Лекция 7. Обработка естественного языка, эмбеддинги слов

Денис Деркач, Дмитрий Тарасов

Использовались слайды Кати Трофимовой, Антона Кленицкого, Лены Войта

10 марта 2025 года

Обработка текстовой информации

Задачи NLP

Text classification

- Сопоставляем метки всему тексту как целому
- Sentiment analysis
- анализ тональности (положительная, нейтральная, отрицательная)
- Spam / not spam

Word classification

- Сопоставляем метки каждому слову по отдельности
- Part-of-speech (POS) tagging - частеречная разметка
- Named Entity Recognition (NER) - распознавание именованных сущностей (person, location, organization, date,..

Word classification

- Machine translation
- Text summarization (extractive / abstractive)
- Question Answering
- Dialogue systems
- Text generation

Word embeddings

Представление текста

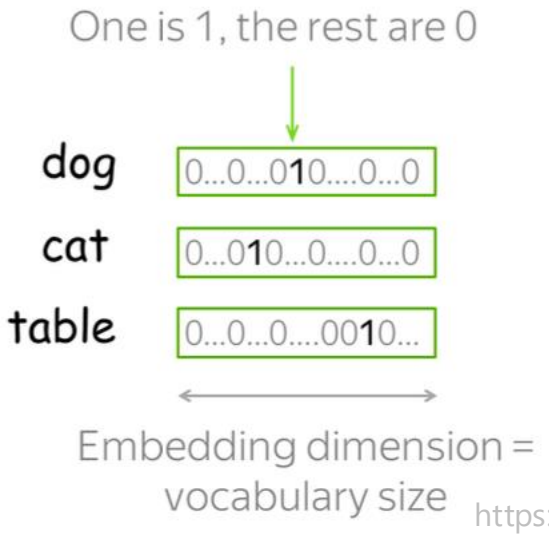
Первый шаг - токенизация (разбиение текста на отдельные слова - токены)

Дальше нужно перевести слова в числовой вид

Самый простой вариант - one-hot encoding

Минусы one-hot encoding

- Большой словарь -> очень большая размерность
- Не учитывает смысл и взаимоотношения между словами (все вектора ортогональны друг другу)



https://lena-voita.github.io/nlp_course/word_embeddings.html

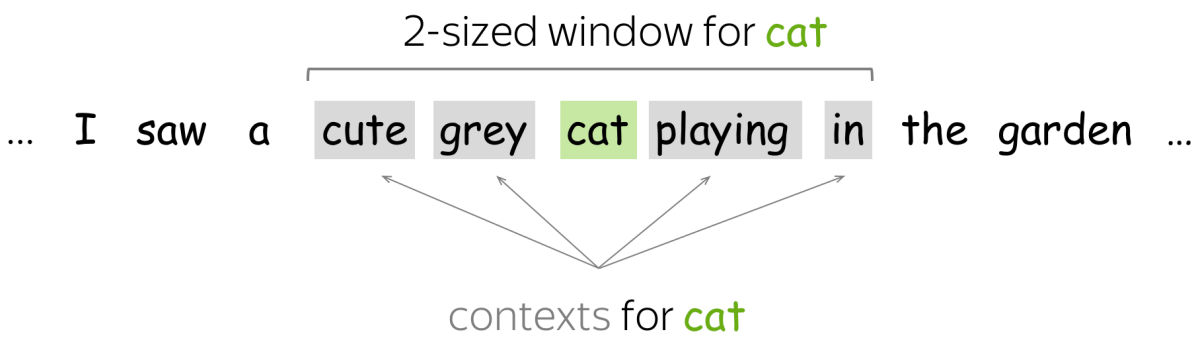
Word embeddings

Хотим представить слова в виде векторов небольшой размерности, которые отражают их смысл, чтобы

- Близкие по смыслу слова имели похожие вектора
- Разные по смыслу слова имели непохожие вектора

Основная идея:

Слова, которые часто встречаются в схожих контекстах, имеют похожее значение (distributional hypothesis)



Context:

- surrounding words in a L-sized window

Matrix element:

- N(w, c) – number of times word w appears in context c

Positive Pointwise Mutual Information (PPMI)

Вероятности P того, что слова окажутся в L-окне.

NB: Было показано, что некоторые из нейронных методов, которые мы рассмотрим (Word2Vec), неявную аппроксимацию факторизации (смещенной) матрицы PMI.

Context:

- surrounding words in a L-sized window

Matrix element:

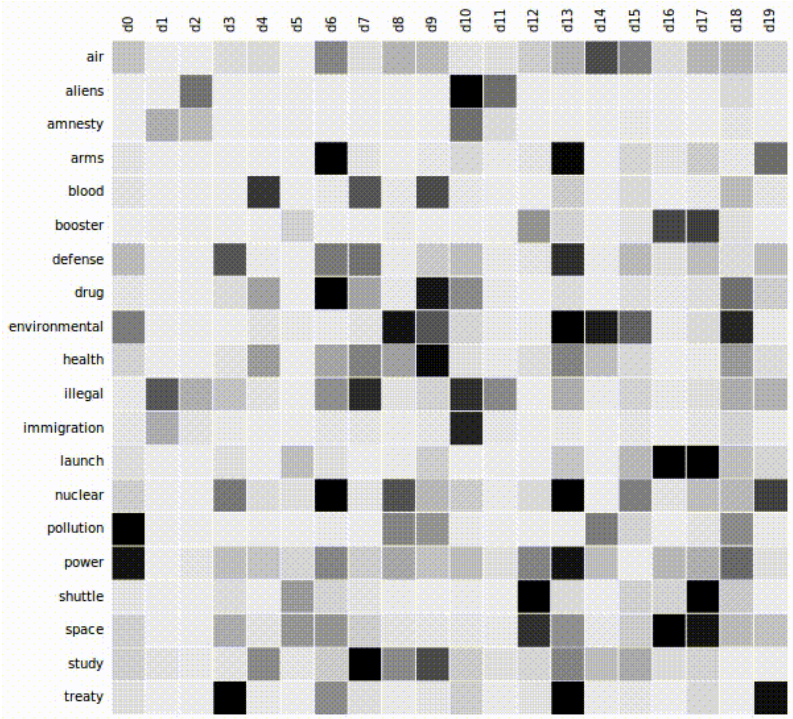
- PPMI(w, c) = max(0, PMI(w, c)),

where

$$\mathrm{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)} = \log \frac{\mathrm{N}(w, c)|(w, c)|}{\mathrm{N}(w)\mathrm{N}(c)}$$

Latent Semantic Analysis (LSA)

Если в предыдущих подходах контексты служили только для получения векторов слов и впоследствии отбрасывались, то здесь нас также интересует контекст или, в данном случае, векторы документов.



