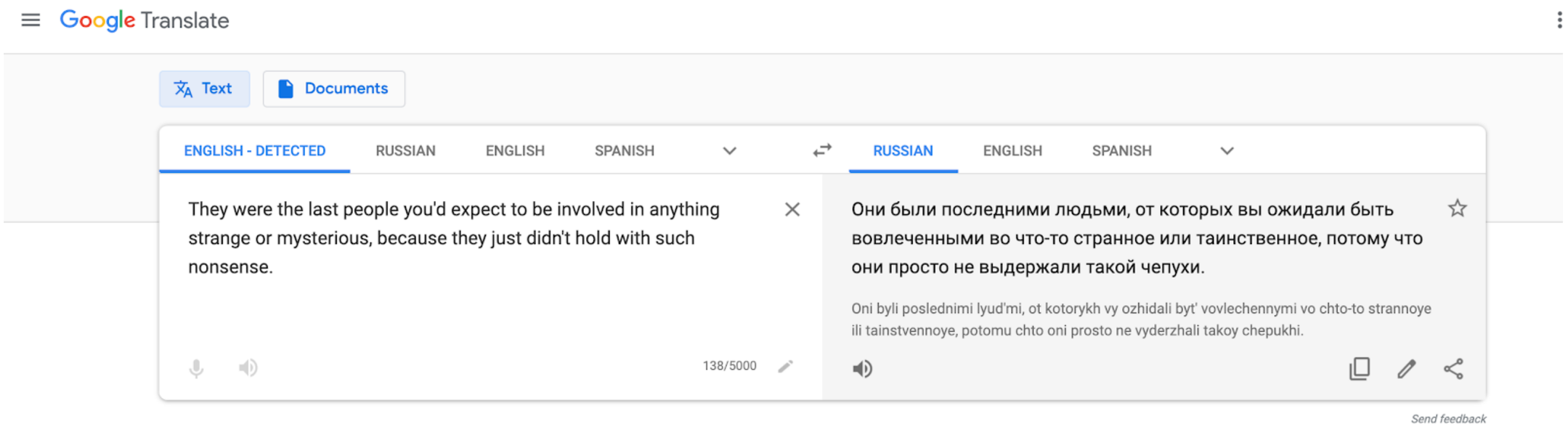
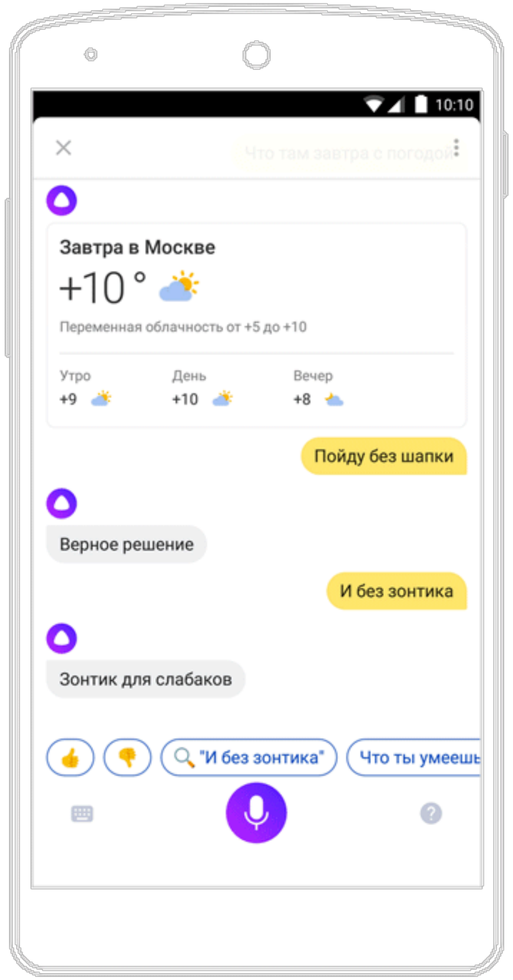
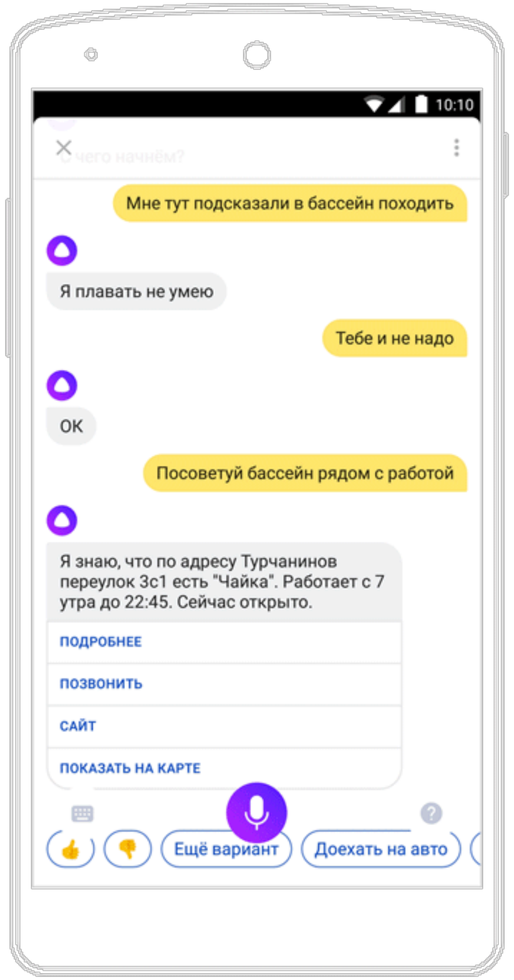
Обработка естественного языка: эмбеддинги и языковые модели

NLP: приложения

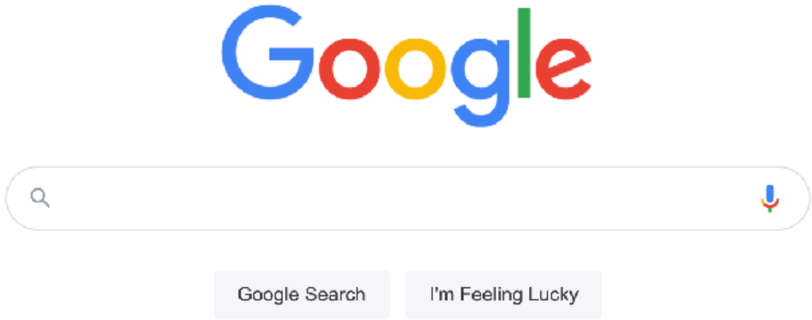


NLP: приложения





NLP: приложения

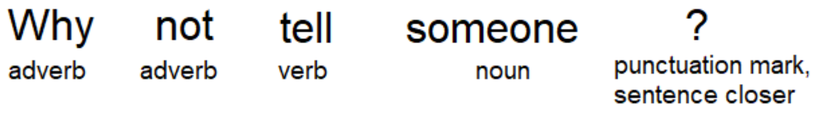


NLP: задачи

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

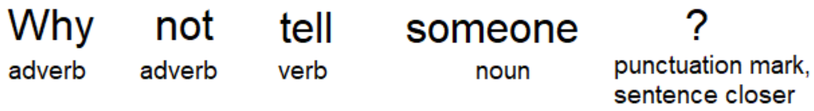
Part-of-Speech Tagging (POS Tagging) — это задача присваивания каждой словоформе в тексте её части речи (например, существительное, глагол, прилагательное и т.д.). Это один из ключевых этапов обработки естественного языка (NLP), поскольку части речи помогают понять структуру и смысл предложения.

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

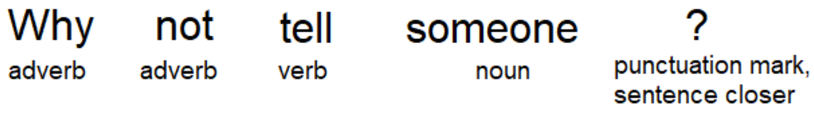
Named-Entity Recognition (NER) — это задача обработки естественного языка (NLP), направленная на автоматическое распознавание именованных сущностей в тексте. Задача NER заключается в идентификации и классификации сущностей (таких как имена людей, организации, локации, даты и т.д.) в предложениях или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

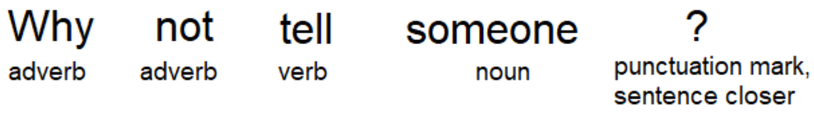
- Part-of-Speech Tagging
- Named-Entity Recognition
- Sentiment Analysis

