Data Gym
word2vec в рекомендательных системах

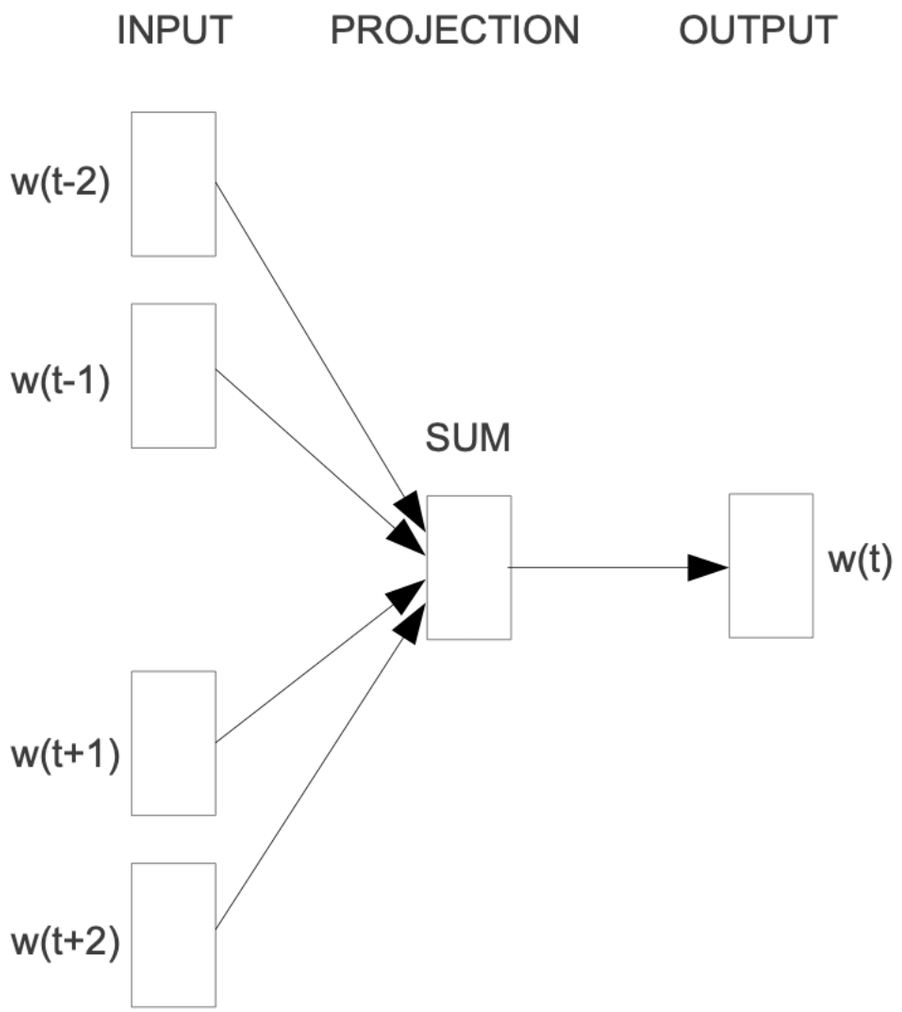
Дмитриев Александр
DS MyBook / Zvooq

Предыстория

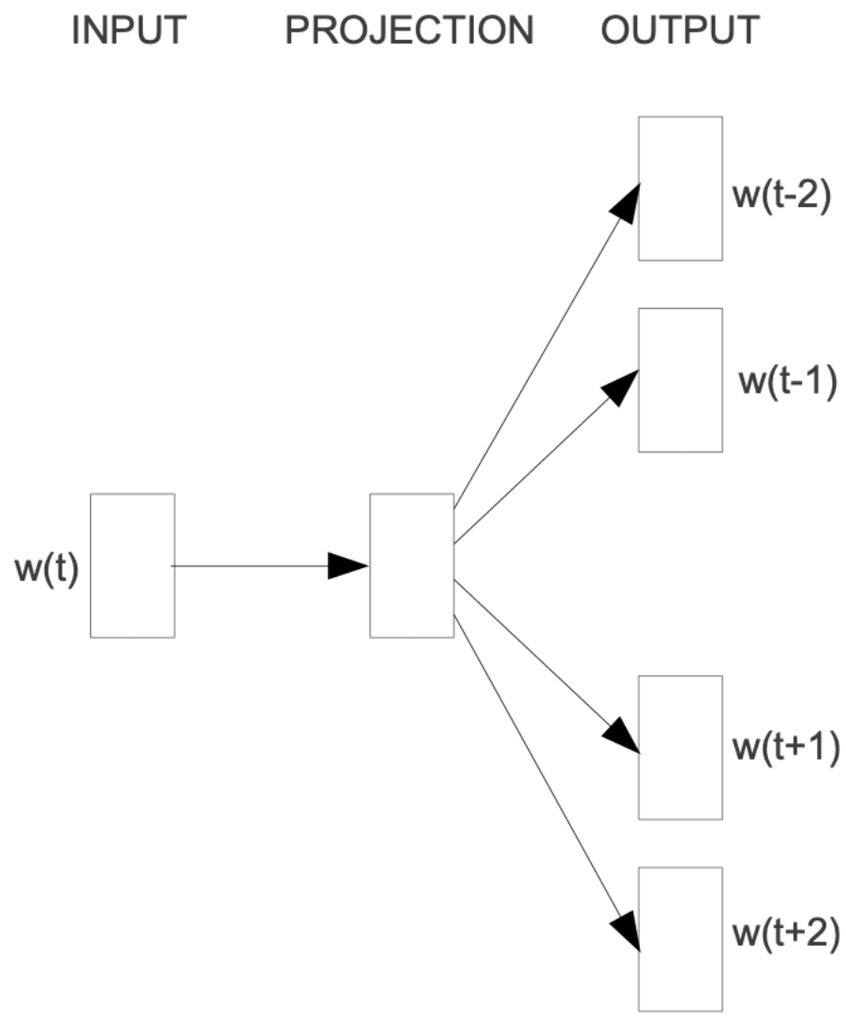
- 2013 год, статья от Tomáš Mikolov et al.
- В основе алгоритма distributional hypothesis
- Начало активного использования эмбедингов в NLP

https://code.google.com/archive/p/word2vec/
https://arxiv.org/pdf/1301.3781.pdf
https://arxiv.org/pdf/1310.4546.pdf

