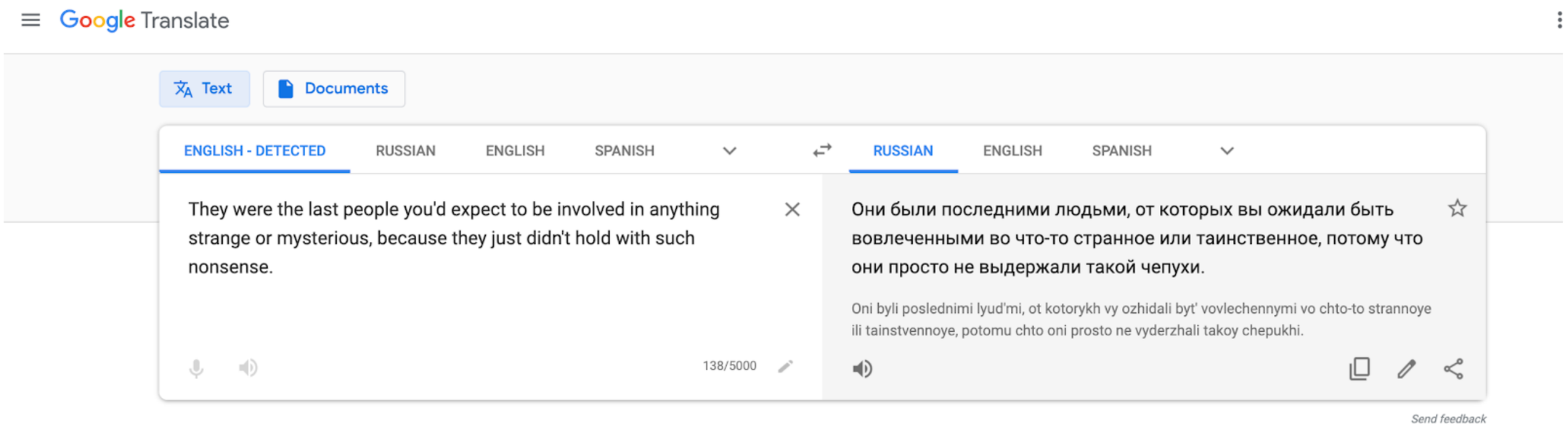
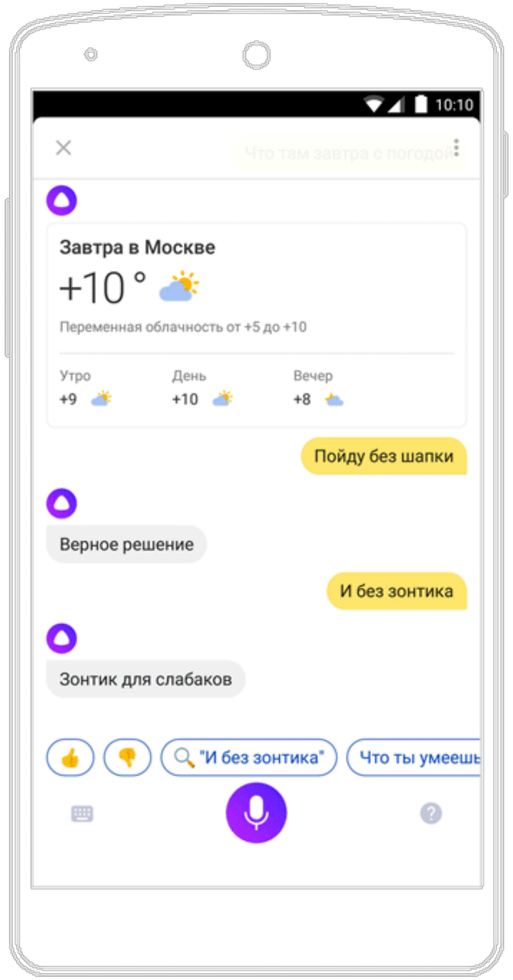
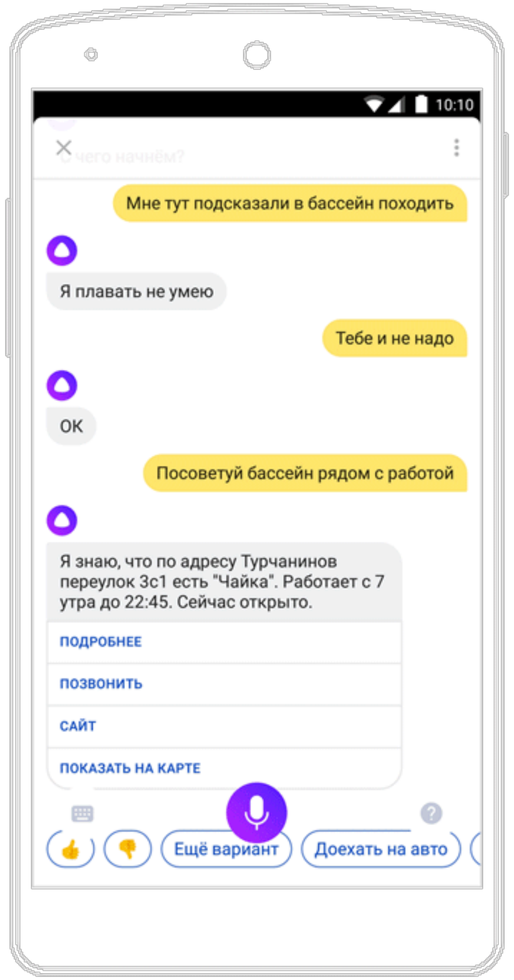
Обработка естественного языка

