

# Немного про нейросети или «Как научить компьютер читать?»

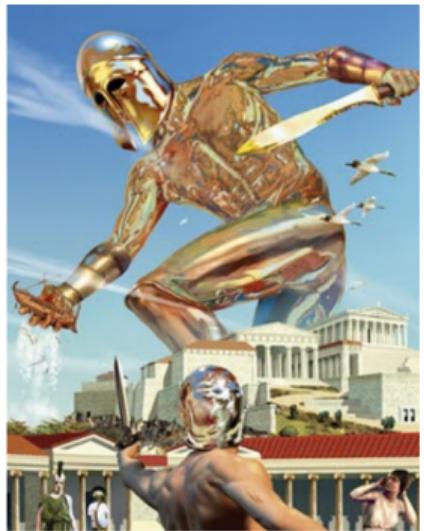
Ульянкин Филипп

28 ноября 2018 г.

# Agenda

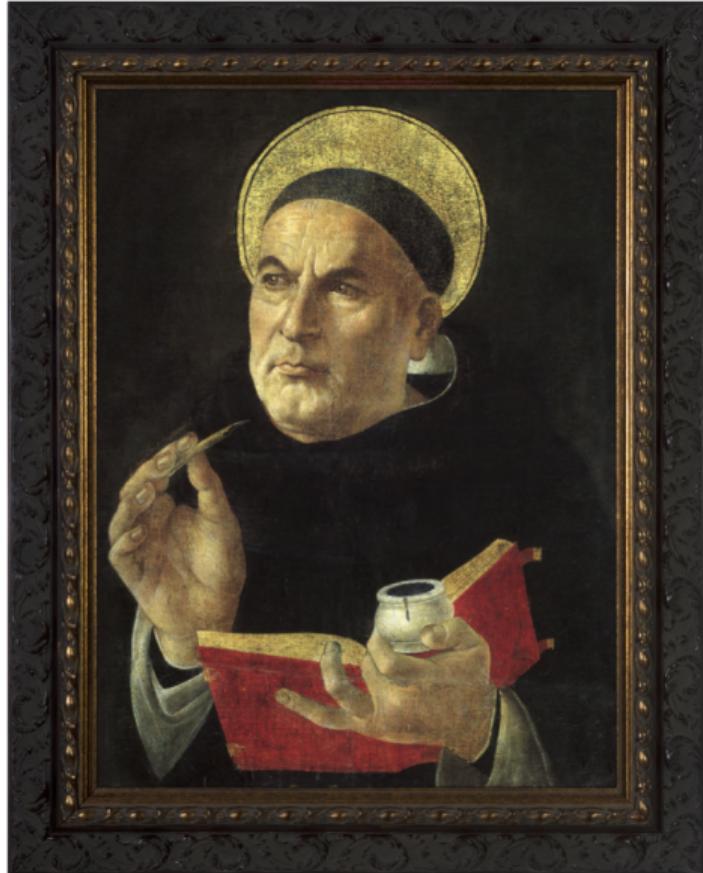
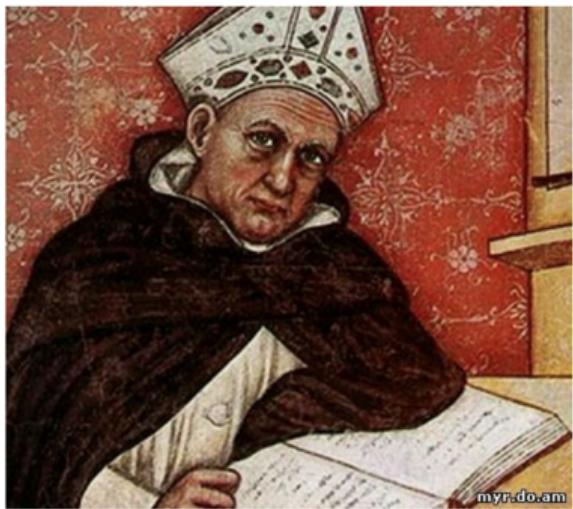
- Немного истории
- Как работает нейросеть
- Как заставить сетку искать смысл
- Учим word2vec на википедии
- Применяем word2vec для сентимент-анализа

