

Лекции 1-2.

Word embeddings

О курсе «Искусственный интеллект»

- Yandex School of Data Analysis
- NLP Course For You

Баллы за курс

- Зачёт
- 60 баллов за Лабораторные работы (6 работ по 10 баллов)
- 40 баллов за Лекционные занятия (в конце каждой лекции мини-тест или мини-задание на 2 балла, остальные баллы можно получить за финальный тест (конец декабря))

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, эмбеddинг) — 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».

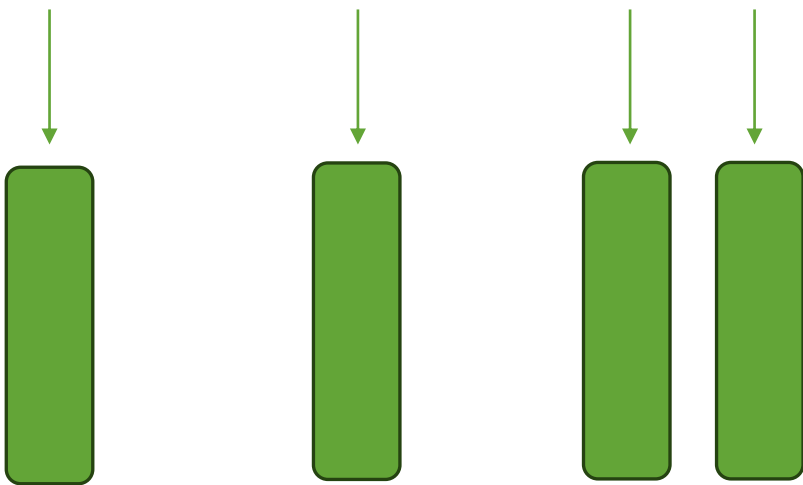
Зачем нужно векторное представление слов?

Предложение: Сегодня хороший день.

Разбиваем предложение на последовательность токенов:

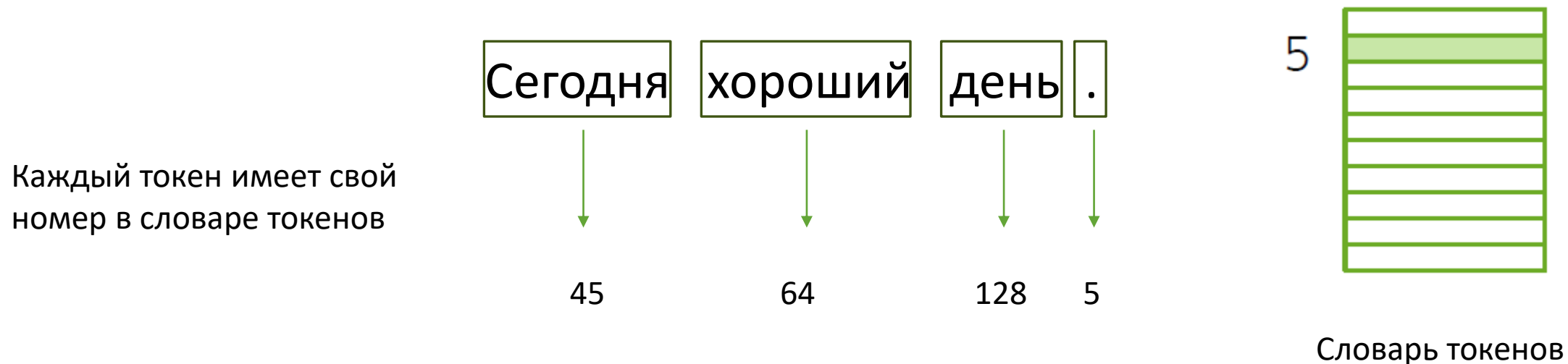
Сегодня хороший день .

Ставим в соответствие каждому слову вектор



Зачем нужно векторное представление слов?

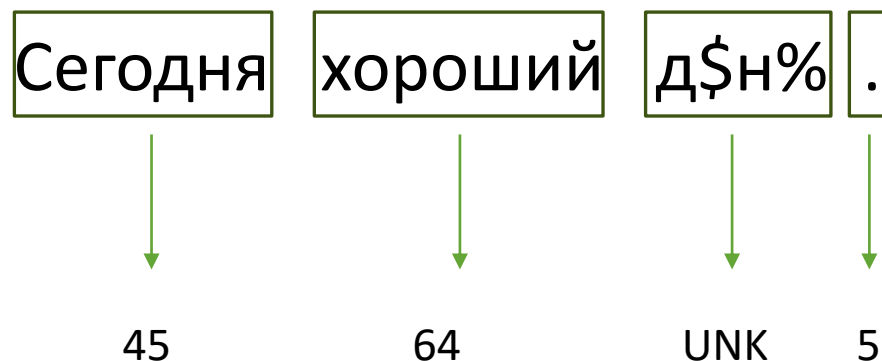
Каждый токен имеет свой номер в словаре токенов:



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

Какие могут возникать проблемы при таком представлении слов?

Размерность векторного представления = размеру словаря

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

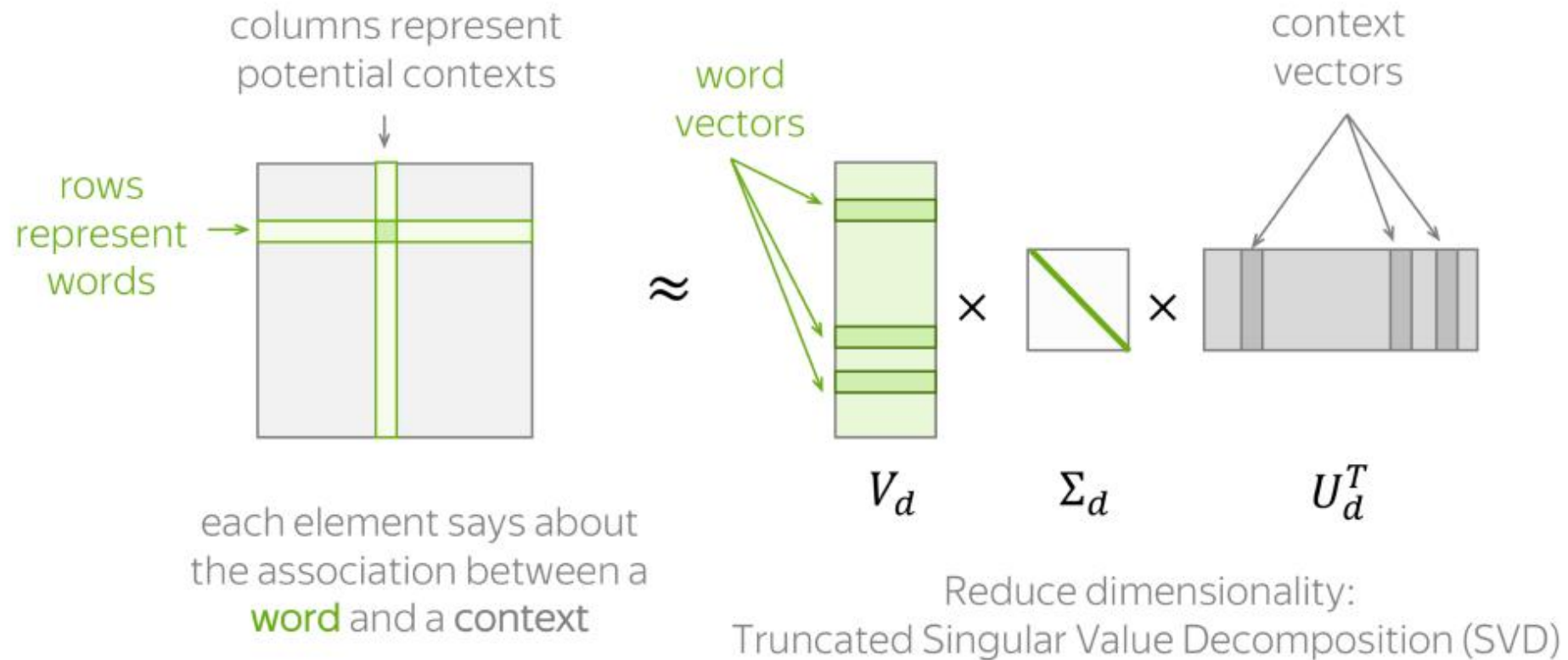
Идея методов, основанных на подсчёте (Count-Based Methods)

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

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

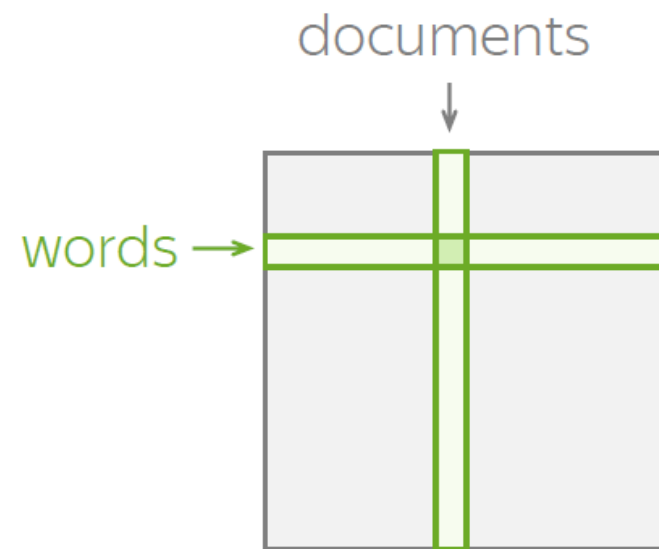
Анализ скрытых семантических структур (Latent Semantic Analysis (LSA))

Контекст: документ d из коллекции документов D

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

$N(w, d)$

$$\log \frac{|D|}{|\{d \in D: w \in d\}|}$$



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)

Две основные архитектуры для обучения: CBOW и Skip-gram.