Context:

- document d (from a collection D)

Matrix element:

- $\text{tf-idf}(w, d, D) = \text{tf}(w, d) \cdot \text{idf}(w, D)$

$\text{tf}(w, d) = N(w, d)$ — term frequency

$\text{idf}(w, D) = \log \frac{|D|}{|\{d \in D: w \in d\}|}$ — inverse document frequency

https://en.wikipedia.org/wiki/File:Topic_model_scheme.webm

Word2Vec

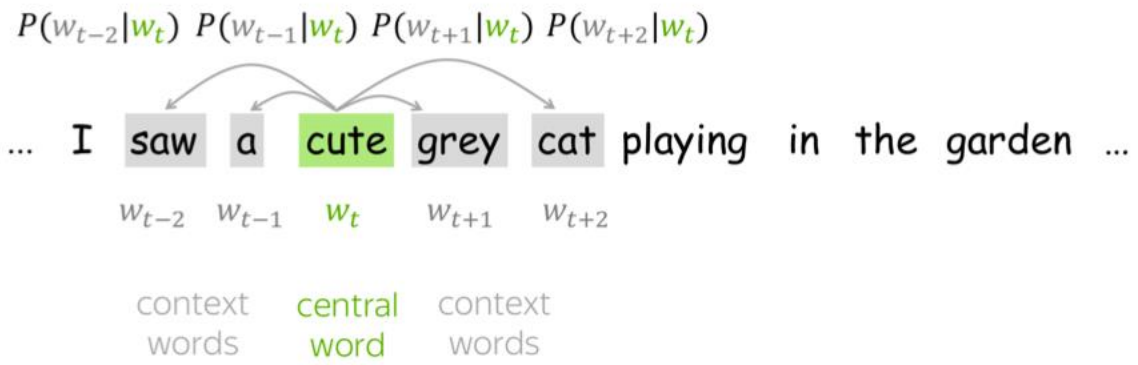
Word2Vec

Общая идея:

- Вектора слов, которые встречаются в одном контексте, должны быть близки

Реализация:

- Учимся предсказывать контекст (окружающие слова) по вектору данного слова

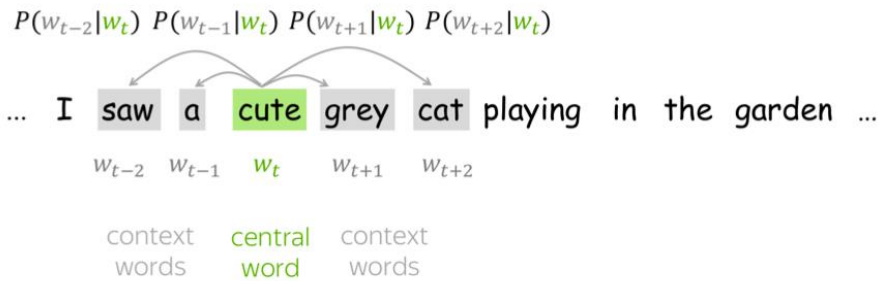


Word2Vec

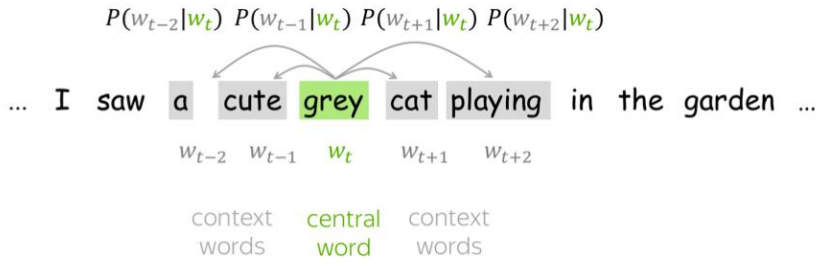
Алгоритм:

- Берем большой корпус текстов
- Проходим по текстам скользящим окном, смещаемся на одно слово на каждом шаге
- В каждом окне есть центральное слово и слова контекста (остальные слова в окне)
- Предсказываем вероятность окружающих слов на основе вектора центрального слова

1.



2.

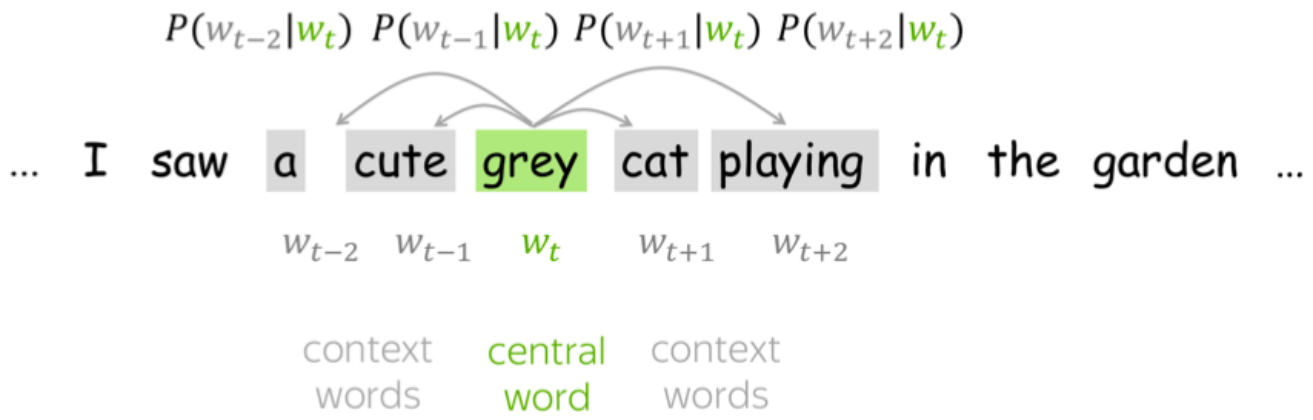


https://lena-voita.github.io/nlp_course/word_embeddings.html

Word2Vec – функция потерь

Функция потерь – отрицательное лог-правдоподобие:

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



Word2Vec – Обучение

Для каждого слова обучаются два вектора - v_w когда слово в центре и u_w когда слово в контексте.

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

o - outside word, c - central word

Обучаем стохастическим градиентым спуском:

$$L_{t,j}(\theta) = -\log P(o|c) = -u_o^T v_c + \log \sum_{w \in V} \exp(u_w^T v_c)$$

$$v_c = v_c - \alpha \frac{\partial L_{t,j}(\theta)}{\partial v_c}$$

$$u_w = u_w - \alpha \frac{\partial L_{t,j}(\theta)}{\partial u_w}$$

Увеличиваем близость между v_c и u_o и уменьшаем близость между v_c и всеми остальными u_w.