# История нейронных сетей











*Fig. 1.*



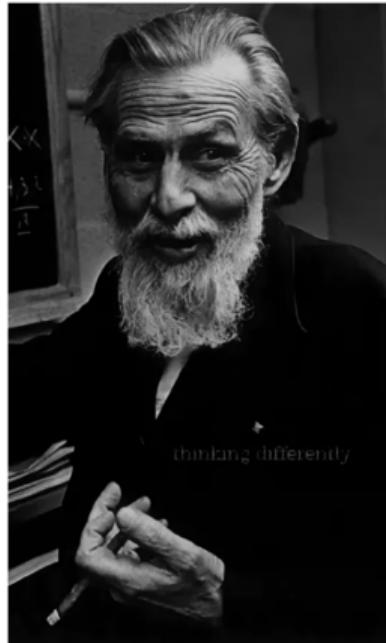
*Fig. 2.*



# Первый формальный нейрон, 1943 год



Уоррен Маккуллок



Уолтер Питтс

# Первый искусственная нейросеть, 1958 год



Фрэнк Розенблатт





Марвин Минский

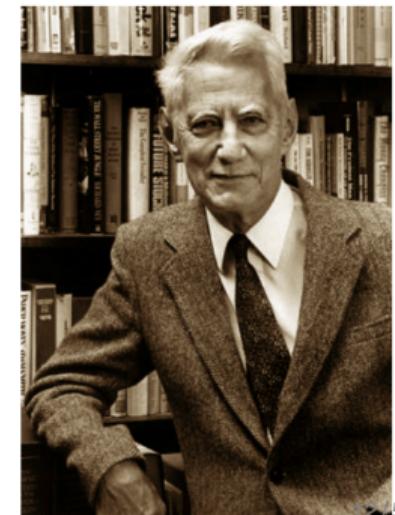
Джон Маккарти



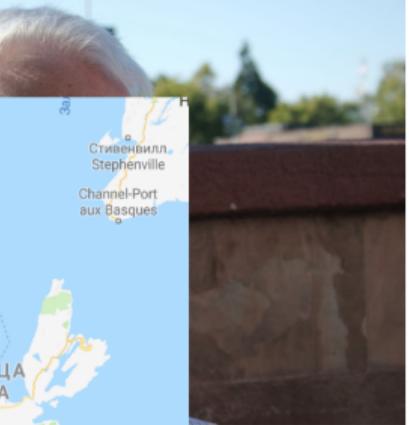
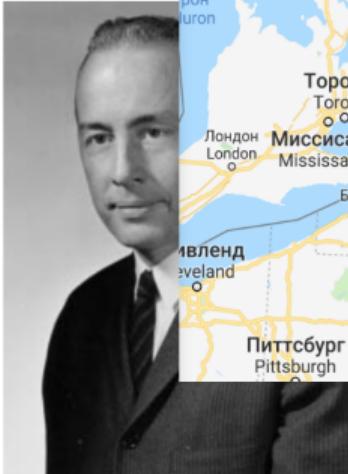
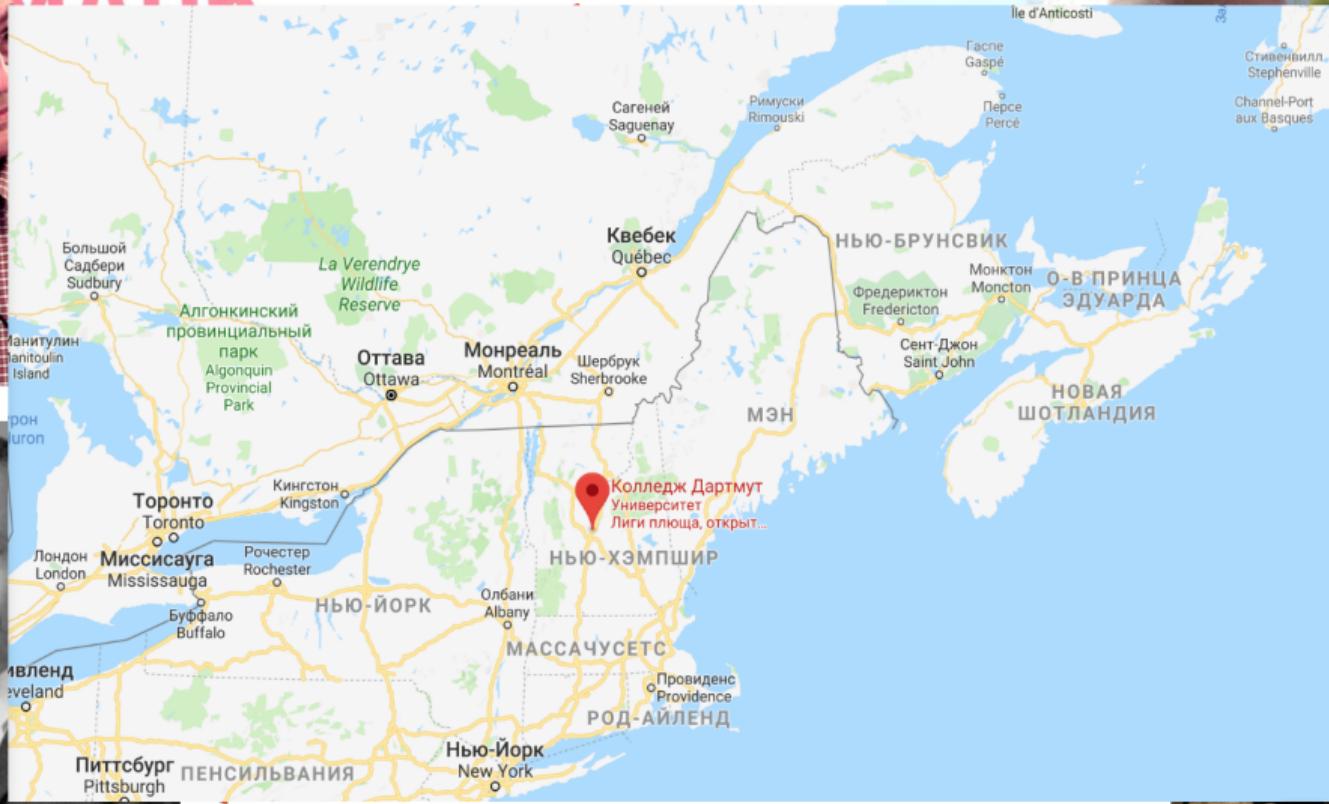
1956 год

Клод Шеннон

Натаниэль Рочестер



# Марвин Минский



# Зима близко

- 1956 – Дартмунтский семинар, море оптимизма
- 1958 – Перцептрон Розенблатта
- середина 1960-х – провал крупного проекта по машинному переводу с русского на английский и наоборот
- 1969 – Марвин Минский и Сеймур Пейперт опубликовали книгу «Перцептроны» с критикой



memegenerator.net

# Оттепель

- 1970-е — Расцвет экспертных систем, принимающих решения на основе большого числа правил и знаний о предметной области
- MYCIN накопила около 600 правил для идентификации вирусных бактерий и выдачи подходящего метода лечения (угадывала в 69% случаев, лучше любого начинающего врача)
- 1980-е — появилось много разных архитектур
- 1980-е — backpropagation позволил обучать сети за линейное время
- Ренессанс нейронных сетей

# Зима близко

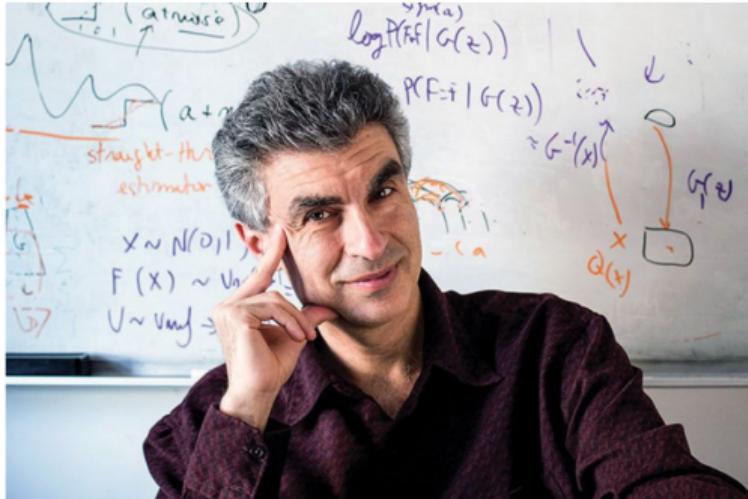
- Новая волна оптимизма
- 1986 — один из первых AI-отделов экономил компании DEC около 10 миллионов долларов в год
- Завышенные ожидания снова лопнули
- 1990-е — ударными темпами развивается классическое машинное обучение



# 2005-2006: начало революции



Джеффри Хинтон  
университет Торонто



Йошуа Бенджи  
университет Монреяля

# Революция

- 2005-2006 — группы Хинтона и Бенджи научились обучать глубокие нейросетки
- Накопилось больше данных! Огромные данные!
- Компьютеры стали на порядки мощнее! Появились крутые GPU!
- На больших данных и мощностях заработали старые архитектуры
- Появились новые алгоритмы, эвристики и подходы
- Ящик Пандоры открыт!

