Лекция 1.
Word embeddings

О курсе «Искусственный интеллект»

- Yandex School of Data Analysis
- NLP Course For You

Word embeddings

- Зачем нужно векторное представление слов?
- Работа с неизвестными токенами
- Векторы с одним активным компонентом (One-Hot Vectors)
- Методы, основанные на подсчёте (Count-Based Methods)
- Word2Vec (Word to Vector)
- GloVe
- Внутренняя и внешняя оценки качества
- О Лабораторной работе 1 «Базовое использование LLM»

Зачем нужно векторное представление слов?

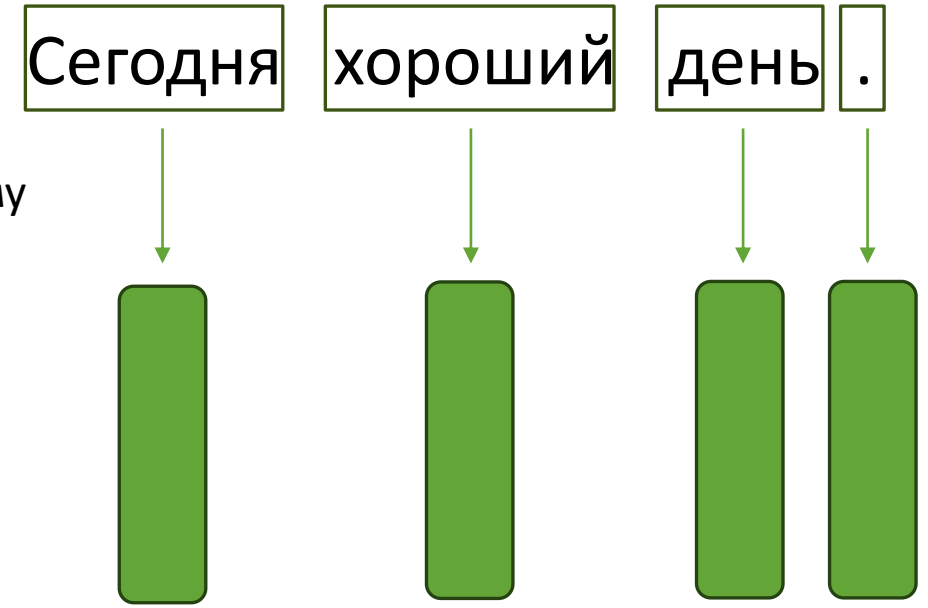
Векторное представление слов (векторное вложение слов, эмбеддинг) — это способ представления слов в виде числовых векторов, который используется компьютерными моделями для обработки естественного языка (NLP - Natural Language Processing).

Чтобы получить возможность представить семантическую близость, было предложено использовать embedding, то есть сопоставить слову некий вектор, отображающий его значение в «пространстве смыслов».

Зачем нужно векторное представление слов?

Предложение: Сегодня хороший день.

Разбиваем предложение на последовательность токенов:

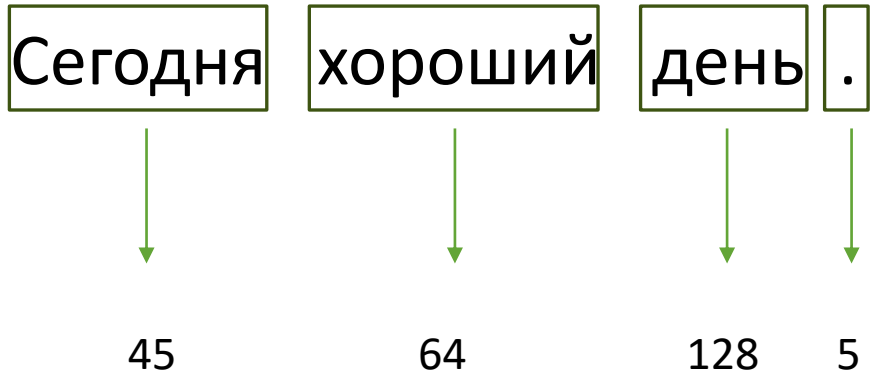


Ставим в соответствие каждому слову вектор

Зачем нужно векторное представление слов?

Каждый токен имеет свой номер в словаре токенов:

Каждый токен имеет свой номер в словаре токенов

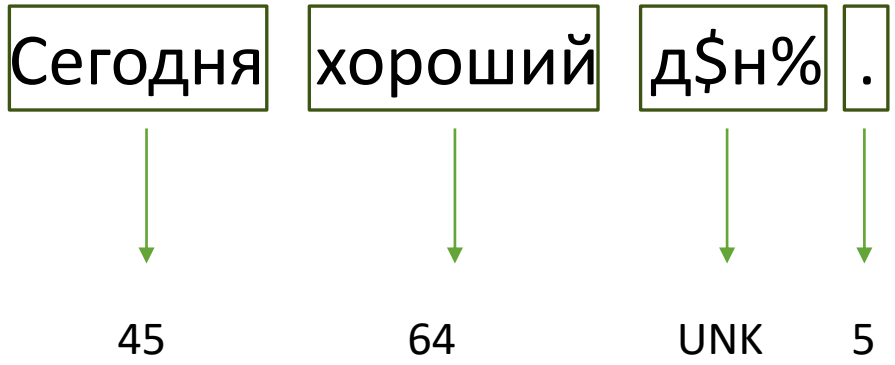


Словарь токенов

Неизвестные токены: Слова/единицы вне словаря

Некоторые токены могут не содержаться в словаре:

Каждый токен имеет свой номер в словаре токенов



Векторы с одним активным компонентом (One-Hot Vectors)

одна 1, остальные - 0

сегодня	0…0…0010…0…0….0…0
хороший	0…0…0000…0…1….0…0
день	0…0…1000…0…0….0…0

Размерность векторного представления = размеру словаря

Какие могут возникать проблемы при таком представлении слов?

Что такое смысл слова?

Например, испанское слово «***estante***».

Что такое смысл слова?

На всю стену стоит большой ***estante***.
В этом ***estante*** есть специальное место для верхней одежды.
Ребёнок не может достать до верхней полки ***estante***.
Мы купили этот ***estante*** в мебельном магазине.

Вы можете понять смысл слова с помощью контекста.

Что такое смысл слова?

На всю стену стоит большой _____.
В этом _____ есть специальное место для верхней одежды.
Ребёнок не может достать до верхней полки _____.
Мы купили этот _____ в мебельном магазине.

Вы можете понять смысл слова с помощью контекста.

Что такое смысл слова?

На всю стену стоит большой _____.
В этом _____ есть специальное место для верхней одежды.
Ребёнок не может достать до верхней полки _____.
Мы купили этот _____ в мебельном магазине.

Шкаф	1	1	1	1
Комод	1(?)	0	1	1

Похожие контексты, значит слова имеют похожие значения.

Дистрибутивная гипотеза

Слова, которые часто встречаются в похожих контекстах, имеют похожий смысл (Harris 1954).

Таким образом, основная идея представления слов:

разместить в векторы информацию про контексты

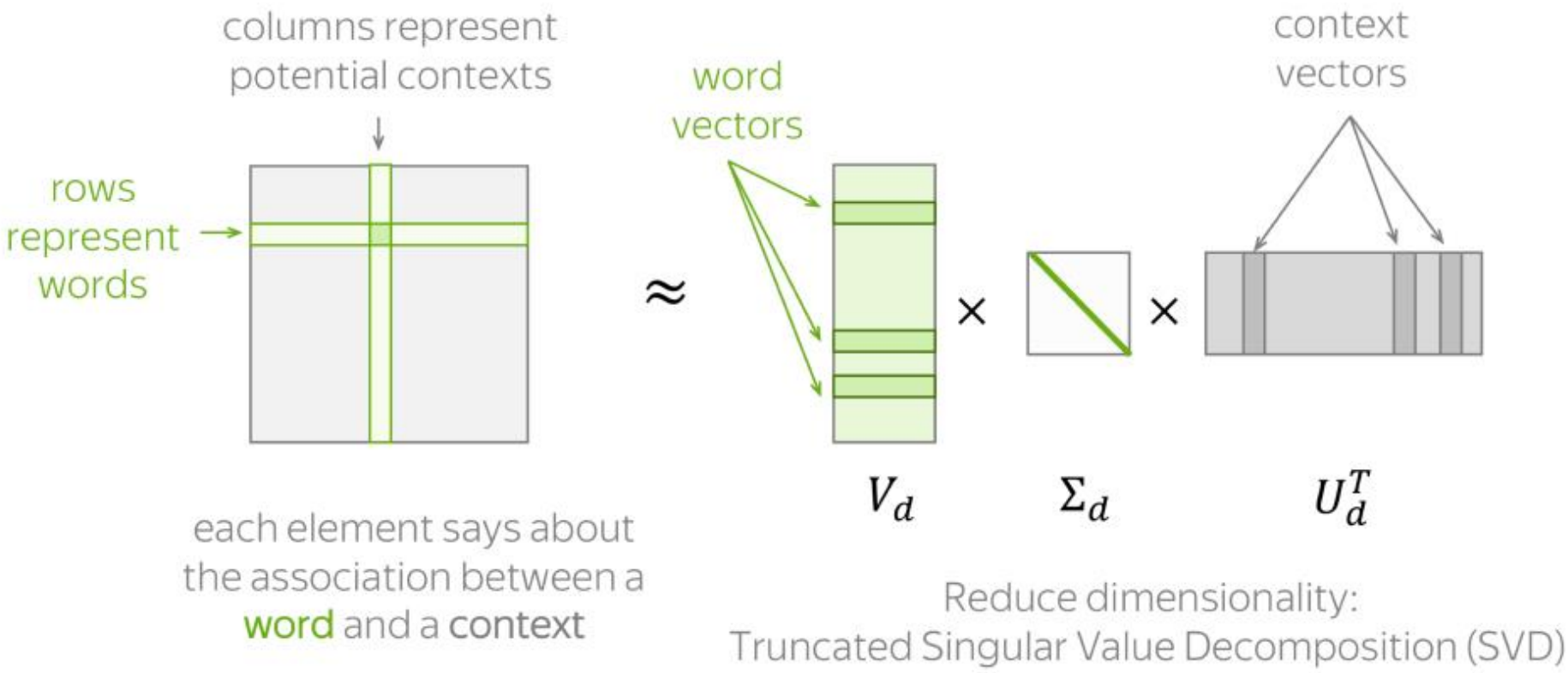
Идея методов, основанных на подсчёте (Count-Based Methods)

Основная идея:

разместить в векторы информацию про контексты

Данные методы подразумевают «вручную» задать эту информацию на основе глобальной статистики текстов.