NLP: приложения

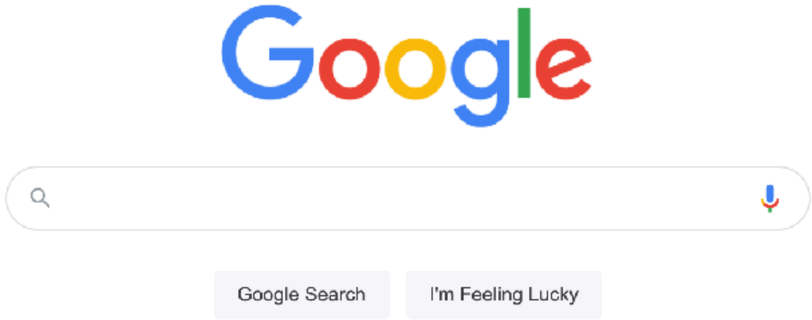


NLP: приложения





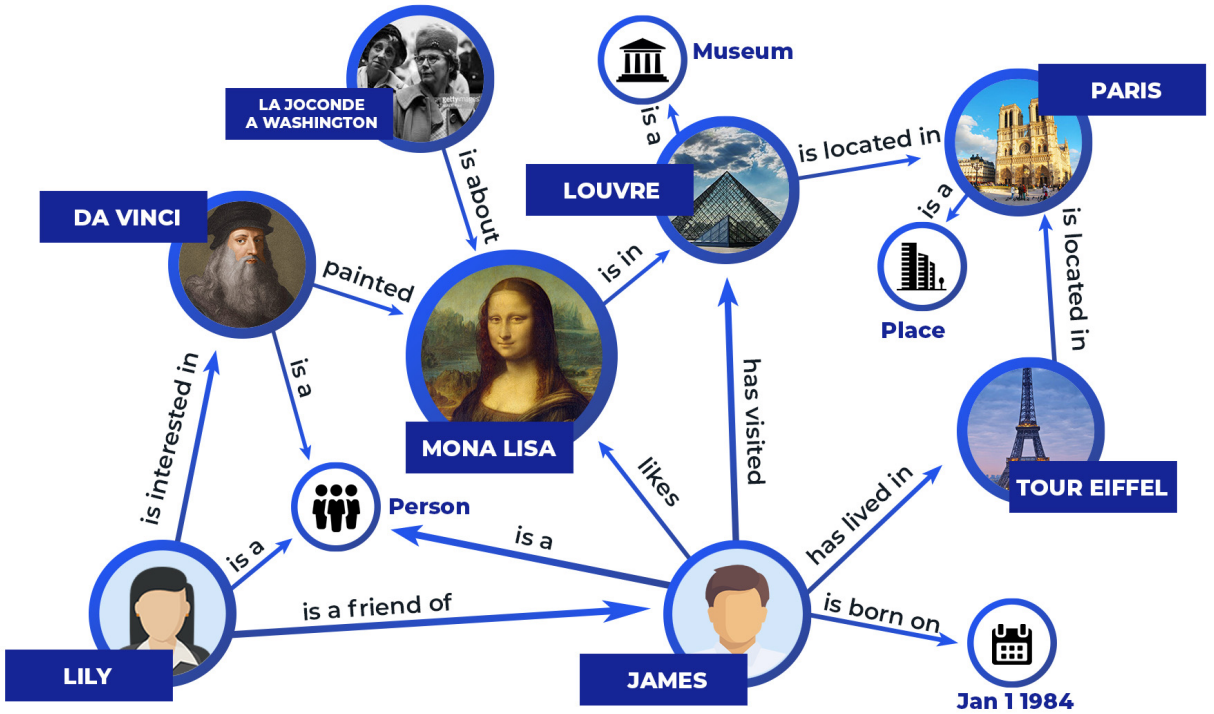
NLP: приложения



Что такое NLP?

Текстовые данные

- Большая часть данных в мире представлена в текстовом виде
- Текстовые данные могут быть:
  - структурированными (графы знаний, базы данных)



https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/

Текстовые данные

- Большая часть данных в мире представлена в текстовом виде
- Текстовые данные могут быть:
  - структурированными (графы знаний, базы данных)
  - неструктурированными (сырые тексты)

```
Введение в обработку естественного языка.
Под авторством Сидорова Ивана Петровича.
Вступление.
Настоящее пособие предназначено для ...
```

```
Чек
Магазин канцелярских товаров
1. Шариковая ручка (син) 23 руб.
2. Тетрадь клет (12 л) 5 руб.
...
```

Текстовые данные

- Большая часть данных в мире представлена в текстовом виде
- Текстовые данные могут быть:
  - структурированными (графы знаний, базы данных)
  - неструктурированными (сырые тексты)
  - частично структурированными (JSON, XML)

```
{
    "type": "учебник",
    "title": "Введение в обработку естественного языка.",
    "author": "Сидоров Иван Петрович",
    "introduction": "Настоящее пособие предназначено для \dots
",
    \dots
}

```

Естественный язык

- Естественный язык – способ общения между людьми
- Можем противопоставить его *формальным* и *искусственным* языкам:
  - Языки программирования – Programming Language Processing (PLP)
- Немного формализма:
  - Язык – это множество допустимых цепочек символов из некоторого алфавита
  - Текст – это цепочка, построенная по некоторым правилам
  - Алфавит – это множество символов, из которых строятся тексты
  - Каждая цепочка должна нести некоторую информацию (на деле не всегда)

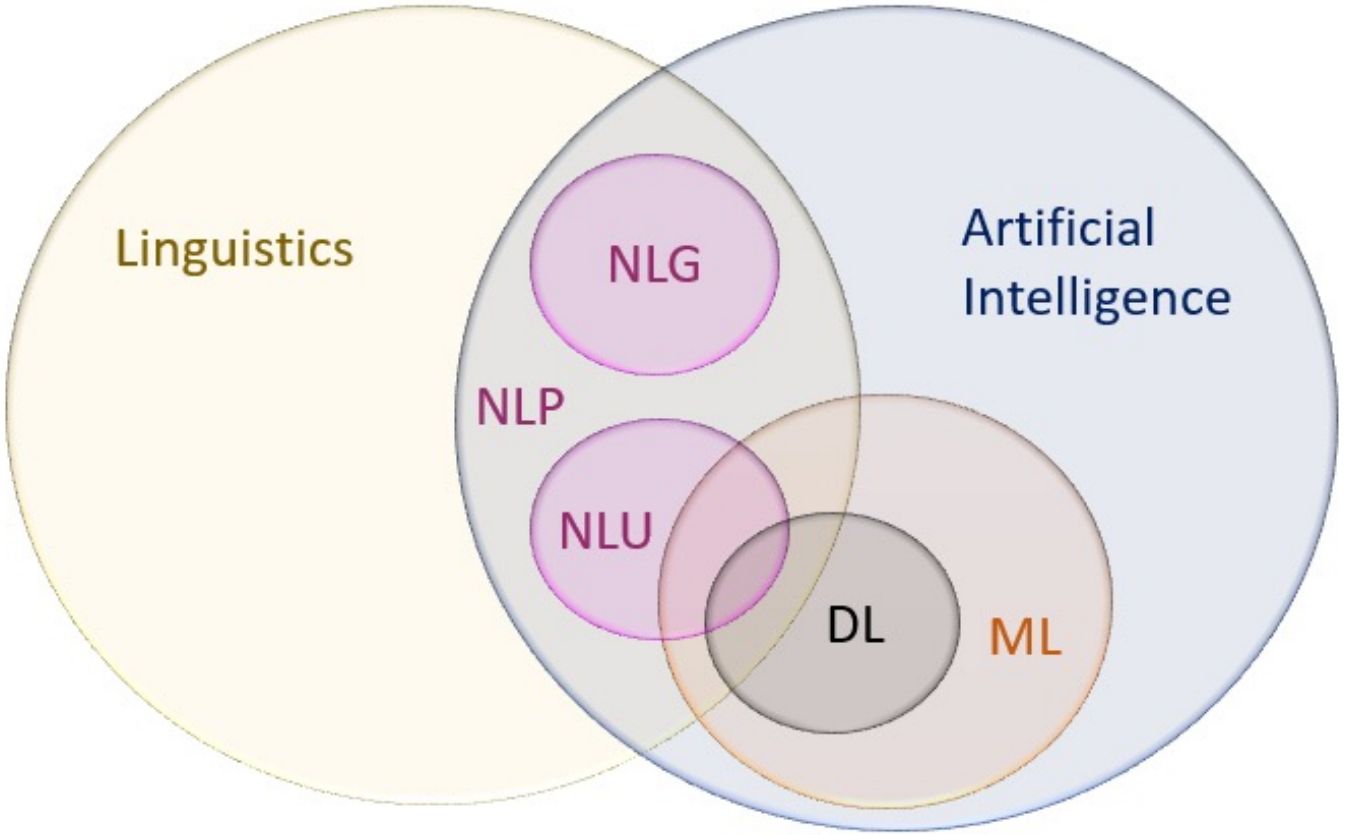
- «Глокая куздра штеко будланула бокра и кудрячит бокренка» (Л.В. Щерба, 1930-е)

Правила языка

- Выстраивается некоторая иехрархия:
  - Графематические – как разделять слова и предложения между собой
  - Морфологические – как строить и изменять слова
  - Синтаксические – как согласовывать словоформы друг с другом
  - Семантические – как применять все предыдущие правила, чтобы сообщить необходимую информацию
  - Стилистические – «уместность» словоупотребления в конкретной ситуации
  - И т.п.

