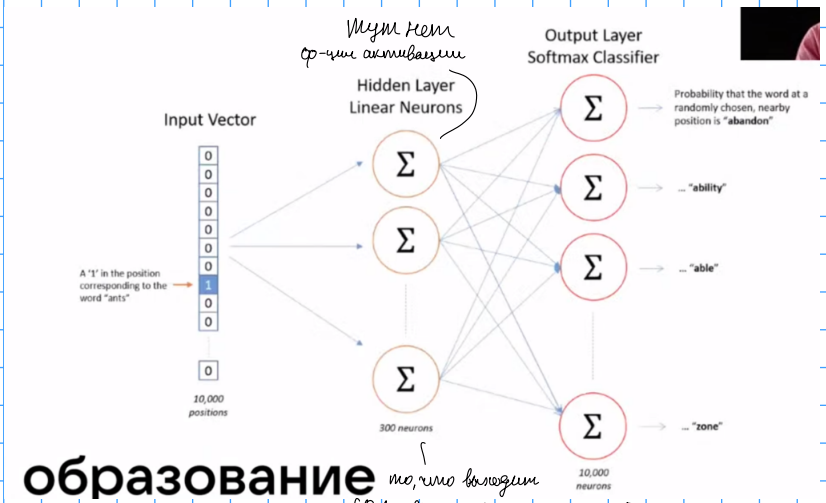
предсказываем слово по контексту

б) Skip-gram

предсказываем контекст по слову

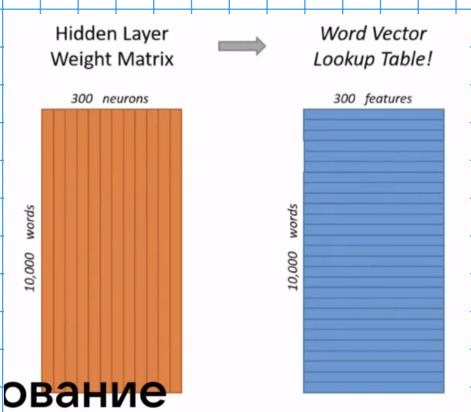


3) обучаем 1 слойную нейронку на классификацию



образование

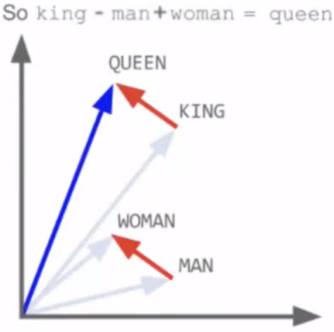
то, что выходит со слоя перед классификацией считаем эмбеддингом



ование

- W2V легко дообучать

- св-во контекстов emb.



Инженерные моменты при обучении W2V

проблема: основная проблема - дисбаланс "классов" и перекос токенов в тех, на чём обучаемся

решение:

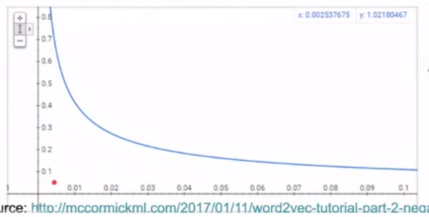
а) перекос токенов

Subsampling - оставляем пару эс-у в зависимости от того как часто встречается эс

Subsampling frequent words.

w_i is the word, $z(w_i)$ is the fraction of this word in the whole text

Graph for (sqrt(x/0.001)+1)*0.001/x



$P(w_i)$ is the probability of *keeping* the word:

$$P(w_i) = (\sqrt{\frac{z(w_i)}{0.001}} + 1) \cdot \frac{0.001}{z(w_i)}$$

Source: http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/

52

б) дисбаланс классов

негатив семплинг - берём не все нули, а только часть

и чем чаще класс встречается, чем чаще мы показываем как 0 в других случаях