Word2Vec – Negative Sampling

$$L_{t,j}(\theta) = -\log P(o|c) = -u_o^T v_c + \log \sum_{w \in V} \exp(u_w^T v_c)$$

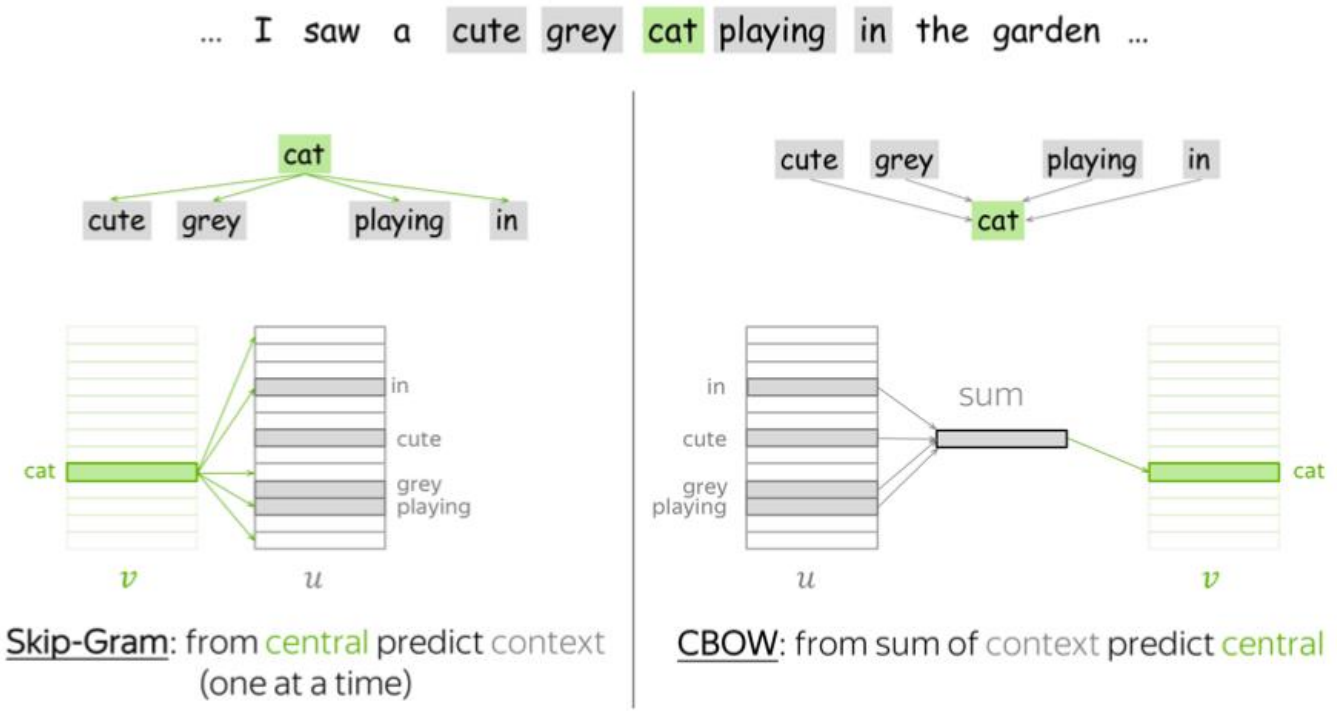
Словарь очень большой, на каждом шаге SGD обновляем вектора всех слов u_w - долго.

Выход - сэмплирование отрицательных примеров

$$\sum_{w \in V} \exp(u_w^T v_c) \rightarrow \sum_{w \in \{w_{i_1}, .., w_{i_K}\}} \exp(u_w^T v_c)$$

Skip-gram vs CBOW

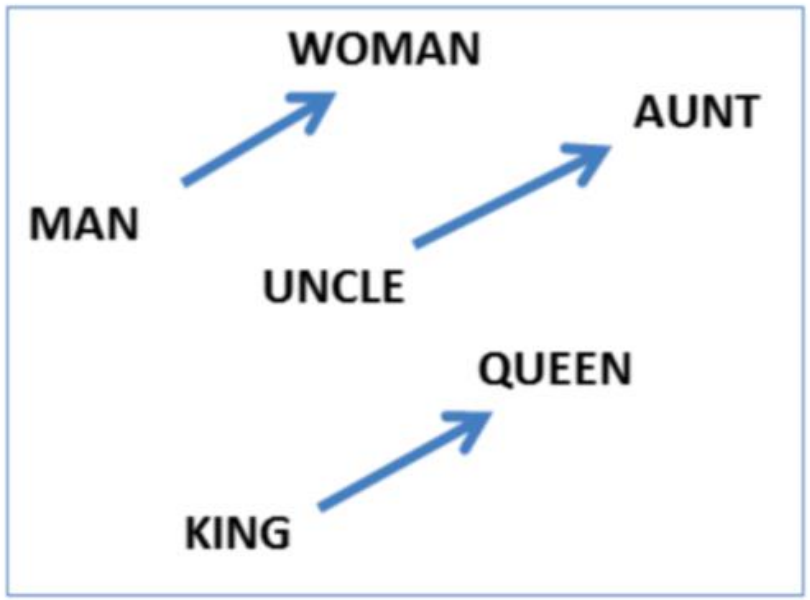
- Skip-gram - предсказание контекста по центральному слову
- CBOW (Continuous Bag-of-Words) - предсказание центрального слова по контексту