Skip gram / CBOW

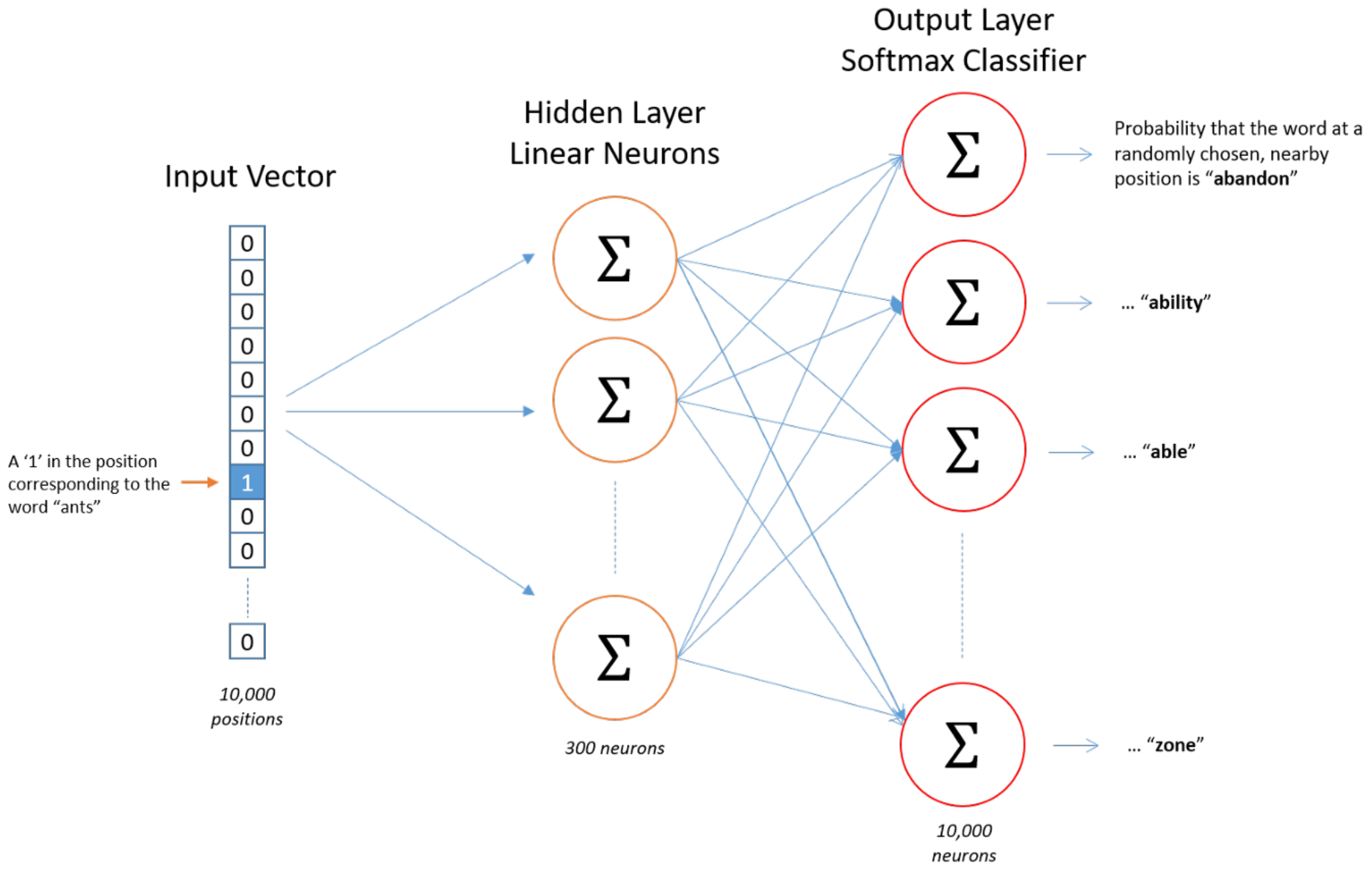


CBOW

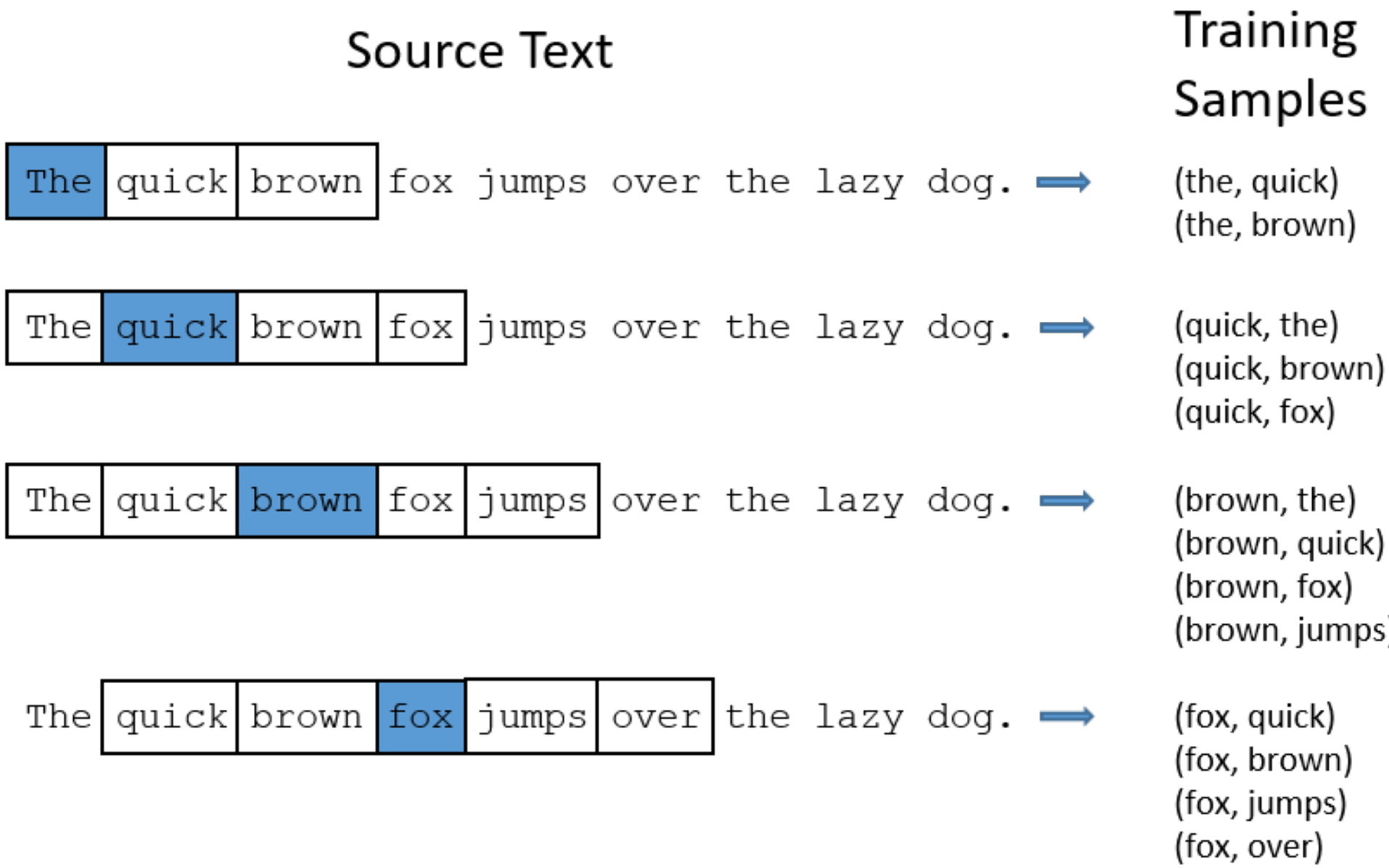


Skip-gram

Skip gram


Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
A '1' in the position corresponding to the word "ants"
0
0
0
0
0
1
0
0
0
10,000 positions
300 neurons
10,000 neurons

Обучение


Source Text
Training Samples
The quick brown fox jumps over the lazy dog.
(the, quick)
(the, brown)
The quick brown fox jumps over the lazy dog.
(quick, the)
(quick, brown)
(quick, fox)
The quick brown fox jumps over the lazy dog.
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)
The quick brown fox jumps over the lazy dog.
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)

Сабсэмлинг

$$p = 1 - \sqrt{\frac{t}{f}}$$

$$p = \frac{f - t}{f} - \sqrt{\frac{t}{f}}$$

Реализация gensim

- Although, in words of word2vec's authors, the toolkit is meant for "research purposes", it's actually optimized C, down to cache alignments, memory look-up tables, static memory allocations and a penchant for single letter variable names. Somebody obviously spent time profiling this, which is good news for people *running it*, and bad news for people wanting to *understand it, extend it* or *integrate it* (as researchers are wont to do).