Идея методов, основанных на подсчёте (Count-Based Methods)



Co-Occurrence Counts (как часто два слова стоят рядом)

Сегодня замечательный солнечный день, идеальный для лекции

Контекст для слова «день»: окружающие слова в окне размера L.

$N(w, c)$ — элемент матрицы, количество раз, которое слово w встречается в контексте c.

PPMI (Positive Pointwise Mutual Information)

Это мера, используемая для оценки того, насколько чаще два события (например, два слова) встречаются вместе по сравнению с тем, если бы они были статистически независимы.

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

where

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

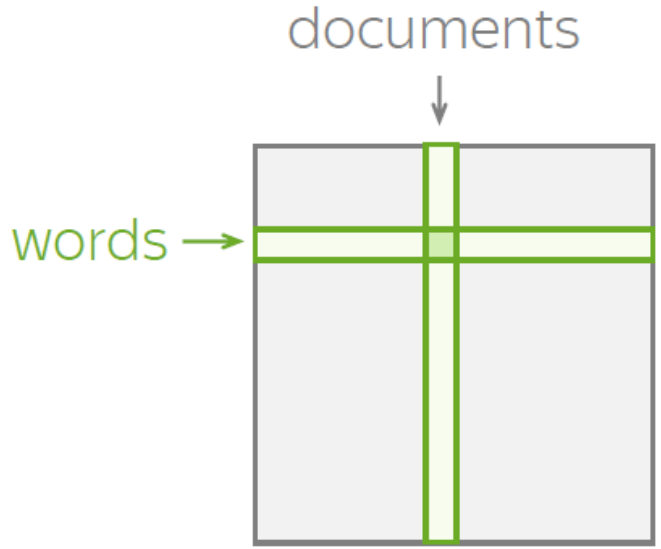
Анализ скрытых семантических структур (Latent Semantic Analysis (LSA))

Контекст: документ d из коллекции документов D

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

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

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



Count-Based Methods. Заключение

Основная идея:

- Положить в векторы информацию о контекстах.
- Заполнить эту информацию вручную на основе глобальной статистики корпуса.

Word2Vec (Word to Vector) (Prediction-Based Method)

Основная идея:

- положить информацию о контекстах в векторы
- обучение векторов слов через предсказание контекста

Изучаемые параметры: векторы слов

Цель: заставить каждый вектор «знать» о контекстах своего слова

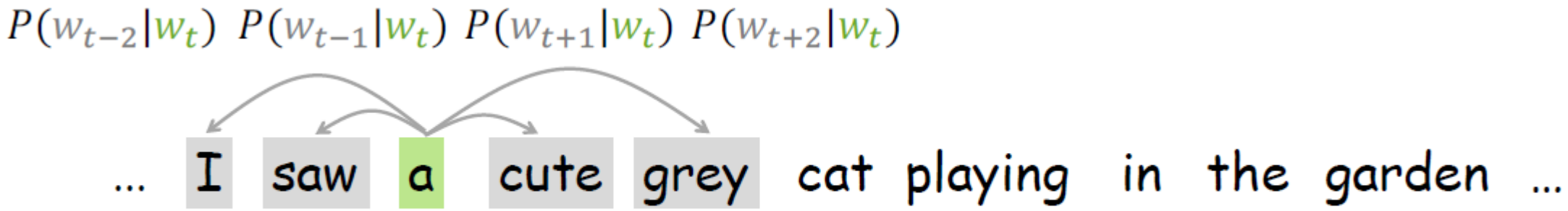
Способ: обучить векторы предсказывать возможные контексты по словам (или, альтернативно, по словам из контекстов)

Word2Vec: High-Level pipeline

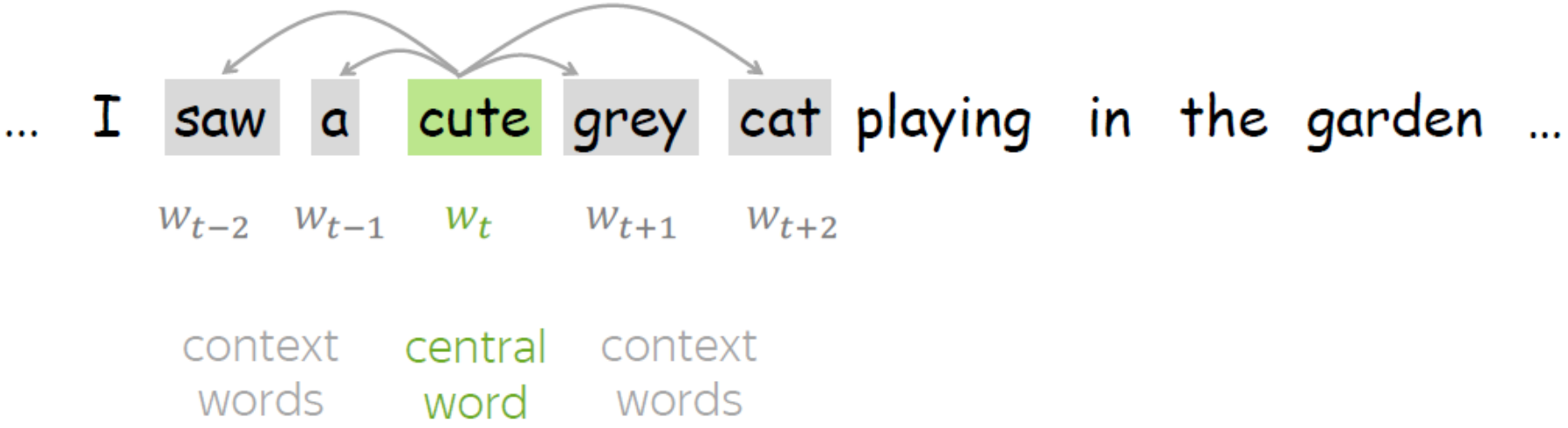
- взять большой текстовый корпус
- пройдитесь по тексту с помощью скользящего окна, перемещаясь по одному слову за раз
- для центрального слова вычислить вероятности контекстных слов
- скорректировать векторы, чтобы увеличить эти вероятности.

Word2Vec: High-Level pipeline

1.

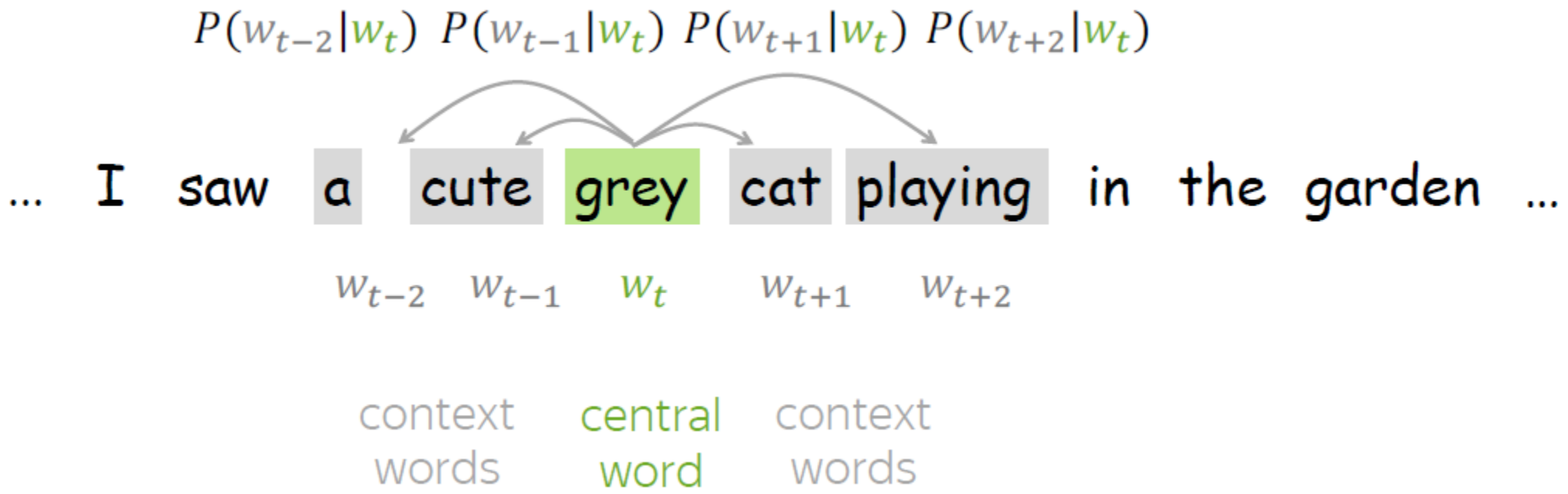


2.



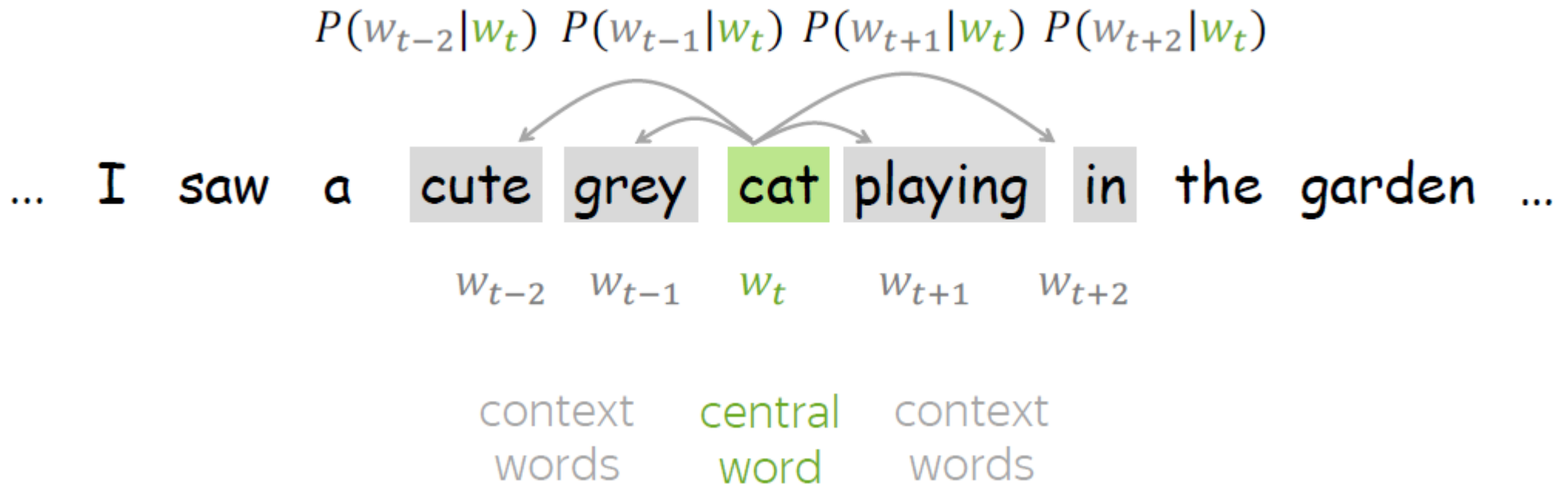
Word2Vec: High-Level pipeline

3.