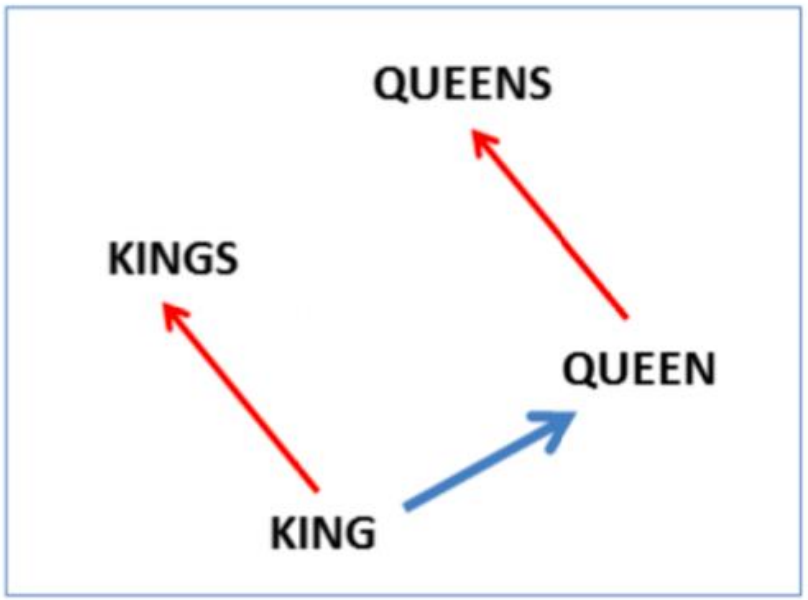


Соотношения между эмбеддингами

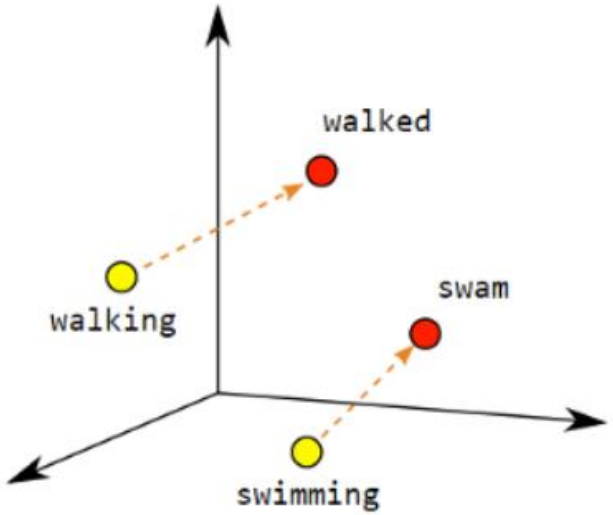
semantic: $v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen})$

syntactic: $v(\text{kings}) - v(\text{king}) + v(\text{queen}) \approx v(\text{queens})$

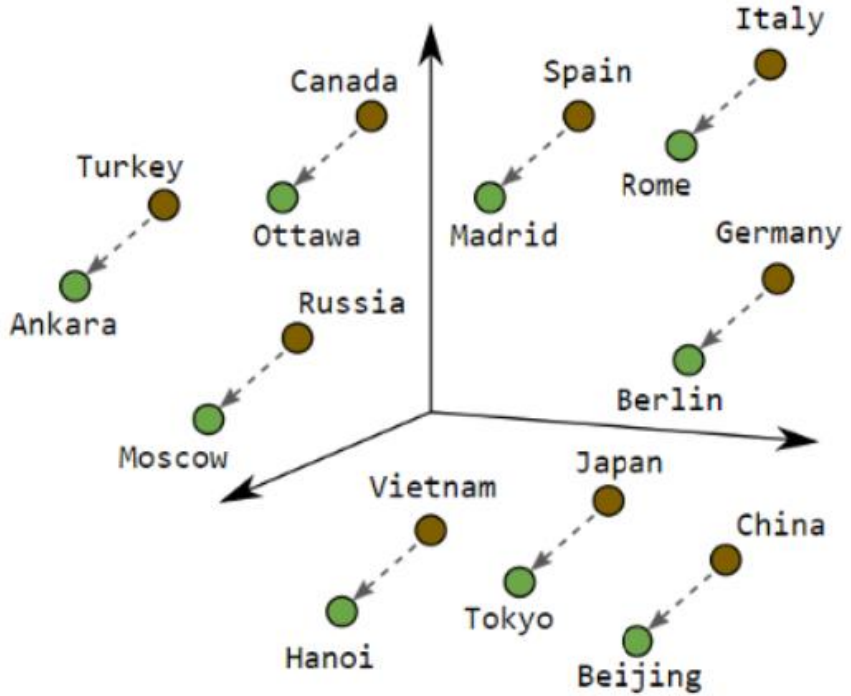




Соотношения между эмбеддингами

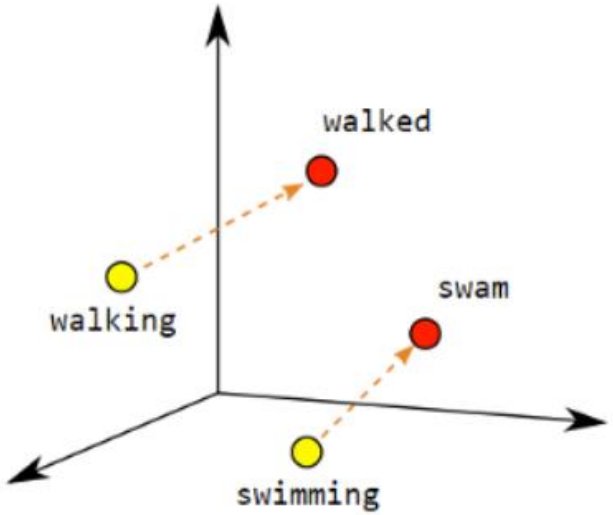


Verb Tense

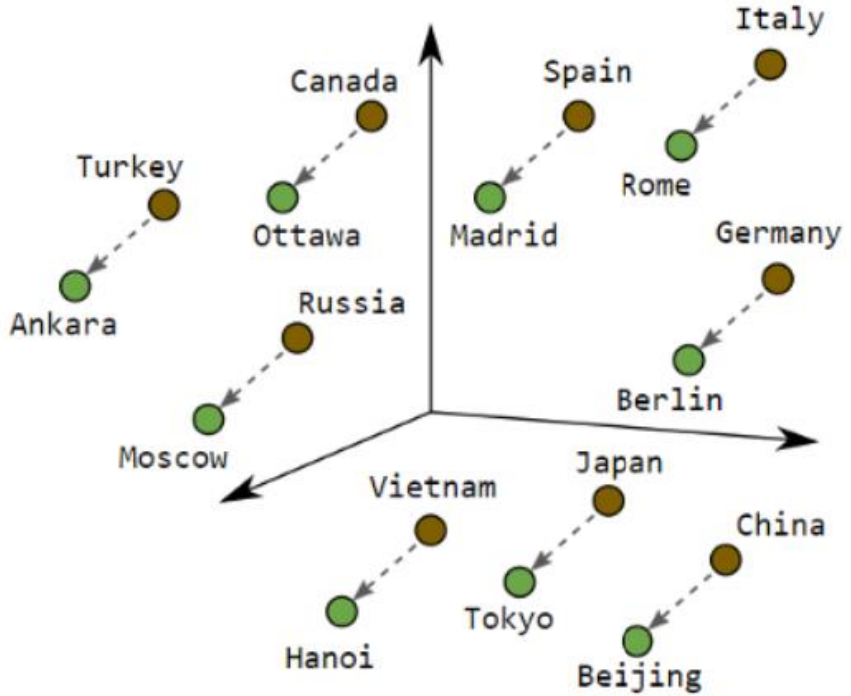


Country-Capital

Соотношения между эмбеддингами



Verb Tense



Country-Capital

Стандартные гиперпараметры

- **Модель**: Skip-Gram с negative sampling;
- **Количество отрицательных примеров**: для малых наборов, 15-20; для больших наборов 2-5.
- **Размерность эмбеддинга**: обычно 300, другие варианты (e.g., 100 or 50) также используются.
- **Размер окна (контекст)**: 5-10.