RADIM ŘEHŮŘEK

- 29K words per second
- Einstein proved nothing is faster than C

Статья про параллелизацию и видео про оптимизацию:
https://rare-technologies.com/parallelizing-word2vec-in-python/
https://www.youtube.com/watch?v=vU4TlwZzTfU

Реализация gensim

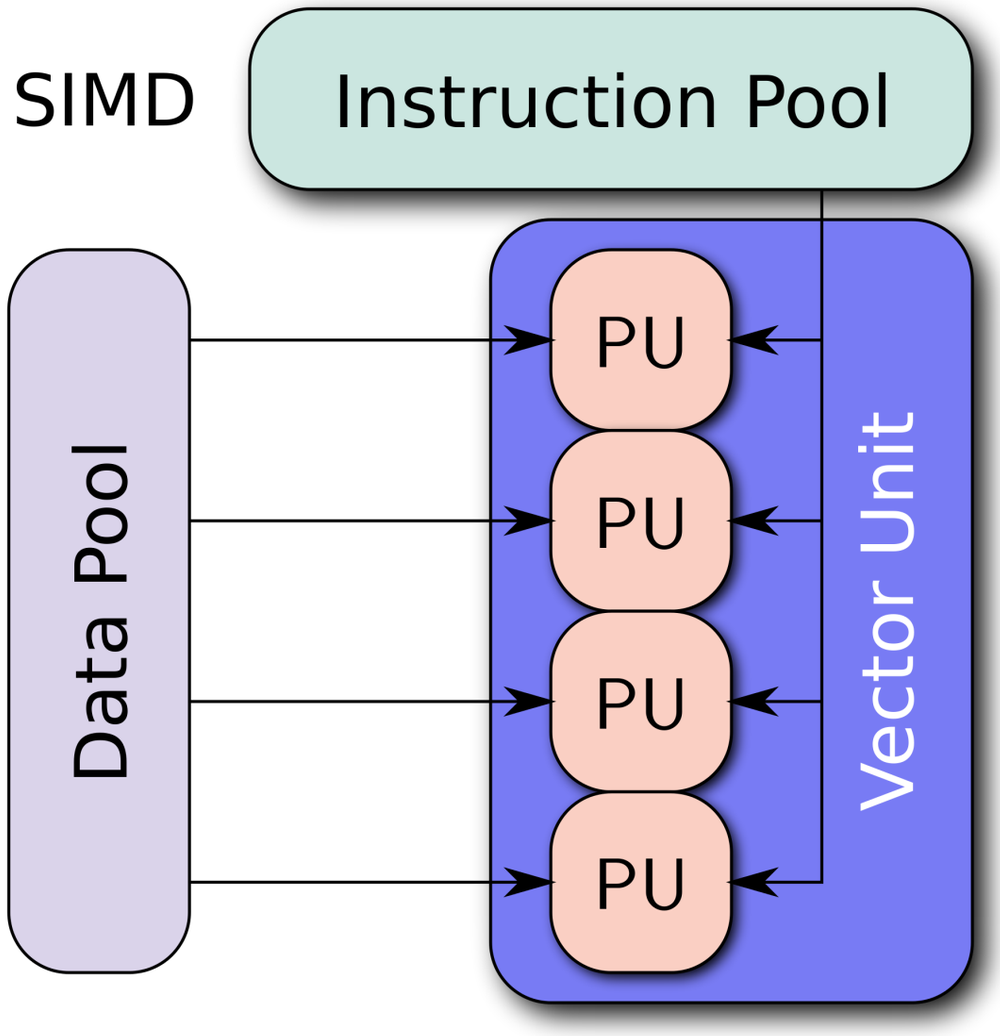
Naive Python 100 LOC	12-28 w/s	x1/120 or x1/50
NumPy	1.4K w/s	x1
Original C code	29K w/s	x21
Cython	33K w/s	x24
Cython + BLAS	89K w/s	x64
Cython + sigmoid table	35K w/s	x25
Cython + BLAS + sigmoid table	102K w/s	x73