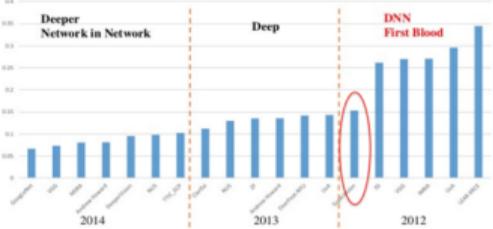
# ImageNet



AlphaGo 4:1



ImageNet Classification error throughout years and groups



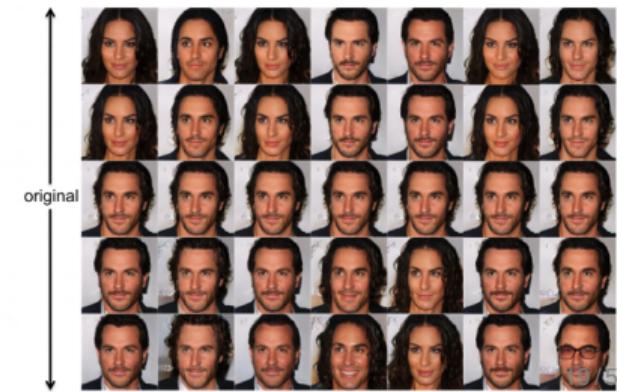
Пабло Пикассо



Винсент Ван Гог

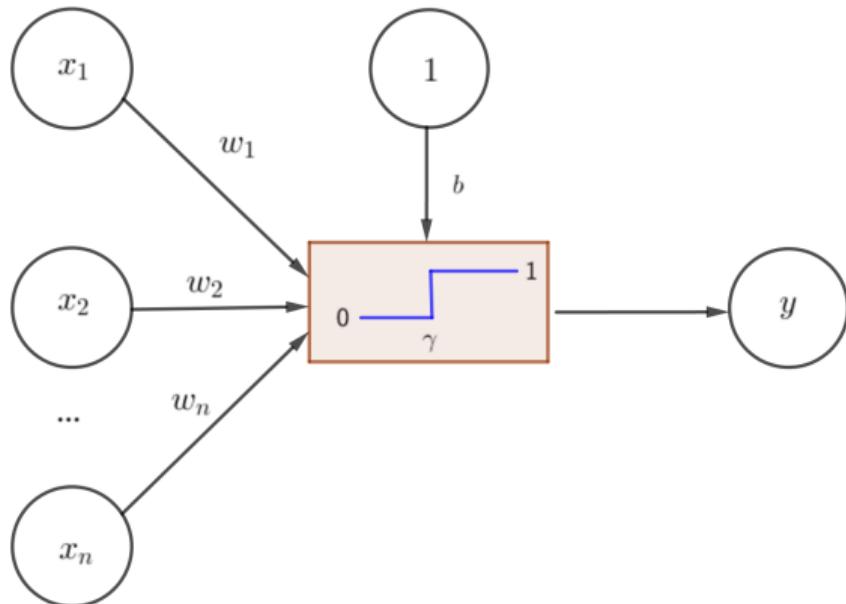


Василий Кандинский



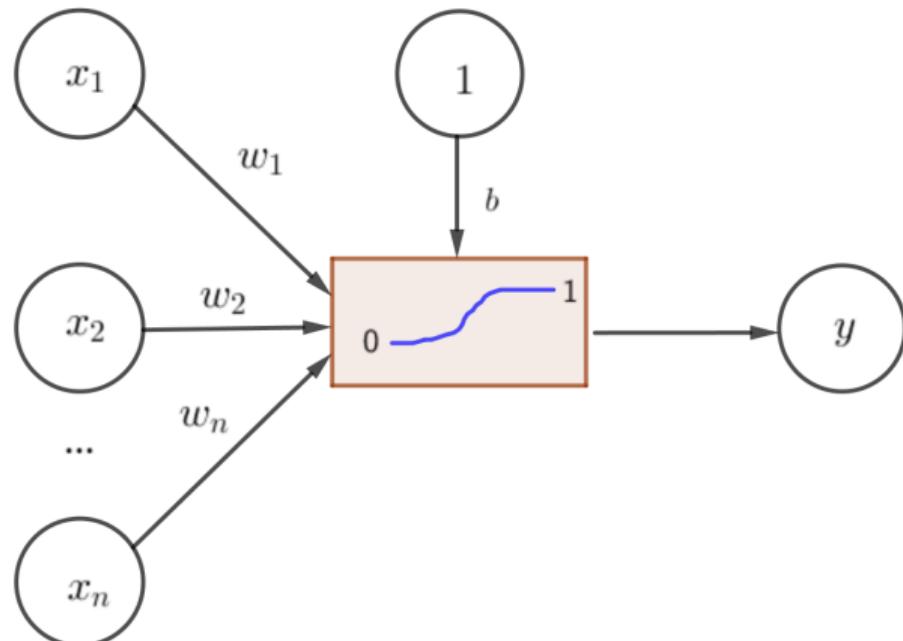
У нас есть армия.  
А у тебя нет даже Халка.

# Перцептрон Розенблатта

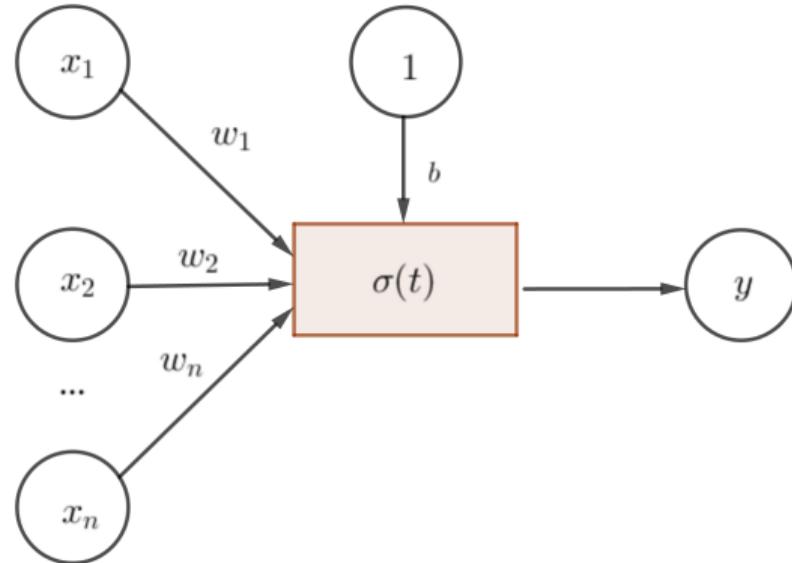


$$y = \begin{cases} 1, & \text{если } \sum w_i x_i \geq \gamma \\ 0, & \text{если } \sum w_i x_i < \gamma \end{cases}$$

# Немного иной перцептрон Розенблатта



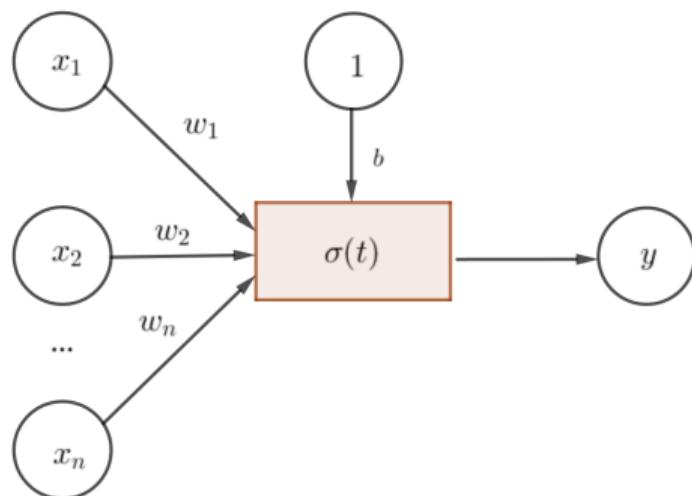
# Функция активации



- Функция активации  $\sigma(t)$  вносит нелинейность, она может быть любой

# Линейная регрессия

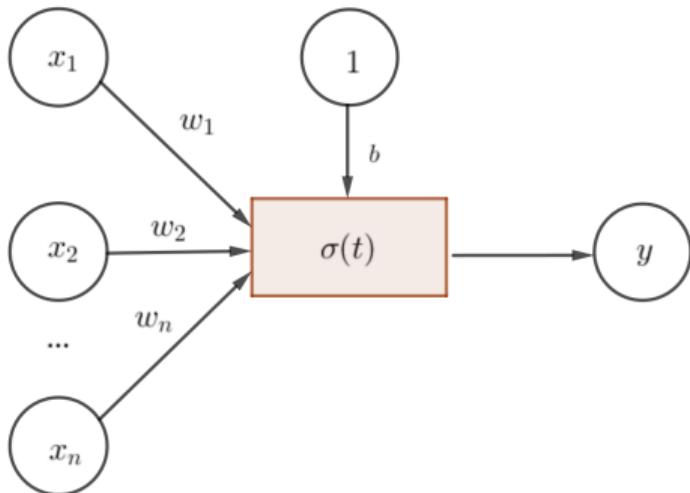
Нейрон с линейной функции  
активации — это линейная регрессия...