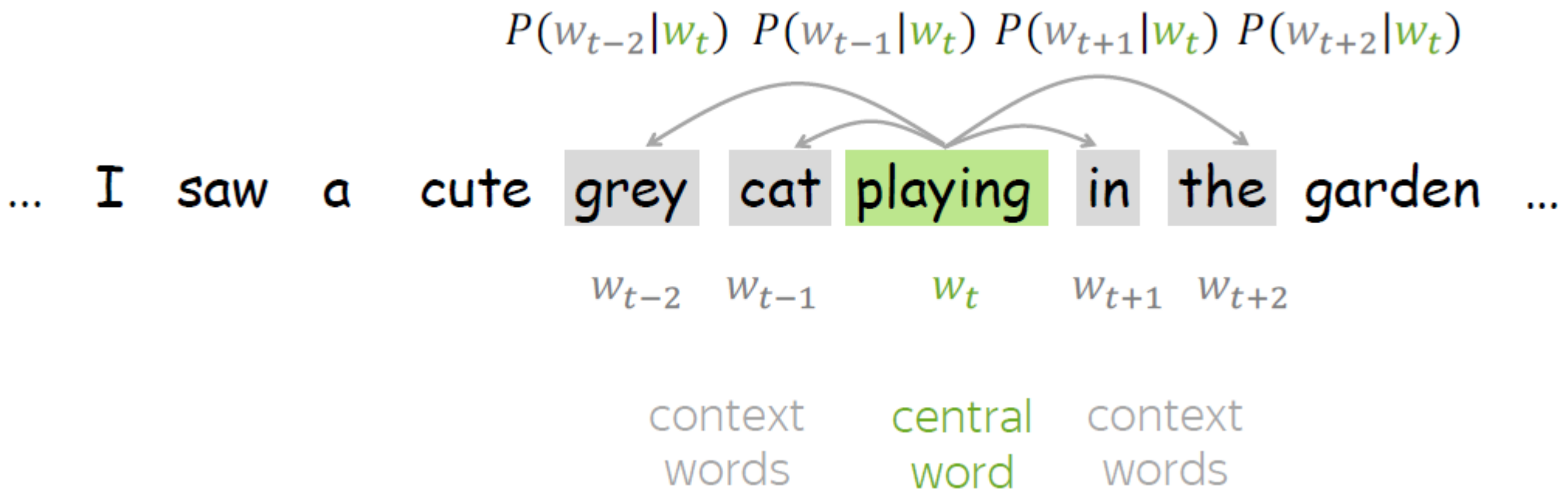
Word2Vec: High-Level pipeline

4.



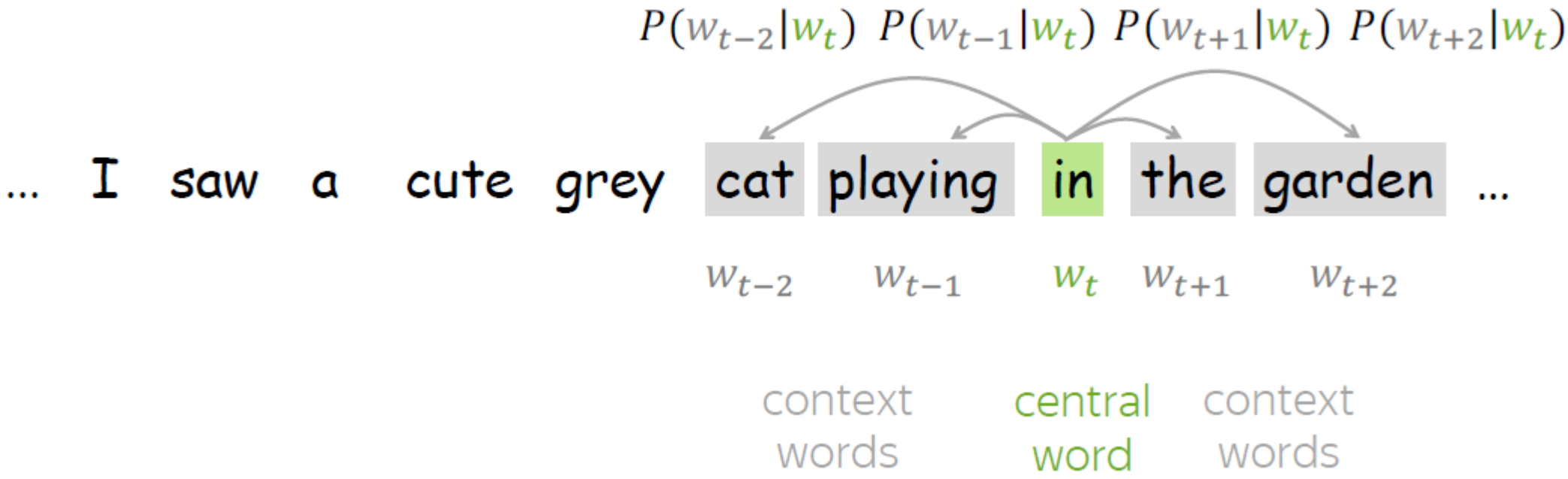
Word2Vec: High-Level pipeline

5.



Word2Vec: High-Level pipeline

6.



Целевая функция: отрицательный логарифм правдоподобия

Word2Vec пытается найти параметры, которые максимизируют вероятность данных:

Функция правдоподобия: $$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m, \\ j \neq 0}} P(w_{t+j} | w_t, \theta)$$

где θ — оптимизируемые векторы

Целевая функция: отрицательный логарифм правдоподобия

Используем отрицательное (логарифмическое) правдоподобие в качестве функции потерь:

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

Вычисление вероятности в логарифмической функции правдоподобия

$P(w_{t+j}|w_t, \theta)$

Для каждого слова w существует два вектора:

- v_w для центральной позиции слова
- u_w для контекстной позиции слова

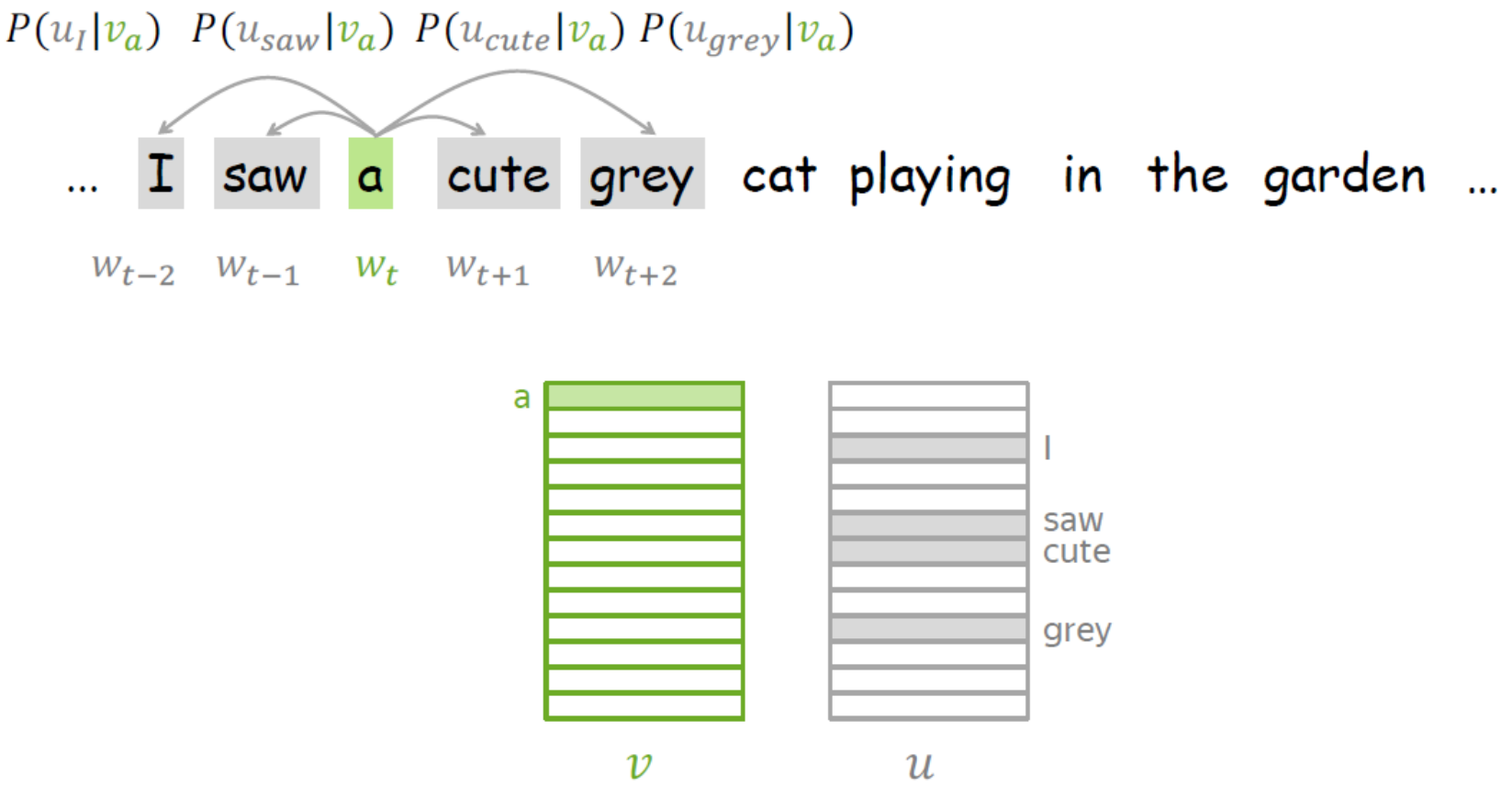
Для центрального слова c и контекстного слова o:

