

Лекция 7. Natural Language Processing. Word embeddings.

Глубинное обучение

Антон Кленицкий

Задачи NLP

—

Text classification

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

Word-level 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,...)

- 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)

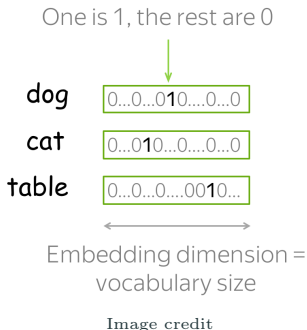
Дальше нужно перевести слова в числовой вид

Представление текста

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

Дальше нужно перевести слова в числовой вид

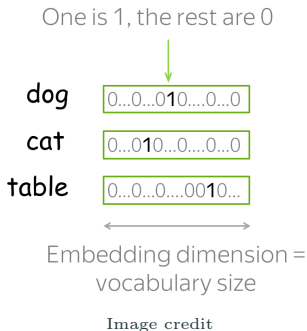
Самый простой вариант - one-hot encoding



One-hot encoding

Минусы one-hot encoding

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



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)

Общая идея:

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

Реализация

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

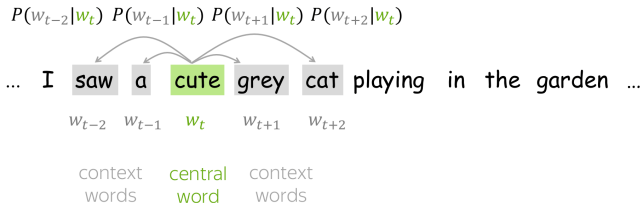


Image credit

- 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



Image credit

- 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

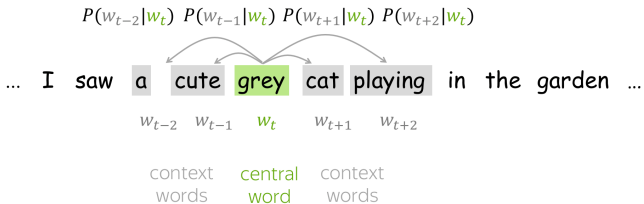


Image credit

Функция потерь - negative log-likelihood:

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

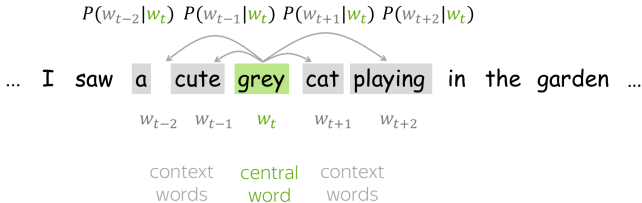


Image credit

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

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

о - outside word, с - central word

Обучаем стохастическим градиентным 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 .

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

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

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.

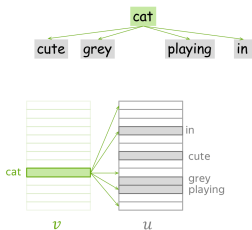
Выход - сэмплирование отрицательных примеров

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

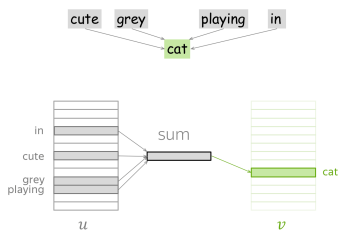
CBOW vs Skip-Gram

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

... I saw a cute grey cat playing in the garden ...



Skip-Gram: from **central** predict context
(one at a time)



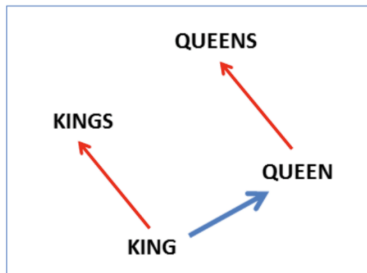
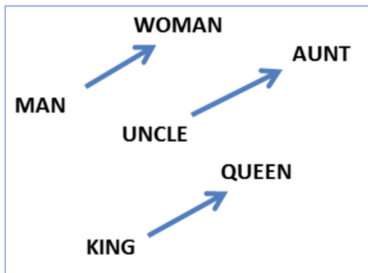
CBOW: from sum of context predict **central**