…

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

Представления текста

Представления текста

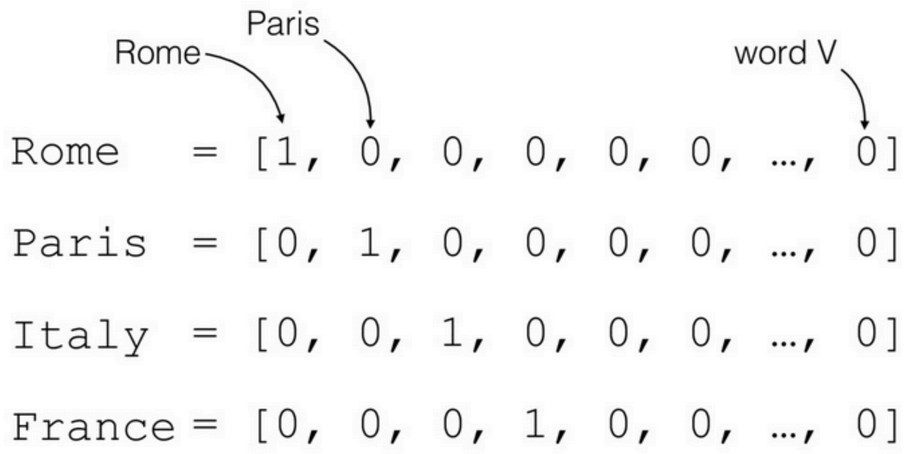
Хотим: перевести слова в числовой вид

Представления текста

Хотим: перевести слова в числовой вид

One-hot encoding

Создаем словарь всех слов (размера V) и нумеруем их

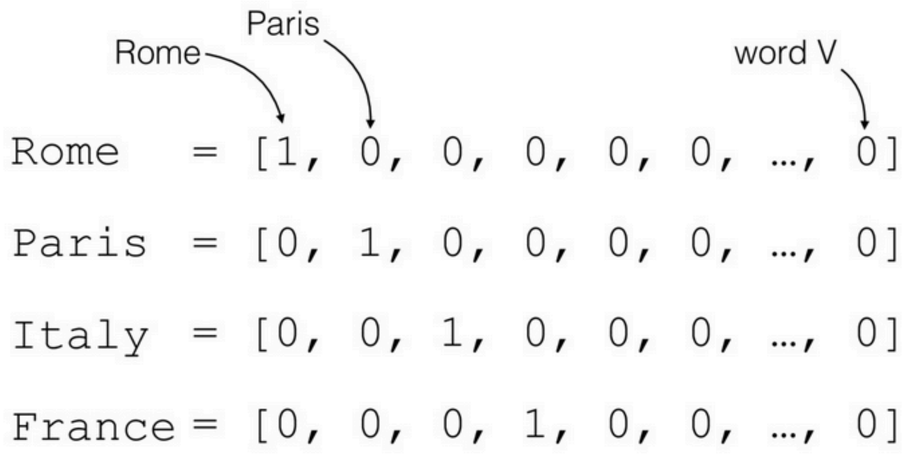


Представления текста

Хотим: перевести слова в числовой вид

One-hot encoding

Создаем словарь всех слов (размера V) и нумеруем их



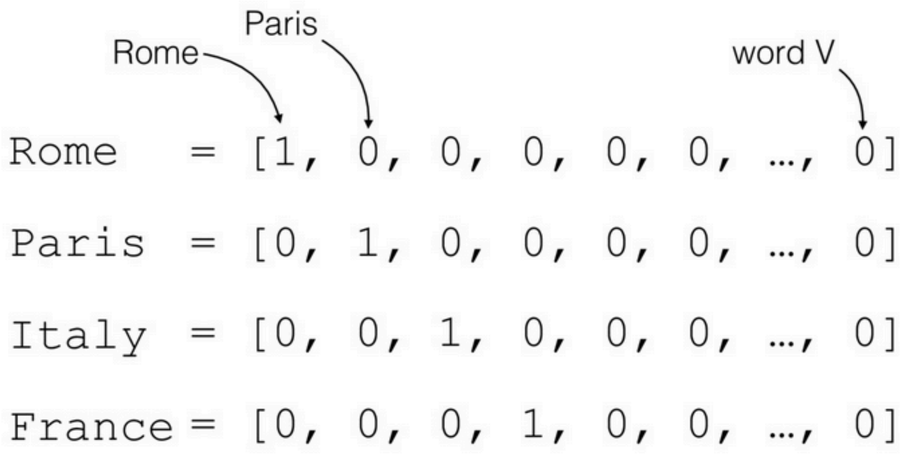
Все вектора ортогональны друг другу $\text{cos\_sim}(w_1, w_2) = \frac{w_1^T w_2}{\|w_1\|^2 \|w_2\|^2} = 0$

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

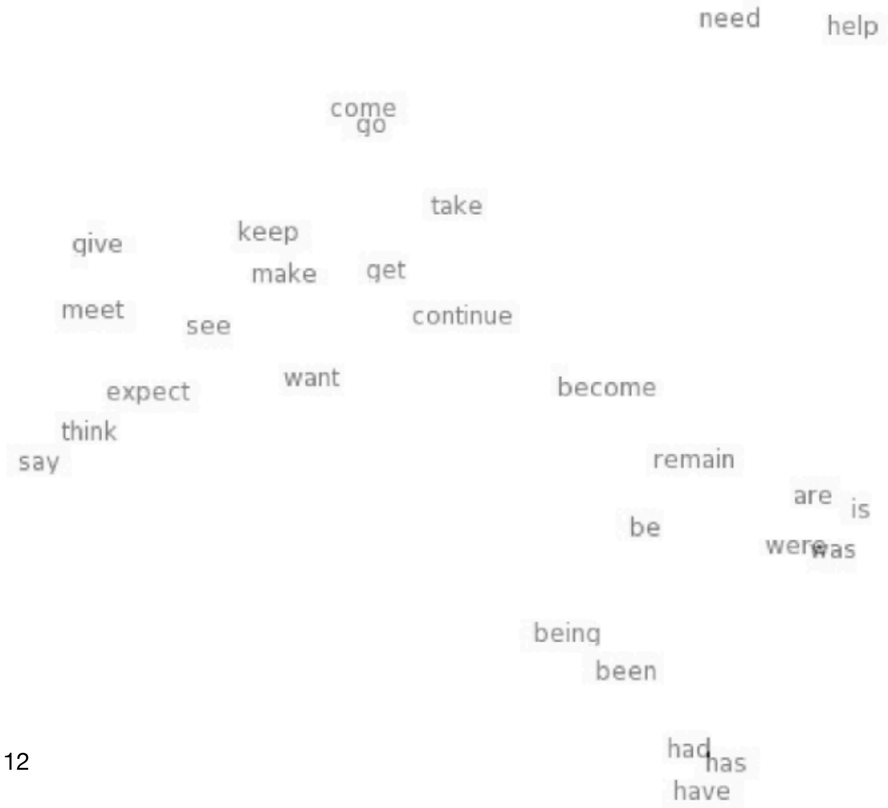
Rome? Paris?