Word2Vec - summary

Преимущества:

- Эффективно фиксирует семантические связи.
- Эффективно для больших наборов данных.
- Предоставляет осмысленные представления слов.
- Простая архитектура.

Недостатки:

- Могут возникнуть проблемы с редкими словами.
- Игнорирует порядок слов.

GloVe

GloVe

Global Vectors (GloVe)

Обучение на основе глобальных статистик совместных встречаемостей слов, посчитанных по всему корпусу текстов

context vector

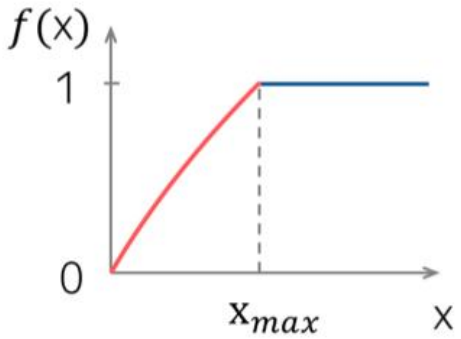
word vector

bias terms (also learned)

$$J(\theta) = \sum_{w,c \in V} f(\mathrm{N}(w,c)) \cdot (u_c^T v_w + b_c + \overline{b_w} - \log \mathrm{N}(w,c))^2$$

Weighting function to:

- penalize rare events
- not to over-weight frequent events



$$\begin{cases} (x/x_{max})^{\alpha} & \text{if } x < x_{max}, \\ 1 & \text{otherwise.} \end{cases}$$

$\alpha = 0.75, x_{max} = 100$

GloVe - summary

Преимущества:

- Эффективно фиксирует глобальную статистику корпуса.
- Хорошо отображает как семантические, так и синтаксические отношения.
- Эффективно фиксирует аналогии слов.
- Простая архитектура (без нейронной сети).

Недостатки:

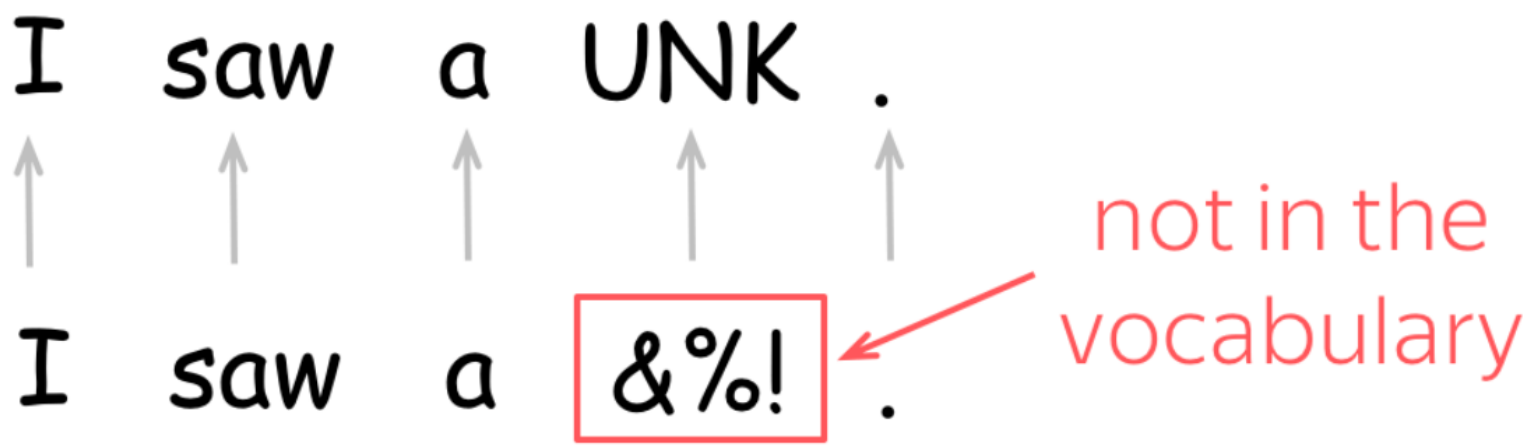
- Требует больше памяти для хранения матриц совместной встречаемости.
- Менее эффективен при очень маленьких корпусах.

fastText

Out-of-vocabulary words

Обучились с фиксированным словарем.

Что делать, если появилось новое слово?



fastText

Идея - считать вектора по n-граммам, составляющим данное слово.