$$\sigma(t) = t$$

$$y = b + w_1x_1 + \dots + w_nx_n$$

# Логистическая регрессия

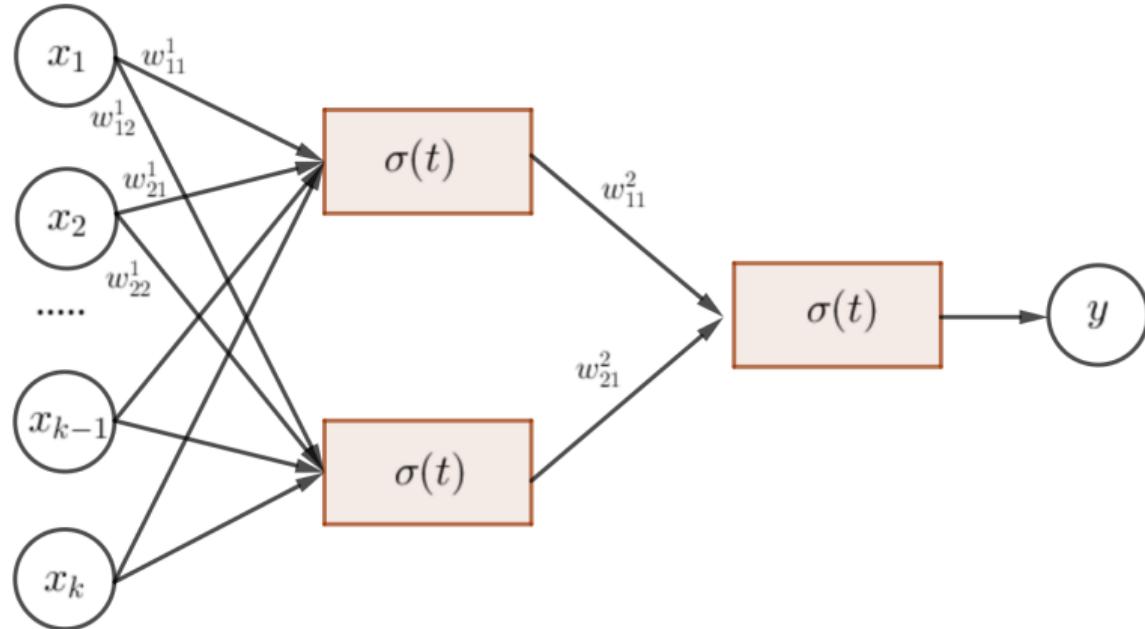


Нейрон с сигмоидом в качестве функции активации — это логистическая регрессия...

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

$$P(y = 1 | x) = \sigma(b + w_1x_1 + \dots + w_nx_n)$$

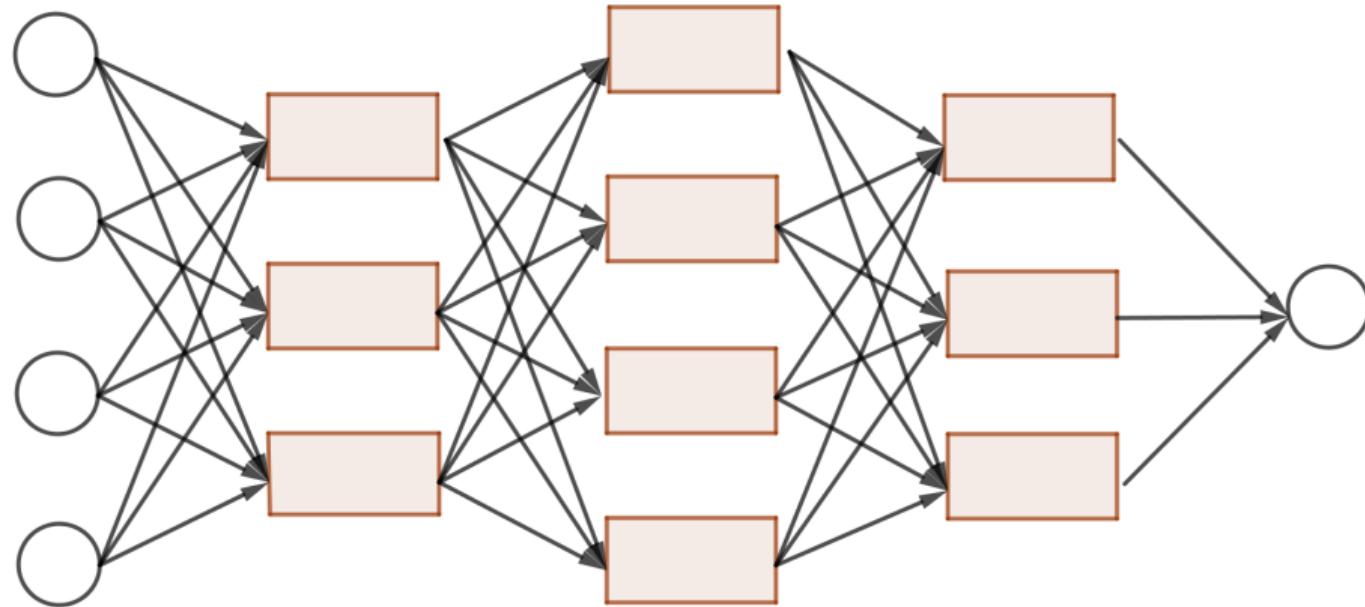
# Две регрессии скрепили третьей



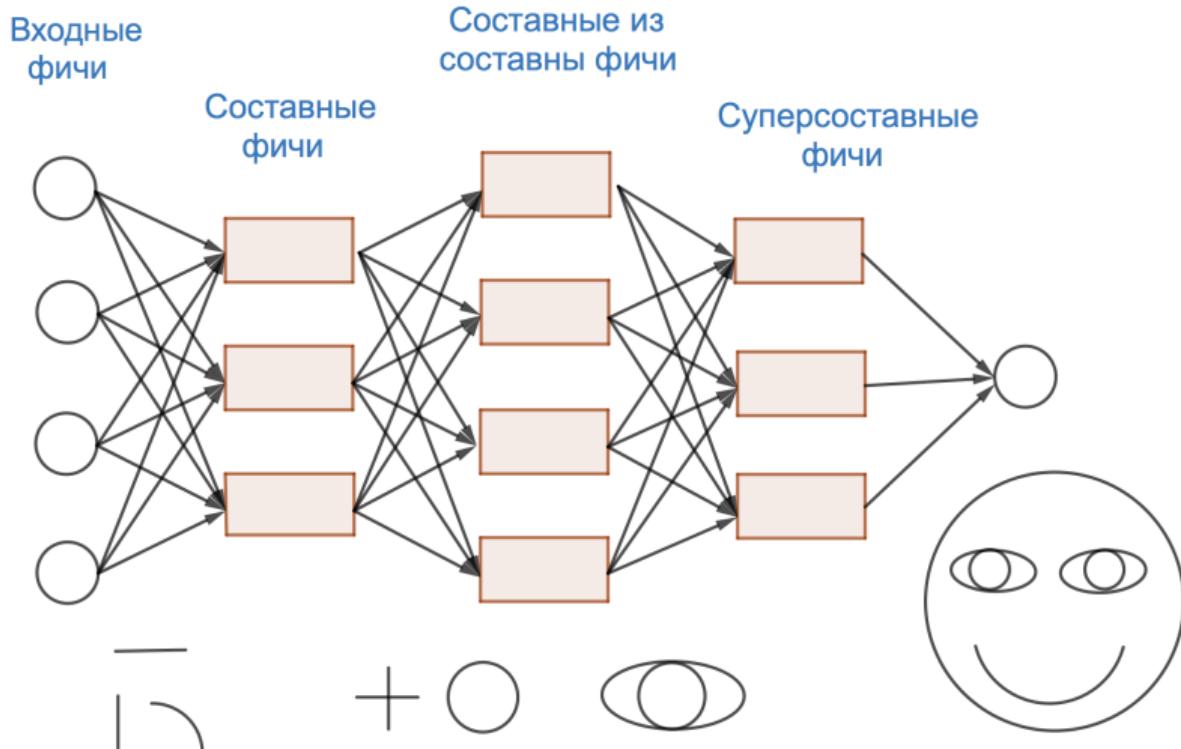
$$y = \sigma(w_{11}^2 \cdot h_1 + w_{21}^2 \cdot h_2)$$