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**

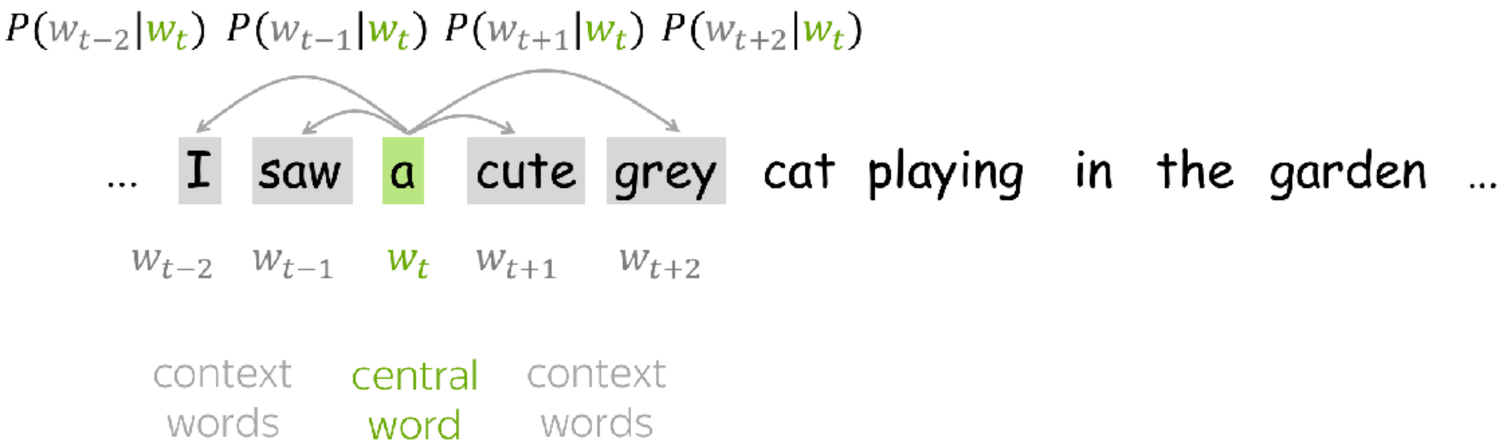


Image credit

Word2Vec

Хотим: перевести слова в числовой вид

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

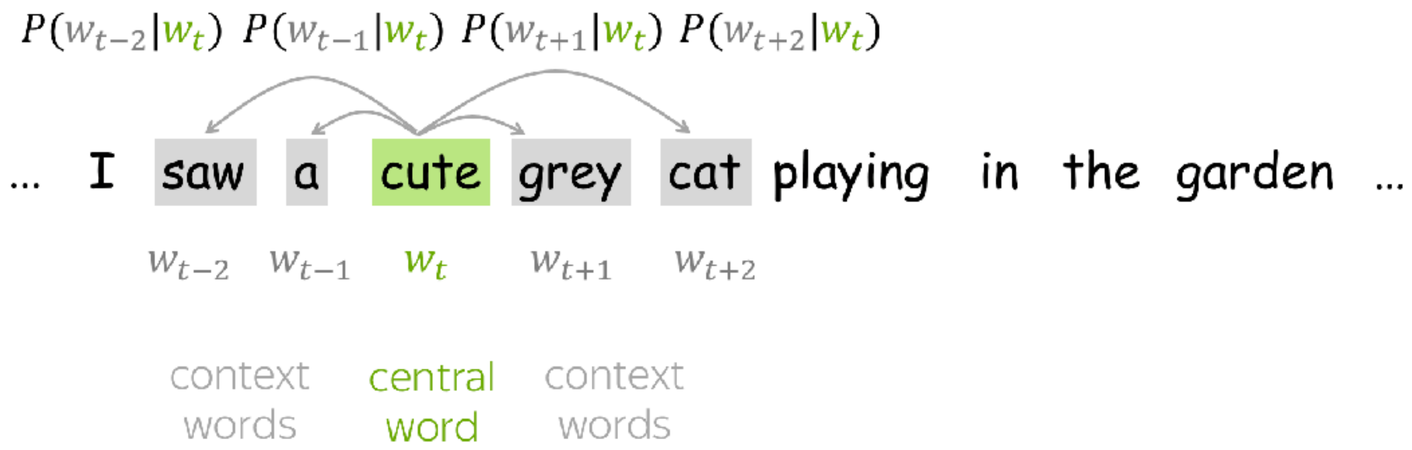


Image credit

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

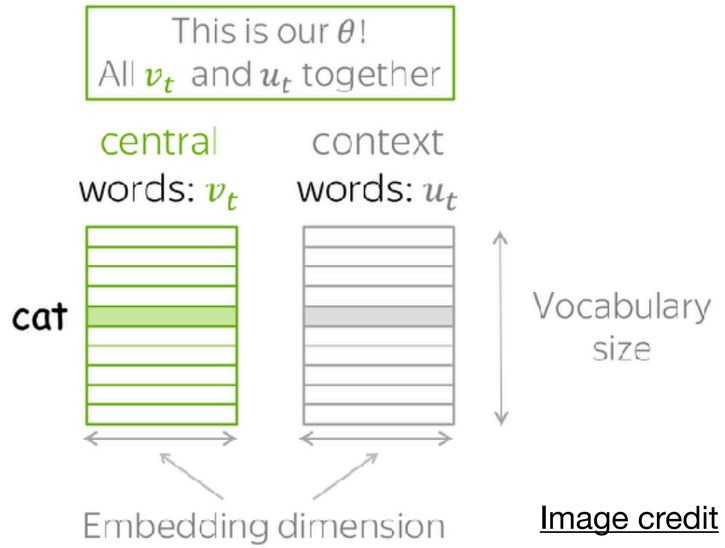
Как задать вероятность?

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**:

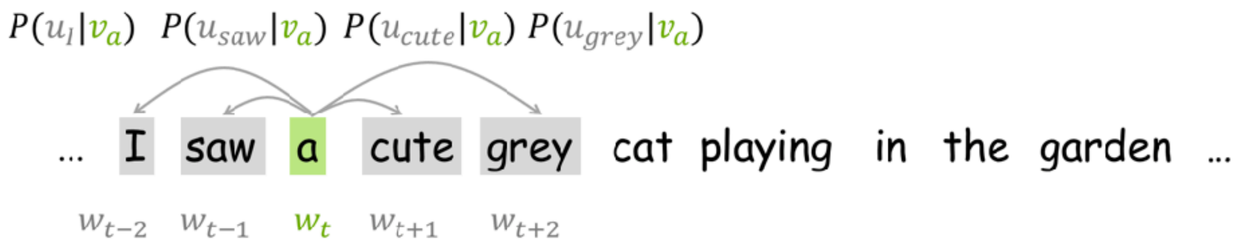
$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

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**



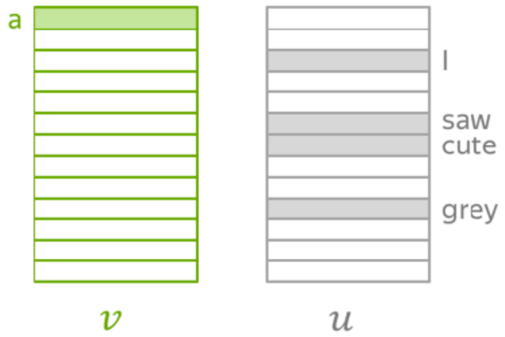
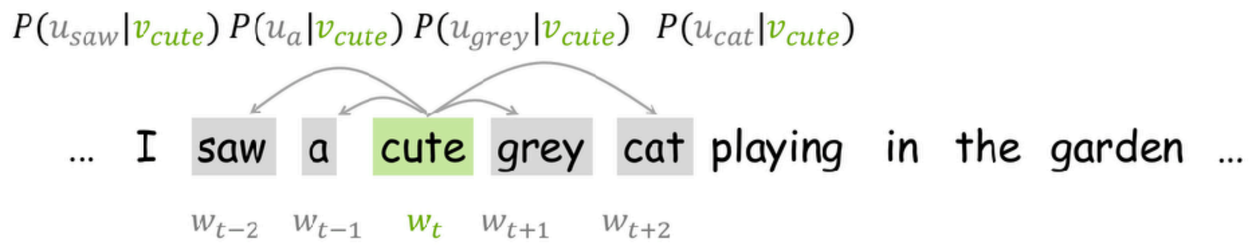


Image credit

Word2Vec

Хотим: перевести слова в числовой вид

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



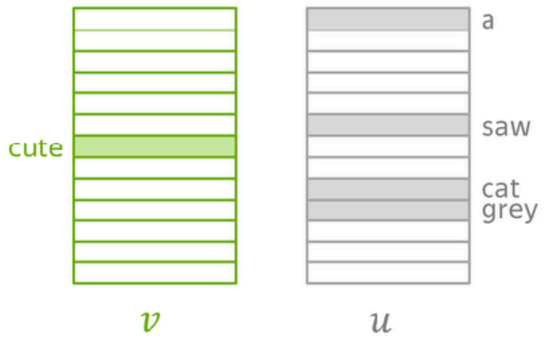


Image credit

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)

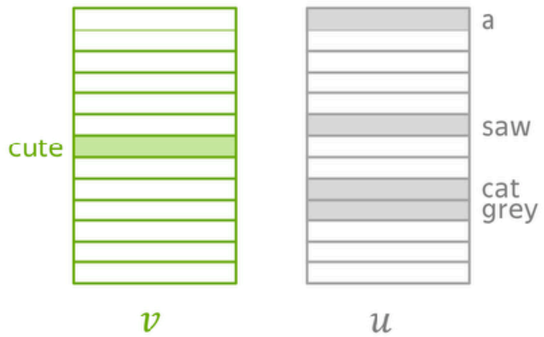


Image credit

Word2Vec

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

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

Нормализация **по словарю** - долго

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

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

Нормализация **по словарю** - долго

Улучшения:

- Hierarchical Softmax
- Negative Sampling $$\sum_{w\in V}\exp(u_w^T v_c) \longrightarrow \sum_{w\in\{o\}\cup S\_k}\exp(u_w^T v_c)$$

Нормализация по выбранным “negative” словам

Word2Vec

Hierarchical Softmax — это метод для ускорения вычисления вероятностей в моделях глубокого обучения, таких как нейронные сети для языкового моделирования. В стандартной реализации Softmax при вычислении вероятности для каждого класса необходимо пройти через все классы, что может быть дорогостоящим, особенно когда классов много (например, в задаче предсказания слов). Hierarchical Softmax решает эту проблему, заменяя полный Softmax на иерархическую структуру, что снижает вычислительные затраты. Вместо того чтобы рассматривать каждый класс (например, слово) как отдельный элемент, классы организуются в виде бинарного дерева. Листья этого дерева представляют классы, а внутренние узлы определяют промежуточные решения.

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

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

Нормализация **по словарю** - долго

Улучшения:

- Hierarchical Softmax
- Negative Sampling $$\sum_{w\in V}\exp(u_w^T v_c) \longrightarrow \sum_{w\in\{o\}\cup S\_k}\exp(u_w^T v_c)$$

Нормализация по выбранным “negative” словам

Word2Vec

Хотим: перевести слова в числовой вид

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

Преимущества Skip-Gram

Устойчивость к редким словам: Skip-Gram хорошо работает даже с редкими словами, поскольку он фокусируется на предсказании контекстных слов.
Хорошее представление: Skip-Gram создает векторы слов, которые хорошо захватывают семантические и синтаксические связи.
Гибкость: Модель может быть легко адаптирована к различным задачам и использует простые методы оптимизации.

Недостатки

Вычислительная сложность: В сравнении с CBOW (другой архитектурой Word2Vec) Skip-Gram требует больше вычислительных ресурсов, особенно при большом размере словаря, из-за необходимости вычисления softmax для всех слов.
Не учитывает порядок слов: Skip-Gram рассматривает контекстные слова без учета их порядка, что может привести к некоторой потере информации.

Word2Vec

Хотим: перевести слова в числовой вид

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

Преимущества Skip-Gram

Устойчивость к редким словам: Skip-Gram хорошо работает даже с редкими словами, поскольку он фокусируется на предсказании контекстных слов.
Хорошее представление: Skip-Gram создает векторы слов, которые хорошо захватывают семантические и синтаксические связи.
Гибкость: Модель может быть легко адаптирована к различным задачам и использует простые методы оптимизации.

Недостатки

Вычислительная сложность: В сравнении с CBOW (другой архитектурой Word2Vec) Skip-Gram требует больше вычислительных ресурсов, особенно при большом размере словаря, из-за необходимости вычисления softmax для всех слов.
Не учитывает порядок слов: Skip-Gram рассматривает контекстные слова без учета их порядка, что может привести к некоторой потере информации.

Word2Vec

Хотим: перевести слова в числовой вид

CBOW (Continuous Bag of Words) — это одна из двух основных архитектур модели Word2Vec, используемая для обучения векторных представлений слов. В отличие от Skip-Gram, который предсказывает контекстные слова на основе целевого слова, CBOW работает наоборот: он предсказывает целевое слово на основе его контекстных слов.
Принцип работы

Центровое слово и контекст:

В CBOW каждое слово в предложении рассматривается как контекстное слово (context word), из которого модель должна предсказать целевое слово (target word).

Например, в предложении "кот играет с мячом", если "играет" является центровым словом, то контекстными словами будут "кот", "с", "мячом".