Обработка естественного языка (NLP)

- Положение NLP среди наук по анализу и обработке данных:



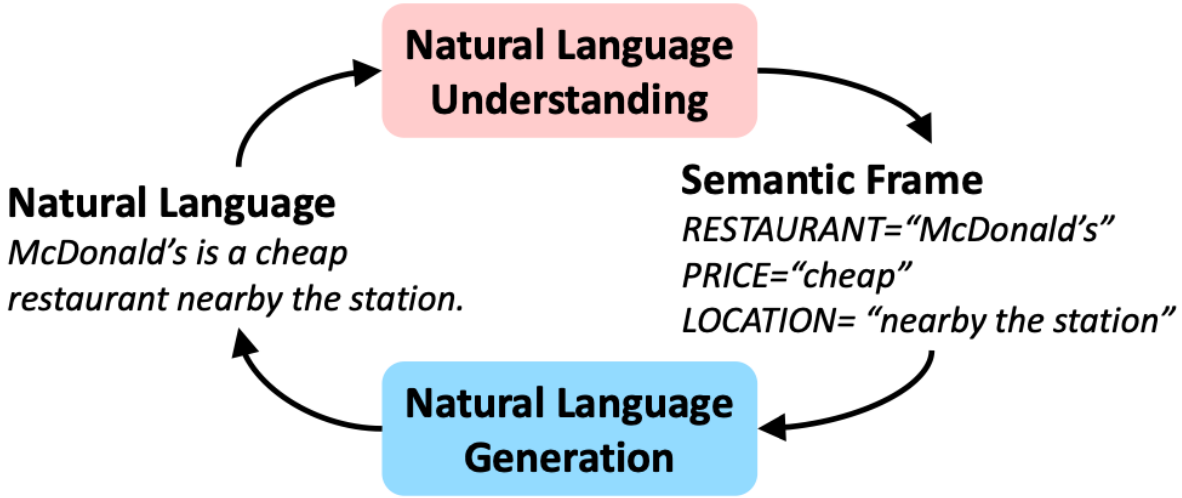
https://medium.com/analytics-vidhya/natural-language-processing-nlp-based-chatbots-7b2436428256

Структура NLP

- Внутри NLP условно выделяются два направления:
  - понимание языка (NLU)
  - генерация языка (NLG)

- Текст –> NLU –> смысл –> NLG –> текст

- Смежные области:
  - распознавание (ASR)
  - генерация (TTS) речи



https://arxiv.org/abs/1905.06196

Пирамила NI P

P.S. В самом низу графематический уровень

Особенности NLP

- Базовая структурная единица языка — слово
  - Даже вне контекста оно несет полезную информацию
  - У слов есть различные словоформы в зависимости от контекста
  - Многозначность слова (полисемия, омонимия)

- Текст без дополнительной разметки имеет внутреннюю структуру, определяемую языком на разных уровнях:
  - текст (порядок реплик)
  - предложение (синтаксис)
  - слова и словосочетания (морфология, синтаксис)

- Наличие огромных массивов сырых текстов и структуры в них позволяет обучать большие общеязыковые модели

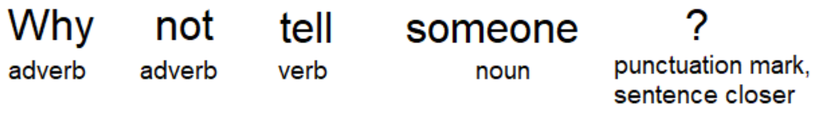
- Существует много лингвистических ресурсов, которые помогают в различных задачах обработки текстов

NLP: задачи

Классификация:

- Part-of-Speech Tagging
- Named-Entity Recognition
- Sentiment Analysis

…



Finished fixing my twitter...I had to unfollow and follow everyone again	Negative
@DinahLady I too, liked the movie! I want to buy the DVD when it comes out	Positive

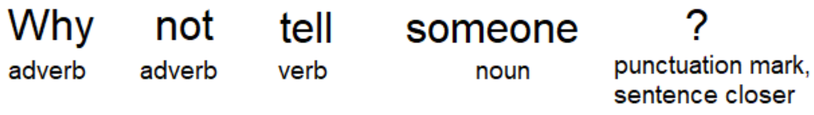
Part-of-Speech Tagging (POS Tagging) — это задача присваивания каждой словоформе в тексте её части речи (например, существительное, глагол, прилагательное и т.д.). Это один из ключевых этапов обработки естественного языка (NLP), поскольку части речи помогают понять структуру и смысл предложения.

NLP: задачи

Классификация:

- Part-of-Speech Tagging
- Named-Entity Recognition
- Sentiment Analysis

…



Finished fixing my twitter...I had to unfollow and follow everyone again	Negative
@DinahLady I too, liked the movie! I want to buy the DVD when it comes out	Positive

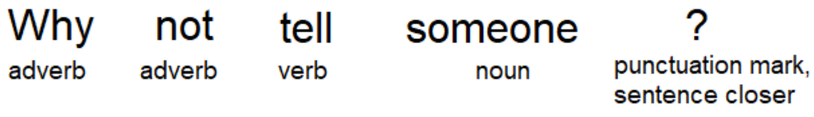
Named-Entity Recognition (NER) — это задача обработки естественного языка (NLP), направленная на автоматическое распознавание именованных сущностей в тексте. Задача NER заключается в идентификации и классификации сущностей (таких как имена людей, организации, локации, даты и т.д.) в предложениях или документах.

NLP: задачи

Классификация:

- Part-of-Speech Tagging
- Named-Entity Recognition
- Sentiment Analysis

…



Finished fixing my twitter...I had to unfollow and follow everyone again	Negative
@DinahLady I too, liked the movie! I want to buy the DVD when it comes out	Positive

Sentiment Analysis (анализ тональности) — это задача обработки естественного языка (NLP), направленная на определение эмоциональной окраски текста. Целью анализа тональности является понимание того, выражает ли текст положительное, отрицательное или нейтральное мнение. Это важно для таких областей, как анализ отзывов, мониторинг социальных сетей, анализ мнений и т.д.