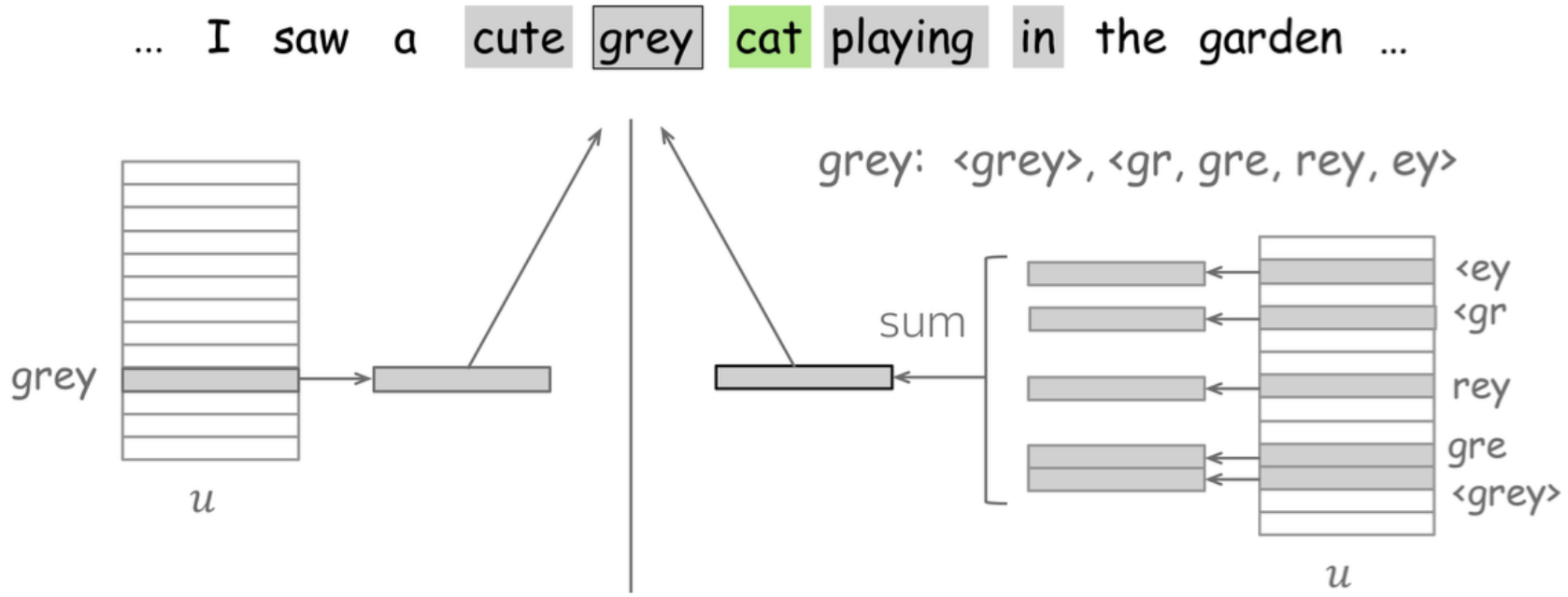


Image credit

Для любого слова можно получить какое-то представление.

FastText - summary

Преимущества:

- Лучшее представление редких слов.
- Способность обработки слов, не входящих в словарный запас.
- Более богатое представление слов благодаря информации о подсловах.

Недостатки:

- Увеличенный размер модели из-за информации об n-граммах.
- Более длительное время обучения по сравнению с Word2Vec.

Методы построения эмбеддингов

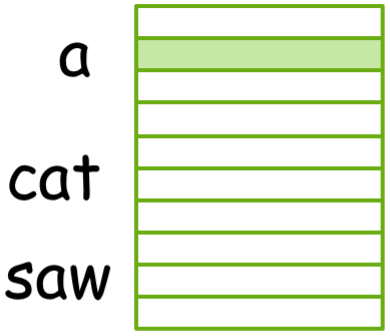
- Word2Vec: используйте, когда семантические связи имеют решающее значение, и у вас большой набор данных.
- GloVe: подходит для разнообразных наборов данных и когда важен охват глобального контекста.
- FastText: выбирайте морфологически богатые языки или когда обработка слов, не входящих в словарный запас, имеет решающее значение.

Свойства эмбеддингов

Внутренняя и внешняя оценка

Внутренняя
рассматривает внутренние свойства эмбеддингов, то есть насколько хорошо они передают смысл.

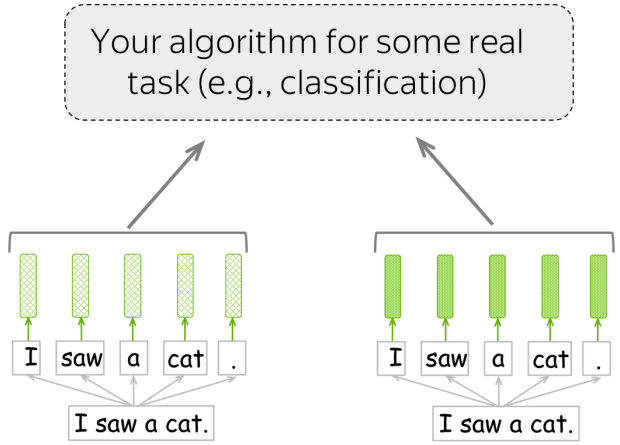
Внешняя
показывает, какие эмбеддинги лучше всего подходят для задачи, которая вас действительно интересует (например, классификация текста, разрешение кореферентности и т. д.).



Look-up table

How well do embeddings capture **meaning**?

- word similarity
- word analogy
- ...



some embeddings

some other embeddings

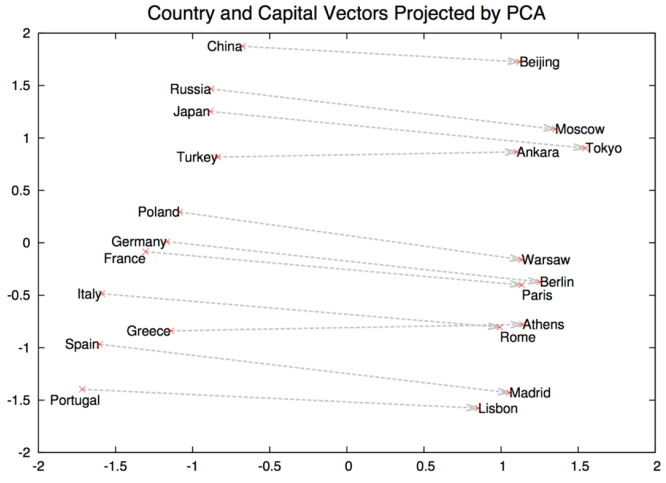
Model with which embeddings performs better?

Train the same model several times: one model for each embedding set

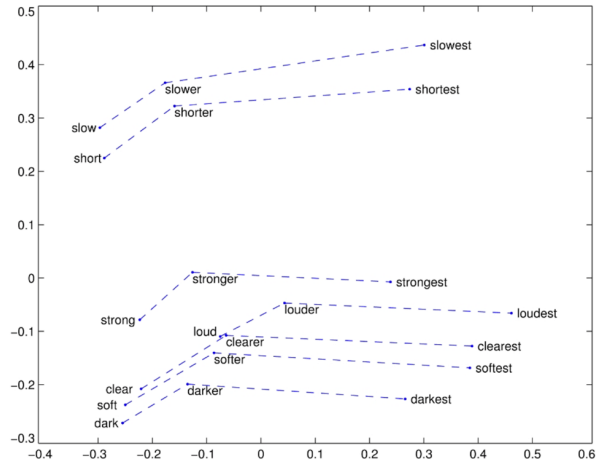
For the same dataset, you can get representations using different word embeddings

Линейная структура

Даны две пары слов для одного и того же отношения, например (мужчина, женщина) и (король, королева), задача состоит в том, чтобы проверить, можем ли мы идентифицировать одно из слов на основе остальных.