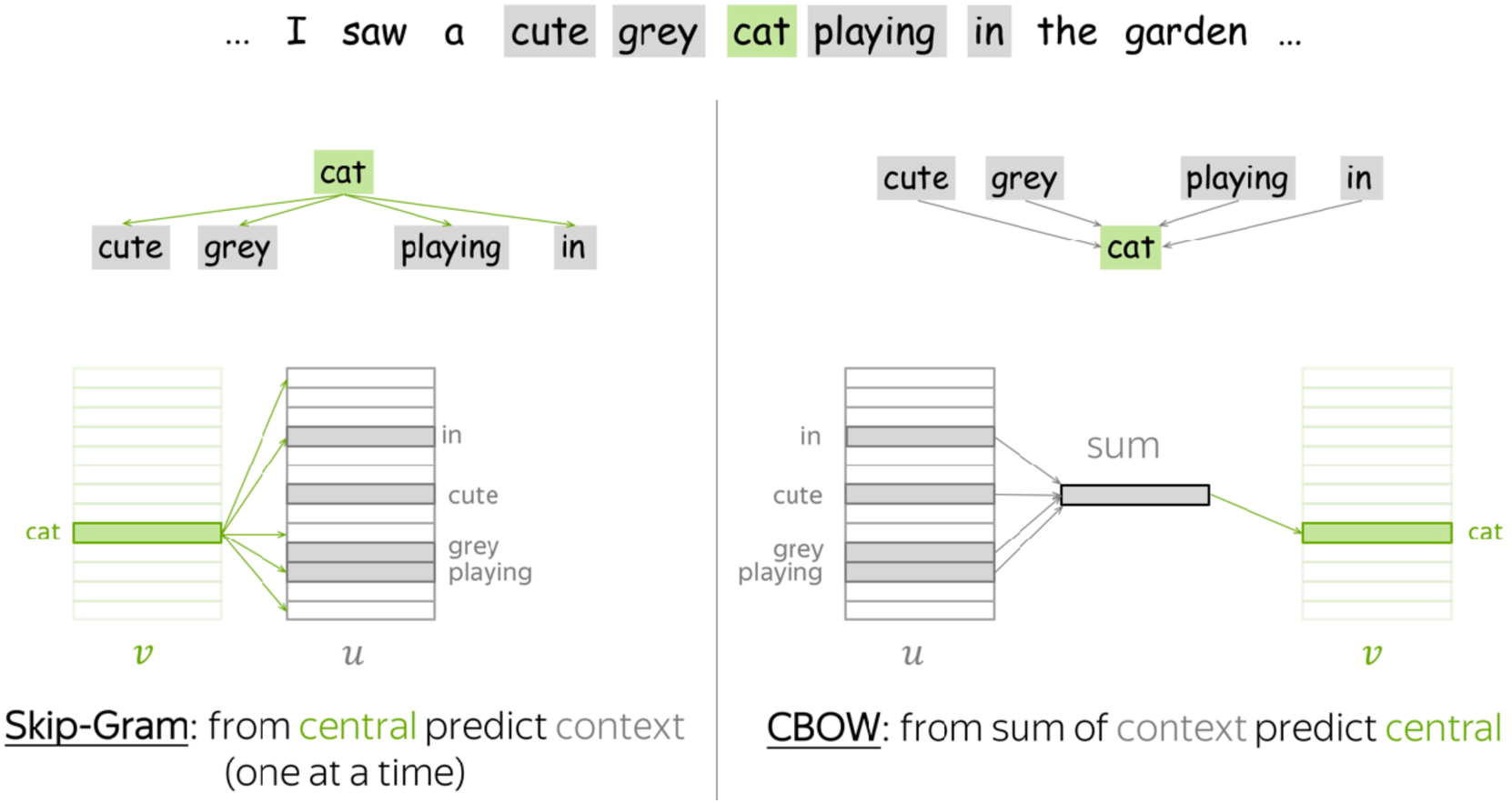


Image credit

Word2Vec

Преимущества CBOW

Эффективность: CBOW обычно быстрее и требует меньше вычислительных ресурсов, чем Skip-Gram, поскольку он использует контекстные слова для предсказания одного целевого слова.

Хорошее представление: CBOW создает векторы слов, которые хорошо захватывают семантические и синтаксические связи между словами.

Устойчивость к редким словам: CBOW лучше работает с частыми словами и может эффективно обрабатывать большие объемы текста.

Недостатки

Игнорирование порядка: CBOW также не учитывает порядок контекстных слов, что может привести к потере информации о семантических отношениях.

Предсказание для редких слов: CBOW может иметь трудности с предсказанием для редких или уникальных слов, так как они могут не встречаться часто в контексте.

Хотим: перевести слова в числовой вид

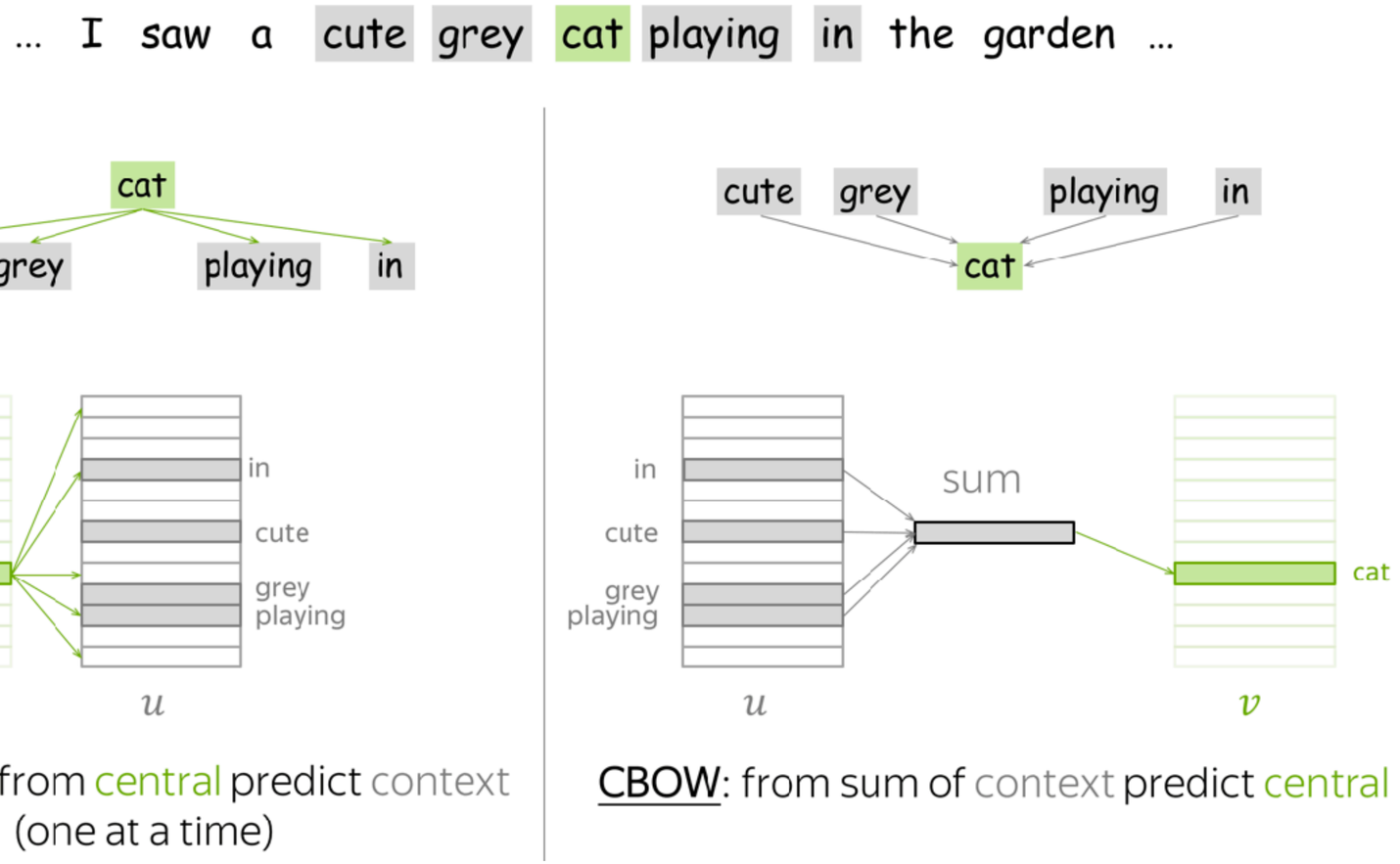
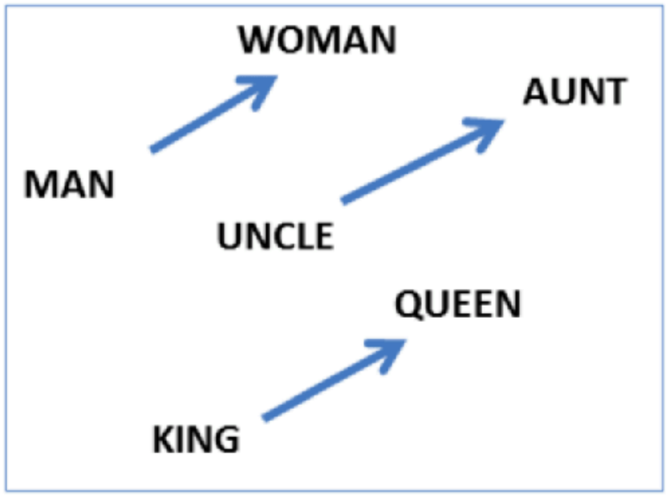