Word2Vec: High-Level pipeline

- взять большой текстовый корпус
- пройтись 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.

$$P(w_{t-2}|w_t) \quad P(w_{t-1}|w_t) \quad P(w_{t+1}|w_t) \quad P(w_{t+2}|w_t)$$

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

2.

$$P(w_{t-2}|w_t) \quad P(w_{t-1}|w_t) \quad P(w_{t+1}|w_t) \quad P(w_{t+2}|w_t)$$

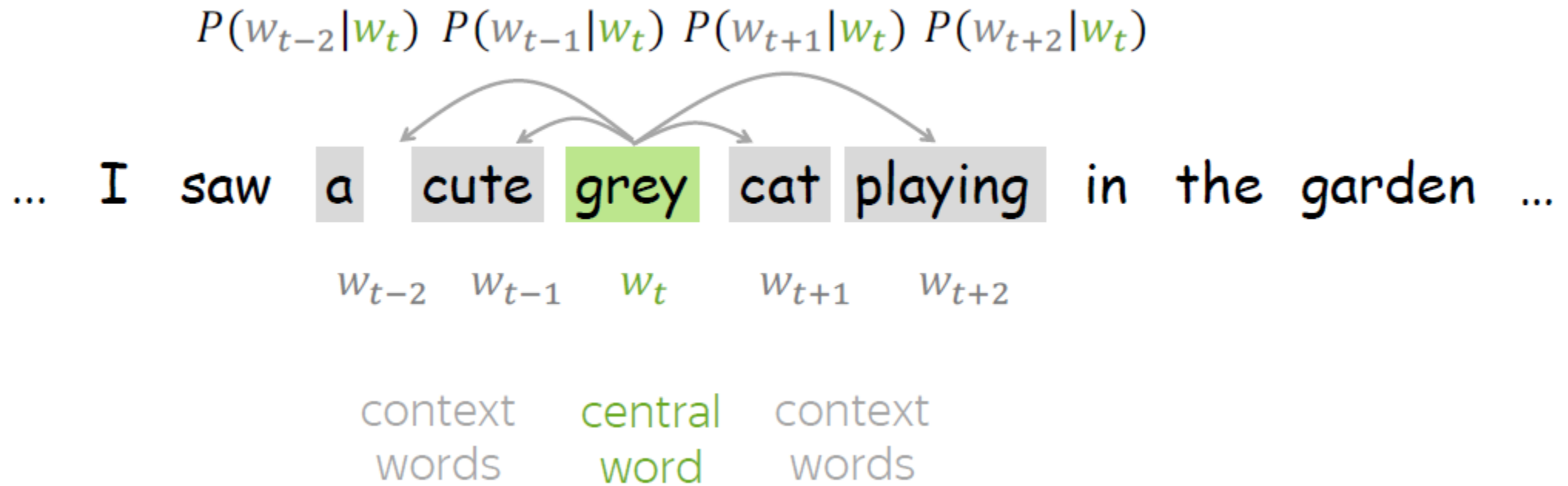
... I saw a cute grey cat playing in the garden ...

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context central context
words word words

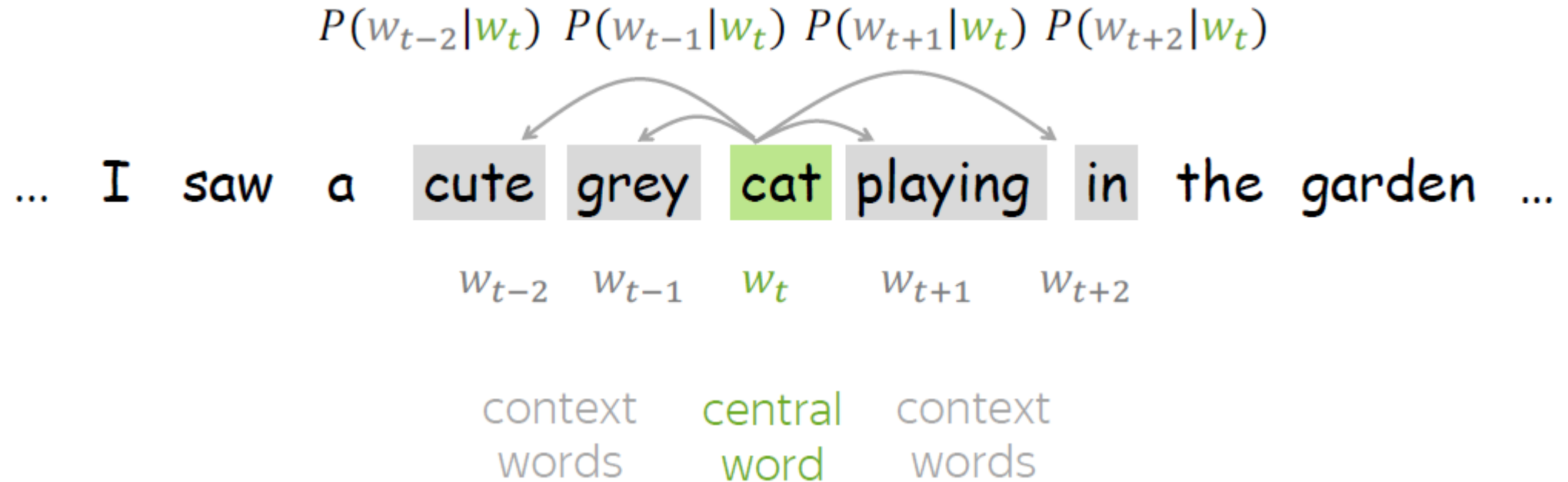
Word2Vec: High-Level pipeline

3.



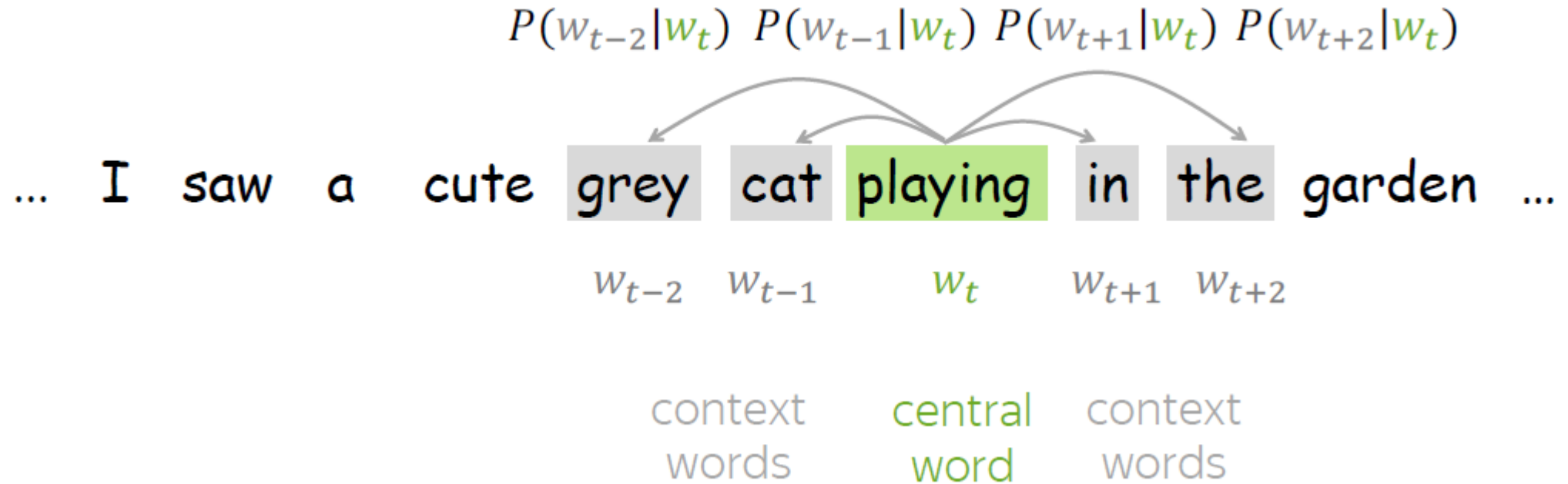
Word2Vec: High-Level pipeline

4.