Image credit

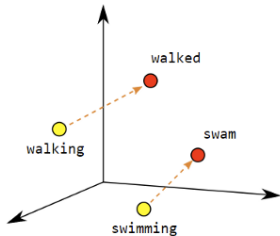
Соотношения между эмбедингами

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

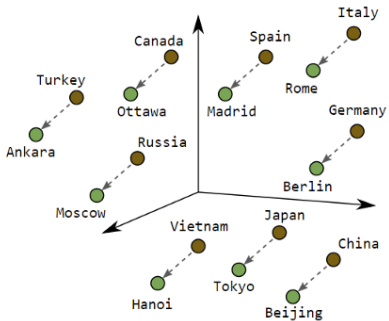
syntactic: $v(\text{kings}) - v(\text{king}) + v(\text{queen}) \approx v(\text{queens})$



Соотношения между эмбедингами



Verb Tense



Country-Capital

Global Vectors (GloVe)

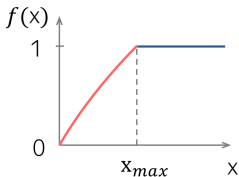
Обучение на основе глобальных статистик совместных встречаемостей слов, посчитанных по всему корпусу текстов

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

context vector
word vector
bias terms (also learned)

Weighting function to:

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



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

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

Out-of-vocabulary words

Обучились с фиксированным словарем.

Что делать, если появилось новое слово?

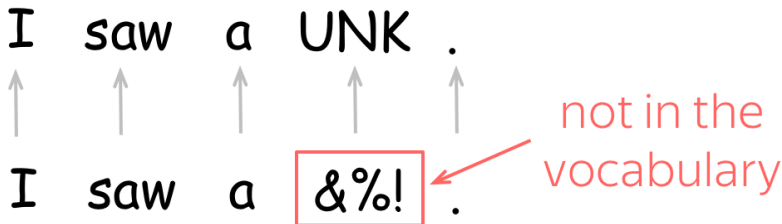


Image credit

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

... I saw a cute grey cat playing in the garden ...

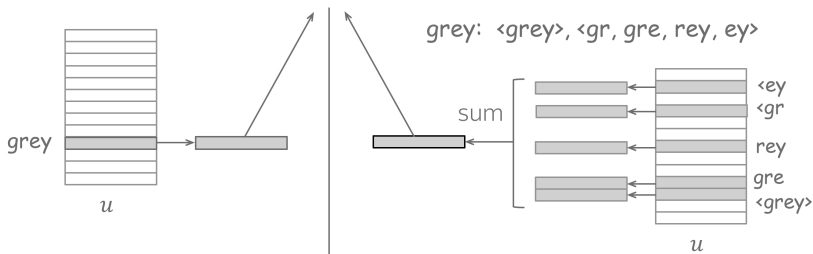


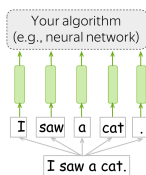
Image credit

Для любого слова можно получить какое-то представление.

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

Word embeddings + нейросети

Общая схема



Any algorithm for solving a task

Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)

Image credit

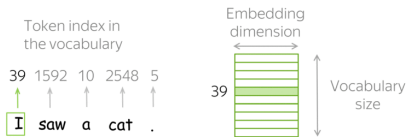


Image credit

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

Freeze

- Замораживаем эмбединги, учим только другие слои

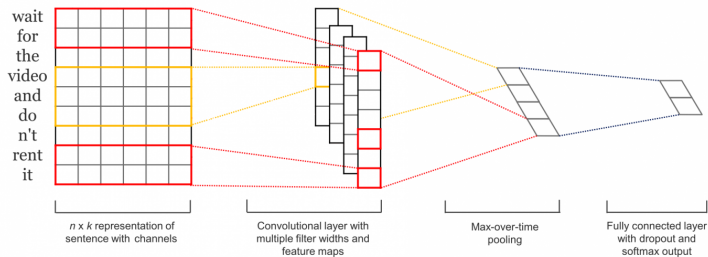
Finetune

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

По сути это пример transfer learning

1D CNN

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



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

- 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

- 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

Byte Pair Encoding

AABABCABBAABAC

AA - 2

AB - 4 **AB = D**

BA - 3

BC - 1

CA - 1

BB - 1

AC - 1

A**DD****C****D****B****A****D****A****C**

AD - 2 **AD = E**

DD - 1

DC - 1

CD - 1

DB - 1

DA - 1

AC - 1

E**D****C****D****E****A****C**

- WordPiece
- Unigram Language Model

В следующий раз

- Рекуррентные нейронные сети (RNN)
- LSTM, GRU
- Seq2seq models