Image credit

Word2Vec

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

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



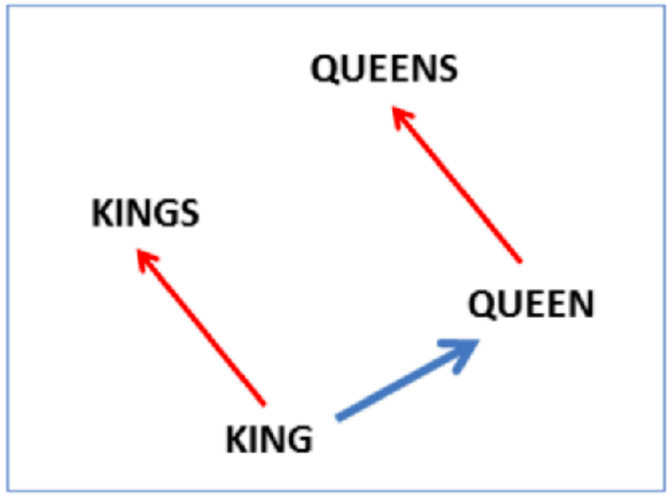
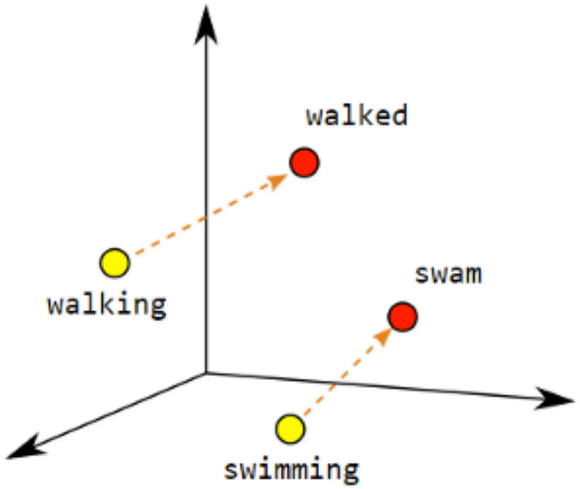
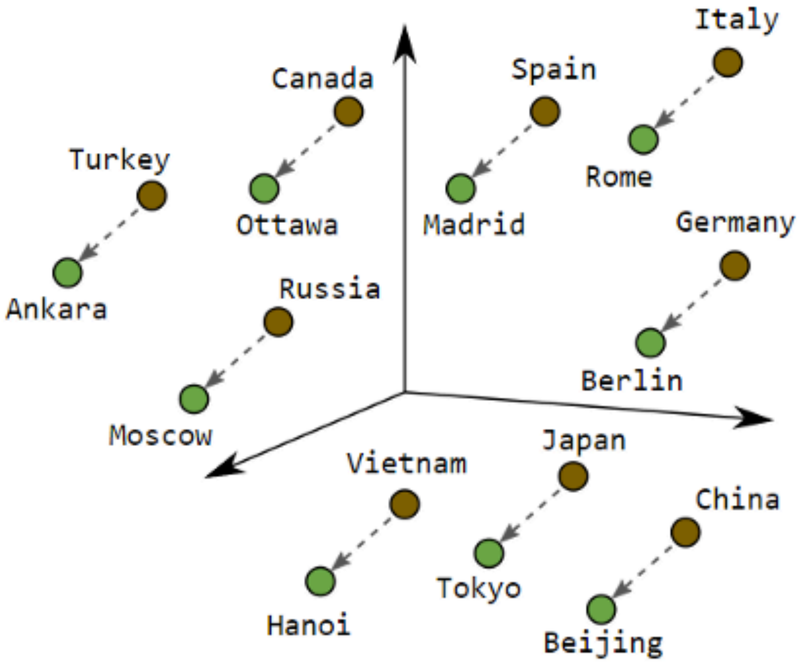


Image credit

Word2Vec



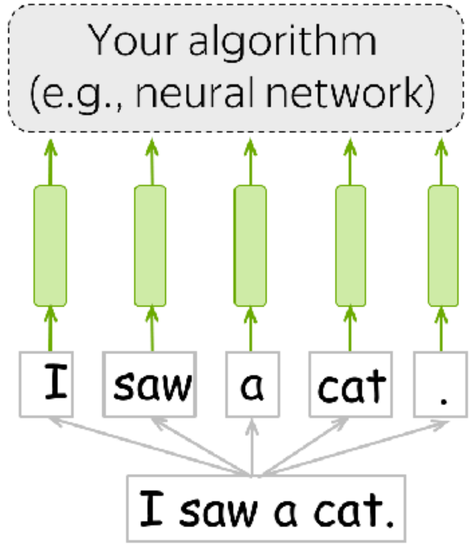
Verb Tense



Country-Capital

Image credit

Word2Vec



Any algorithm for solving a task

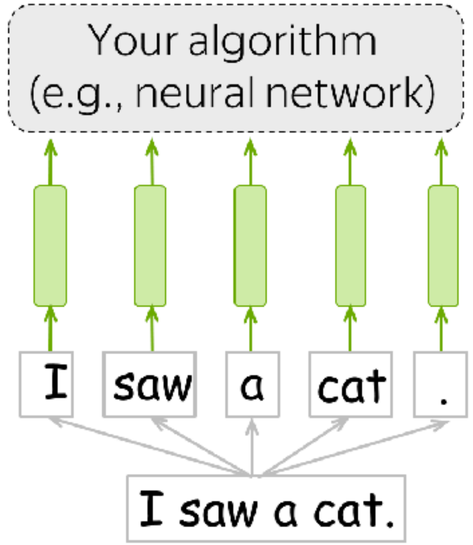
Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)

Image credit

Word2Vec

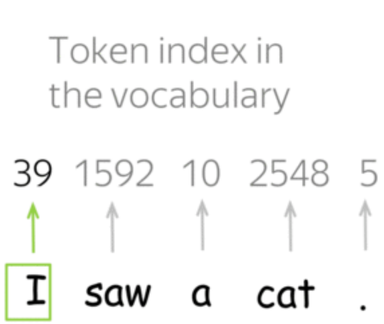


Any algorithm for solving a task

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

Sequence of tokens

Text (your input)



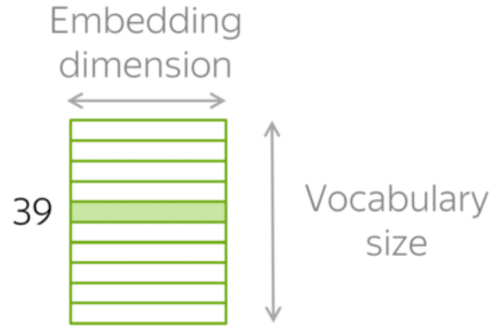
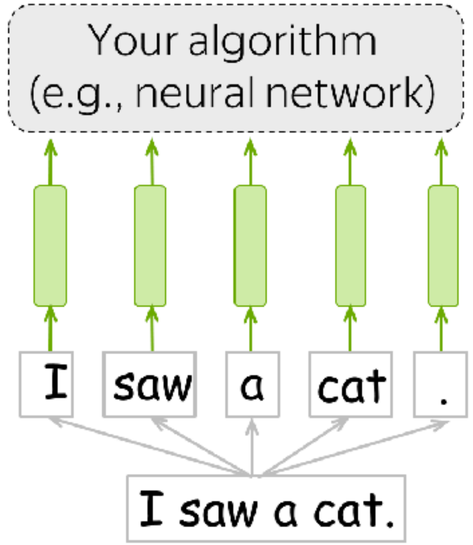


Image credit

Word2Vec

Что делать со словами не из словаря (out-of-vocab)?



Any algorithm for solving a task

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

Sequence of tokens

Text (your input)

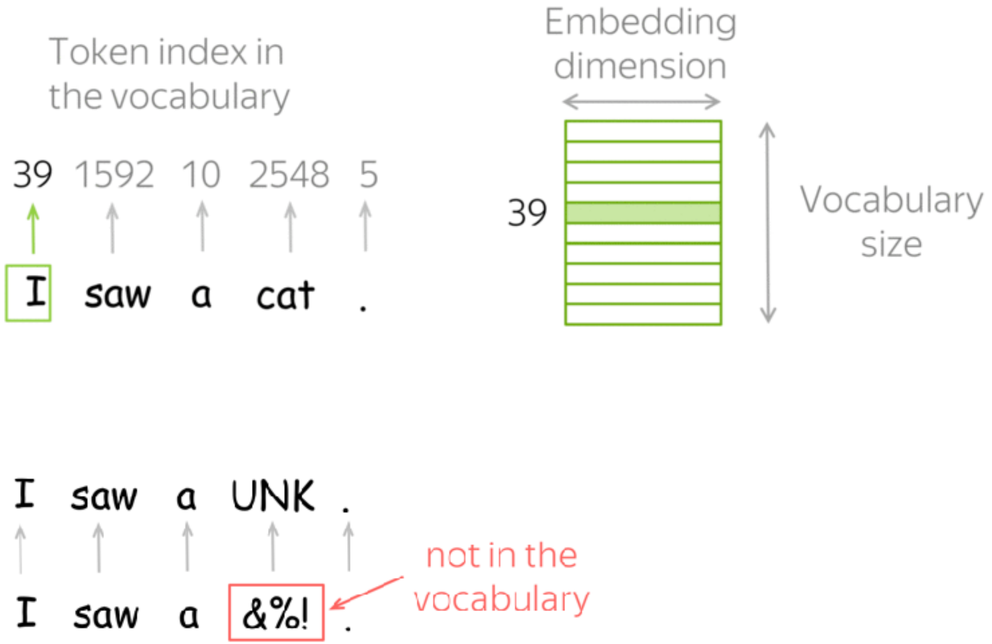


Image credit

FastText

Что делать со словами не из словаря (out-of-vocab)?

Идея: использовать n-grams

Подход на основе n-grams:

FastText учитывает не только отдельные слова, но и их подстроки (n-grams), что позволяет лучше захватывать морфологические свойства слов. Это особенно полезно для языков с богатой морфологией, где одно слово может иметь множество форм (например, падежи, времена).