NLP: задачи

Классификация:

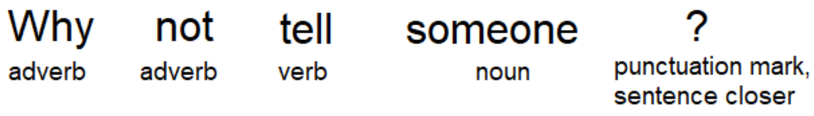
- Part-of-Speech Tagging
- Named-Entity Recognition
- Sentiment Analysis

…

Генерация

- Machine Translation
- Question Answering

…



Finished fixing my twitter...I had to unfollow and follow everyone again	Negative
@DinahLady I too, liked the movie! I want to buy the DVD when it comes out	Positive

Language Modeling

Этапы решения NLP задач

Этапы решения NLP-задачи

- Всё так же, как и при обработке других типов данных:
  - Выбор верной метрики качества
  - Сбор обучающих и тестовых данных
  - ***Предобработка данных***
  - ***Формирование признакового описания текста***
  - Выбор подхода и класса моделей
  - Обучение моделей и настройка решения

- Предположим, что данные есть в некотором подходящем для работы формате

Инструменты для работы с текстами

- В обработке текстов часто полезны библиотеки общего назначения:
  - re/regex — модули для работы с регулярными выражениями
  - numpy/pandas/scipy/sklearn — базовые библиотеки для анализа данных и ML
  - pytorch — одна из основных библиотек для обучений нейросетей

- Специализированные библиотеки:
  - nltk
  - gensim
  - transformers

Регулярные выражения

- Регулярные выражения появились от т.н. регулярных автоматов (классификация грамматик по Хомскому)
- По факту это некоторый строковый шаблон, на соответствие которому можно проверить текст

- С синтаксисом можно ознакомиться на странице выбранного инструмента, но основные правила одинаковы, например:
  - . — означает наличие одного любого символа
  - [a-zA-Z0-9] — означает множество символов из заданного диапазона
  - +, *, ? — показывают, что следующий перед ними символ или последовательность символов должны повториться >= 1 раз (r+), >= 0 раз ((xa-)*) и 0/1 раз (r?)

https://habr.com/ru/articles/545150/

Предобработка текстов

- Пусть дана коллекция текстовых документов - текст представляет собой одну строку и алфавитных и неалфавитных символов
- Обрабатывать его в таком виде неудобно, сперва нужно выделить числовые признаки
- Базовые шаги предобработки:
  1. токенизация
  2. приведение к нижнему регистру
  3. удаление пунктуации
  4. удаление стоп-слов
  5. фильтрация слов по частоте/длине/регулярному выражению
  6. нормализация слов - лемматизация или стемминг

Токенизация

- Токенизацию можно производить по словам и/или предложениям
- Используются как подходы, основанные на правилах, так и ML-модели
- В nltk есть много различных токенизаторов, например RegexpTokenizer и sent_tokenize
- Часто слова грубо выделяют разделением по пробелам с помощью метода split

```
text = 'Набор слов, составляющий какое-то предложение.'
print(text.split(' '))
#['Набор', 'слов,', 'составляющий', 'какое-то', 'предложение.']
```

Регистр и пунктуация

- Есть задачи, в которых пунктуация и регистр несут важную информацию
- Это важно для определения границ предложений, для решения задачи выделения именованных сущностей
  - в комнату вошел ***лев*** и, потянувшись, достал из кармана сигару
  - ***лев*** обитает в саванне, в арктике не обитает
- В задаче анализа тональности существенное значение имеют смайлы (текстовые или символы в Unicode):
  - ❌ Одежда у вас в магазине очень своеобразная***:/***
  - ✅ Одежда у вас в магазине очень своеобразная***:)***

Нормализация слов

- Слова в тексте могут иметь различную формы, часто такая информация скорее мешает, чем помогает анализу
- Для нормализации применяется один из подходов:
  - *лемматизация* (pymorphy2, pymystem3) – приводит слова к нормальной форме
  - *стемминг* (реализации в nltk) – стемминг приводит слова к псевдооснове (убирает окончания и формообразующие суффиксы)

```
1 import pymorphy2
2 text = 'я хотел бы поговорить с вами'.split(' ')
3 lemmatizer = pymorphy2.MorphAnalyzer()
4 print([lemmatizer.parse(t)[0].normal_form for t in text])
5 # ['я', 'хотеть', 'бы', 'поговорить', 'с', 'вы']
```

Фильтрация словаря

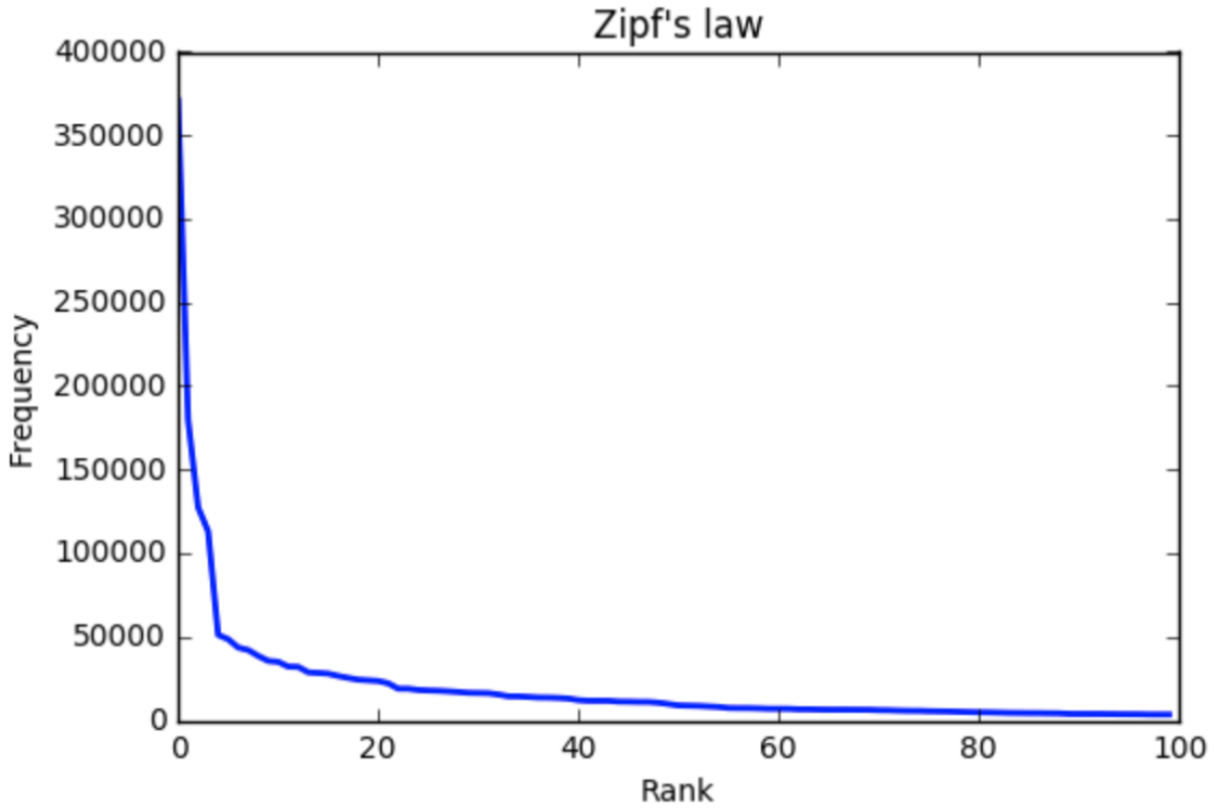
- Часто из текстов нужно удалять лишние слова
- Обычно это стоп-слова - очень редкие и очень частые слова
- К стоп-словам относятся союзы, предлоги, модальные глаголы, местоимения, вводные слова
- Большой набор стоп-слов есть в nltk:

```
from nltk.corpus import stopwords
sw = set(stopwords.words('russian'))

for w in ['я', 'хотеть', 'бы', 'поговорить', 'с', 'вы']:
    if w not in sw:
        print(w, end=' ')
# хотеть поговорить
```