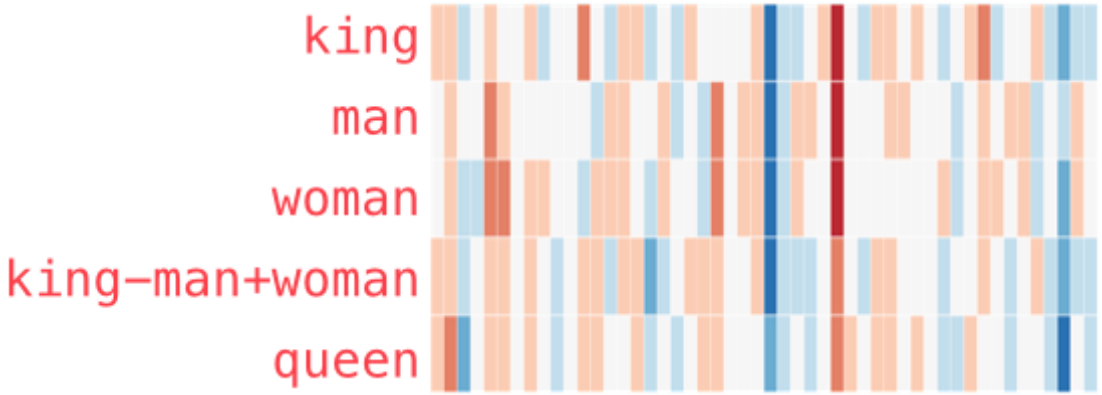


Word2Vec



GloVe

king − man + woman ~= queen



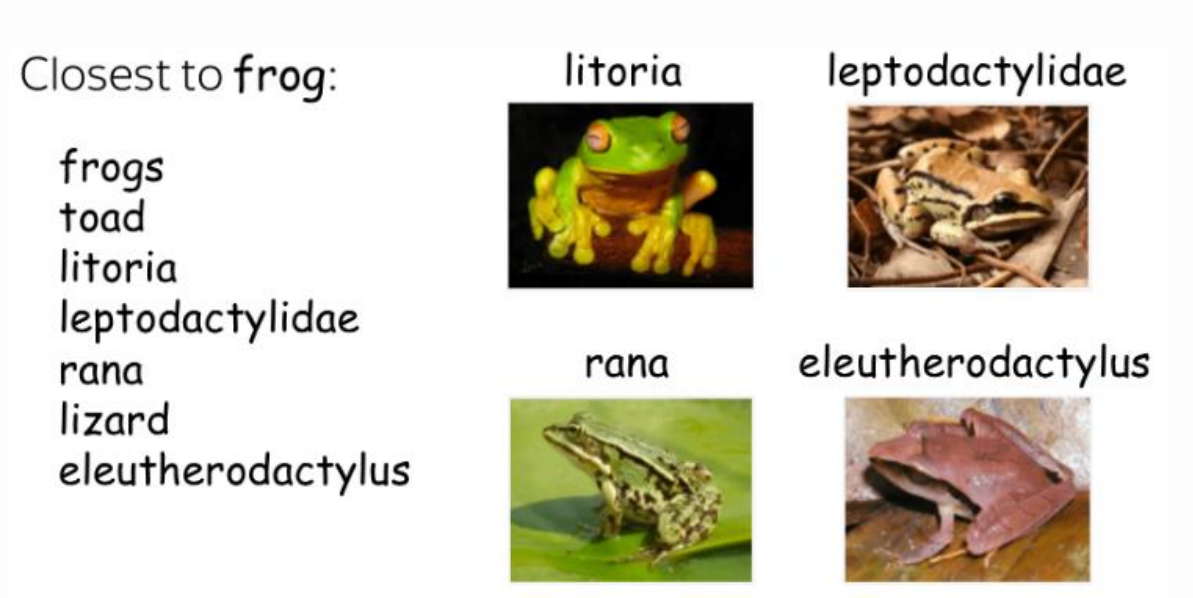
Analogy: **a** is to **a*** as **b** is to __

Task: $v(\mathbf{a}^*) - v(\mathbf{a}) + v(\mathbf{b}) \approx ?$ ⟶

- find the closest vector
- check if it corresponds to the correct word

Ближайшие соседи

Точки (векторы), которые находятся рядом, обычно имеют близкое значение. Иногда даже редкие слова понимаются очень хорошо.



The recipe for building large dictionaries from small ones

https://nlp.stanford.edu/projects/glove/

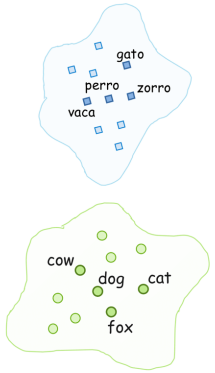
Ingredients:

- corpus in one language (e.g., English)
- corpus in another language (e.g., Spanish)
- very small dictionary

cat ↔ gato
cow ↔ vaca
dog ↔ perro
fox ↔ zorro
...

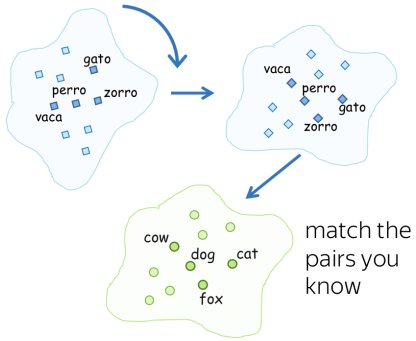
Step 1:

- train embeddings for each language



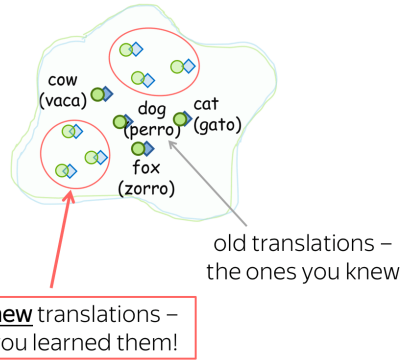
Step 2:

- linearly map one embeddings to the other to match words from the dictionary



Step 3:

- after matching the two spaces, get new pairs from the new matches



Можно дополнить редкие словари и другие языки

Word embeddings + нейросети

Общая концепция

Берем обученные на большом корпусе текстов эмбеддинги, используем в своей задаче

Freeze

- Замораживаем эмбеддинги, учим только другие слои

Finetune

- Инициализируем слой эмбеддингов, потом дообучаем этот слой вместе с остальными слоями