Например, для слова "машина" FastText может учитывать такие n-grams, как "маш", "аши", "шина", что позволяет модели лучше справляться с редкими словами и словами, которых нет в словаре.

Обработка редких и неизвестных слов:

Благодаря учету подстрок, FastText может эффективно обрабатывать слова, которые не были встречены в обучающем наборе данных (out-of-vocabulary words). Если слово не присутствует в словаре, его вектор может быть вычислен как сумма векторов его подстрок.

Image credit

FastText

Что делать со словами не из словаря (out-of-vocab)?

Идея: использовать n-grams

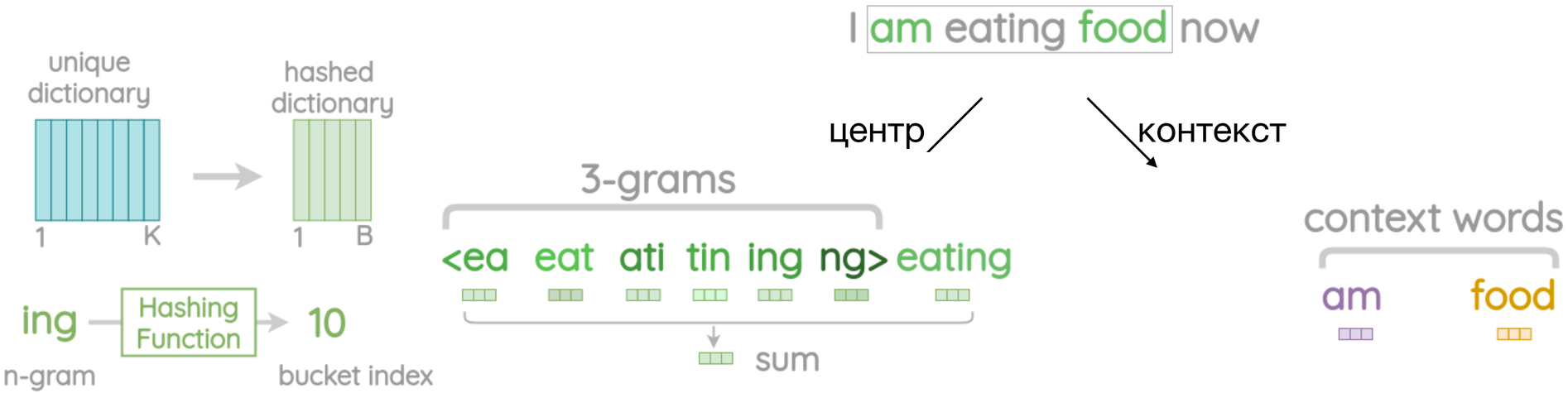


Image credit

FastText

Что делать со словами не из словаря (out-of-vocab)?

Идея: использовать n-grams

Медленнее обучается, но лучше представления для OOV слов

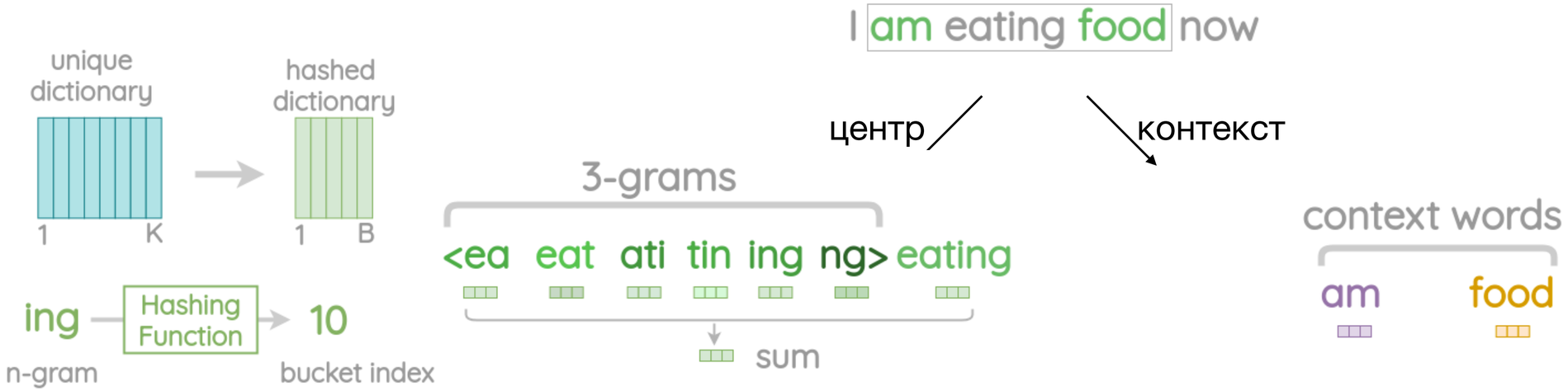


Image credit

Представления текста

Теперь знаем разные векторные представления слов (word embeddings)!

Word2Vec, FastText, Glove есть предобученные для разных языков!

Есть имплементации, можно обучить на своем корпусе (gensim и др.)

GloVe (Global Vectors for Word Representation) — это метод для обучения векторных представлений слов, который был разработан в Стэнфордском университете. Он основывается на статистических свойствах корпуса текста и стремится захватить семантические и синтаксические отношения между словами.
Основные характеристики GloVe

Контекстные окна:
GloVe использует статистику соотношений частоты слов (co-occurrence statistics), что позволяет оценить, насколько часто слова встречаются вместе в контексте. Это даёт возможность моделировать смысл слов, основываясь на их окружении.

Матрица совместной встречаемости:
Создаётся матрица, где строки представляют слова, а столбцы представляют контекстные слова. Каждая ячейка матрицы содержит количество совместных встреч (co-occurrences) для пары слов. Например, если слова "кошка" и "собака" часто встречаются вместе, значение в соответствующей ячейке будет высоким.

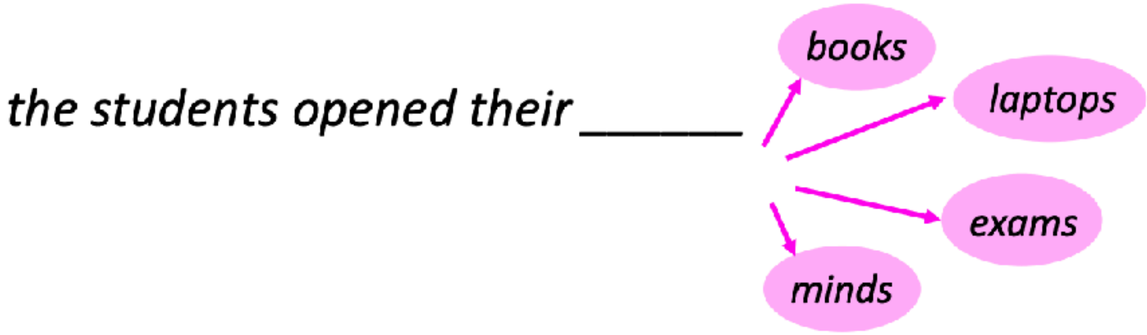
Эмбеддинги слов:
GloVe обучает векторные представления слов таким образом, чтобы расстояние между векторами отражало семантическую близость. Например, вектора для "король" и "королева" будут близки, как и вектора для "мужчина" и "женщина".

Глобальная статистика:
GloVe использует глобальную информацию о частотах слов, в отличие от некоторых других методов (например, Word2Vec), которые учитывают только локальный контекст.

Языковые модели (LM)

Языковые модели (Language Models)

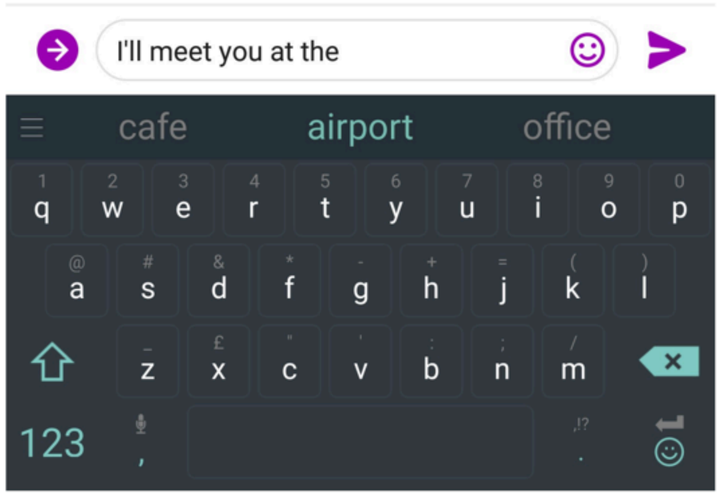
Задача: предсказать следующее слово по предыдущим

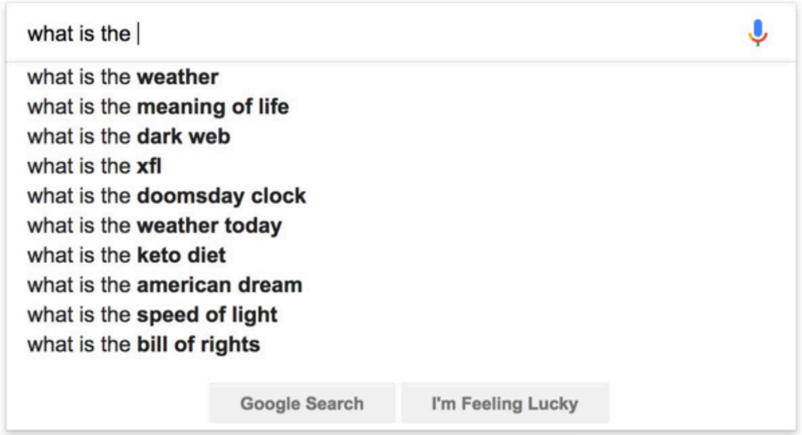


Авторегрессионная модель $p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$

Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим





Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим

Используем нейросеть и проход окном

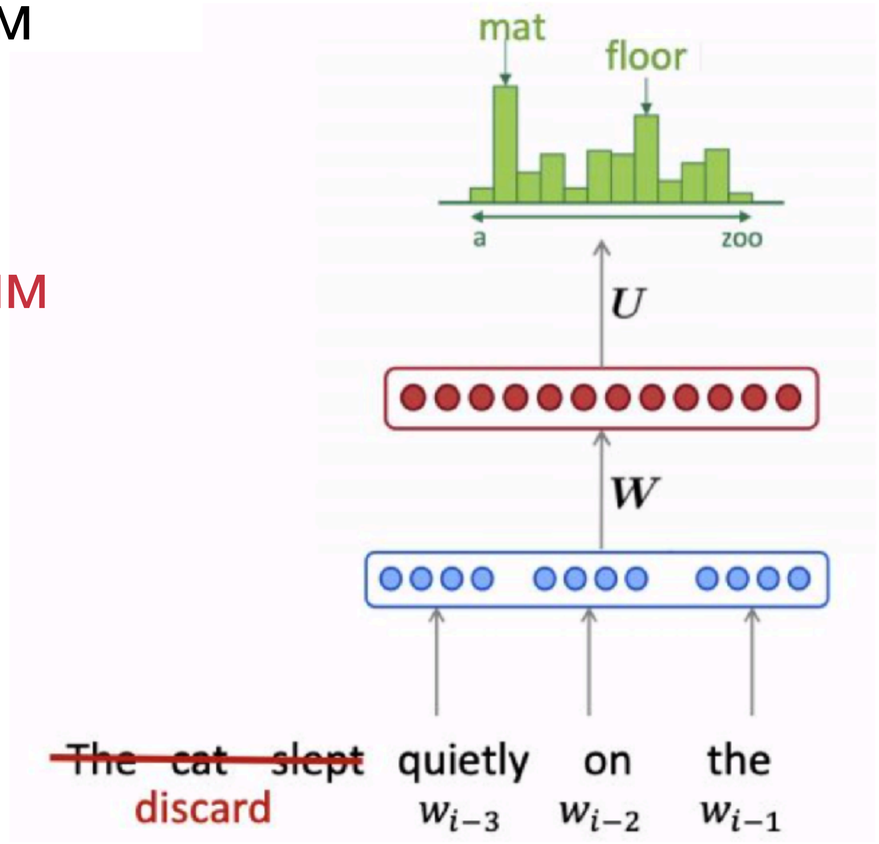
Языковые модели (Language Models)

Задача: предсказать следующее слово по предыдущим

Используем нейросеть и проход окном

Проблемы:

- Размер окна не может быть большим
- Не получаем весь контекст



Языковые модели (Language Models)

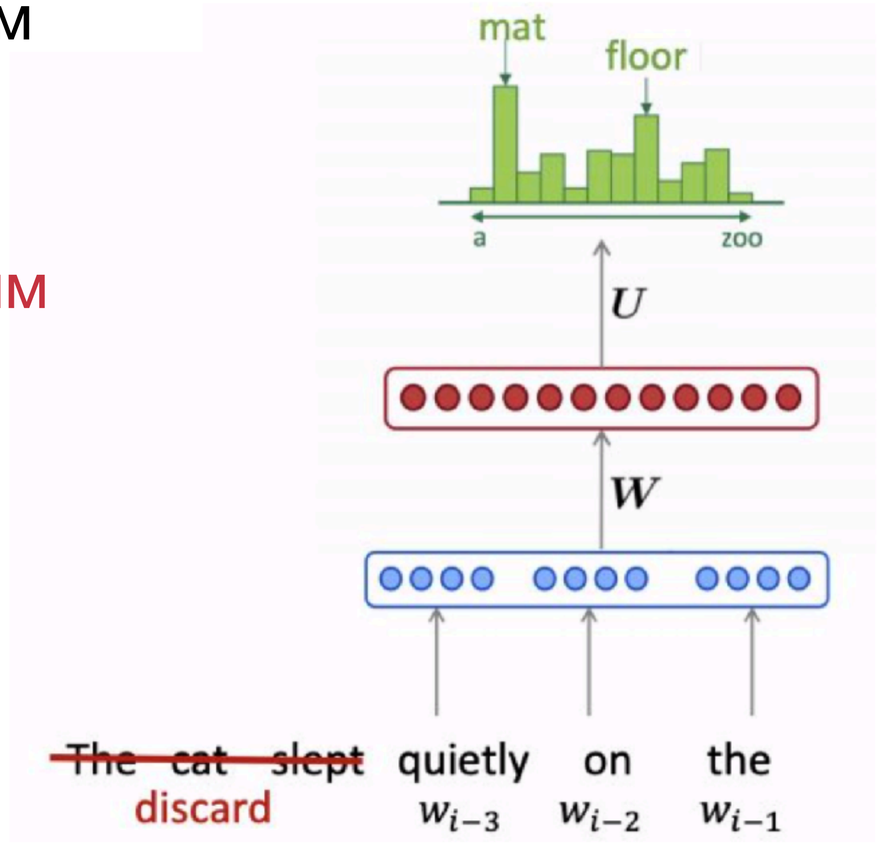
Задача: предсказать следующее слово по предыдущим

Используем нейросеть и проход окном

Проблемы:

- Размер окна не может быть большим
- Не получаем весь контекст

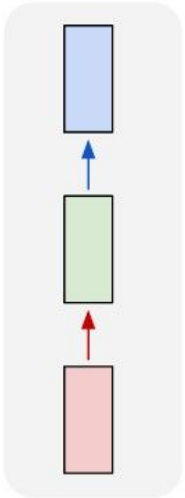
Нужна архитектура, которая может обрабатывать текст любой длины



Recurrent Neural Networks

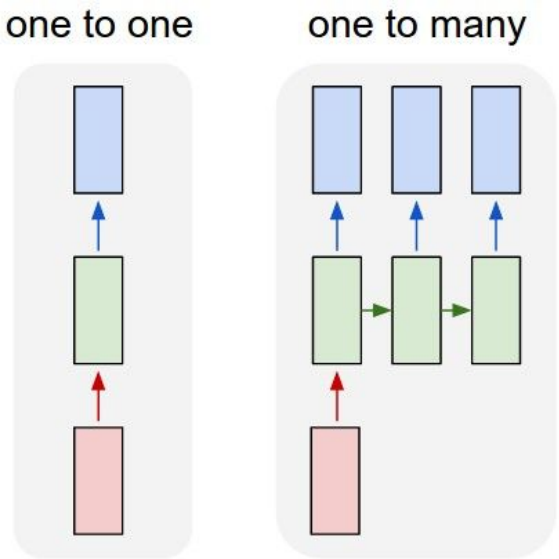
“Vanilla” Neural Network

one to one



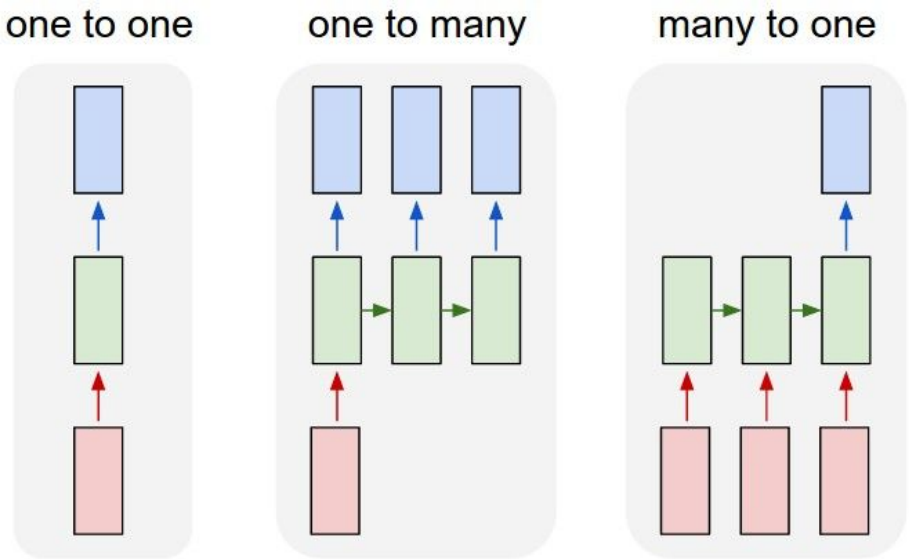
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences



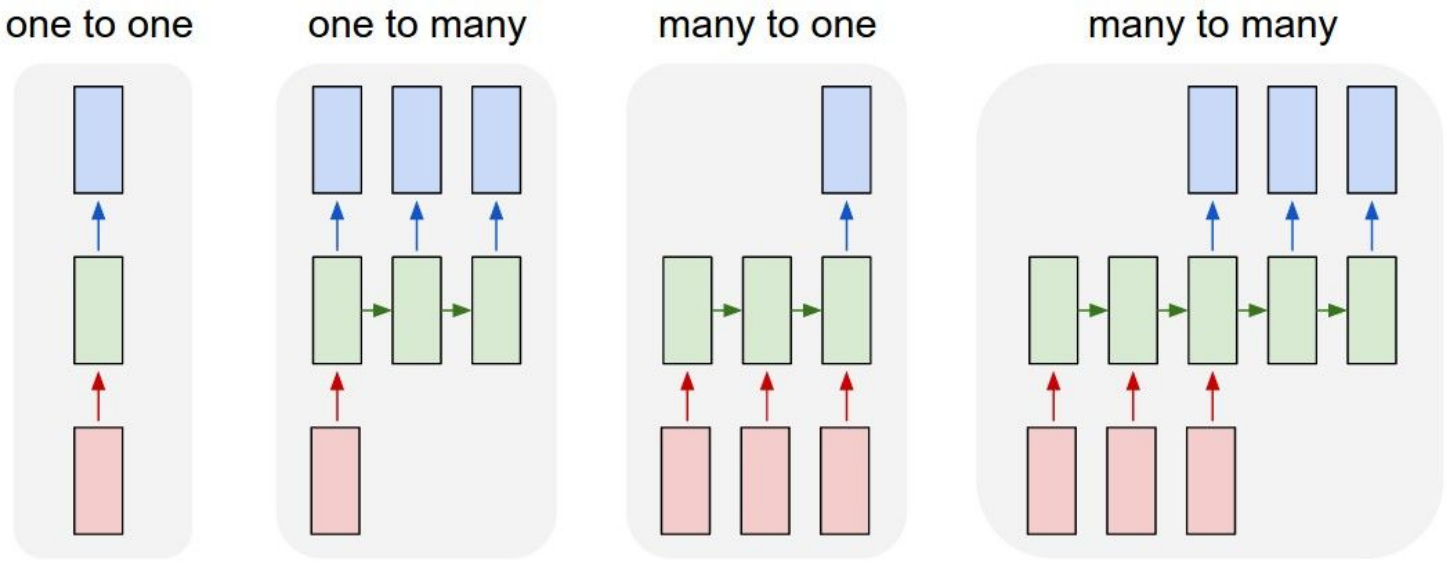
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



e.g. **action prediction**
sequence of video frames -> action class

Recurrent Neural Networks: Process Sequences



E.g. **Video Captioning**
Sequence of video frames -> caption

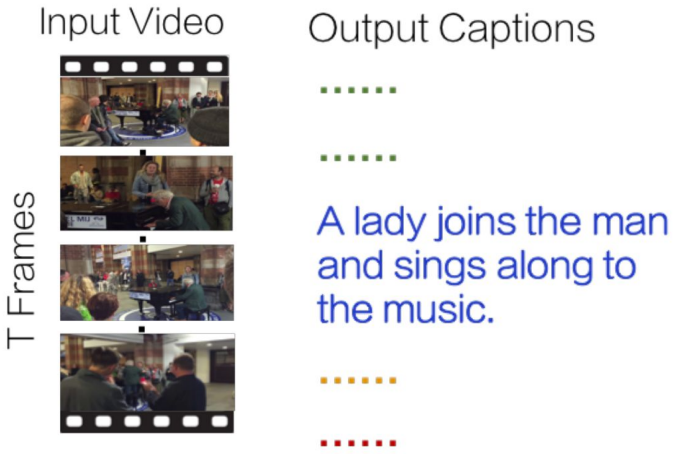
Why existing convnets are insufficient?

Variable sequence length inputs and outputs!

Example task: video captioning

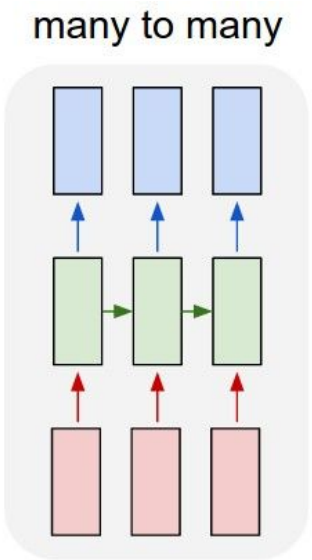
Input video can have variable number of frames

Output captions can be variable length.

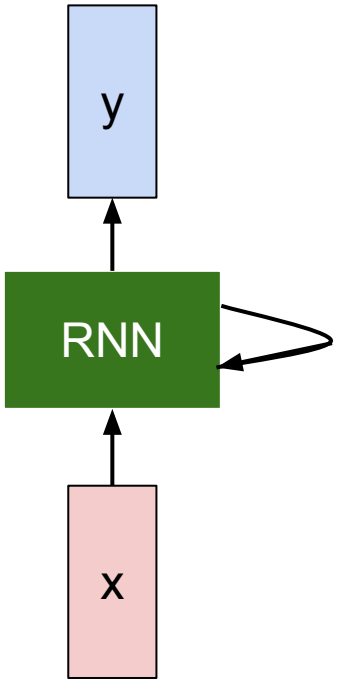


Krishna, Hata, Ren, Fei-Fei, Niebles. Dense captioning Events in Videos. ICCV 2019

Let's start with a task that takes a variable input and produces an output at every step



Recurrent Neural Network



Recurrent Neural Network

Ключевая идея рекуррентных нейронных сетей (RNN) заключается в том, что они имеют внутреннее состояние, которое обновляется по мере обработки последовательности. Это позволяет RNN эффективно справляться с задачами, где порядок данных имеет значение, такими как прогнозирование временных рядов, обработка естественного языка и многое другое. Вот основные моменты о том, как работает это внутреннее состояние и почему оно важно:
Как работают RNN

Секвенциальная обработка:

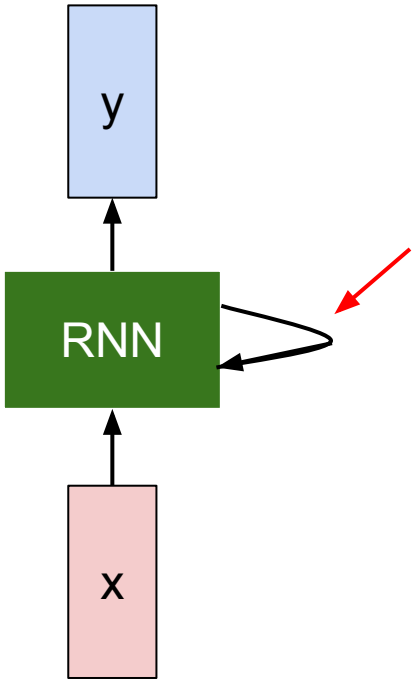
RNN обрабатывают последовательности по одному элементу за раз. Например, в обработке естественного языка RNN может обрабатывать предложение слово за словом.

На каждом временном шаге tt RNN получает входные данные xtxt (например, вектор эмбеддинга для слова в предложении).

Внутреннее состояние (Скрытое состояние):

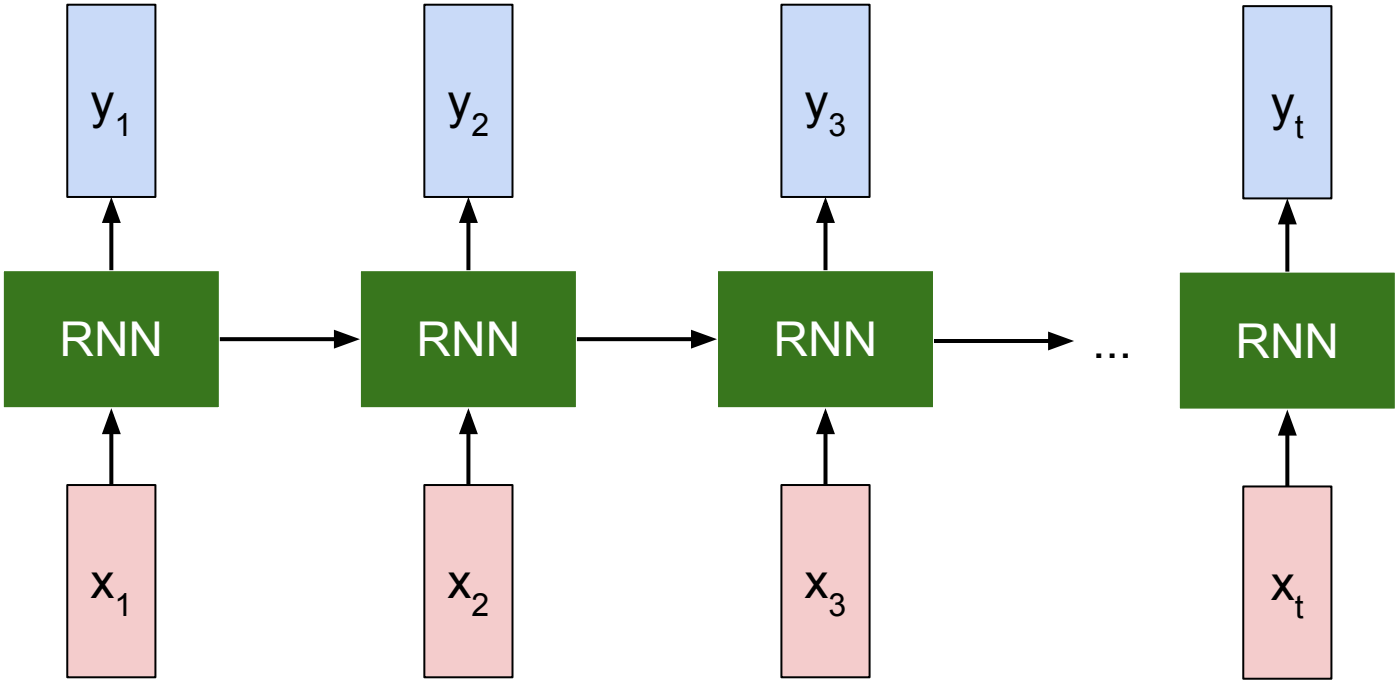
У RNN есть внутреннее состояние (часто обозначаемое как скрытое состояние, htht), которое обновляется на каждом временном шаге. Это состояние содержит информацию о предыдущих входах в последовательности.

Скрытое состояние инициализируется (обычно нулями) и обновляется с каждым входом



Key idea: RNNs have an “internal state” that is updated as a sequence is processed

Unrolled RNN



RNN hidden state update

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

some function with parameters W