Фильтрация словаря

- Слишком частые или редкие слова тоже могут оказаться вредными
- Такие слова могут и мешать обучению модели, и увеличивать затраты ресурсов памяти и времени счета
- При обработке коллекции стоит проверить выполнение закона Ципфа:



Признаковые описания документов

Признаковые описания документов

- Обычно в ML данные представляют собой матрицу «объекты-признаки»:

Номер автомобиль	Тип топлива	Мощность двигателя	...	Масса
1	Бензин	120	...	1700
...	...	...	...	...
N	Дизель	160	...	2100

- Для текстов тоже нужно как-то получить такую матрицу

Модель мешка слов

- Можно проверять наличие всех возможных слов из некоторых словаря:

Номер текста	Содержит «абрикос»	. . .	Содержит «яблоко»
1	0	. . .	1
. . .	. . .	. . .	. . .
N	1	. . .	0

- Пусть значением признака будет не наличие слова, а число его вхождений в документ («мешок слов»):

Номер текста	Вхождений «абрикос»	. . .	Вхождений «яблоко»
1	0	. . .	23
. . .	. . .	. . .	. . .
N	2	. . .	0

TF-IDF

- Представление «мешка слов» часто используется при обработке текстов, но частота встречаемости слов не самый информативный признак
- Идея: хотим выделить слова, которые часто встречаются в данном тексте, и редко — в других текстах - используем значения TF-IDF:

$$v_{wd} = tf_{wd} \times \log \frac{N}{df_w}$$

- tf_{wd} — доля слова w в словах документа d
- df_w — число документов, содержащих w
- N — общее число документов

«Мешок слов» и TF-IDF в Python

```
1  from sklearn.feature_extraction.text import CountVectorizer
2  from sklearn.feature_extraction.text import TfidfVectorizer
3
4  c_vectorizer, t_vectorizer = CountVectorizer(), TfidfVectorizer()
5  corpus = [
6       'This is the first document.',
7       'This is the second second document.',
8       'And the third one.',
9       'Is this the first document?',
10 ]
11 X_c = c_vectorizer.fit_transform(corpus)
12 X_t = t_vectorizer.fit_transform(corpus)
```

Коллокации

- N-граммы — устойчивые последовательности из N слов, идущих подряд («машина опорных векторов»)
- Коллокация — устойчивое сочетание слов, не обязательно идущих подряд («Он сломал своему противнику руку»)
- Часто коллокациями бывают именованные сущности (но не всегда)
- Методы получения N-грамм:
  - на основе частот встречаемости (sklearn, nltk)
  - на основе морфологических шаблонов (Томита, YARGY-парсер)
  - с помощью ассоциации и статистических критериев на основе частот совместных встречаемостей (nltk, TopMine)
  - иные подходы (RAKE, TextRank)

Меры ассоциации биграмм

- Поточечная совместная информация (Pointwise Mutual Information, PMI):

$$\mathrm{PMI}(w_1, w_2) = \log \frac{f(w_1, w_2)}{f(w_1) f(w_2)}$$

- T-Score (по сути — тест Стьюдента):

$$\mathrm{T}_{\mathrm{score}}(w_1, w_2) = \frac{f(w_1, w_2) - f(w_1) f(w_2)}{\sqrt{f(w_1, w_2)/N}}$$