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

Цель: скорректировать векторы для увеличения вероятностей

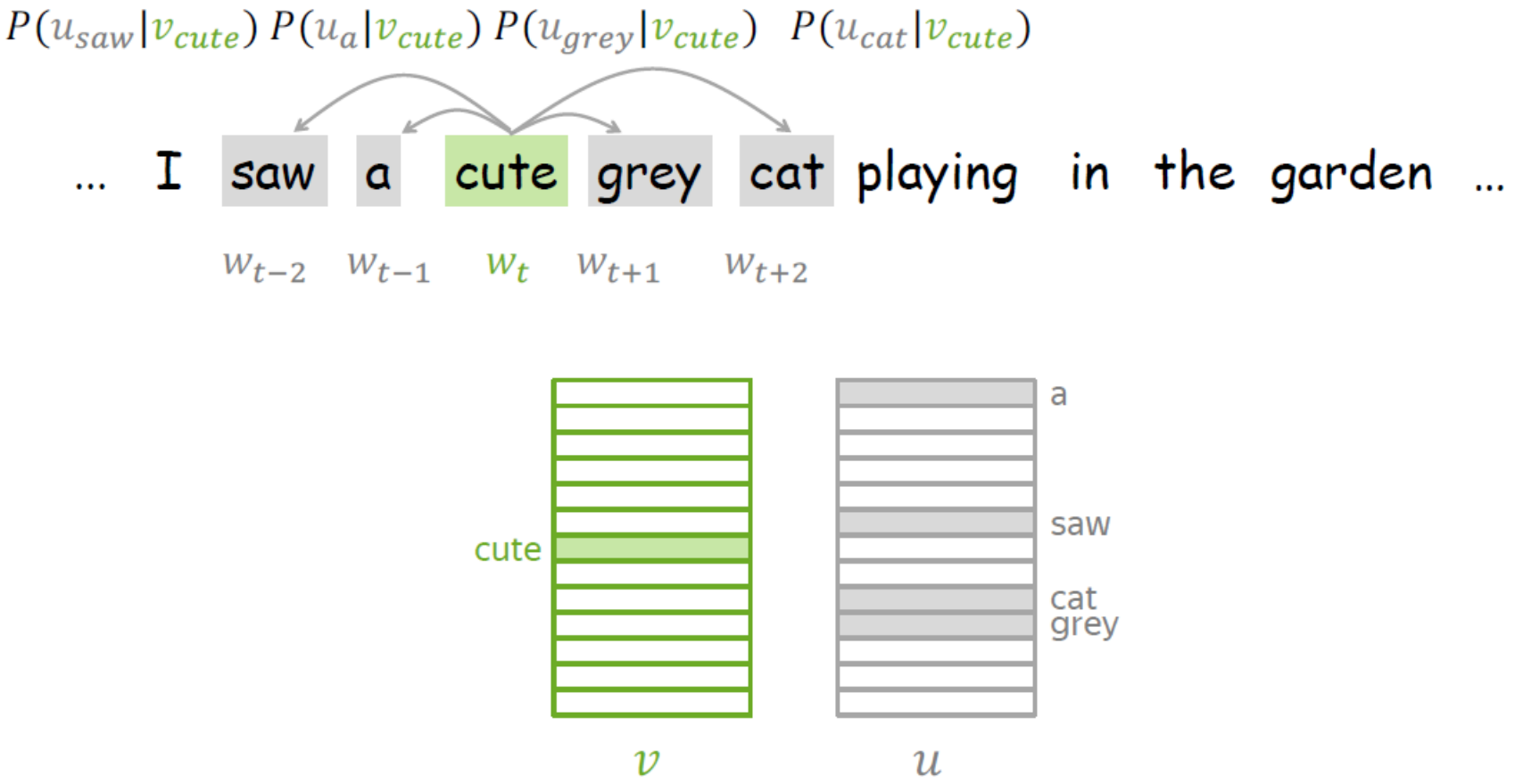
Два вектора для каждого слова. Пример

1.



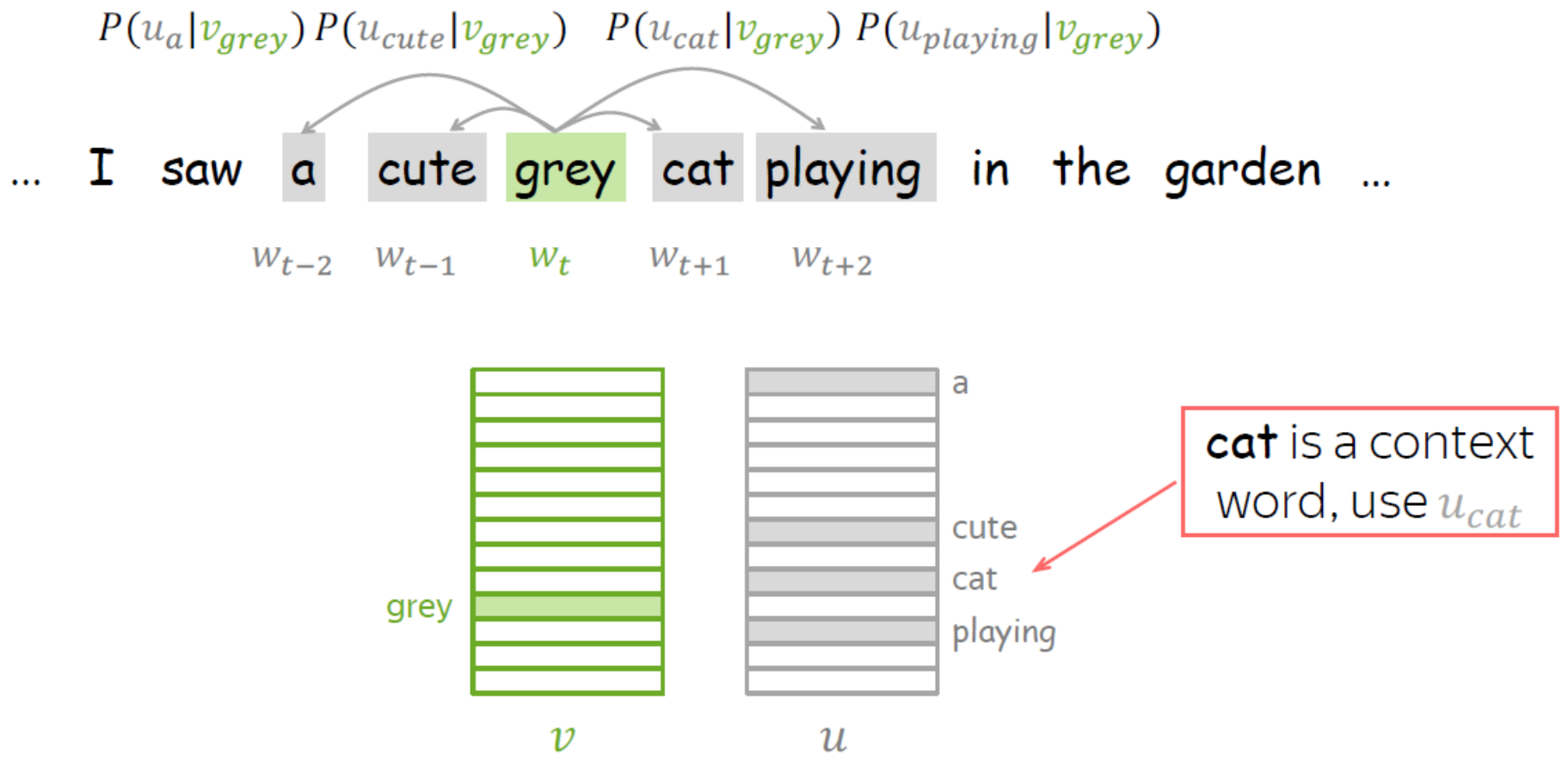
Два вектора для каждого слова. Пример

2.