BLAS

- Спецификация набора примитивов линейной алгебры
- Дефакто стандарт для библиотек, активно работающих с векторами, матрицам и т.п.
- Изначально библиотека на фортране 1979 года
- Много имплементаций: ACML (AMD), MKL (Intel), OpenBLAS, ATLAS
- Активно использует SIMD инструкции и кэш, быстро работает только на том железе для которого написана

SIMD

- Основа векторных суперкомпьютеров 70-х (Cray etc.)
- Сейчас в кармане каждой домохозяйки
- Параллельные вычисления не как в GPU



Использование w2v в рекомендациях

- Айтемы - слова
- Контест - предложения

https://astro.temple.edu/~tuc17157/pdfs/grbovic2015kddB.pdf

https://youtu.be/yImkZN-C5Dc?t=246

Тюнинг гиперпарметров

- Товары это не совсем слова
- NLP это совсем не RecSys

- Статья от Deezer: взяли несколько датасетов и перебрали много гиперпараметров

Тюнинг гиперпарметров

Датасеты:

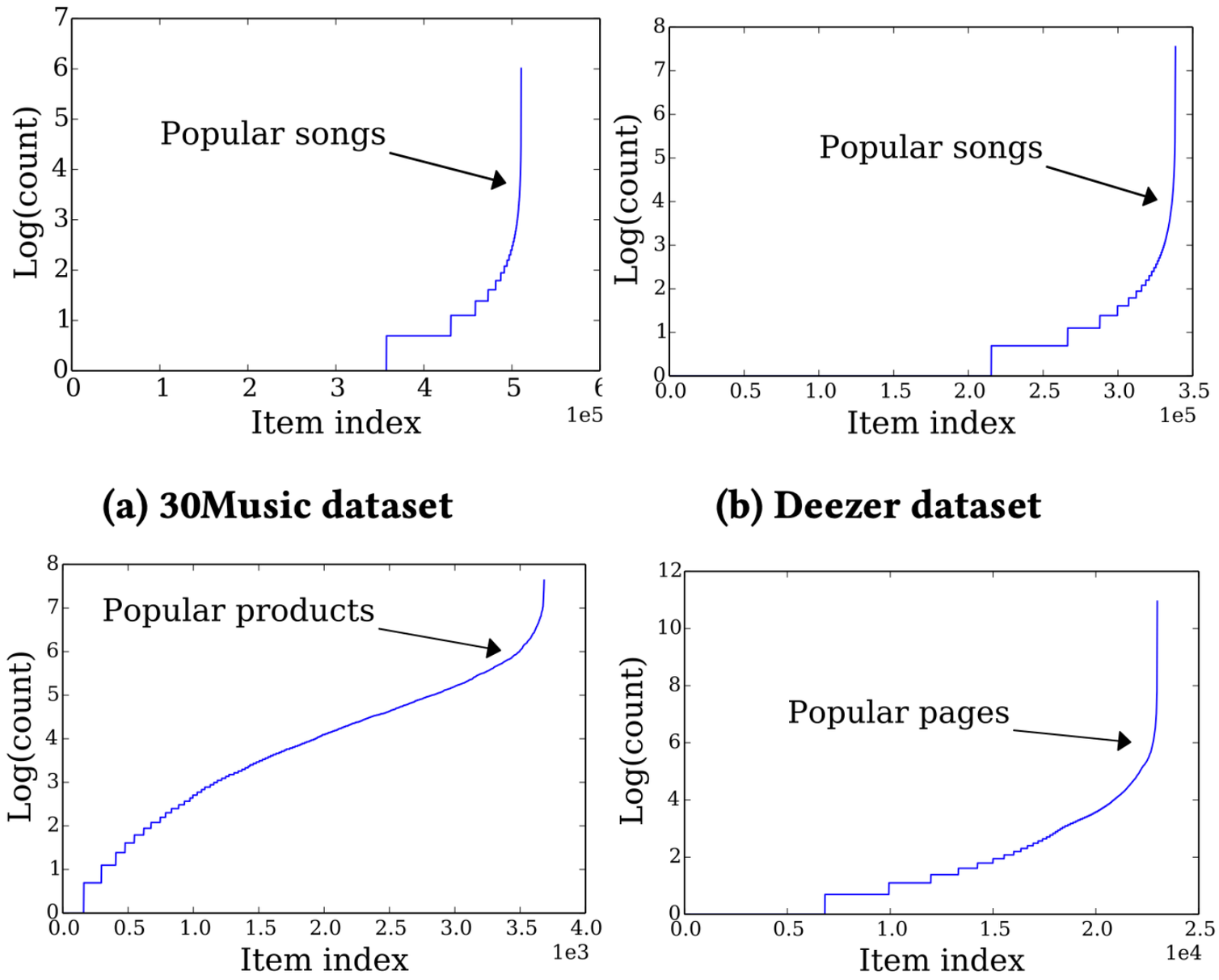
- 2 музыкальных, 100k сессий в каждом
- e-comerce, покупки 4234 пользователей
- новостной, 83625 кликстрима пользователей