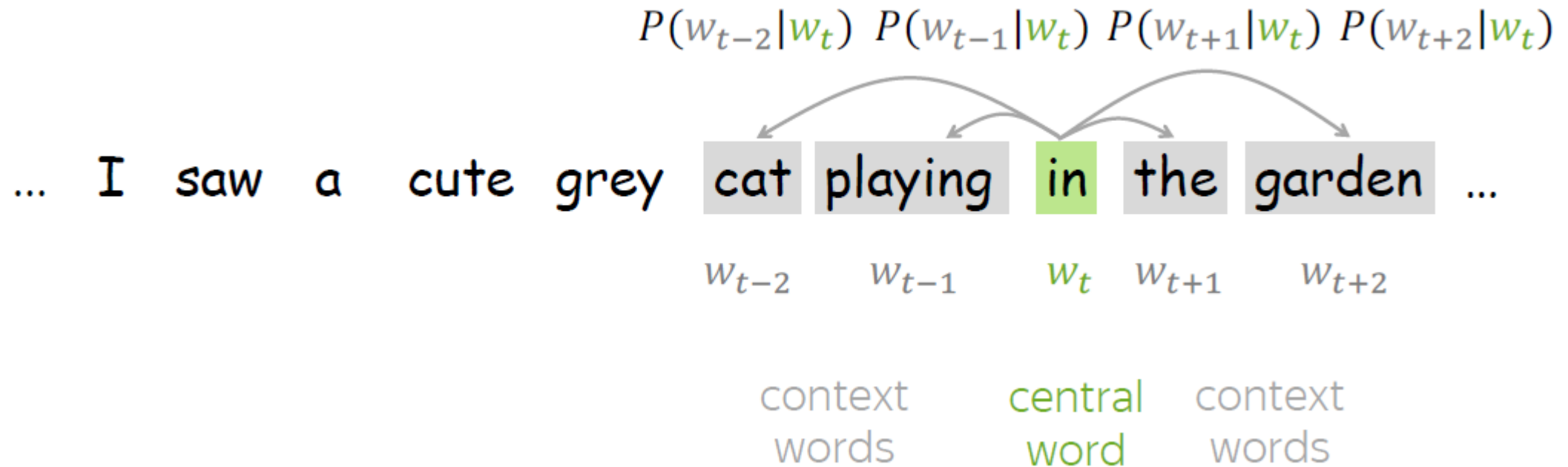
Word2Vec: High-Level pipeline

5.



Word2Vec: High-Level pipeline

6.



Целевая функция: отрицательный логарифм правдоподобия

Word2Vec пытается найти параметры, которые максимизируют вероятность данных:

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

где θ — оптимизируемые векторы

Целевая функция: отрицательный логарифм правдоподобия

Используем отрицательное (логарифмическое) правдоподобие в качестве функции потерь:

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

Вычисление вероятности в $P(w_{t+j}|w_t, \theta)$ логарифмической функции правдоподобия

Для каждого слова w существует два вектора:

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

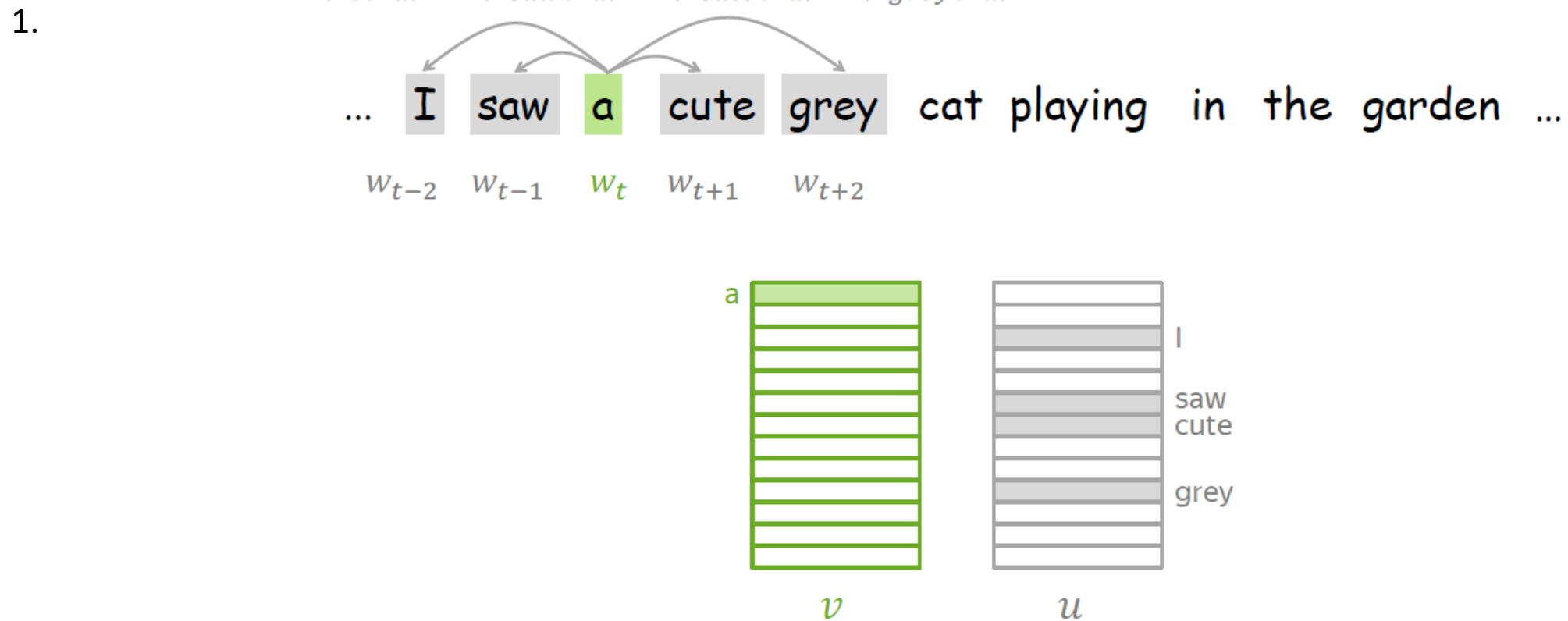
Для центрального слова c и контекстного слова o :

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

Цель: скорректировать векторы для увеличения вероятностей

Два вектора для каждого слова.

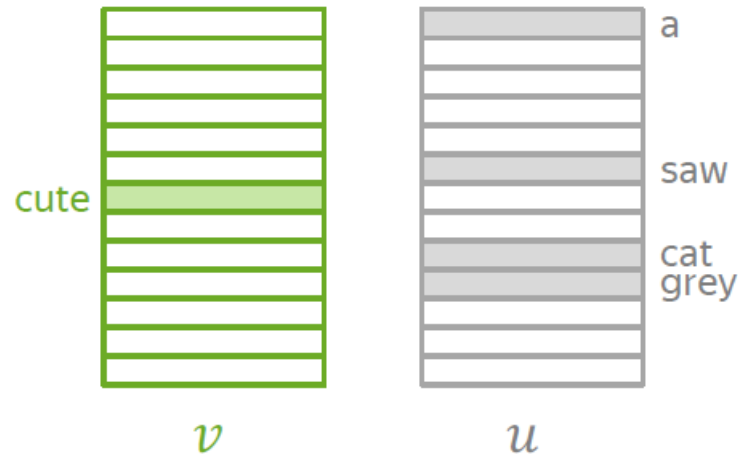
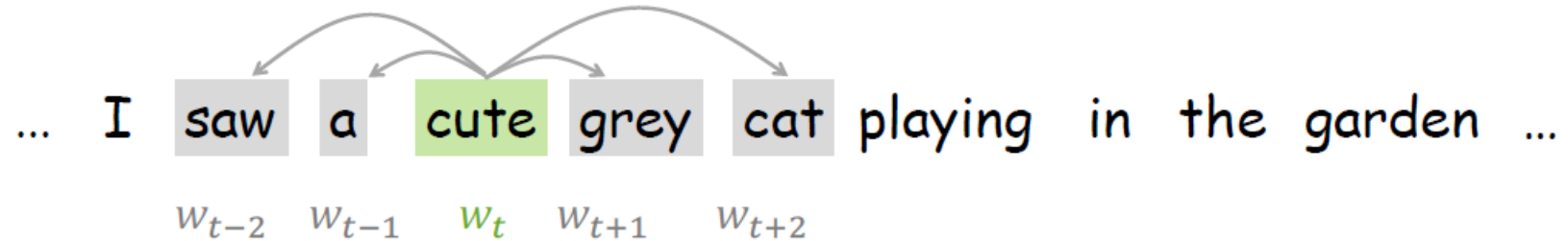
Пример



Два вектора для каждого слова. Пример

2.