$$h_j = \sigma\left(\sum_i w_{ij}\right)$$

# Армия из регрессий

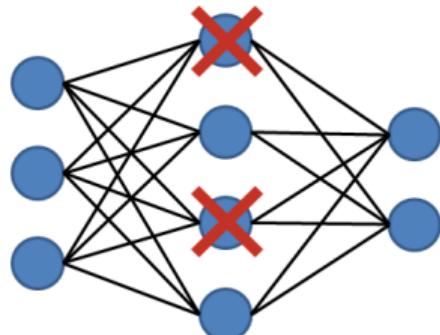


# Армия из регрессий



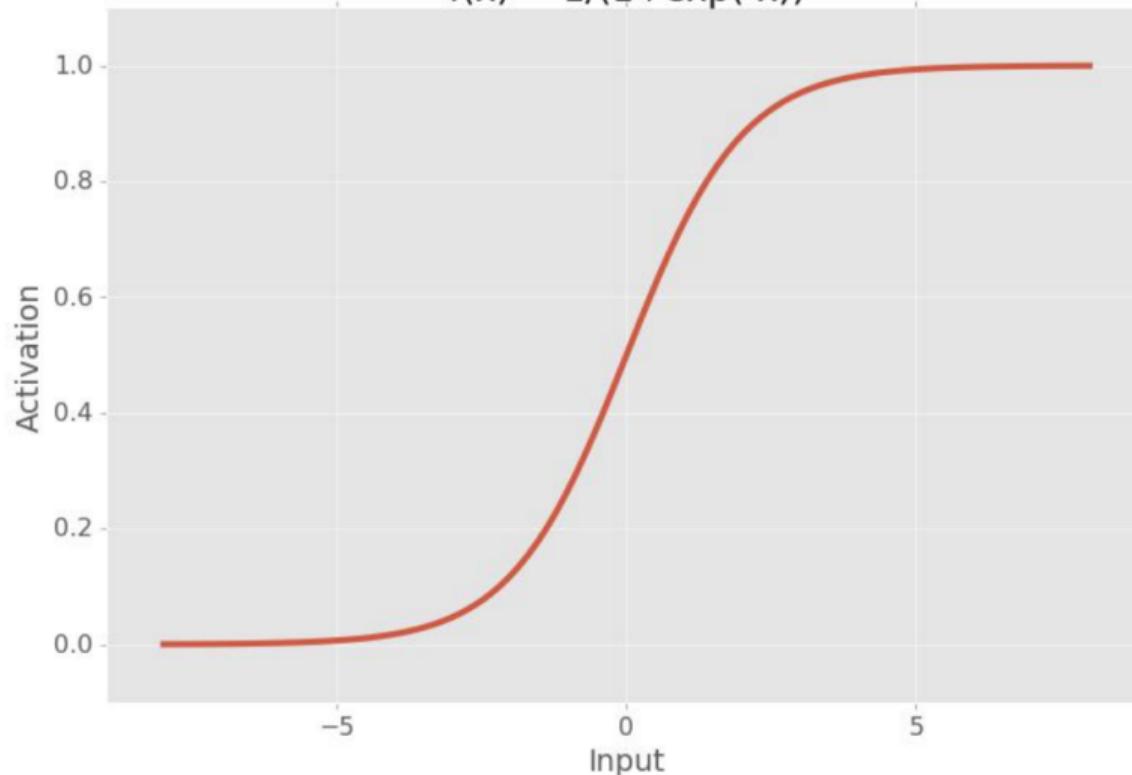
## Слои бывают разными

- Слой, который просто взвешивает входы называется **полносвязным**.
- Слои бывают очень разными. Например, **Dropout**: с вероятностью  $p$  отключаем нейрон. Такой слой препятствует переобучению и делает нейроны более устойчивыми к случайным возмущениям.