  - w_i — слово
  - $f(\cdot)$ — частота слова или биграммы

Представления текста

Представления текста

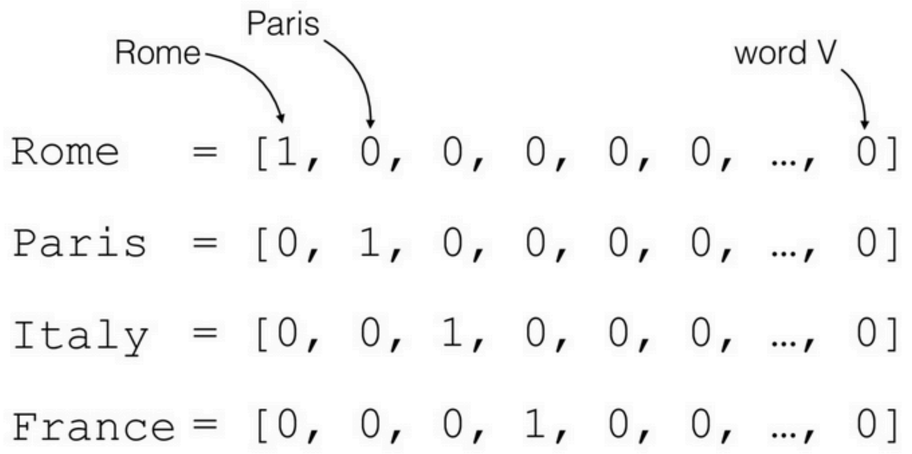
Хотим: перевести слова в числовой вид

Представления текста

Хотим: перевести слова в числовой вид

One-hot encoding

Создаем словарь всех слов (размера V) и нумеруем их

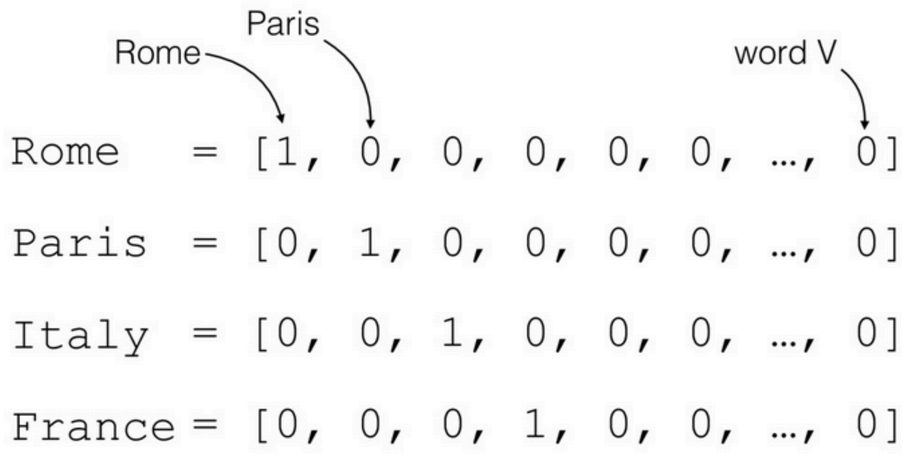


Представления текста

Хотим: перевести слова в числовой вид

One-hot encoding

Создаем словарь всех слов (размера V) и нумеруем их



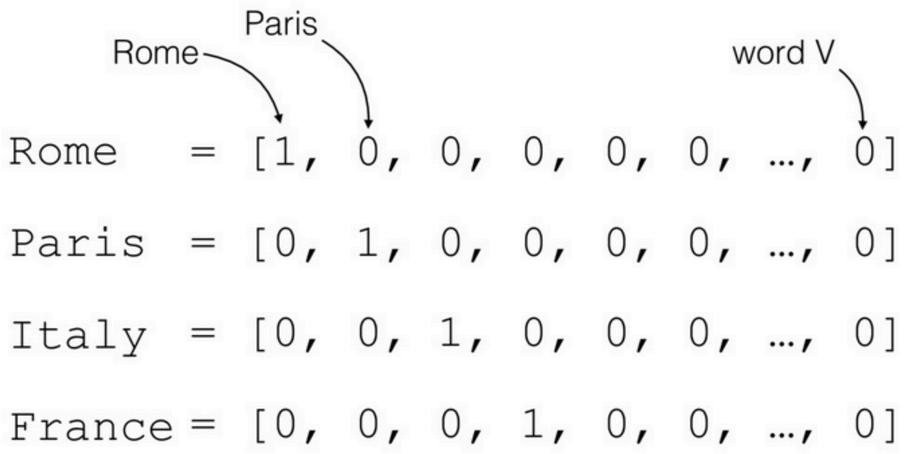
Все вектора ортогональны друг другу $\text{cos\_sim}(w_1, w_2) = \frac{w_1^T w_2}{\|w_1\|^2 \|w_2\|^2} = 0$

Представления текста

Хотим: перевести слова в числовой вид

One-hot encoding

Создаем словарь всех слов (размера V) и нумеруем их



Все вектора ортогональны друг другу $\cos\_sim(w_1, w_2) = \frac{w_1^T w_2}{\|w_1\|^2 \|w_2\|^2} = 0$

Ортогональные вектора — это такие вектора, угол между которыми равен 90 градусов, что значит, что их скалярное произведение равно нулю.
Косинусное сходство (cosine similarity) — это мера, используемая для вычисления схожести между двумя векторами в многомерном пространстве. Оно определяется как косинус угла между векторами.
Основное преимущество косинусного сходства в том, что оно позволяет сравнивать векторы, игнорируя их длину, и фокусируется исключительно на их направлениях.
Неправильная интерпретация: One-hot encoding не учитывает семантические связи между словами (например, "кот" и "собака" рассматриваются как равные, хотя они имеют разное значение).

Представления текста

Хотим: перевести слова в числовой вид

Идея: вектора слов, которые встречаются в одном контексте, должны быть близки $\text{cos\_sim}(w_1, w_2) \approx w_1^T w_2$

____ is the most beautiful capital city in Europe

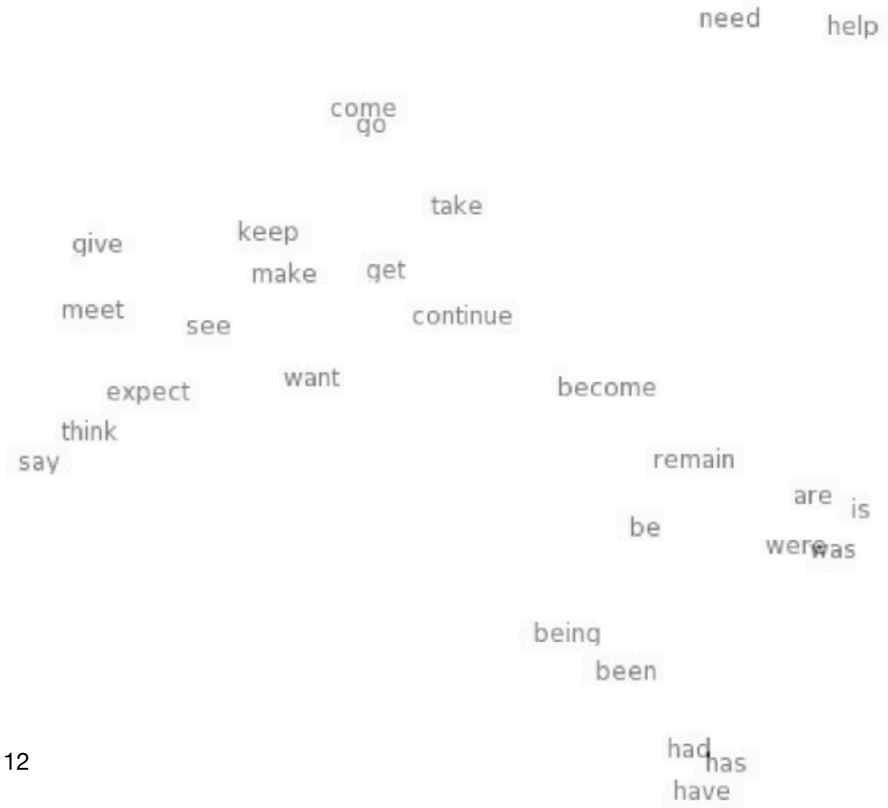
Rome? Paris?

Представления текста

Хотим: перевести слова в числовой вид

Идея: вектора слов, которые встречаются в одном контексте, должны быть близки $\text{cos\_sim}(w_1, w_2) \approx w_1^T w_2$