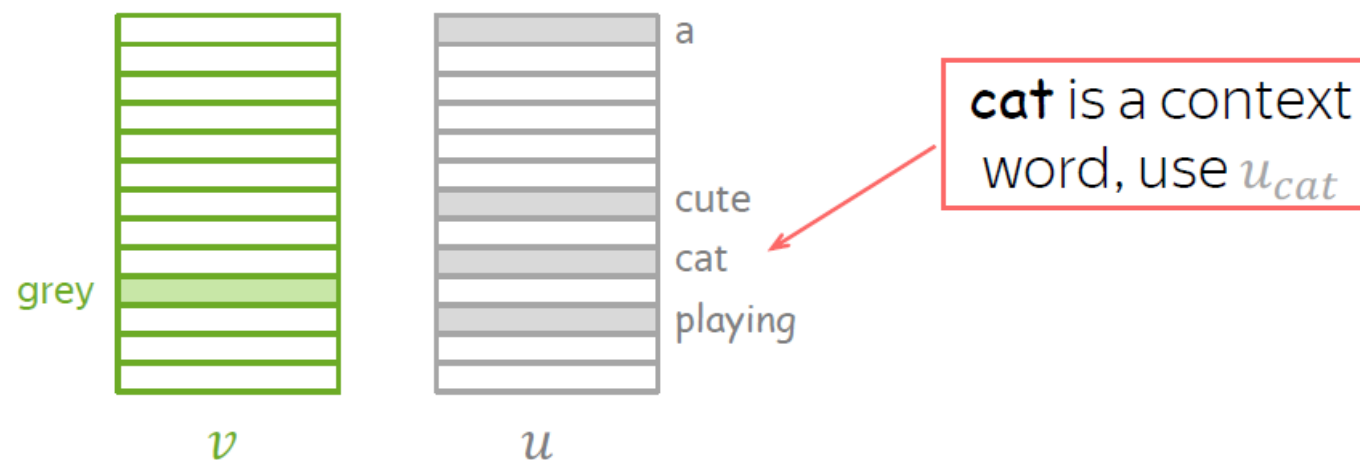
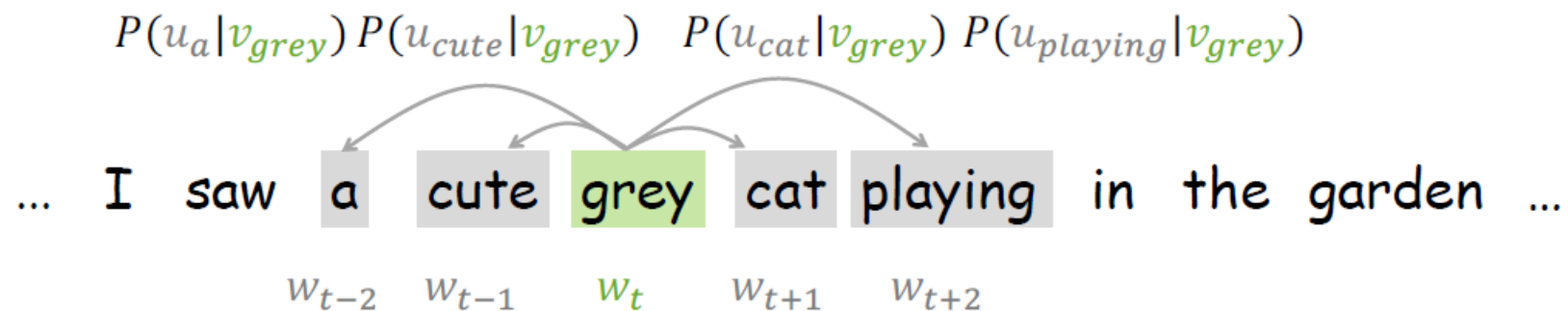
$$P(u_{\text{saw}}|v_{\text{cute}}) P(u_{\text{a}}|v_{\text{cute}}) P(u_{\text{grey}}|v_{\text{cute}}) P(u_{\text{cat}}|v_{\text{cute}})$$



Два вектора для каждого слова.

Пример

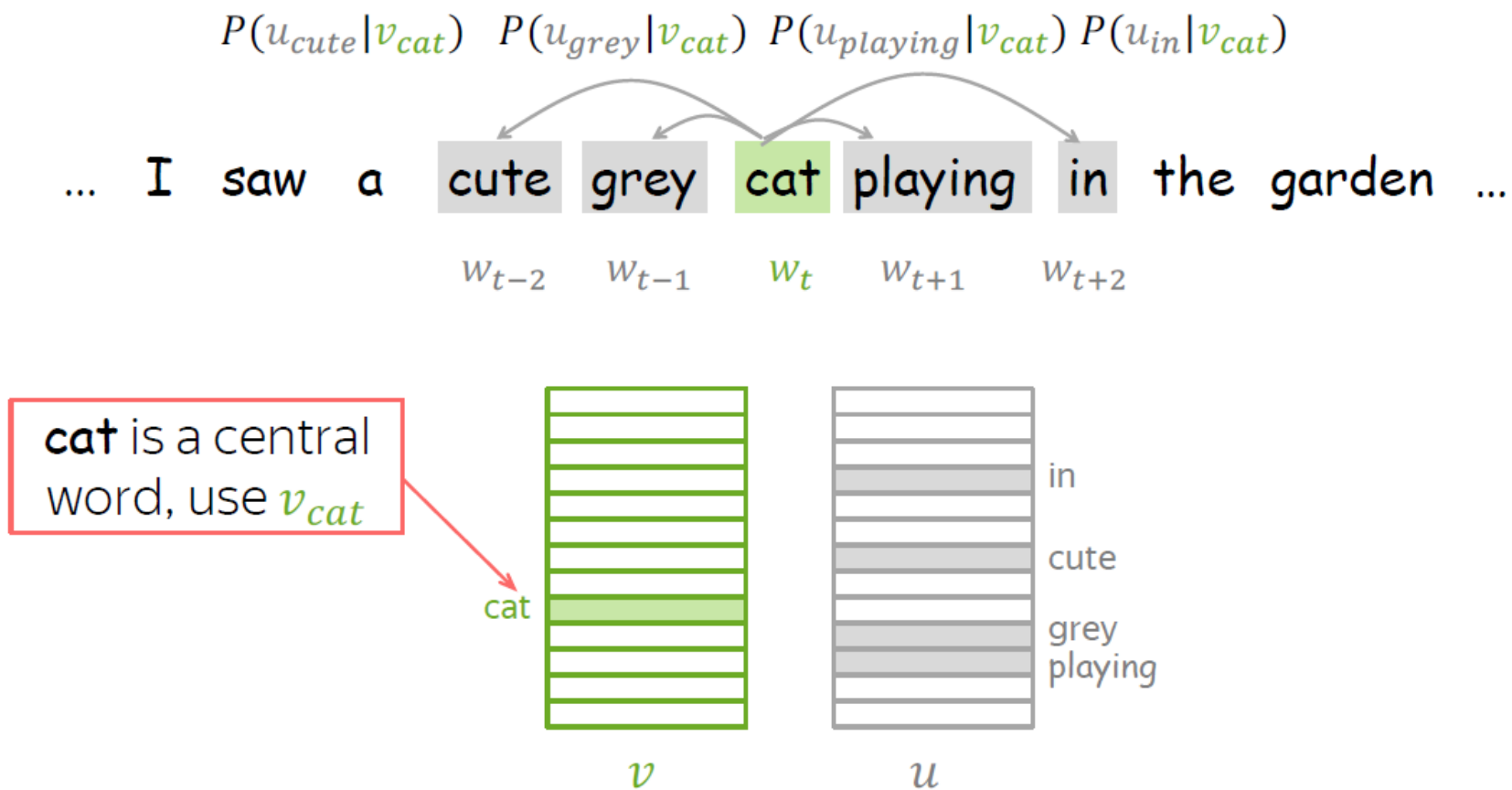
3.



Два вектора для каждого слова.

Пример

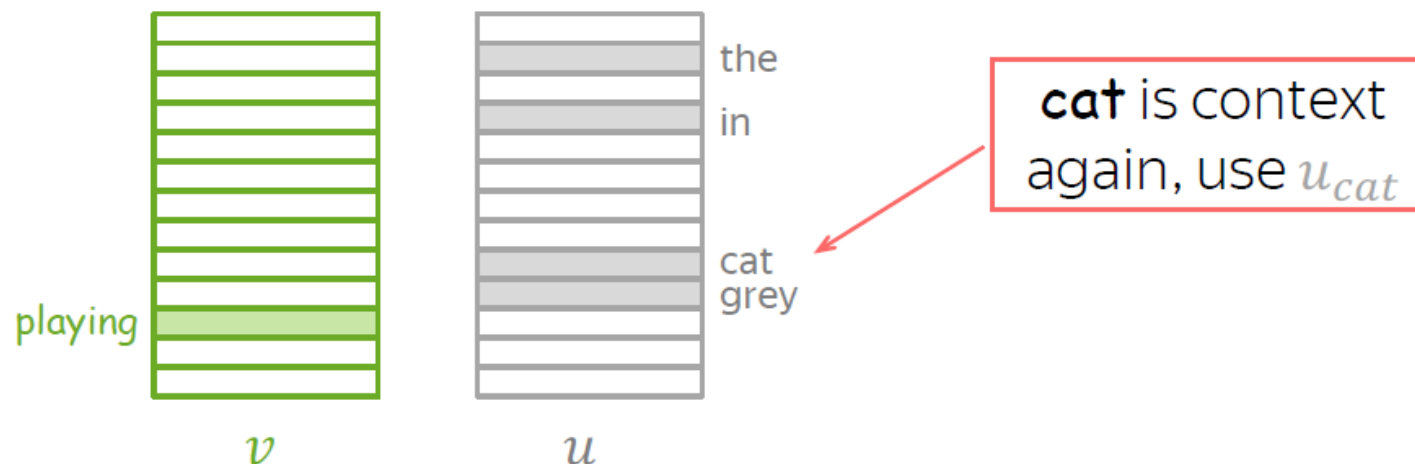
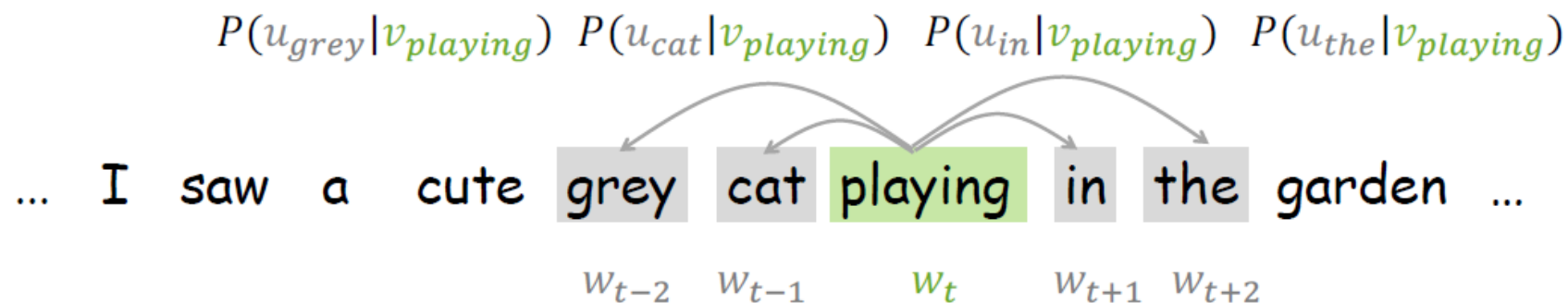
4.