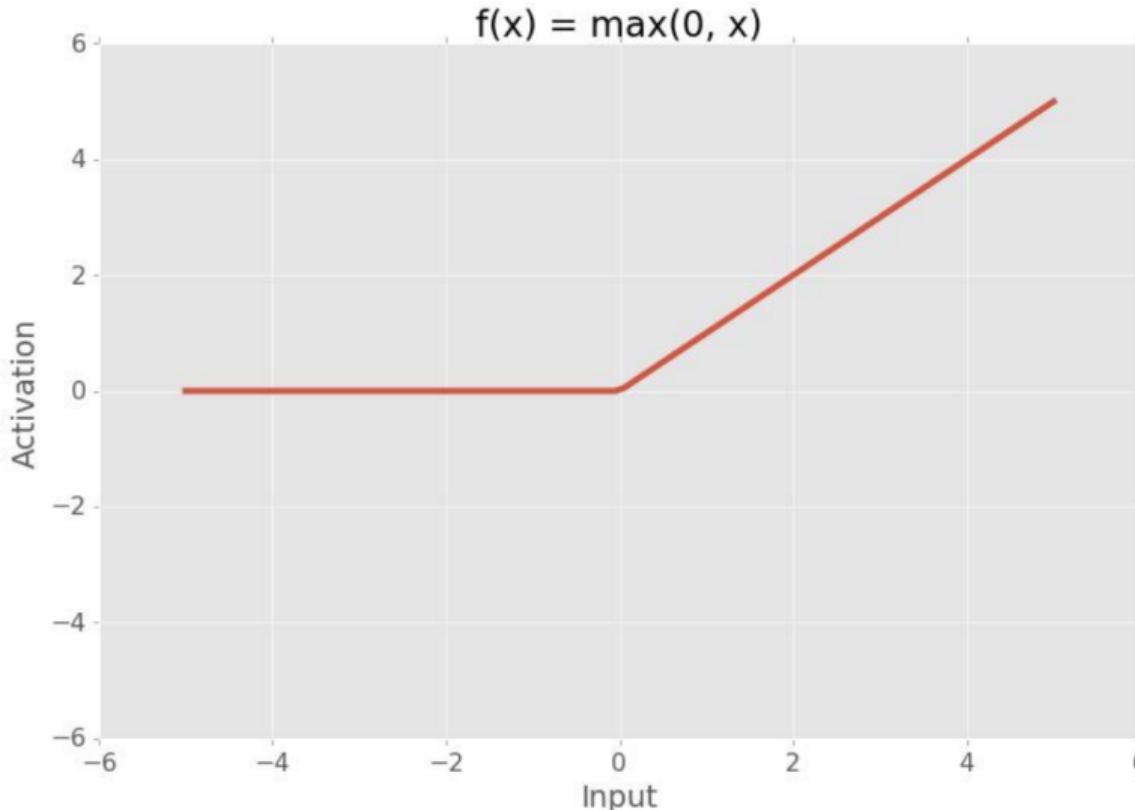


## Функции активации бывают разными (сигмоид)

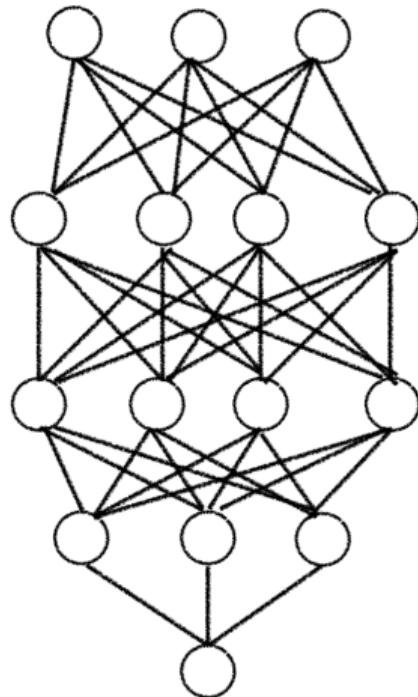
$$f(x) = 1/(1+\exp(-x))$$



## Функции активации бывают разными (ReLU)



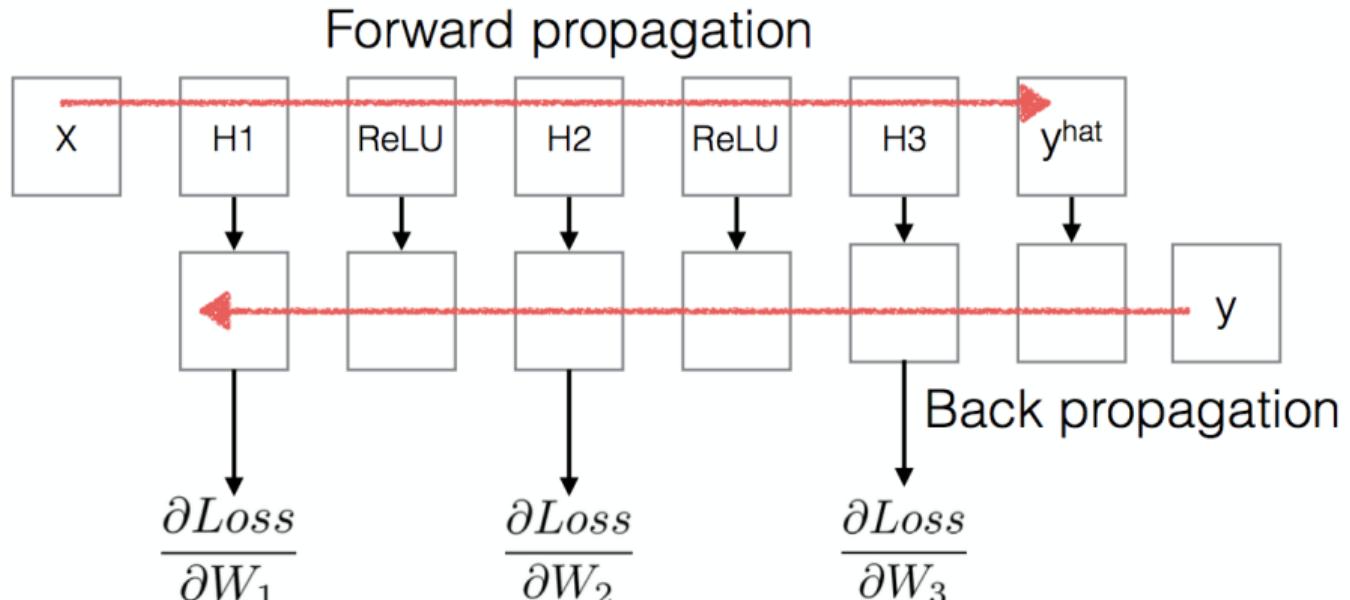
# Архитектуры бывают разными



Input	
Fully connected layer (FC)	$XW + b$
ReLU	$\max(0, x)$
Dropout	$Bern(p)$
FC	$XW + b$
ReLU	$\max(0, x)$
FC	$XW + b$
Output	

Каждый слой — просто функция, каждая сетка — конструктор LEGO

# Обучение сетки



Сетки обучают также как и обычную регрессию — градиентным спуском.  
Алгоритм, который позволяет делать это оптимально — **алгоритм обратного распространения ошибки (backpropagation)**.

Из слов в вектора и обратно

# Весь анализ текстов в одном слайде!



Порядок слов неважен  
Неважен слов порядок  
Слов порядок неважен

Он был хорошим человеком и джедаем.  
Люди держат деньги в банке.  
Падаван, дай мне огурец из банки.



1. токенизация
2. очистка от стоп-слов

[~~он~~, был, хорошим, человеком, ~~и~~, джедаем]  
[люди, держат, деньги, ~~в~~, банке]  
[падаван, дай, мне, огурец, ~~из~~, банки]

лемматизация

[быть, хороший, человек, джедай]  
[человек, держать, деньги, банк]  
[падаван, дать, огурец, банка]

3. нормализация

стемминг

[бы, хорош, чел, джед]  
[люд, держ, ден, банк]  
[падаван, дай, огур, банк]

4. очистка от слишком редких слов

5. ОНЕ или tf-idf,  
а затем  
моделирование

## Ключевые мысли предыдущего слайда

- **Гипотеза мешка слов:** нам плевать на взаимное расположение слов. Порядок слов в предложении никак не сказывается на его смысле.  
**Следуя гипотезе, мы теряем часть информации.**
- Рассматриваем каждое слово, как переменную  $\Rightarrow$  большое пространство признаков. Нужно его урезать  $\Rightarrow$  **Теряем ещё информацию.**
- Хотим маленькое пространство признаков и много информации в нём!
- **Дистрибутивная гипотеза:** слова с похожим смыслом будут встречаться в похожих контекстах.

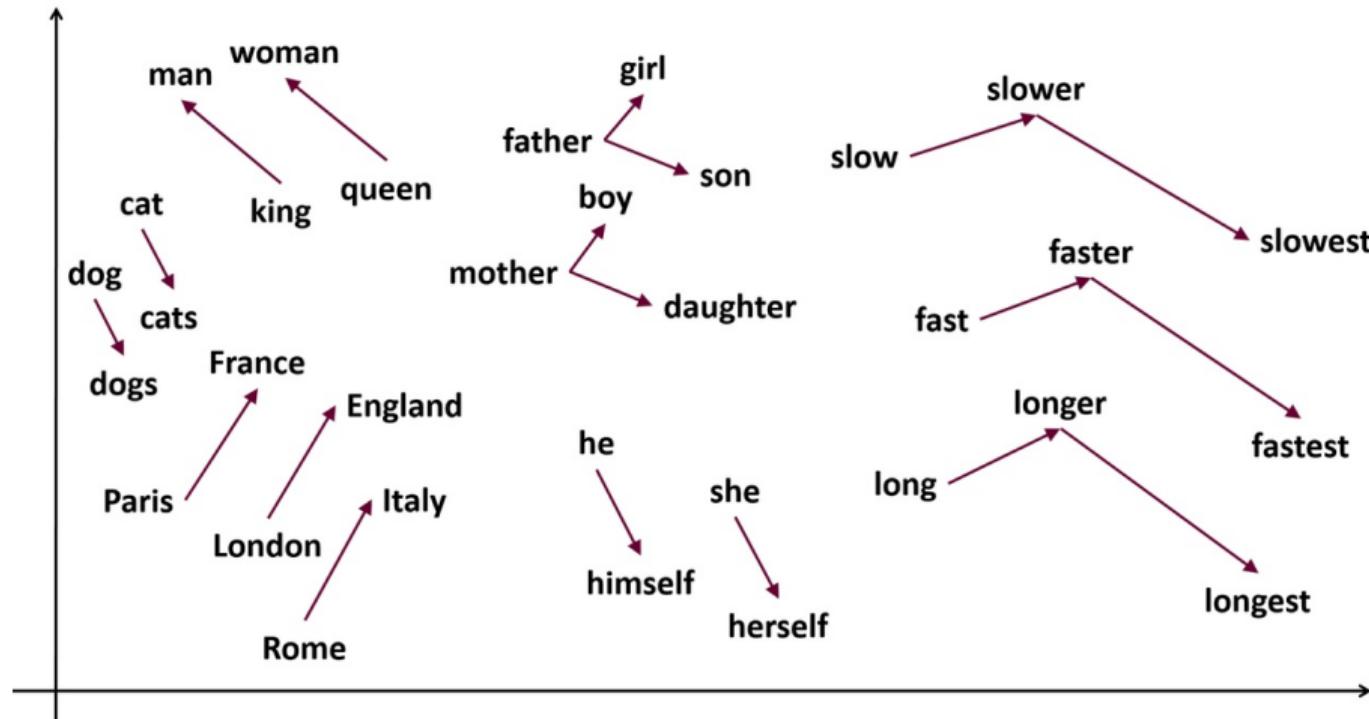
# Words embeddings

- **Наша цель:** хотим научить компьютер понимать слова
- Идея! Давайте превратим наши слова в вектора размера  $d$  и назовём их embeddings
- На вектора понакладываем хотелок!