Что делали:

- next event prediction
- HR@k, NDCG@k

https://arxiv.org/pdf/1804.04212.pdf
https://github.com/deezer/w2v_reco_hyperparameters_matter

Тюнинг гиперпарметров



(a) 30Music dataset

(b) Deezer dataset

(c) E-commerce dataset

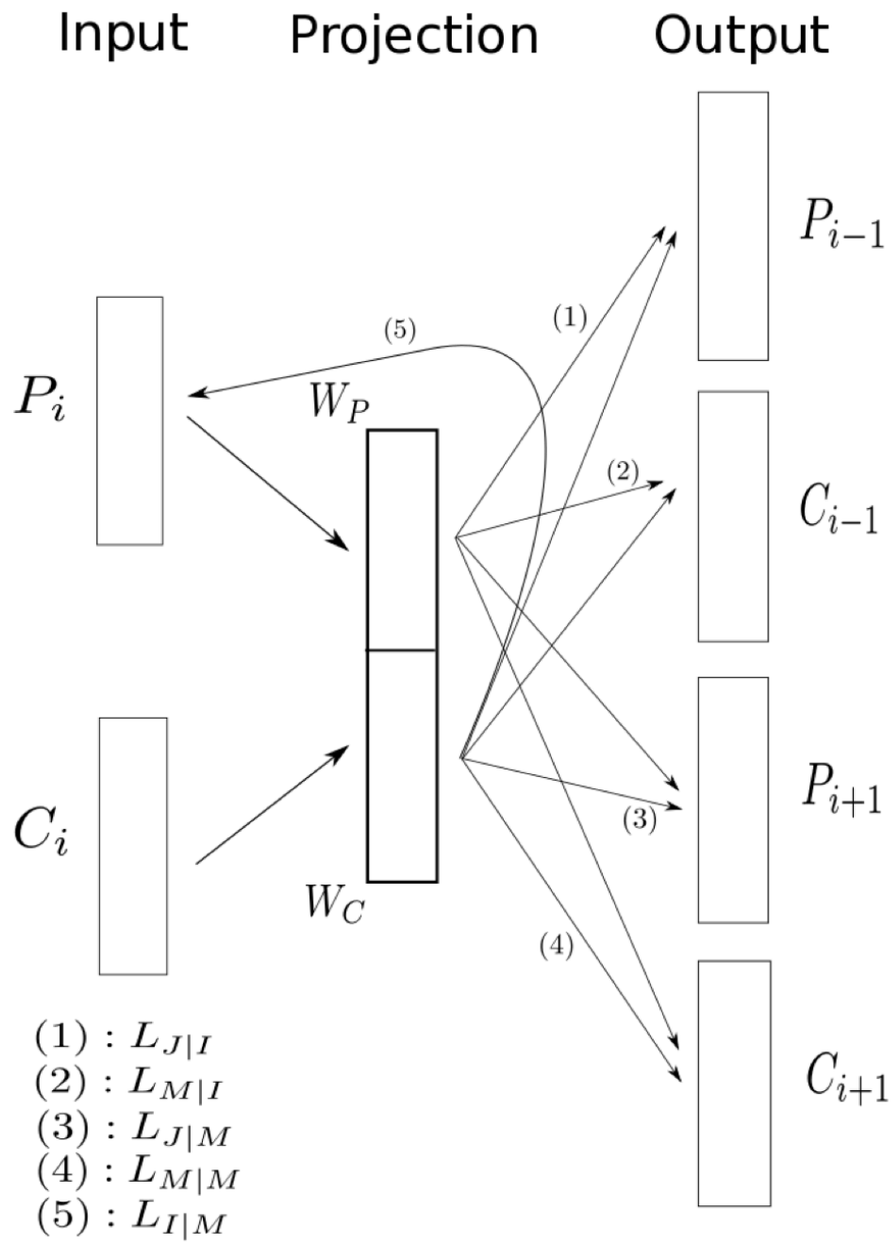
(d) Click-Stream dataset

Тюнинг гиперпарметров

- По метрикам x2 в музыке, x10 на кликстриме, новости ~x1.2
- Не сработали сильно: размер эмбединга (50-200), learning rate, кол-во негативных примеров
- t - отсечка для удаления популярных айтемов
- размер окна сэмплили случайно от 1 до L
- α - константа, влияющая на подбор отрицательных примеров (ns_exponent)

Использование меты

- Статья от Criteo
- По сути выучили эмбединг категории
- Лучше всего перформил MetaProd2Vec + CoCounts



https://arxiv.org/pdf/1607.07326.pdf
Meta-Prod2Vec: Simple Product Embeddings with Side-Information

Манипуляции с векторами

Можно реализовать:

- Учитывать отрицательный фидбэк
- Собирать вектора категорий/подкатегорий и тп
- Манипуляции с вектором пользователя налету

Различия с ALS/MF

- Можно использовать окно
- Учитывает многократное использование айтемы
- Используем негативные примеры
- Не собираем вектор пользователя -> сильно дешевле по памяти
- Фильтрация из коробки
- Удобная работа с ID