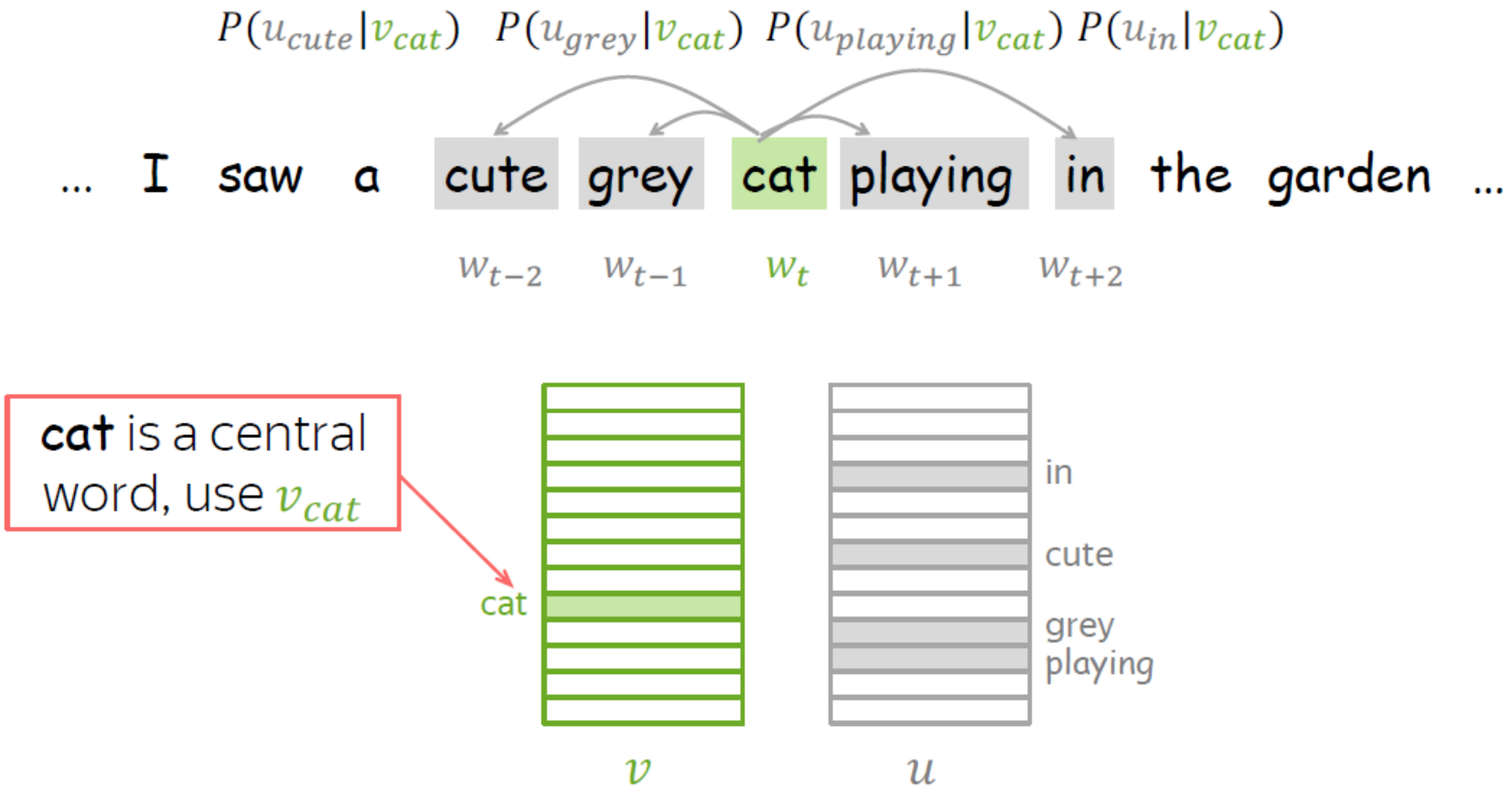
Два вектора для каждого слова. Пример

3.



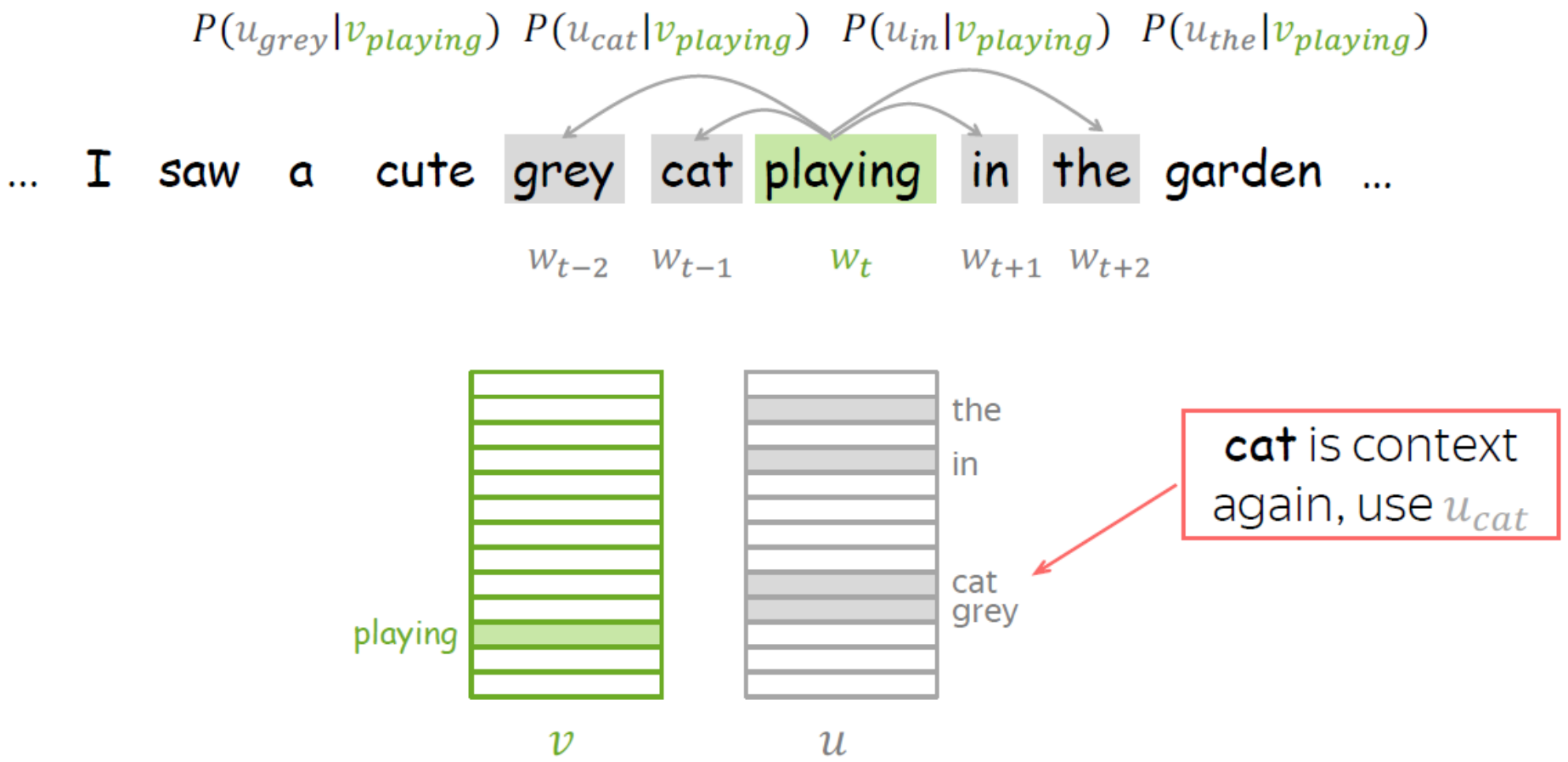
Два вектора для каждого слова. Пример

4.



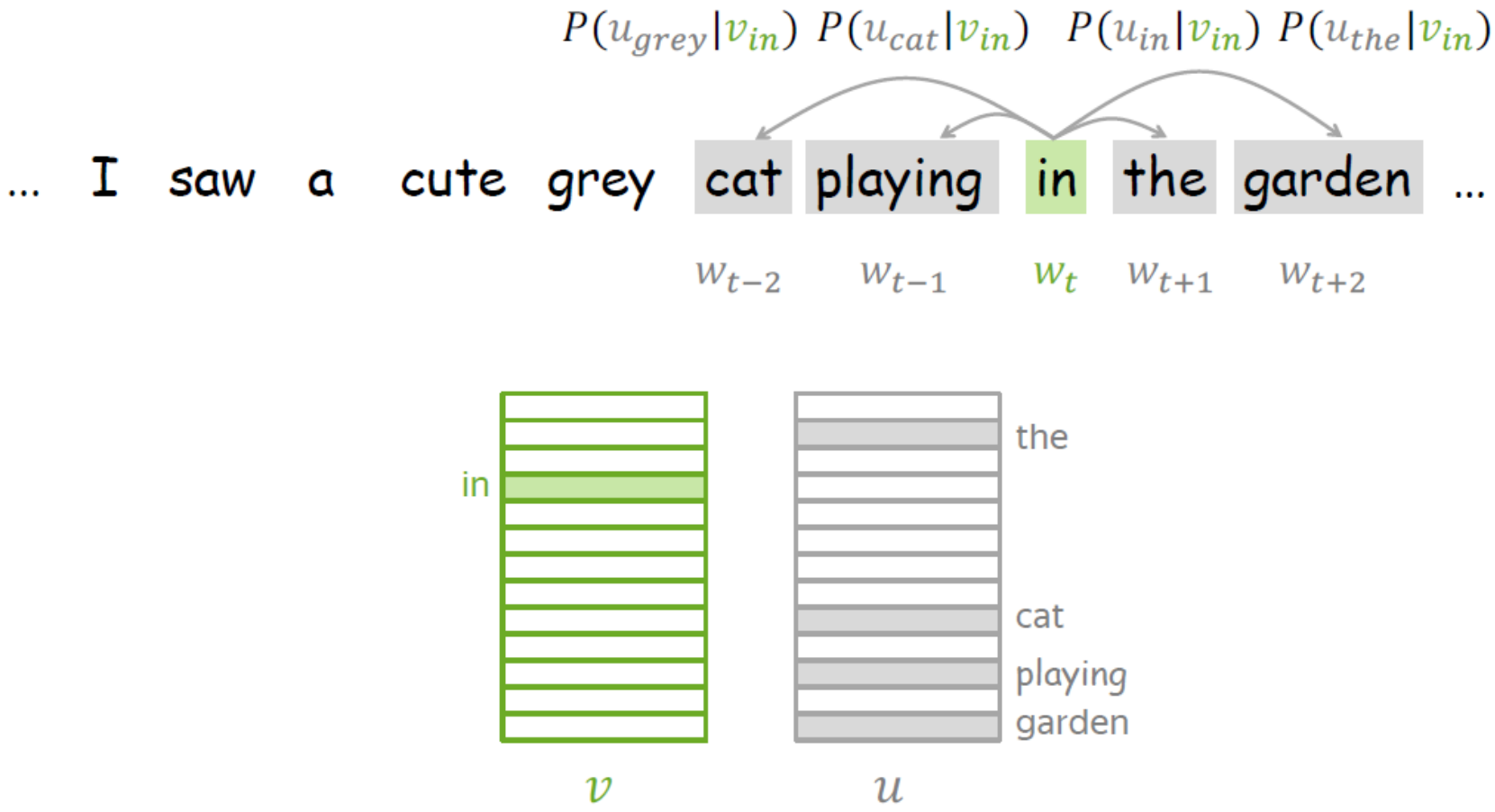
Два вектора для каждого слова. Пример

5.



Два вектора для каждого слова. Пример

6.