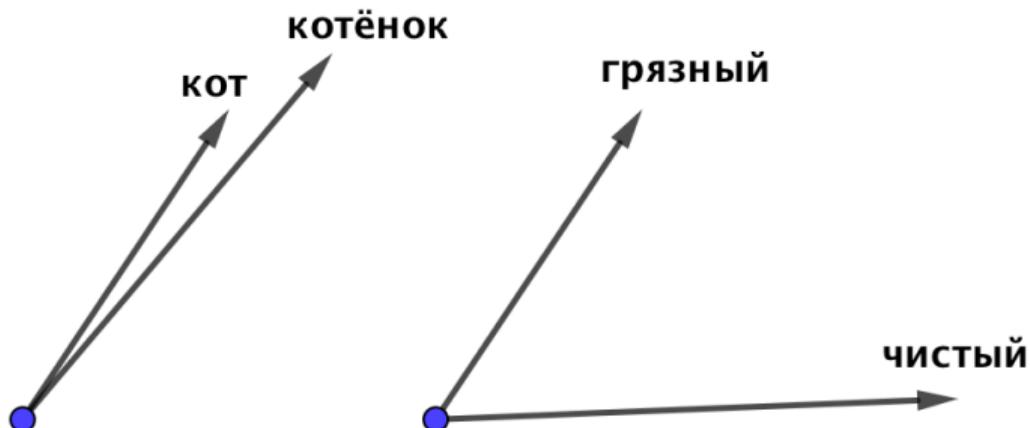
# Хотелка первая

- Хотим, чтобы модель улавливала семантические свойства слов



## Хотелка вторая

- Модель понимала, где близкие по смыслу слова: кот, котёнок, кошка, тигр, лев, ...



# Хотелка третья

- Арифметика!



$$- \text{ (pink silhouette)} + \text{ (blue silhouette)} = \text{ (Khal Drogo)}$$



# Томаш Миколов

- Звучит как магия, но работает
- В 2013 году модель предложена чешским аспирантом Томашем Миколовым
- После работал в Google, сейчас ушёл в Facebook



# Постановка задачи

контекст  
к слов до  
к слов после

$$w_c = \frac{w_1 + w_2 + w_3 + w_4}{4}$$

Вчера на обед Король Лев съел Пумбу

W1 W2

W0

W3 W4

- Контекст —  $k$  слов до рассматриваемого и  $k$  после него.
- Вероятность встретить наше слово в контексте  $c$ :

$$P(w_0 | w_c) = \frac{e^{s(w_0, w_c)}}{\sum e^{s(w_i, w_c)}}$$

## Максимально правдоподобно

- Вероятность встретить наше слово в контексте  $c$ :

$$P(w_0 \mid w_c) = \frac{e^{s(w_0, w_c)}}{\sum e^{s(w_i, w_c)}}$$

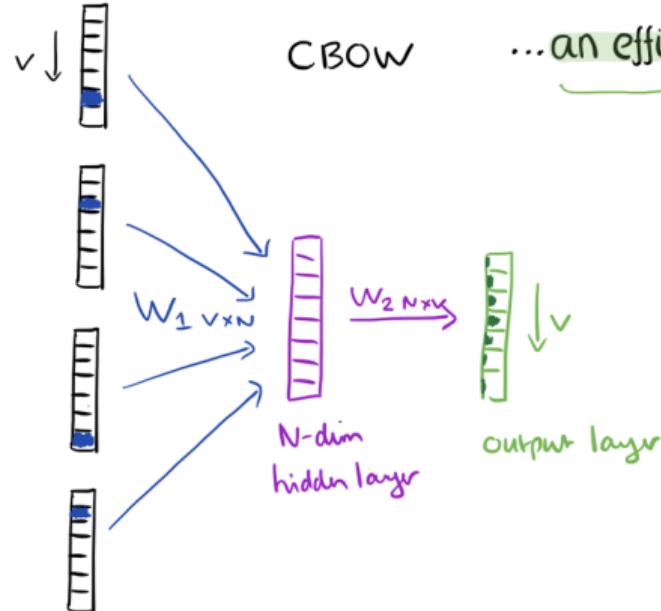
- Мы хотим настроить вектора для слов так, чтобы эти вероятности были большими, если слово встречается в контексте и маленькими, если не встречаются.
- Воспользуемся методом максимального правдоподобия:

$$\ln L = \sum_{(i,c)} \ln P(w_i \mid w_c) \rightarrow \max_w .$$

# Проблемы

- Очень много параметров  $W$ . По каждому брать производную?!
- Хочется побыстрее. Любая взвешенная сумма - нейросеть. Давайте запишем правдоподобие в виде нейросетки.
- Есть два подхода: непрерывный мешок слов. В нём мы прогнозируем слово по контексту.
- Skip-gram. В нём мы прогнозируем контекст по слову.

# CBOW



...an efficient method for learning high quality distributed vector ...

content

focus  
word

content

Пробуем предсказать  
целевое слово по контексту

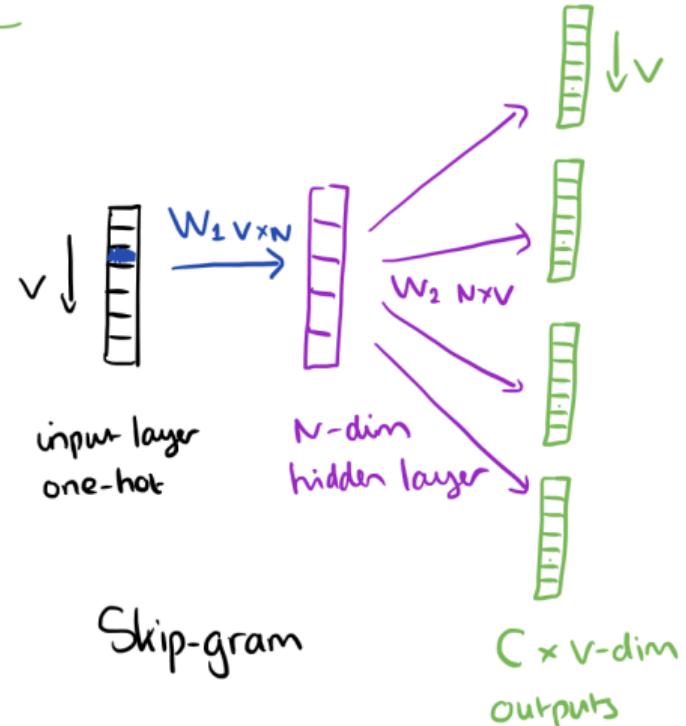
one-hot  
content word  
input vectors

# Skip-gram

...an efficient method for learning high quality distributed vector ...



Пробуем предсказать  
Контекст по целевому слову



# Обучаем свою w2v-нейронку

# Откуда взять данные (дампы Википедии)

## Wikimedia Downloads

If you are reading this on Wikimedia servers, please note that we have rate limited downloaders and we are capping the number of per-ip connections to 2. This will help to ensure that everyone can access the files with reasonable download times. Clients that try to evade these limits may be blocked. Our mirror sites do not have this cap.

### Data downloads

The Wikimedia Foundation is requesting help to ensure that as many copies as possible are available of all Wikimedia database dumps. Please [volunteer to host a mirror](#) if you have access to sufficient storage and bandwidth.

#### Database backup dumps

A complete copy of all Wikimedia wikis, in the form of wikitext source and metadata embedded in XML. A number of raw database tables in SQL form are also available.