Два вектора для каждого слова.

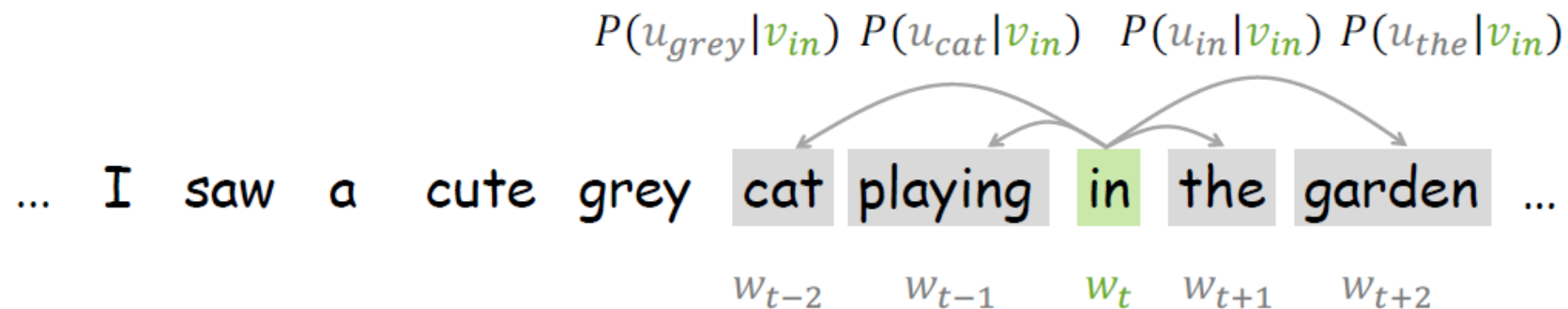
Пример

5.



Два вектора для каждого слова. Пример

6.



Процедура обучения

Есть функция потерь:

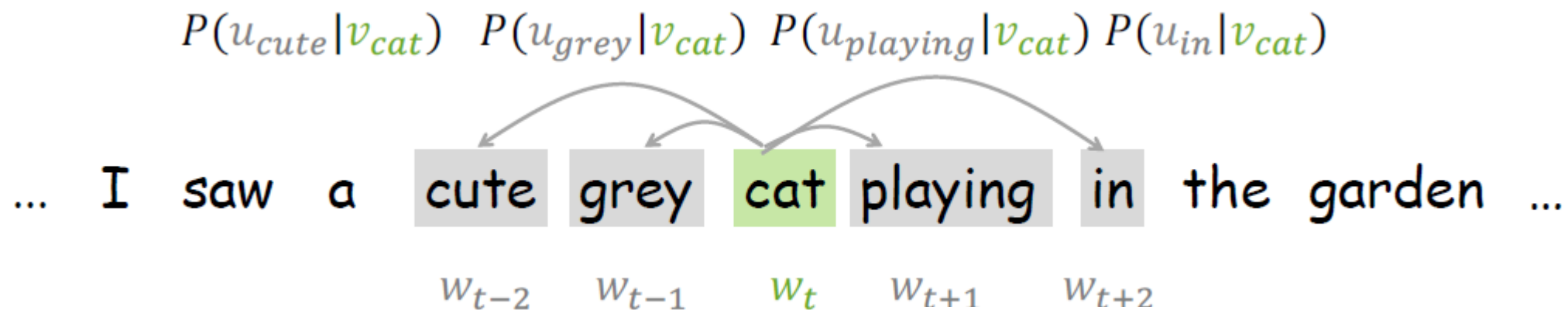
$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m, \\ j \neq 0}} \log P(w_{t+j} | w_t, \theta)$$

Перепишем в виде:

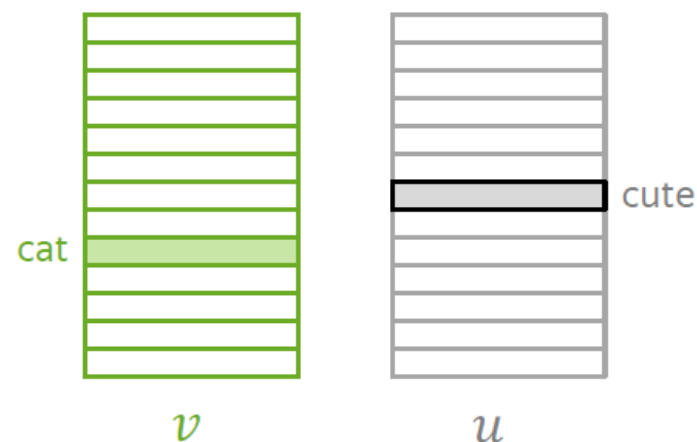
$$-\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m, \\ j \neq 0}} J_{t,j}(\theta)$$

потеря для слова j в контексте t

Процедура обучения. Пример



Выбираем одно из контекстных слов:



Процедура обучения. Пример

Рассмотрим функцию потерь для этого шага:

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

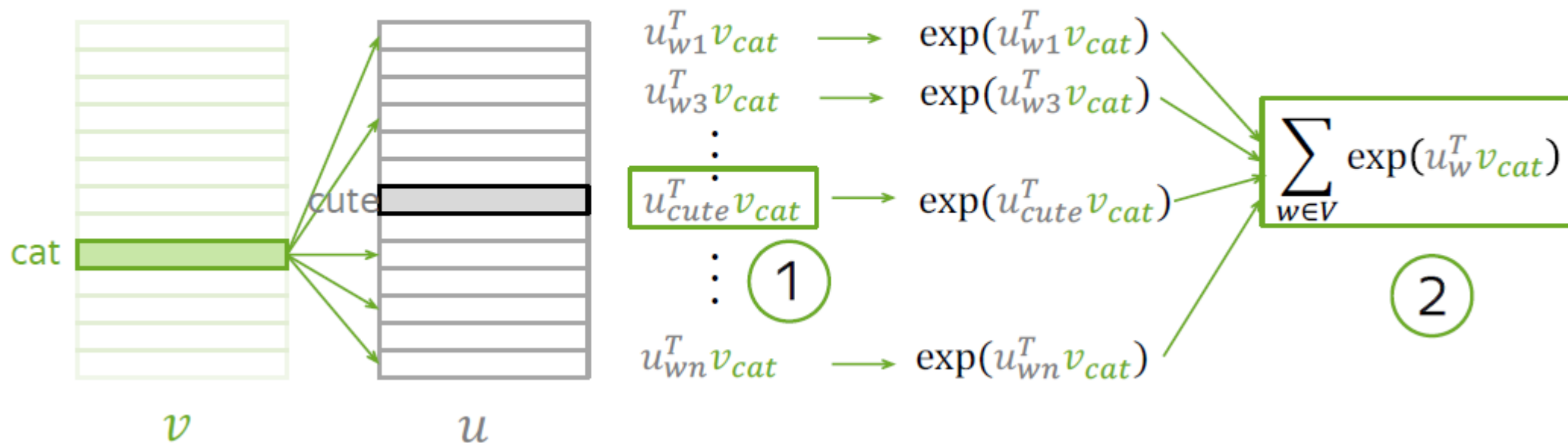
$$= -u_{\textit{cute}}^T v_{\textit{cat}} + \log \sum_{w \in V} \exp(u_w^T v_{\textit{cat}})$$

Процедура обучения. Пример

1. Take dot product of v_{cat} with all u

2. exp

3. sum all



Процедура обучения. Пример

4. get loss (for this one step)