Процедура обучения

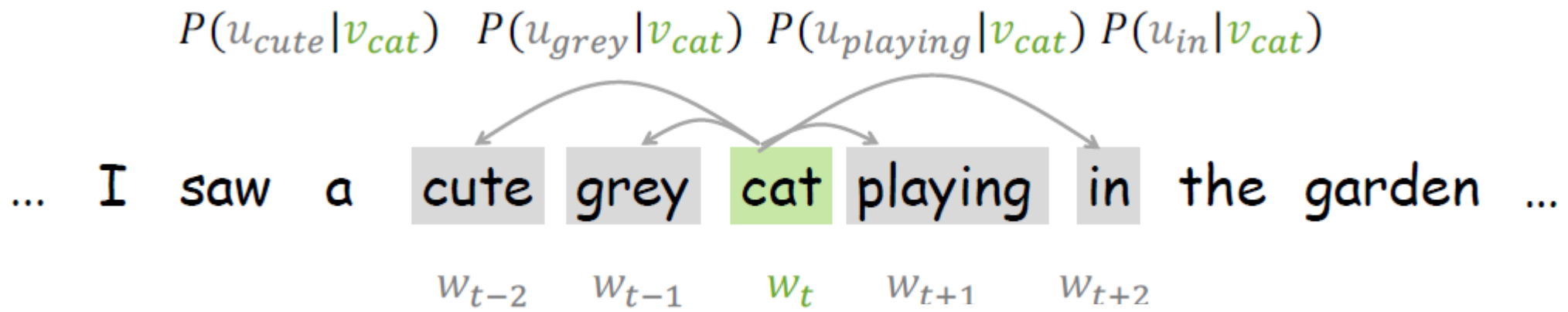
Есть функция потерь:

$$\text{Loss} = J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m,\\ j\neq 0}} \log P\left(w_{t+j}\middle|w_t,\theta\right)$$

Перепишем в виде: $-\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m,\\ j\neq 0}} J_{t,j}(\theta)$

потеря для слова j в контексте t

Процедура обучения. Пример



Выбираем одно из контекстных слов:

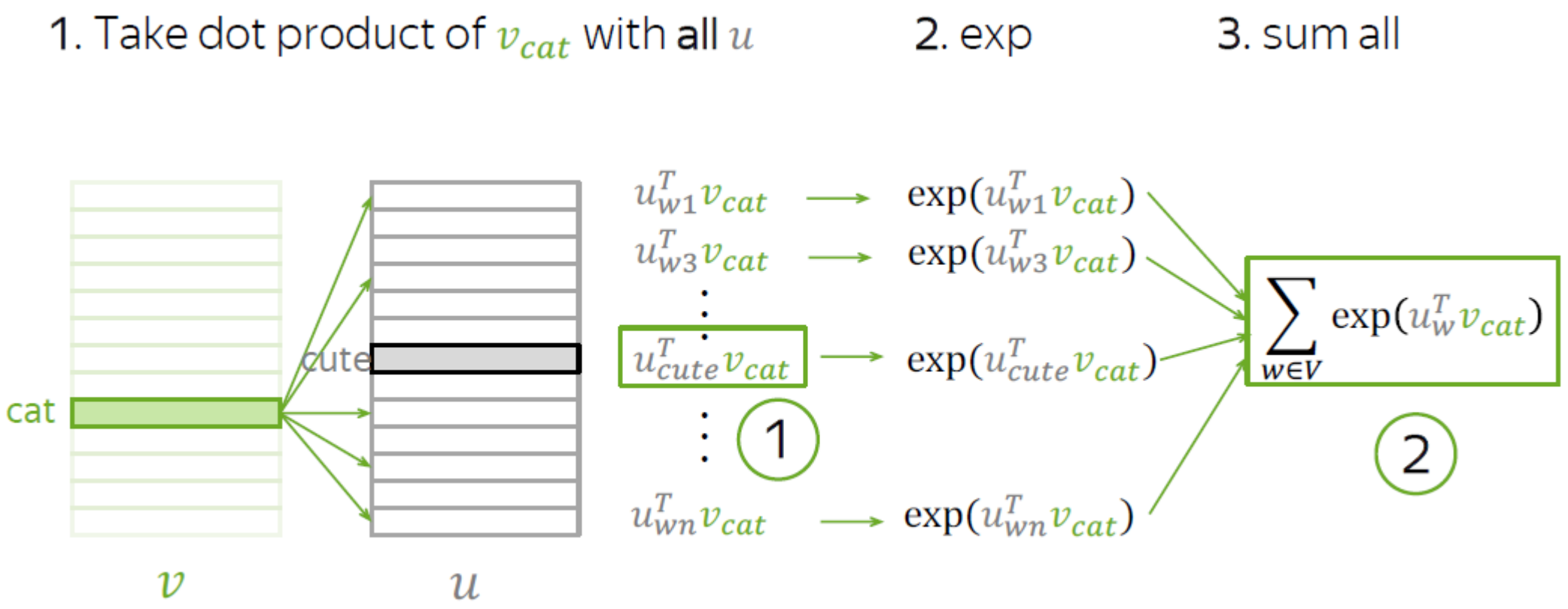
Процедура обучения. Пример

Рассмотрим функцию потерь для этого шага:

$$-\log P(cute|cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{w \in V} \exp(u_w^T v_{cat})}$$

$$= -u_{cute}^T v_{cat} + \log \sum_{w \in V} \exp(u_w^T v_{cat})$$

Процедура обучения. Пример



Процедура обучения. Пример

4. get loss (for this one step)

$$J_{t,j}(\theta) = -\underbrace{u_{cute}^T v_{cat}}_{1} + \underbrace{\log \sum_{w \in V} \exp(u_w^T v_{cat})}_{2}$$

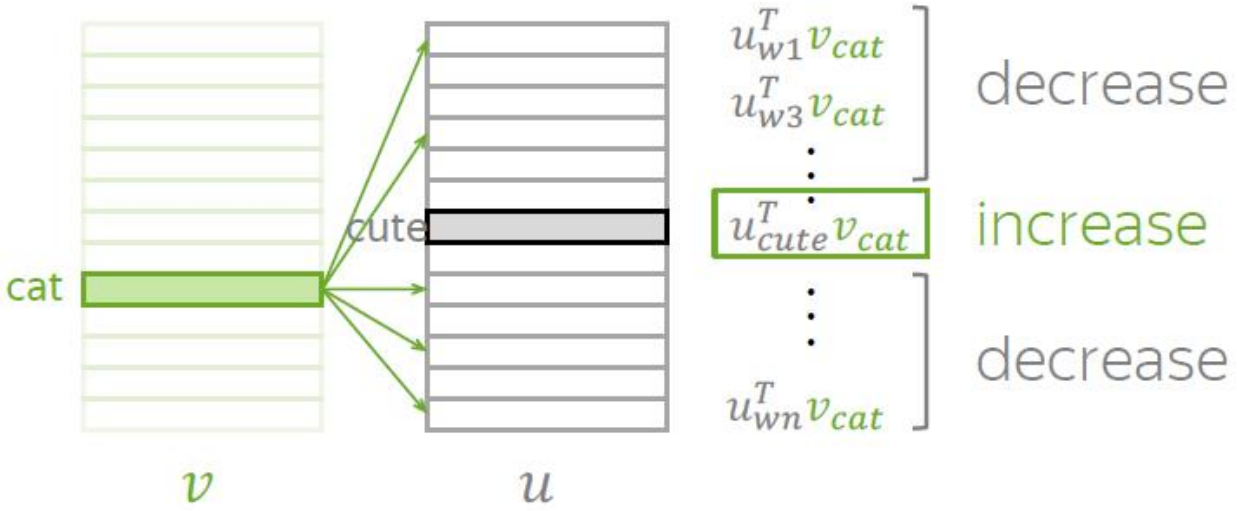
5. evaluate the gradient, make an update

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \quad \forall w \in V$$

Что происходит за одно обновление?

$$-\log P(cute|cat) = -u_{cute}^T v_{cat} + \log \sum_{w \in V} \exp(u_w^T v_{cat})$$

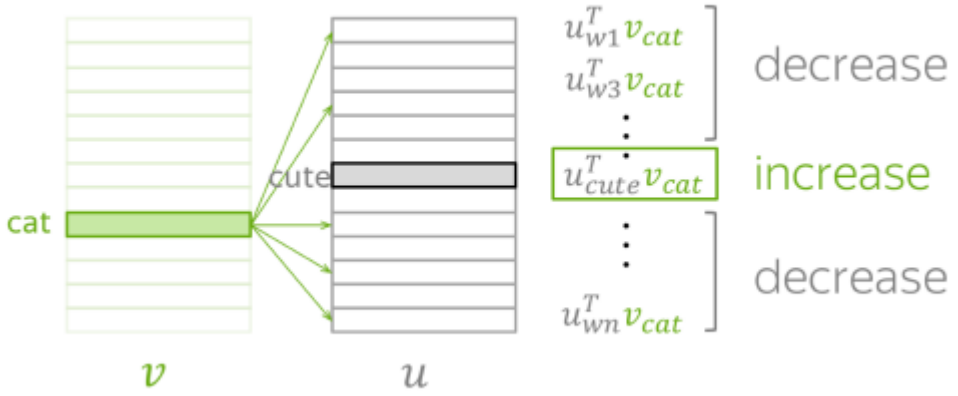


Результат каждого шага

Обычно увеличивается вероятность правильного контекста и уменьшаются остальные.

Необходимо обновить параметры:

- v_{cat}
- u_w для всех w в словаре



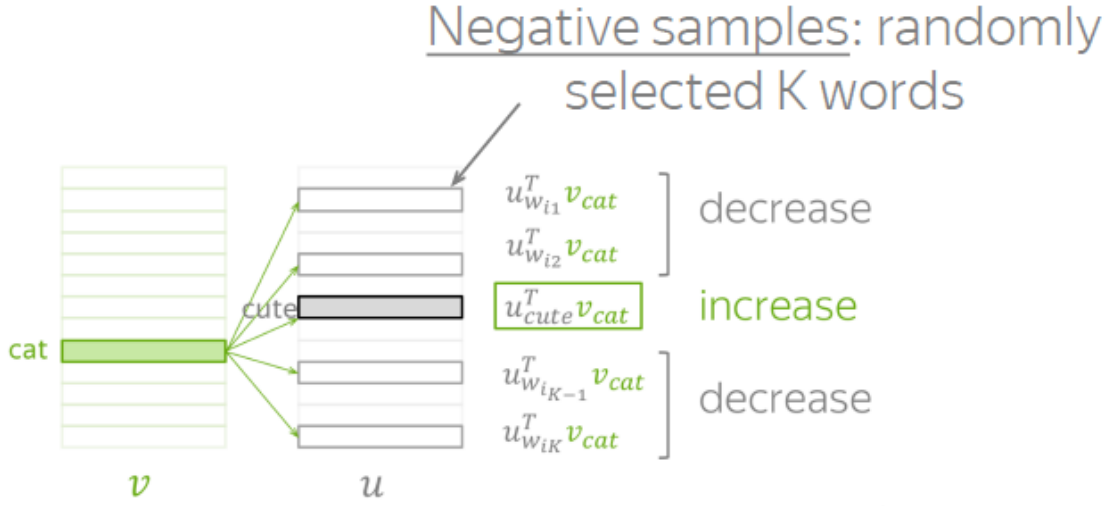
Проблема!