These snapshots are provided at the very least monthly and usually twice a month. If you are a regular user of these dumps, please consider subscribing to [xmldatadumps-l](#) for regular updates.

Дампы википедии для разных языков: <https://dumps.wikimedia.org>

Например, для русского: <https://dumps.wikimedia.org/ruwiki/>

# Откуда взять данные (НКРЯ)

The screenshot shows the homepage of the National Corpus of Russian Language (Национальный корпус русского языка). The header features a logo with stylized letters 'НКРЯ' and the text 'НАЦИОНАЛЬНЫЙ КОРПУС РУССКОГО ЯЗЫКА'. Below the header, there is a navigation bar with links: 'главная' (main), 'архив новостей' (news archive), 'поиск в корпусе' (search in the corpus), 'что такое корпус?' (what is the corpus?), 'состав и структура' (composition and structure), 'статистика' (statistics), 'графики' (charts), 'частоты' (frequencies), and 'морфология' (morphology). The main content area has a title 'Национальный корпус русского языка' and an English link 'English'. Below the title, there is a paragraph about the corpus's purpose and history, followed by a section on its composition and structure.

национальный корпус  
русского  
языка

главная  
архив новостей  
поиск в корпусе  
что такое корпус?  
состав и структура  
статистика  
графики  
частоты  
морфология

Национальный корпус русского языка [English](#)

На этом сайте помещен корпус современного русского языка общим объемом более 600 млн слов. Корпус русского языка — это информационно-справочная система, основанная на собрании русских текстов в электронной форме.

Корпус предназначен для всех, кто интересуется самыми разными вопросами, связанными с русским языком: профессиональных лингвистов, преподавателей языка, школьников и студентов, иностранцев, изучающих русский язык.

Развитие подкорпусов НКРЯ (основного, поэтического, параллельного, акцентологического, диалектного) в 2015 году осуществлялось при поддержке РГНФ, проекты № 15-04-12018 «Развитие специализированных модулей НКРЯ» и № 14-04-12012 «Корпус диалектных текстов Национального корпуса русского языка. Пополнение и разметка».

[Как пользоваться Корпусом \(инструкция в формате PDF\)](#)

[Подробнее о корпусе](#)

Национальный корпус Русского языка: <http://ruscorpora.ru>

# Где взять уже готовенькое (проект RusVectōrēs)

## Модели

В настоящий момент вы можете скачать следующие модели (жирным выделены модели, доступные для использования в веб-интерфейсе):

Таблицу можно (и нужно) пролистывать по горизонтали!

Стационарный идентификатор ▾	Скачать ▾	Корпус ▾	Размер корпуса ▾	Объём словаря	Частотный порог ▾	Таргет ▾	Алгоритм ▾	Размерность вектора ▾	Размер окна ▾
a_upos_skipgram_300_2_2018	<b>331 Мбайт</b>	Тайга	почти 5 миллиардов слов	237 255	200	Universal Tags	Continuous Skipgram	300	2
корпора_upos_skipgram_300_5_2018	<b>191 Мбайт</b>	НКРЯ	260 миллионов слов	195 071	20	Universal Tags	Continuous Skipgram	300	5
ikiuruscorpora_upos_skipgram_300_2_2018	<b>376 Мбайт</b>	НКРЯ и Википедия за декабрь 2017	600 миллионов слов	384 764	40	Universal Tags	Continuous Skipgram	300	2
rs_upos_cbow_600_2_2018	<b>547 Мбайт</b>	Русскоязычные новости, с сентября 2013 до ноября 2016	почти 5 миллиардов слов	289 191	200	Universal Tags	Continuous Bag-of-Words	600	2
арендум_upos_skipgram_300_2_2018	<b>192 Мбайта</b>	Araneum	около 10 миллиардов слов	196 620	400	Universal Tags	Continuous Skipgram	300	2
арендум_none_fasttextcbow_300_5_2018	<b>1 Гбайт</b>	Araneum	около 10 миллиардов слов	195 782	400	Нет	fastText CBOW (3.-5-граммы)	300	5
арендум_none_fasttextskipgram_300_5_2018	<b>675 Мбайт</b>	Araneum	около 10 миллиардов слов	195 782	400	Нет	fastText Skipgram	300	5

Куча разных моделей для русского языка: <https://rusvectores.org/ru/models/>

# Где взять уже готовенькое (проект RusVectōrēs)

## Семантический калькулятор

Здесь можно вычислять отношения: например, «найти слово **C**, связанное со словом **B**, таким же образом, как слово **A** связано со словом **B**». Таким образом можно определять семантические связи между понятиями и решать задачи на аналогии. В форме ввода приведен пример: какое слово относится к слову «Лондон», так же, как «Россия» относится к «Москве»? Ответ — «Великобритания»: Лондон столица Великобритании, а Москва — столица России.

[Подробнее...](#)

Москва      Лондон

Россия      ???

Выберите модель:

Aneauum fastText  НКРЯ  Новостной корпус  НКРЯ и Wikipedia  Тайга

Показывать только:

Существительные  Наречия  Имена собственные  Прилагательные  Глаголы  Все части речи

**Вычислить!**

Вы также можете попробовать арифметические операции над большим количеством слов.

Введите в «**положительную**» и «**отрицательную**» формы не более 10 слов через пробел. *RusVectōrēs* сложит вектора положительных слов и вычесть из них отрицательные. Затем он выдаст слова, наиболее близкие к получившемуся вектору. Если вы оставите отрицательное поле пустым, *RusVectōrēs* просто найдет центр лексического кластера, образованного положительными словами.

+      телефон мобильный

-      ???

Выберите модель:

Aneauum fastText  НКРЯ  Новостной корпус  НКРЯ и Wikipedia  Тайга

Показывать только:

Существительные  Наречия  Имена собственные  Прилагательные  Глаголы  Все части речи

**Вычислить!**

Куча разных моделей для русского языка: <https://rusvectores.org/ru/calculator/>

# Где взять уже готовенькое (Google)

Google Code Archive

Search this site

Projects Search About

Project word2vec

Source

Issues Tool for computing continuous distributed representations of words.

Wikis

Downloads

**Introduction**

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

**Quick start**

- Download the code: svn checkout <http://word2vec.googlecode.com/svn/trunk/>
- Run 'make' to compile word2vec tool
- Run the demo scripts: `./demo-word.sh` and `./demo-phrases.sh`
- For questions about the toolkit, see <http://groups.google.com/group/word2vec-toolkit>

**Project Information**

The project was created on Jul 30, 2013.

- License: Apache License 2.0
- 945 stars
- svn-based source control

Labels:

NeuralNetwork MachineLearning  
NaturalLanguageProcessing WordVectors  
Google

Гуглловская модель для английского языка: <https://code.google.com/archive/p/word2vec/>

# Подведём итоги

## something2vec

- **Embedding** – это сопоставление произвольной сущности (например, узла в графе или кусочка картинки) некоторому вектору.
- Любую последовательность можно представить в виде эмбединга
- Последовательность банковских транзакций
- Веб-сессии (последовательность перехода по сайтам)
- Графы взаимосвязей между пользователями
- Любая категориальная переменная: порядок, в котором турист посещал города; порядок, в котором юзер отректировал сериалы и тп

# Что почитать про нейронки



# Что посмотреть про нейронки

- Если ничего не знаете про машинное обучение, смотрите вводный курс от Яндекса и МФТИ:
- Серия лекций техносферы:  
<https://www.youtube.com/watch?v=Am82yvUSwRE>
- Введение в нейронные сети от Andrew Ng:  
<https://www.coursera.org/instructor/andrewng>
- Advanced ML от Яндекса: <https://www.coursera.org/specializations/ml>

Я ОСОБО И НЕ  
ЗНАЮ НИЧЕГО

ПОДОЖДИ ЧЕ  
СПРОШУ