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

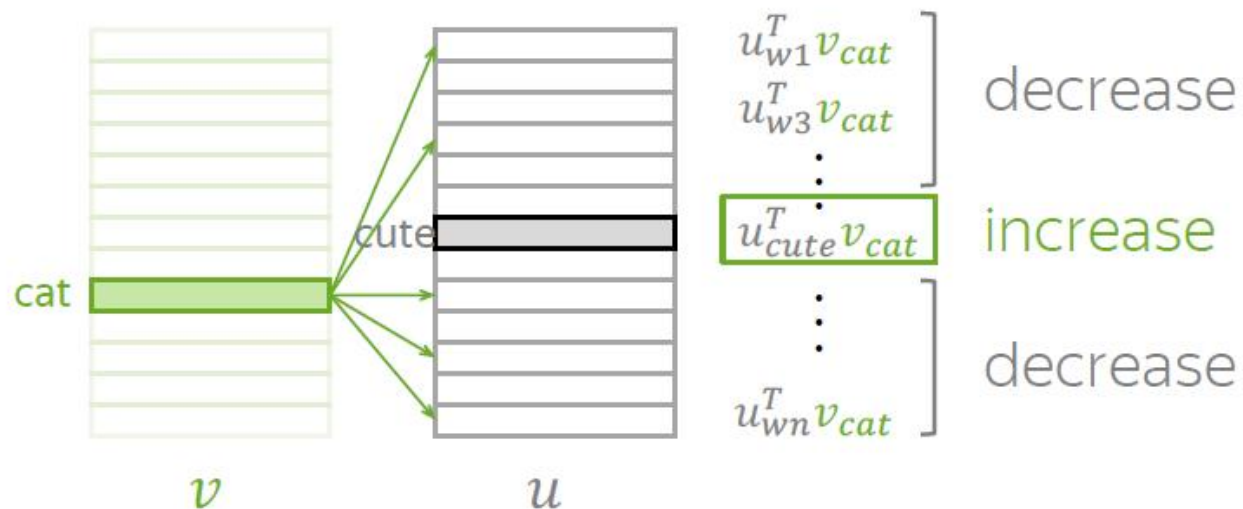
5. evaluate the gradient, make an update

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

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

Что происходит за одно обновление?

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

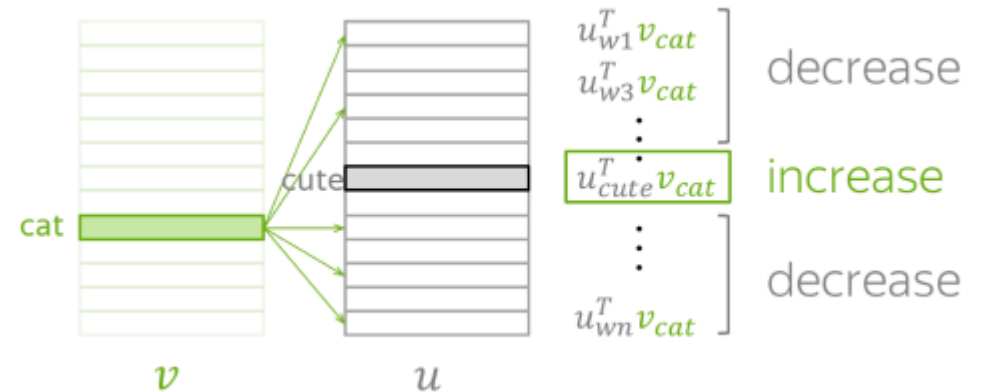


Результат каждого шага

Обычно увеличивается вероятность правильного контекста и уменьшаются остальные.

Необходимо обновить параметры:

- v_{cat}
- u_w для всех w в словаре



Проблема!

Результат. Векторное представление вектор

Например, модель Skip-gram:

Есть слово: **король**

и нужно предсказать слова вокруг него: [король] **сидит на троне**

Модель пытается максимизировать вероятность:

$P(\text{сидит} \mid \text{король})$

$P(\text{трон} \mid \text{король})$

$P(\text{на} \mid \text{король})$

То есть она уменьшает ошибку предсказания. Но важный момент:
после обучения эти вероятности почти не нужны.

Результат. Векторное представление вектор

На выходе получается **матрица векторов слов**.

Например, у нас есть словарь:

[король]
[королева]
[мужчина]
[женщина]
[яблоко]
[апельсин]

И Word2Vec выдаёт:

король → [0.21, -0.34, 0.78, 0.15]
королева → [0.19, -0.31, 0.75, 0.18]
мужчина → [0.44, 0.12, -0.22, 0.67]
женщина → [0.41, 0.15, -0.20, 0.70]
яблоко → [-0.62, 0.81, 0.12, -0.33]

Результат. Векторное представление вектор

Например:

король-королева

близко.

A: король ----- яблоко далеко.

Обычно считают косинусное расстояние:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Например: $\cos(\text{король}, \text{королева})=0.82$, $\cos(\text{король}, \text{яблоко})=0.12$

Значит: король и королева похожи; король и яблоко почти не связаны.

Результат. Векторное представление вектор

король = [0.8, 0.2, 0.5]; мужчина = [0.7, 0.1, 0.4]; женщина = [0.6, 0.1, 0.5]

Делаем: король - мужчина + женщина

Получаем: [0.7, 0.2, 0.6]

И ближайший вектор оказывается: королева

король
-
мужчина
+
женщина
=
королева

Модель сама не знает, что такое «женский род».

Она просто выучила статистические закономерности языка.

Результат. Векторное представление вектор

В процессе обучения есть слой:



Результат. Векторное представление вектор

Например:

Вход: король

Выход:

яблоко 0.01
королева 0.20
трон 0.35
сидит 0.30
машина 0.02

После обучения веса внутри нейросети становятся такими, что их можно использовать как векторы слов.

И вот эти веса:

W =
[0.21 -0.34 0.78 ...
0.19 -0.31 0.75 ...]

и становятся Word2Vec-эмбедингами.

Результат. Векторное представление вектор

На вход:

текст

Во время обучения:

предсказать соседние слова



уменьшить ошибку



изменить веса

На выход:

слово → вектор чисел

«кошка» → [0.12, -0.44, 0.87, ...]

Этот вектор потом используется для:

поиска похожих слов;
классификации текста;
кластеризации;
рекомендаций.

Negative Sampling (негативная выборка)

Вместо того чтобы учиться отличать слово от всех слов в словаре, модель учится отличать правильные слова от нескольких случайно выбранных неправильных.

Допустим, словарь содержит: 1 000 000 слов

Мы хотим обучить Skip-gram.

Есть предложение: "Король сидит на троне"

Берём слово: король

и хотим предсказать: трон

То есть модель должна сделать:

$$P(\text{трон} \mid \text{король})$$

Negative Sampling (негативная выборка)

Модель должна выдать вероятность для **каждого слова словаря**:

король	→
трон	0.35
королева	0.20
замок	0.15
яблоко	0.001
самолёт	0.0001
...	
1 000 000 слов	

Это огромная операция.

Negative Sampling (негативная выборка)

Не будем проверять миллион слов. Возьмём только несколько.

Например:

Правильная пара:

король → трон

Это положительный пример.

Теперь случайно выбираем плохие пары:

король → яблоко

король → самолёт

король → микроскоп

король → футбол

Это негативные примеры.

Negative Sampling (негативная выборка)

Получается задача:

He: «Какова вероятность каждого слова из миллиона?»

A: «Это слово подходит к контексту или нет?»

Пара	Ответ
король — трон	1
король — замок	1
король — яблоко	0
король — футбол	0
король — микроскоп	0

Word2Vec. Skip-Gram

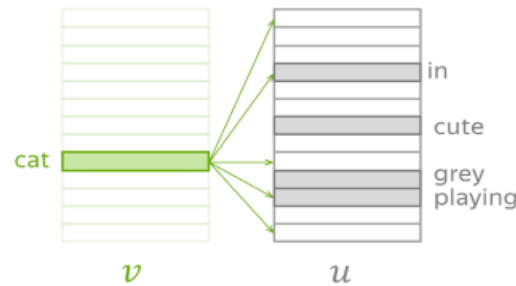
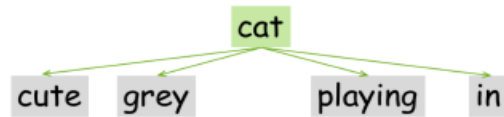
Главная гипотеза: Слова, которые встречаются в сходных контекстах, имеют сходные значения.

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)

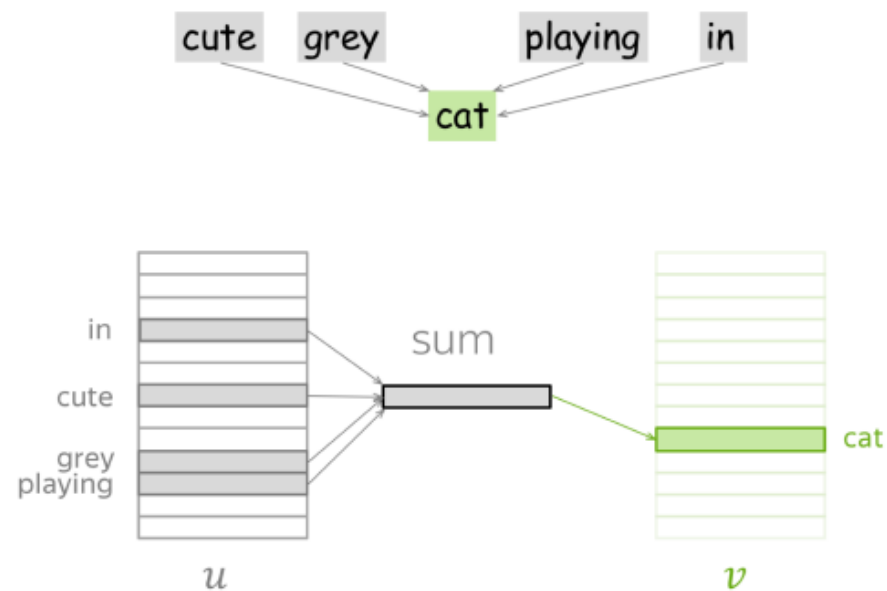
Word2Vec. CBOW

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 Negative Sampling;
- Количество отрицательных примеров: для небольших наборов данных 15–20; для больших наборов данных (которые обычно используются) может быть 2–5.
- Embedding size: часто используемое значение — 300, но возможны и другие варианты (например, 100 или 50).
- Размер скользящего окна (контекста): 5–10.