$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

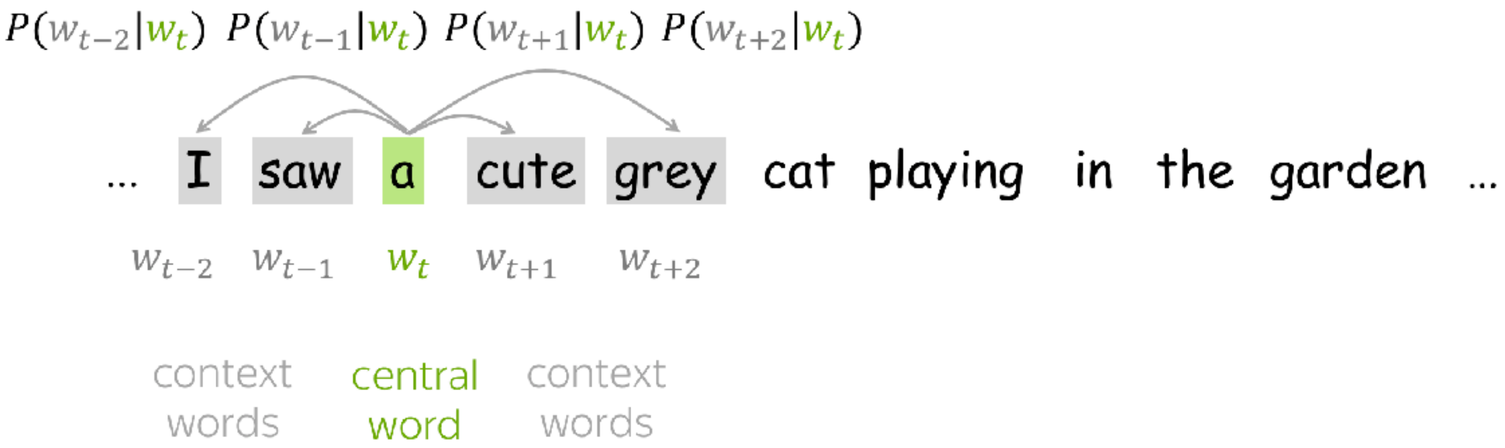


Image credit

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

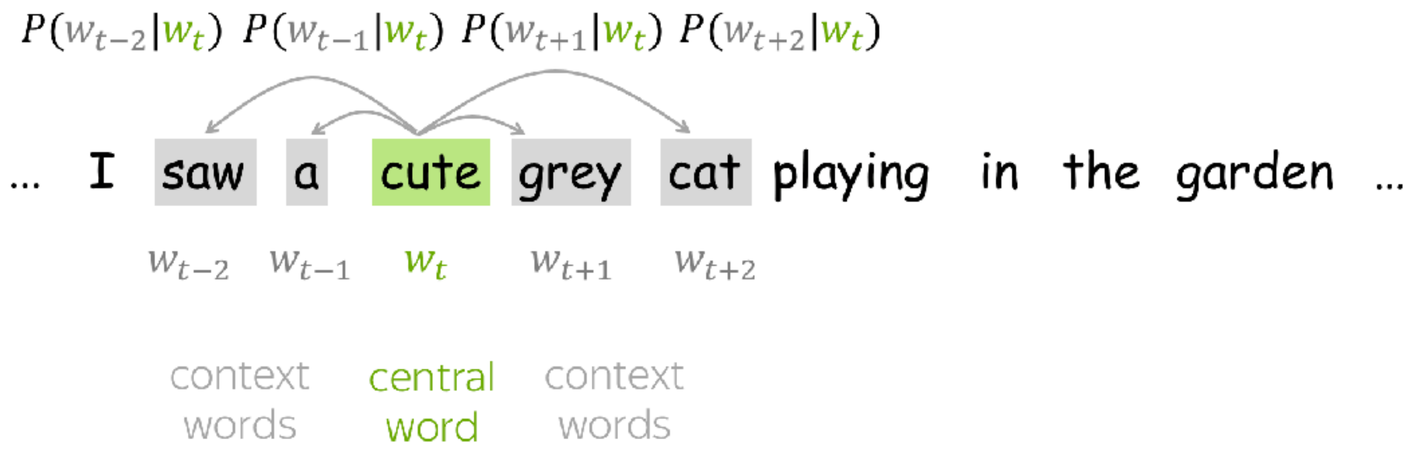


Image credit

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

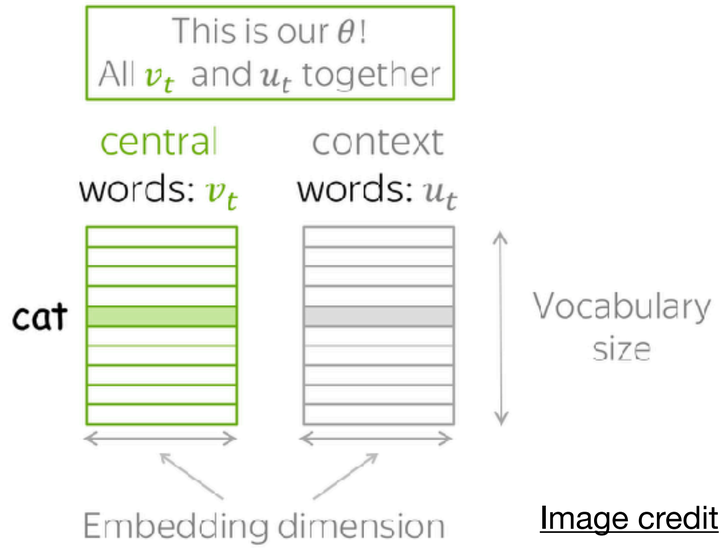
Как задать вероятность?

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$



Для каждого слова в словаре будут два вектора: **v** и **u**

v - если слово в центре, **u** - есть это слово из контекста

v_кот: вектор, когда "кот" является центром.

u_кот: вектор, когда "кот" появляется в контексте, например, в фразах "собака и кот".

Image credit

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Вероятность слова из **контекста o** при условии **центра c**:

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

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Вероятность слова из **контекста o** при условии **центра c**:

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

$\text{cos\_sim}(w_1, w_2) \approx w_1^T w_2$

Больше значение - больше вероятность

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Вероятность слова из **контекста o** при условии **центра c**:

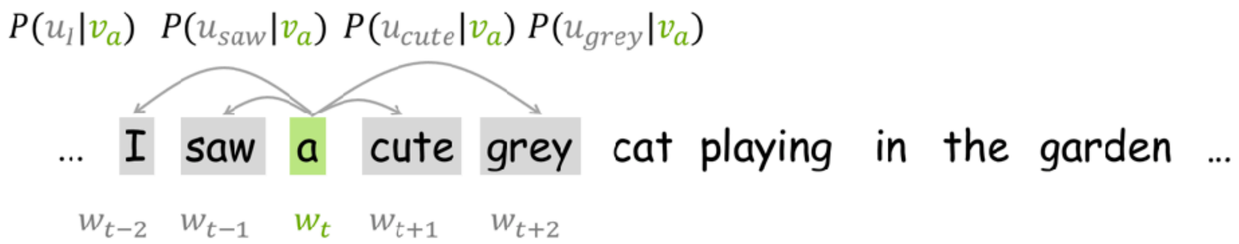
$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

Нормализация по словарю (softmax)

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**



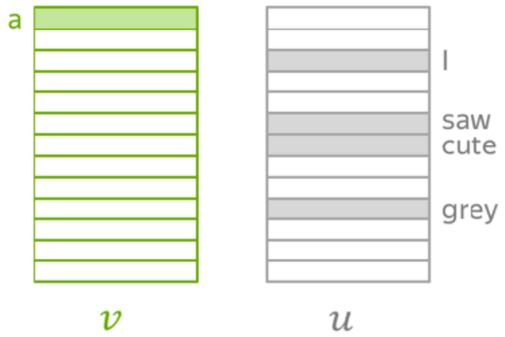
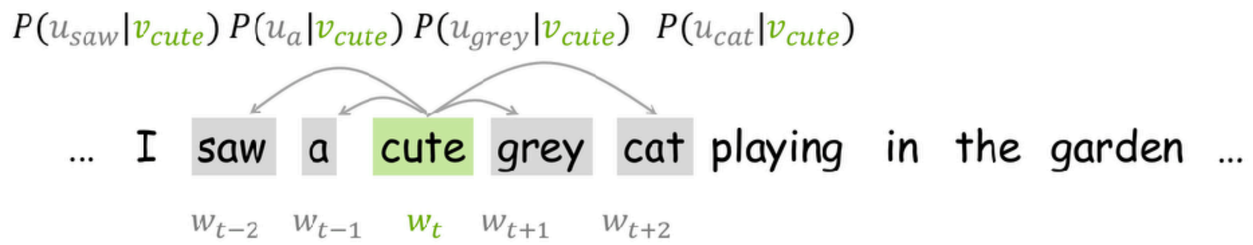