Negative Sampling (негативная выборка)

Идея: уменьшать вероятность какого-то подмножества слов (случайным образом выбирать K слов).

Необходимо обновить параметры:

- v_{cat}
- u_{cute} и u_w для всех w в подмножестве размера K



Word2Vec. Skip-Gram

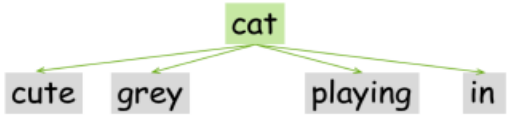
Главная гипотеза: Слова, которые встречаются в сходных контекстах, имеют сходные значения.

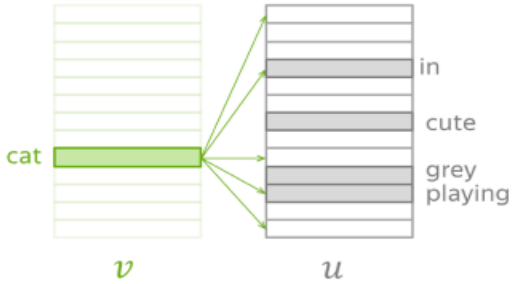
Задача Skip-Gram: Научить модель предсказывать окружающие слова (контекст) по заданному целевому слову.

Word2Vec. Skip-Gram. Пример

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

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





(this is what we did so far)

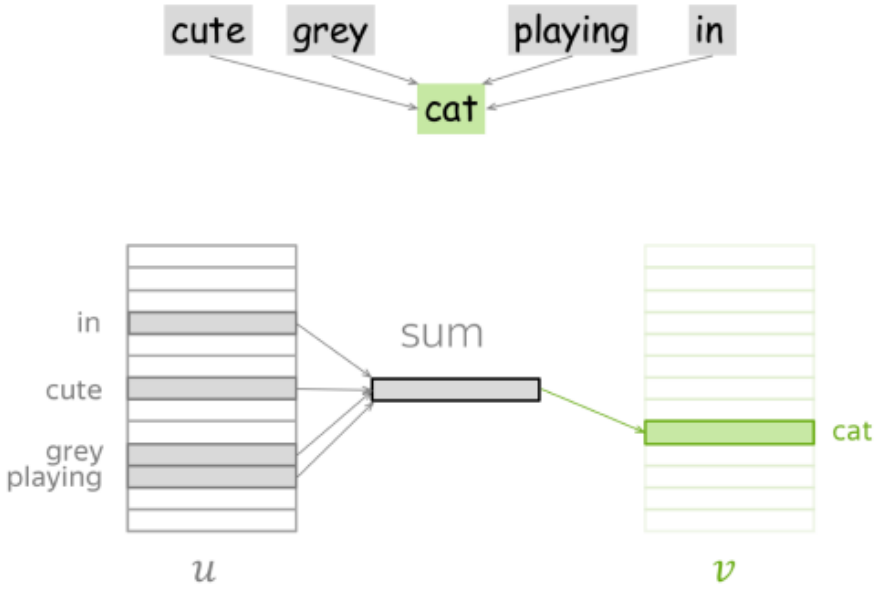
Word2Vec. CBOW

Главная гипотеза: Остается той же — слова в сходных контекстах имеют сходные значения. Но задача обратная.

Задача CBOW: Научить модель предсказывать **целевое (пропущенное) слово** по окружающим его словам (контексту).

Word2Vec. CBOW. Пример

CBOW: from sum of context predict central



(Continuous Bag of Words)

Стандартные гиперпараметры

- Модель: Skip-Gram с отрицательной выборкой;
- Количество отрицательных примеров: для небольших наборов данных 15–20; для больших наборов данных (которые обычно используются) может быть 2–5.
- Размерность встраивания: часто используемое значение — 300, но возможны и другие варианты (например, 100 или 50).
- Размер скользящего окна (контекста): 5–10.

GloVe (Global Vectors for Word Representation)

- В **Count-Based** методах информация приходит из глобальной статистики всего корпуса текстов, а векторы получены путём понижения размерности.
- В **Prediction-based** методах информация приходит из последовательного предсказания локальных контекстов, а векторы обучаются методом градиентного спуска.
- В **GloVe** информация приходит из глобальной статистики всего корпуса текстов, а векторы обучаются методом градиентного спуска.

Внутренняя и внешняя оценки качества

Внутренняя оценка основана на **лингвистических свойствах**, без привязки к какой-либо конкретной прикладной задаче.

Внешняя оценка - это метод оценки качества word embeddings **на основе их производительности в реальных, конечных задачах.**

Внутренняя оценка обычно быстрее, но не говорит, что лучше на практике.

Внешняя оценка говорит, что лучше на практике, но обучение реальных моделей обходится дорого.

Анализ и интерпретируемость

Closest to **frog**:

frogs
toad
litoria
leptodactylidae
rana
lizard
eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

Анализ и интерпретируемость

Критерии оценки качества векторных представлений:

- Содержат пары слов с оценками сходства, полученными от людей.
- Для оценки используется корреляция между двумя оценками сходства: полученными от людей и полученными от векторных представлений.

Анализ и интерпретируемость

word pair		score
vulgarism	profanity	9.62
subdividing	separate	8.67
friendships	brotherhood	7.5
exceedance	probability	5.0
assigned	allow	3.5
marginalize	interact	2.5
misleading	beat	1.25
radiators	beginning	0

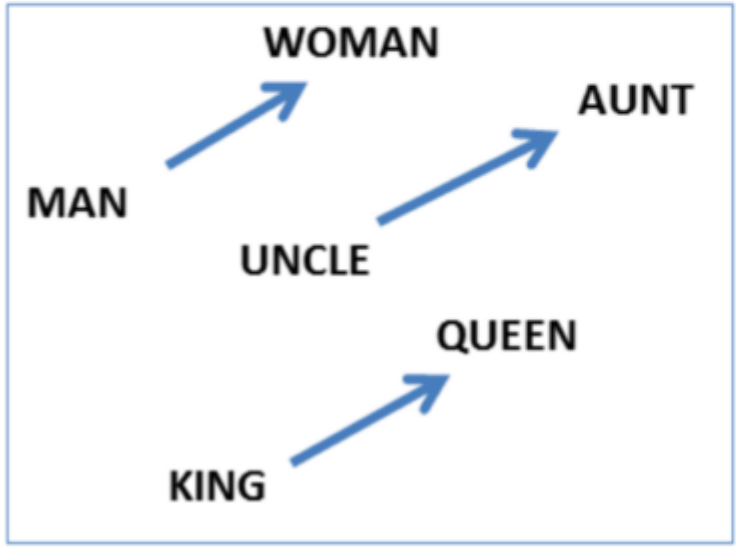
Линейная структура

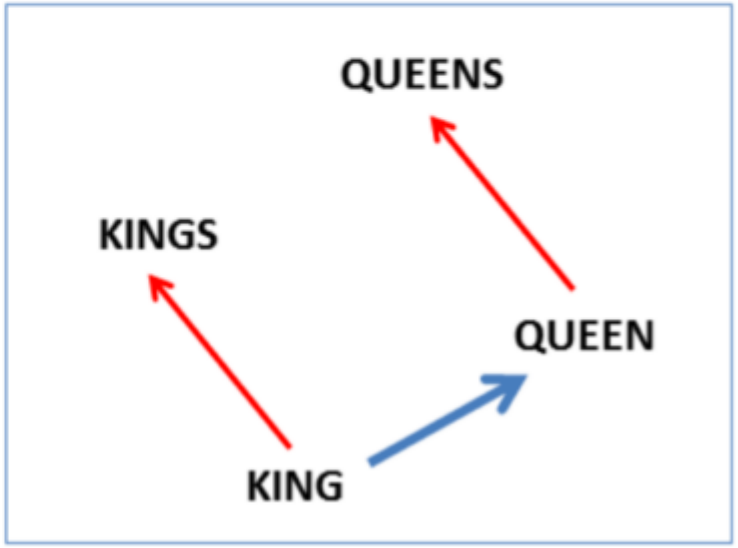
Многие семантические и синтаксические отношения между словами (почти) линейны в пространстве вложений.
Это означает, что сложные семантические или синтаксические связи между словами можно выразить с помощью простых **векторных арифметических операций**, в основном сложения и вычитания.