- Можно говорить друзьям, что у вас нейронки в проде 😎
- Но говорят, что это одно и тоже: https://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf

Разные типы датасетов

- Все айтемы равнозначны
- Наборы айтемов из разных категорий
- Можно добавлять фейковые айтемы для учета контекста

Дообучение

- Учет дополнительных интеракций
- Добавляем айтемы
- Перевыбираем профиль пользователя налету

Inference

- wv.most_similar
- wv.most_similar_cosmul
- wv.doesnt_match
- similar_by_vector == similar_by_word == wv.most_similar
- predict_output_word - выдает вероятности слов в том же контексте (окружающих слов)

Кто использует

- MyBook
- Spotify / Anghami / Deezer
- Airbnb
- Pandao
- Lamoda
- Ozon

spotify https://www.quora.com/How-did-Spotify-get-so-good-at-machine-learning-Was-machine-learning-important-from-the-start-or-did-they-catch-up-over-time
Anghami https://towardsdatascience.com/using-word2vec-for-music-recommendations-bb9649ac2484
airbnb https://medium.com/airbnb-engineering/listing-embeddings-for-similar-listing-recommendations-and-real-time-personalization-in-search-601172f7603e

Код первой версии в gensim

https://github.com/RaRe-Technologies/gensim/commit/6a52630 18c51d9993fb24df584a9eabbffb81642#diff-673fdc31e9aae23291039b143f451b9e