Image credit

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**



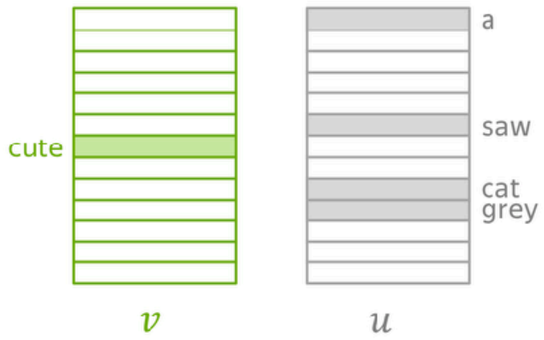


Image credit

Word2Vec

Хотим: перевести слова в числовой вид

Будем проходить **окном** по **большому корпусу текстов** и вычислять вероятность **центрального** слова при условии **контекста**

Loss: negative log-likelihood $$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Обучаем градиентным спуском u, v (θ)

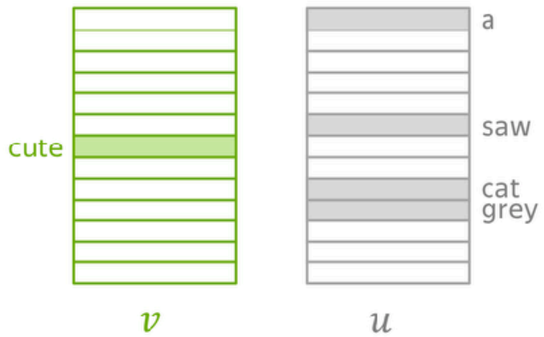
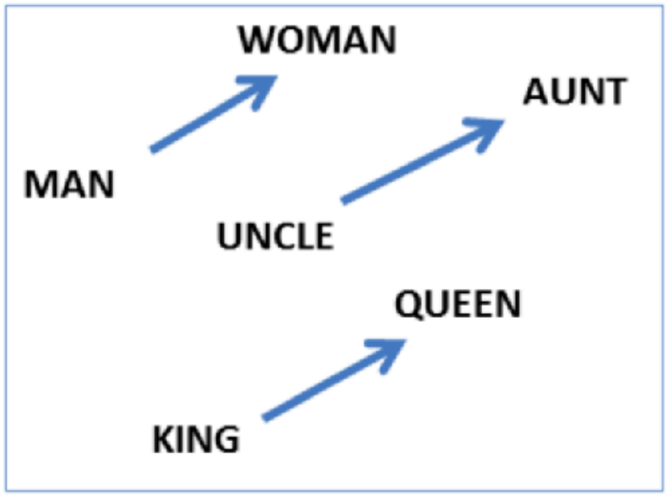


Image credit

Word2Vec

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

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



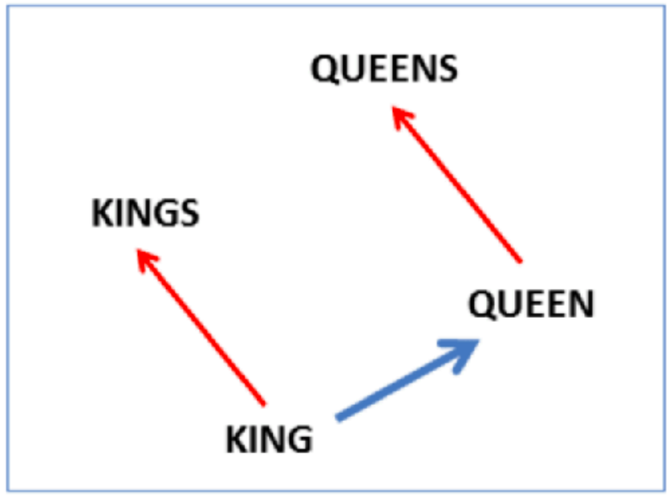
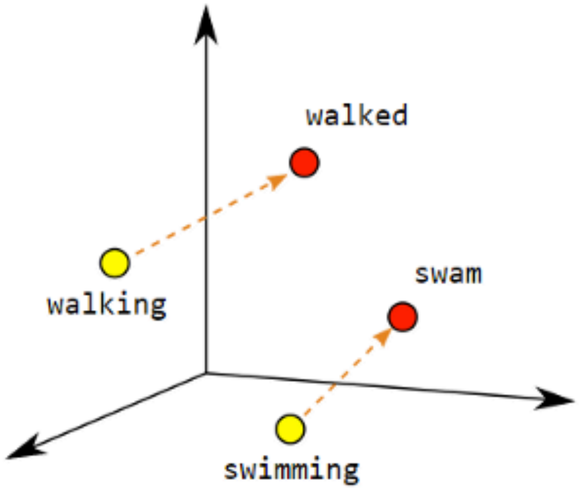
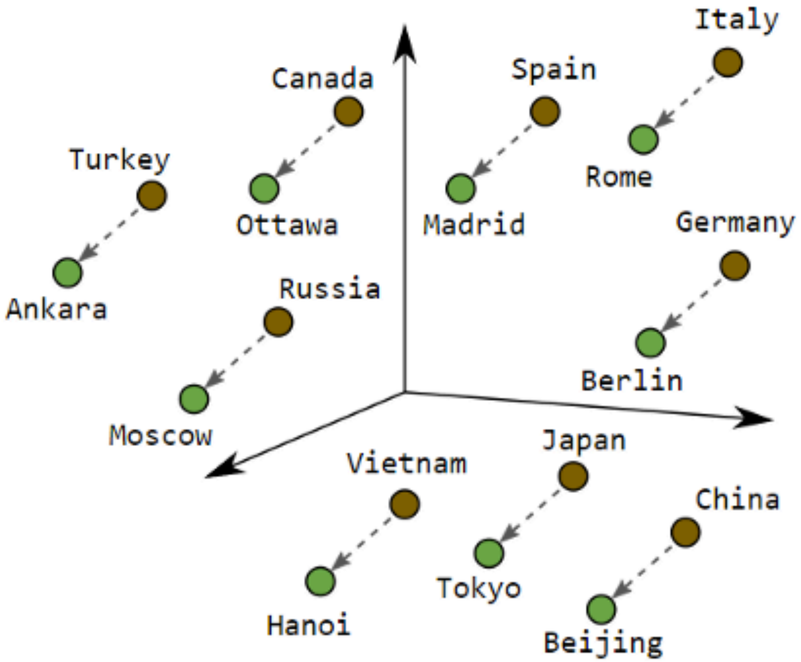


Image credit

Word2Vec



Verb Tense



Country-Capital

Image credit

Итоги занятия

- NLP — очень востребованная и активно развивающаяся область на стыке машинного обучения, анализа данных и лингвистики
- Существуют разнообразные постановки задач обработки текстов, технические и бизнесовые
- Работа с текстами почти всегда требует тщательного изучения и аккуратной предобработки данных
- Можно использовать разнообразные признаковые описания, базовыми являются представления «мешка слов» и TF-IDF