Линейная структура

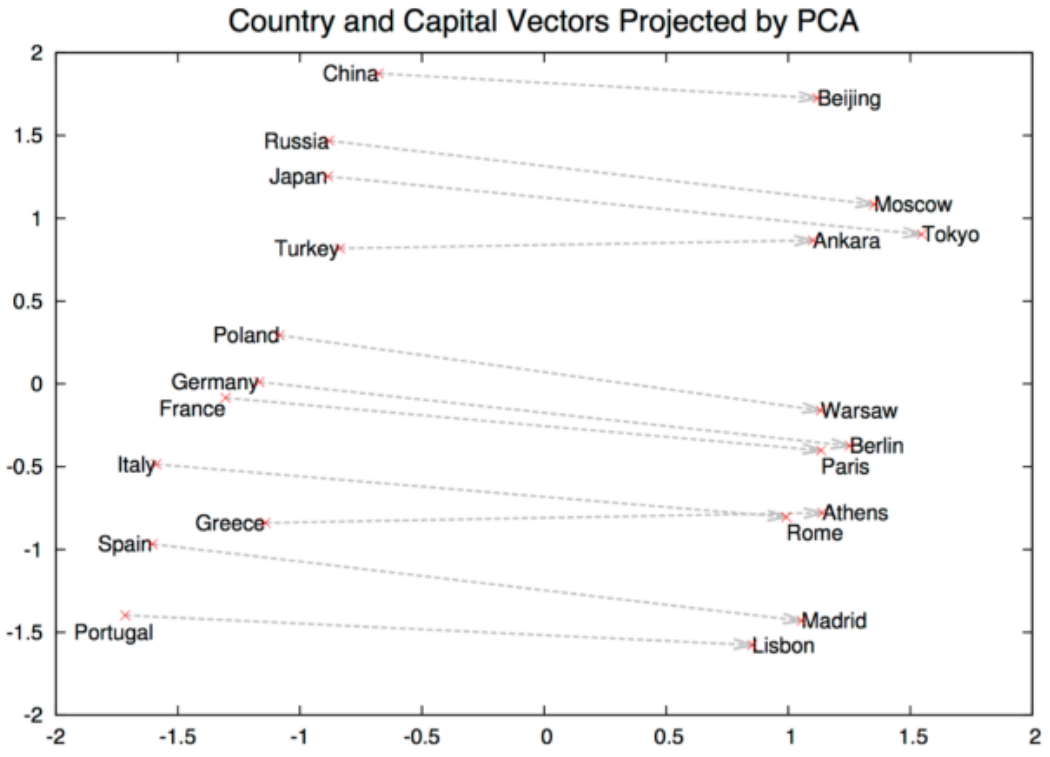
semantic: $v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen})$

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

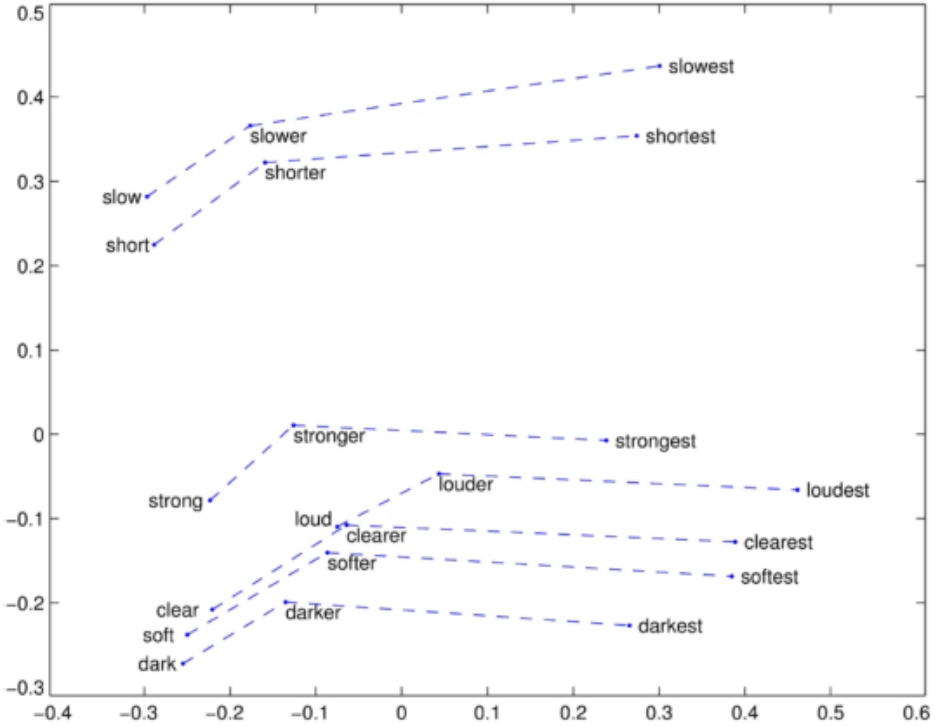




Линейная структура



Word2Vec



GloVe

Сходства в разных языках

Пример создания больших словарей из маленьких

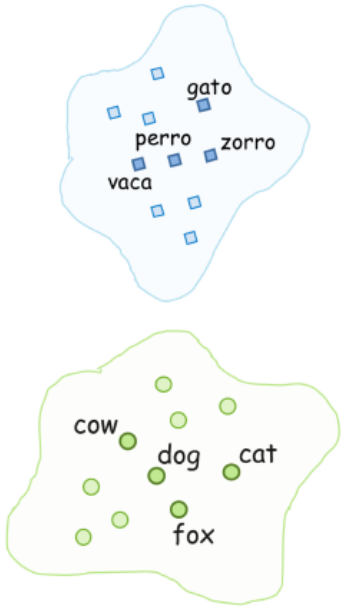
Ingredients:

- 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
…

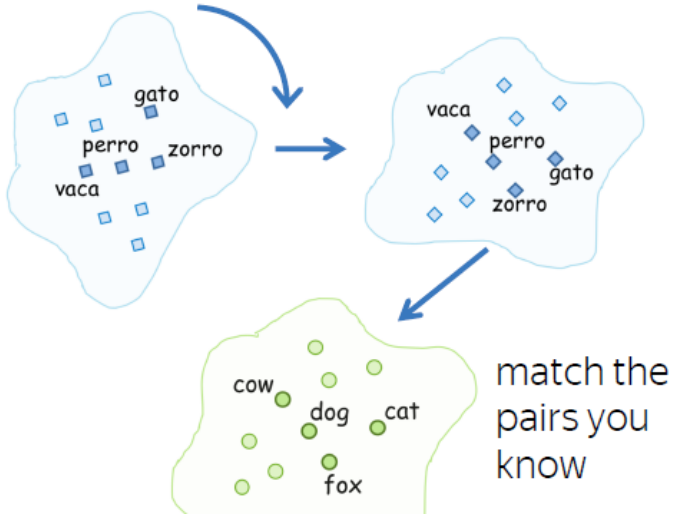
Step 1:

- train embeddings for each language



Step 2:

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



match the pairs you know

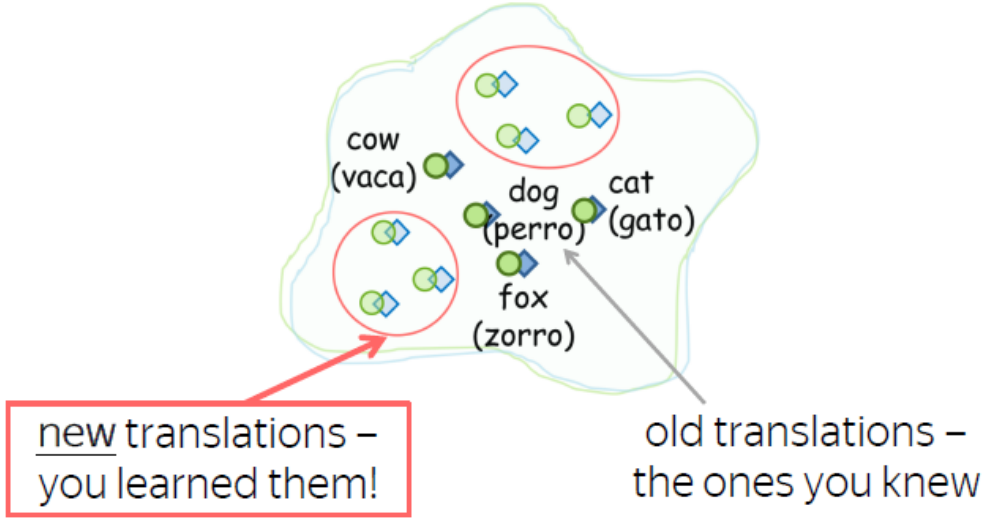
Сходства в разных языках

Steps1-2:

- match words from the vocabulary

Step 3:

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



Лабораторная работа 1

Большая языковая модель (LLM) — это core-технология, которая учится на текстах и предсказывает последующие слова. Она нужна для всего, что связано с пониманием и генерацией человеческого языка: переводчики, помощники, поиск, исправление ошибок и многое другое. Современные большие языковые модели (LLM) на основе архитектуры Transformer (как GPT) стали настолько мощными, что превратились в универсальные инструменты для решения самых разных задач.

Лабораторная работа 1

pip install torch

PyTorch — это одна из самых популярных и мощных библиотек с открытым исходным кодом для машинного обучения. В основе PyTorch лежит объект Tensor. Тензоры PyTorch могут вычисляться на **графическом процессоре (GPU)**. Это ускоряет операции в десятки и сотни раз, что критически важно для обучения больших нейронных сетей.

Лабораторная работа 1

pip install transformers

Эта команда устанавливает библиотеку **Transformers** от **Hugging Face** 🤗. **Transformers — это огромный сборник предварительно обученных моделей-трансформеров**, который позволяет вам за несколько строк кода использовать самые современные модели для задач вроде перевода, генерации текста, анализа тональности и многих других.

Hugging Face Hub — это своего рода "GitHub для моделей ML". Через библиотеку transformers вы можете загрузить и использовать тысячи моделей, обученных как самим Hugging Face, так и сообществом.

Лабораторная работа 1

pip install accelerate

Эта команда устанавливает библиотеку **accelerate** от **Hugging Face**🤗.
Эта библиотека автоматически оптимизирует вычисления под ваше железо, экономит память и ускоряет работу моделей.

Лабораторная работа 1

pip install nltk

Устанавливает библиотеку для обработки естественного языка (Natural Language Processing, или NLP) на Python — **Natural Language Toolkit (NLTK)**. Это **фундаментальный инструментарий для работы с языком с помощью правил и статистики**. Это библиотека для обучения, исследований и решения стандартных задач NLP, где не требуется сверхвысокая точность нейросетей, но важны прозрачность, скорость и контроль.

Лабораторная работа 1

from transformers import AutoModelForCausalLM, AutoTokenizer

AutoTokenizer - класс для автоматической загрузки и работы с **токенизатором**

AutoModelForCausalLM - класс для автоматической загрузки **собственно модели** для генерации текста

Лабораторная работа 1

Qwen/Qwen2-1.5B-Instruct

Это современная компактная языковая модель от Alibaba Group, предназначенная для диалога и выполнения инструкций.

Qwen (*Qianwen*): Бренд больших языковых моделей Alibaba. Переводится с китайского как "Тысяча вопросов".

2: Второе поколение архитектуры и обучения моделей Qwen.

1.5B: 1.5 миллиарда параметров. Это относительно небольшой размер по современным меркам (для сравнения: GPT-3.5 имеет 175B параметров).

Instruct: Модель прошла дополнительное обучение с учителем (Fine-tuning) в отличие от её "базовой" версии (Qwen2-1.5B), которая просто предсказывает текст.