Стандартные гиперпараметры

- Скорость обучения (learning rate): 0.025 (классическое значение из оригинального Word2Vec)
- Количество эпох (сколько раз пройти по всему тексту): 5-10 эпох
- Минимальная частота слова (min_count): 5 (слова, которые встретились меньше 5 раз, убрать)

Типичные настройки Word2Vec. Маленький корпус

- model = Skip-Gram
- vector_size = 100
- window = 5
- negative = 15
- epochs = 10
- min_count = 3
- learning_rate = 0.025

Типичные настройки Word2Vec. Большой корпус

- model = Skip-Gram
- vector_size = 300
- window = 5
- negative = 5
- epochs = 5
- min_count = 5
- learning_rate = 0.025

GloVe (Global Vectors for Word Representation)

Это другая классическая модель получения **векторных представлений слов (word embeddings)**, похожая по назначению на Word2Vec, но с другой философией обучения.

Главная идея:

Значение слова можно понять по тому, с какими словами оно и как часто встречается рядом во всём корпусе.

GloVe (Global Vectors for Word Representation)

GloVe строит **матрицу совместных появлений слов**.

Например, есть текст: король сидит на троне

Берём окно контекста = 2 (заданное количество слов вокруг целевого слова)

Для слова: король

соседи: сидит; на

GloVe (Global Vectors for Word Representation)

Матрица совместной встречаемости (co-occurrence matrix).

Слово	Контекст	Сколько раз
король	сидит	100
король	трон	80
король	замок	50
королева	сидит	90
королева	трон	70
королева	замок	40

GloVe (Global Vectors for Word Representation)

Word2Vec:

Идея: «Я вижу слово. Попробую угадать его окружение.»

Вход: король

Задача: предсказать: трон; замок; корона

То есть: слово → контекст

Word2Vec смотрит на локальные окна:

король сидит на троне



маленькое окно

GloVe (Global Vectors for Word Representation)

GloVe:

Идея: «Посмотрим на всю статистику сразу»

Например: король часто встречается с: трон; корона; замок; править

А: яблоко часто встречается с: есть; сок; фрукт; дерево

GloVe учитывает глобальную структуру языка:



GloVe (Global Vectors for Word Representation)

Есть матрица:

	трон	яблоко	машина
король	100	2	1
королева	80	3	1
яблоко	1	150	5

GloVe пытается найти такие векторы слов, чтобы их произведение объясняло эти числа.

То есть: $\text{вектор(король)} \times \text{вектор(трон)} \approx \text{частота встречи}$

GloVe (Global Vectors for Word Representation)

Авторы GloVe заметили интересную вещь.

Не просто: «какие слова встречаются рядом», а: «как отличаются вероятности появления рядом».

Есть слова: лед; пар

и контекст: твёрдый; газообразный

	твёрдый	газ
лёд	0.8	0.01
пар	0.01	0.7

GloVe (Global Vectors for Word Representation)

Теперь смотрим не на отдельные вероятности, а на отношение.

Например:

$$\frac{P(\text{твёрдый} \mid \text{лёд})}{P(\text{твёрдый} \mid \text{пар})} = \frac{0.8}{0.01} = 80$$

Это означает:

слово «твёрдый» гораздо сильнее связано со льдом, чем с паром.

Теперь:

$$\frac{P(\text{газообразный} \mid \text{лёд})}{P(\text{газообразный} \mid \text{пар})} = \frac{0.01}{0.7} = 0.014$$

То есть:

слово «газообразный» связано с паром, но не со льдом.

GloVe (Global Vectors for Word Representation). Гиперпараметры

Параметр	Типичные значения
размерность вектора	50, 100, 200, 300
размер окна	5–10
количество итераций (сколько раз алгоритм проходит через всю матрицу совместных появлений)	25–100
min_count	5–10
learning rate	0.05

Word2Vec vs GloVe

	Word2Vec	GloVe
Подход	предсказание	статистика
Основа	локальный контекст	глобальная статистика
Обучение	нейросеть	факторизация матрицы (большая матрица → разложение → векторы слов)
Данные	поток текста	матрица слов
Скорость	быстрее	требует памяти
Аналогии	хорошо	хорошо
Семантика	хорошо	хорошо

Word2Vec спрашивает: «какие слова окружают это слово?»

GloVe спрашивает: «какая глобальная статистика связей всех слов в языке?»

GloVe (Global Vectors for Word Representation)

Когда лучше применять GloVe:

- При наличии **большого и стабильного корпуса текстов**.
- Когда важна **глобальная статистика связей между словами**.
- Для задач, где достаточно **общего значения слова**, без учёта конкретного контекста.
- Для **поиска похожих слов и терминов**.
- Для **классификации текстов, кластеризации, анализа тональности**.
- Когда нужны **готовые качественные эмбединги** (например, GloVe 100d/300d).
- Когда модель не нужно часто переобучать и обновлять.

Word2Vec выбирают, когда нужна быстрая, лёгкая и хорошо масштабируемая модель эмбедингов слов, а задача требует понимания связей между словами, но не глубокого понимания контекста.

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

<u>word pair</u>		<u>score</u>
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

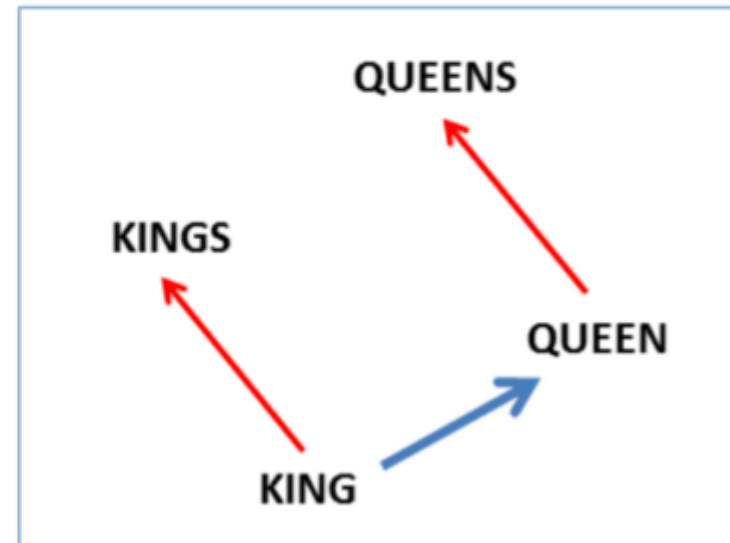
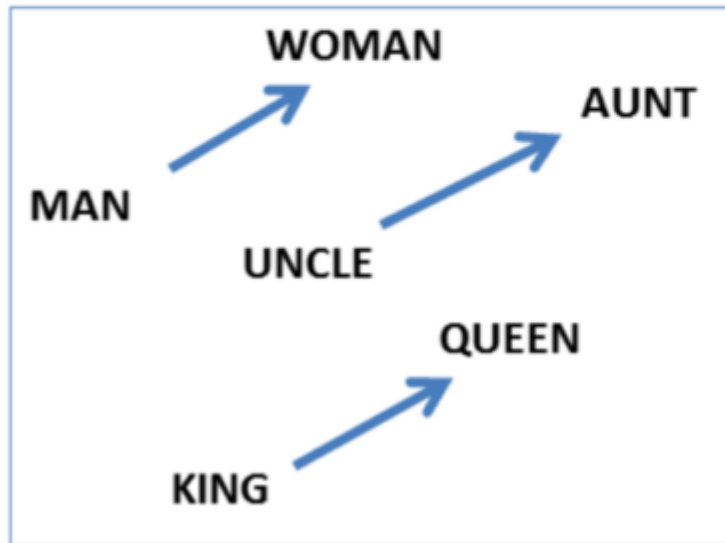
Многие семантические и синтаксические отношения между словами (почти) линейны в пространстве вложений.

Это означает, что сложные семантические или синтаксические связи между словами можно выразить с помощью простых **векторных арифметических операций**, в основном сложения и вычитания.