По сути это пример transfer learning

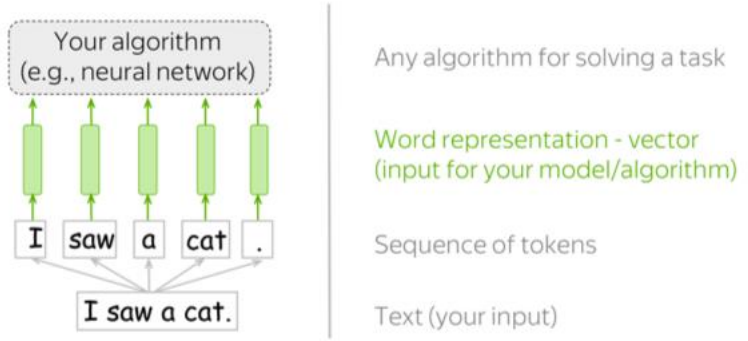


Image credit

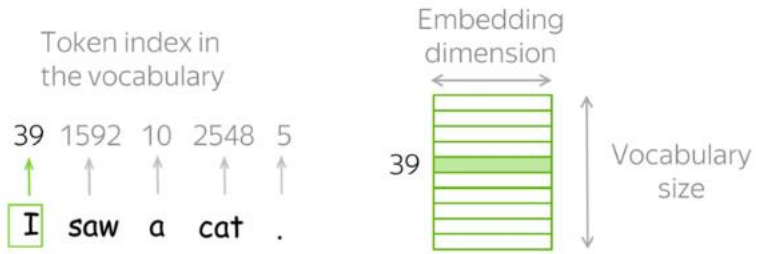
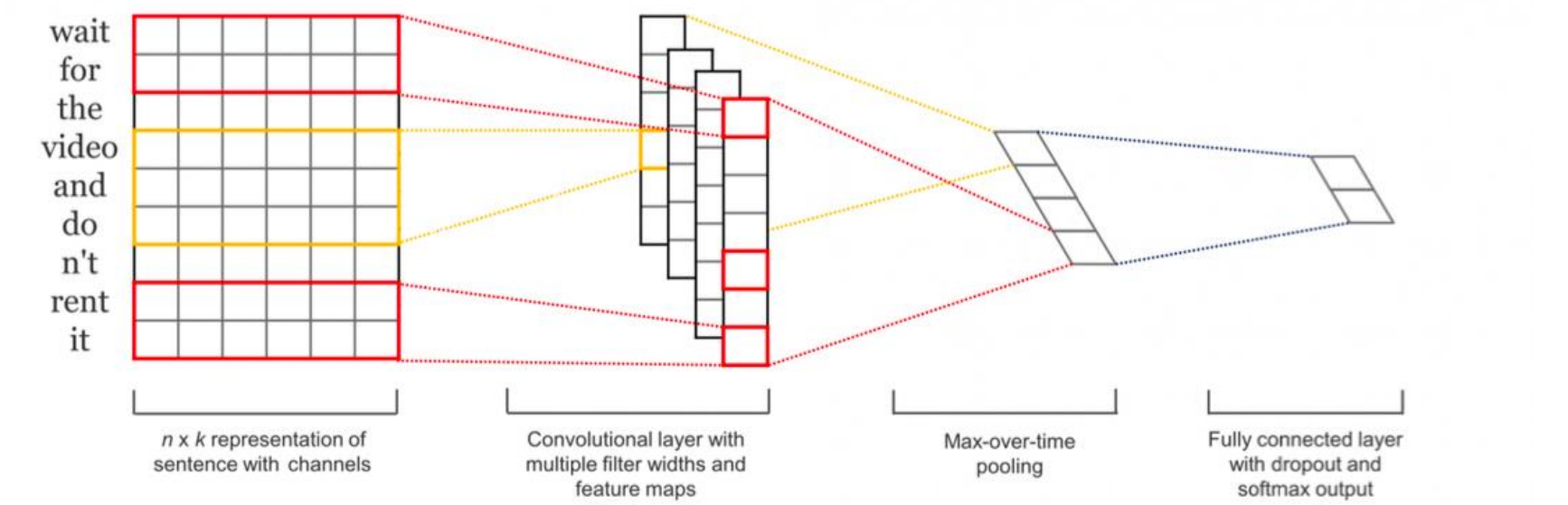


Image credit

Пример: 1D CNN

- Слой эмбеддингов
- 1D свертки с разными kernel size
- Global pooling
- fully connected layers



https://arxiv.org/pdf/1408.5882.pdf

Subword tokenization

Subword Tokenization

Subword tokenization - баланс между представлением на уровне символов и представлением на уровне слов

На уровне символов

- Слишком длинные последовательности
- Теряется семантика, смысл

На уровне слов

- Проблема out-of-vocabulary words
- Слишком большой словарь

Subword Tokenization

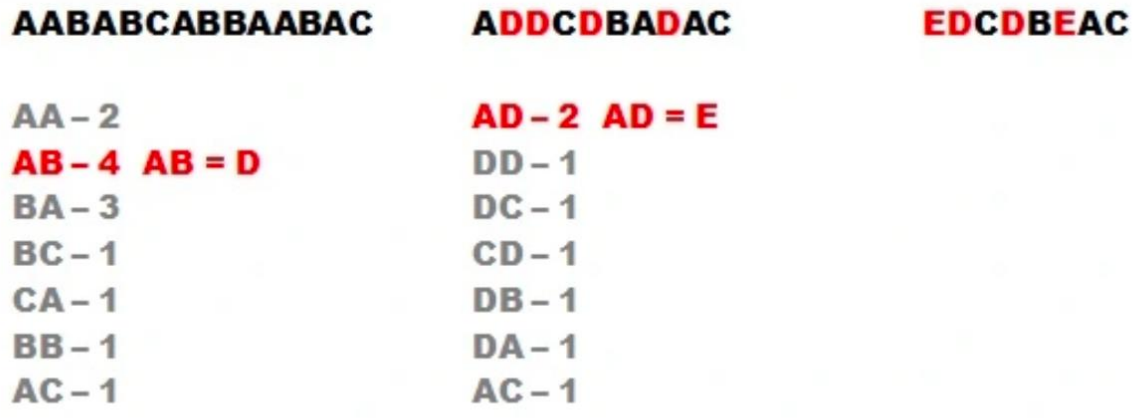
- Часто используемые слова рассматриваются как отдельные токены
- Редкие слова раскладываются на несколько осмысленных подслов

В результате

- Ограниченный размер словаря
- Осмысленные представления
- Способность обработать любое новое слово
- Один токенизатор для всех языков сразу

Byte Pair Encoding

- Берем большой корпус текстов
- Заранее определяем размер словаря (например, 50к)
- Начинаем с отдельных символов как токенов
- На каждой итерации сливаем два токена, которые имеют максимальную частоту совместной встречаемости
- Заканчиваем, когда достигнут размер словаря или максимальная частота равна 1



Anything2Vec

Идею представления объектов в виде эмбеддингов можно расширить на самые разные данные