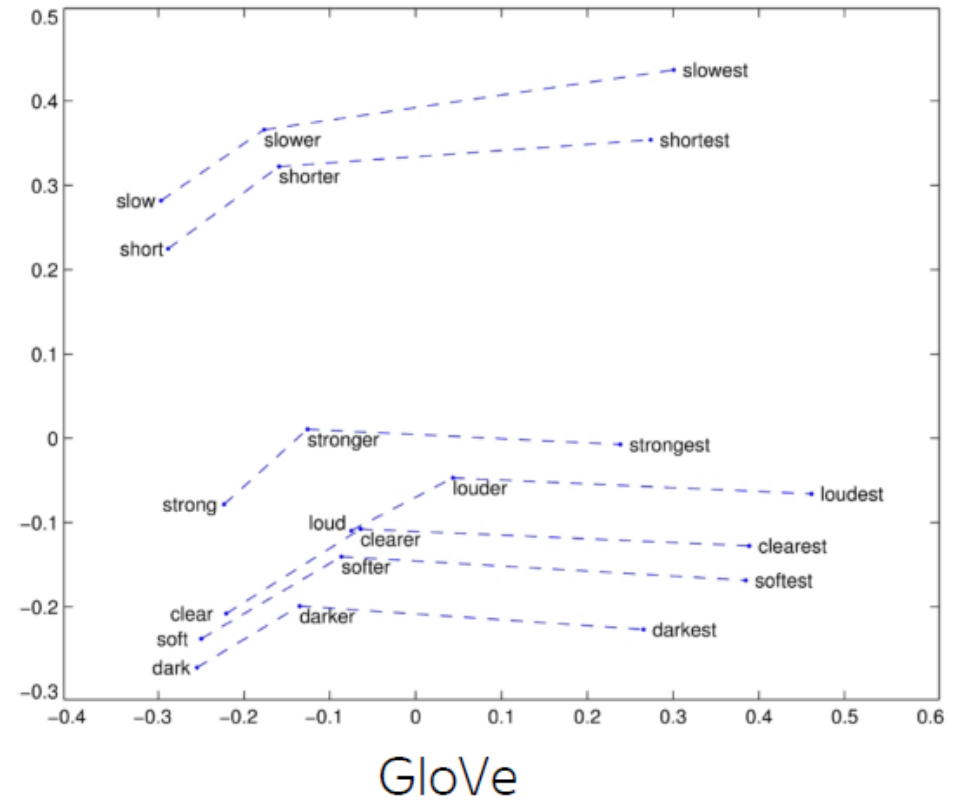
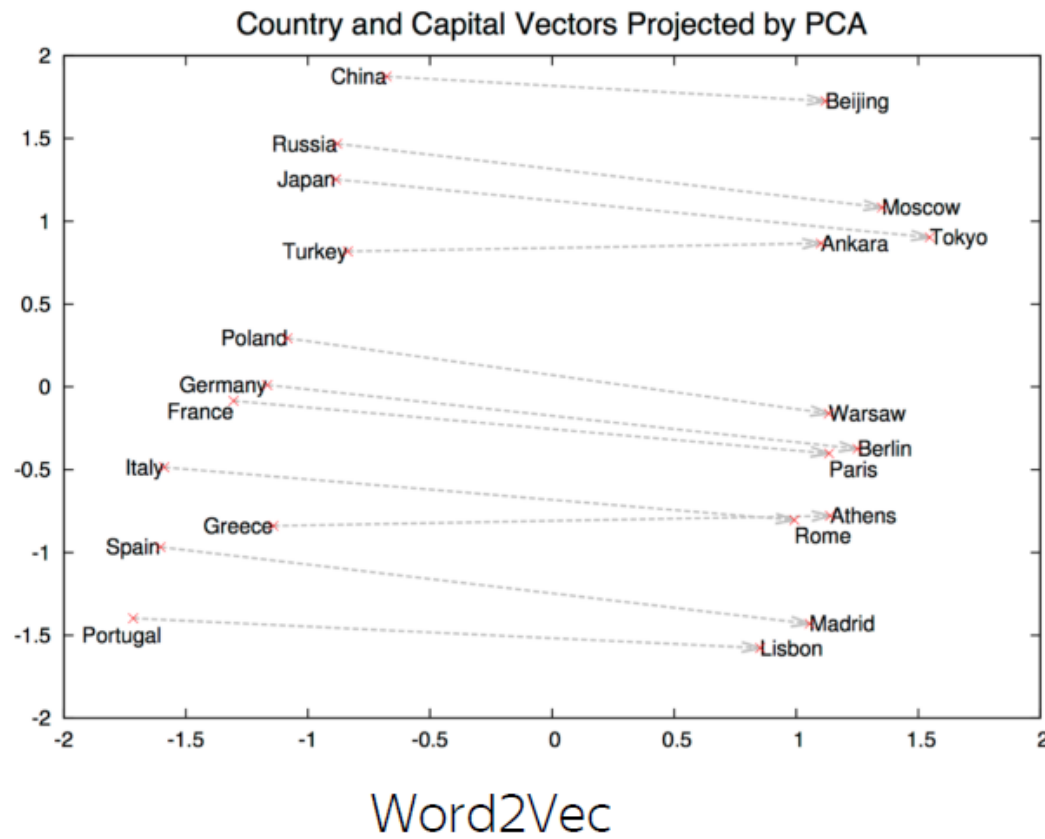
Линейная структура

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



Сходства в разных языках

Пример создания больших словарей из маленьких

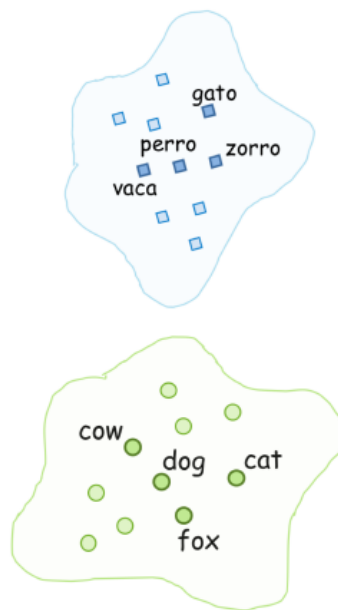
Ingredients:

- 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
...

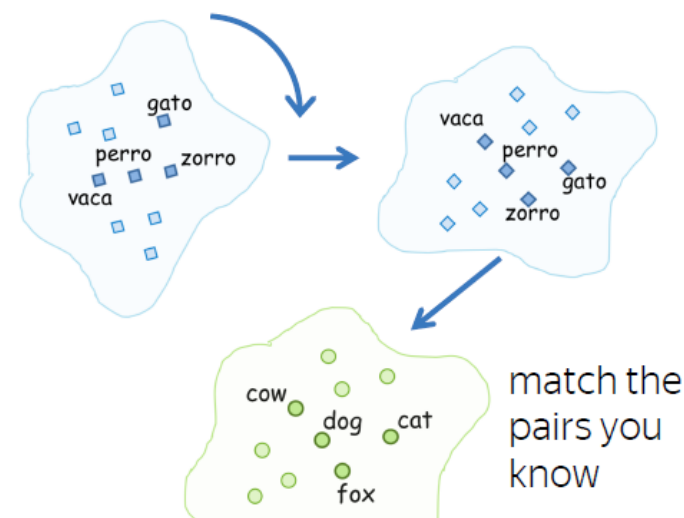
Step 1:

- train embeddings for each language



Step 2:

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



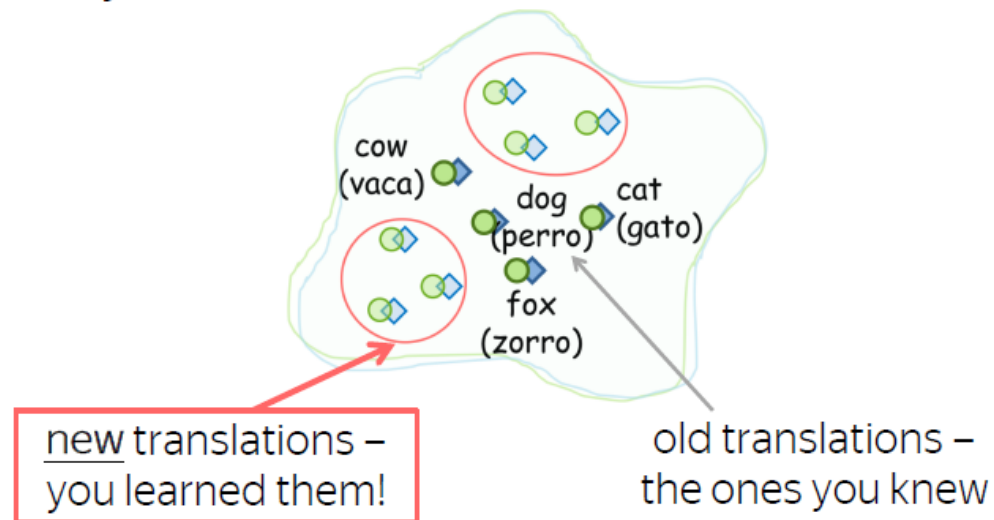
Сходства в разных языках

Steps 1-2:

- match words from the vocabulary

Step 3:

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



Задание 2 (2 балла)

1. Почему после обучения Word2Vec нас интересуют именно веса нейросети (матрица W), а не вероятности предсказания слов?
2. Почему Negative Sampling делает обучение Word2Vec значительно быстрее, но сохраняет качество получаемых векторов?
3. В чём принципиальное отличие Word2Vec и GloVe: почему Word2Vec называют предсказательной моделью, а GloVe — статистической?
4. Почему операция "король – мужчина + женщина $\approx$ королева" возможна, хотя модель не знает понятий "мужской род" и "женский род"?