

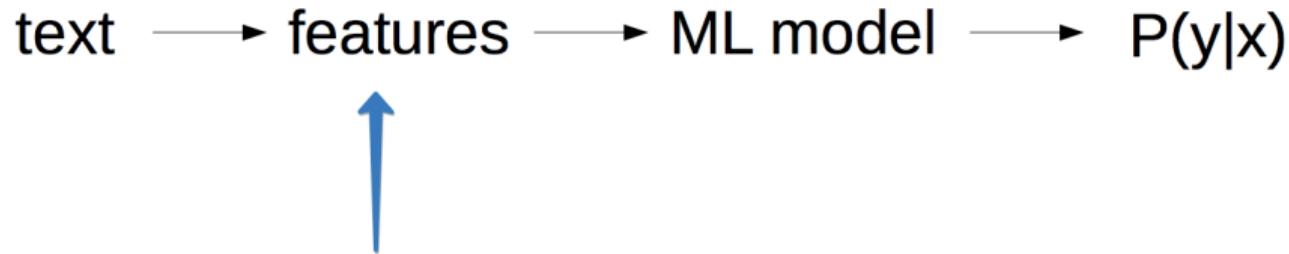
# Глубокое обучение и вообще

Коломейцева Катерина

Занятие 8: Из слов в вектора и обратно

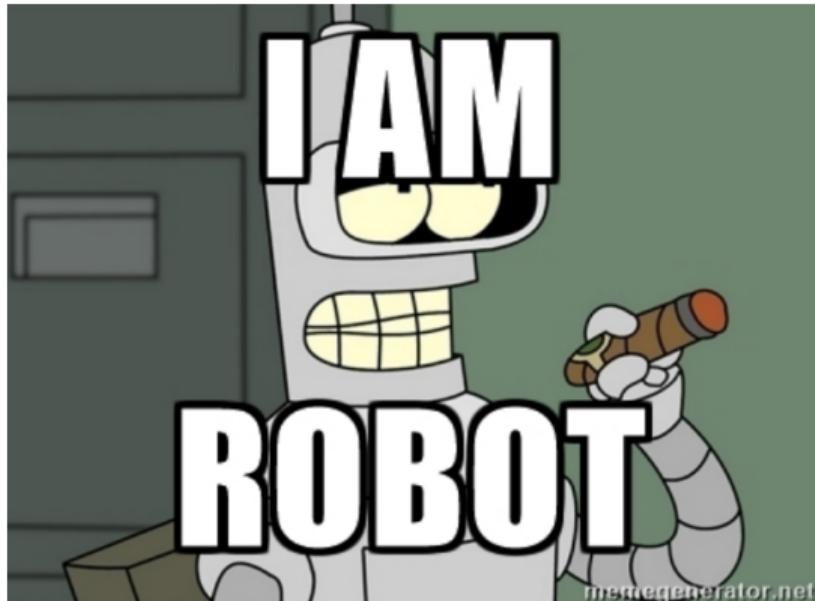
# NLP (Natural Language Processing)

# Модели на текстах



Как представить текст  
в виде, который могла  
бы понять модель?

# Что такое текст?



- Текст (документ) — это последовательность токенов (слов)
- Токен (слово) — это последовательность символов

# Bag of words

1. Нежился на пляже
2. Копали яму на пляже
3. Копал картошку
4. Ел картошки и картошку



	нежиться	пляж	копать	яма	картошка	есть
1	1	1	0	0	0	0
2	0	1	1	1	0	0
3	0	0	1	0	1	0
4	0	0	0	0	2	1

# Lemmatization

Мы хотим уменьшить словарь слов, но сохранив как можно больше информации. Лемматизация - процесс перевода слов по какому-то словарю. Для корректной работы нужен постоянно обновляющийся и большой словарь слов.

# Stemming

Отрубаем по заданному правилу разные куски слова. Всякие приставки суффиксы и тому подобное. Не зависим от словаря слов, но как и любые эвристики могут привести к непредсказуемым результатам - когда-то слишком сильно слово изменим, когда-то наоборот не сможем привести к нормальной форме.

# TF-IDF

Идея подхода просто - слова которые встречаются во всех документах очень часто не несут информацию. И слова, которые встречаются слишком редко тоже не несут информацию. TF(term frequency) - Считаем частоту слова в каждом документе корпуса.  $tf(t, d) = f_{t,d}$

IDF (Inverse Document Frequency) - считаем вес для редких слов в разрезе

всего корпуса текстов.  $idf = \log\left(\frac{N}{D}\right)$ , где N - сколько всего документов в корпусе; D- в скольких документах мы встретили данное слово.

Итоговый вес -  $tf \cdot idf$ .

# 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]

5. ОНЕ или tf-idf,  
а затем  
моделирование

4. очистка от слишком редких слов

# 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.

# Из слов в вектора и обратно



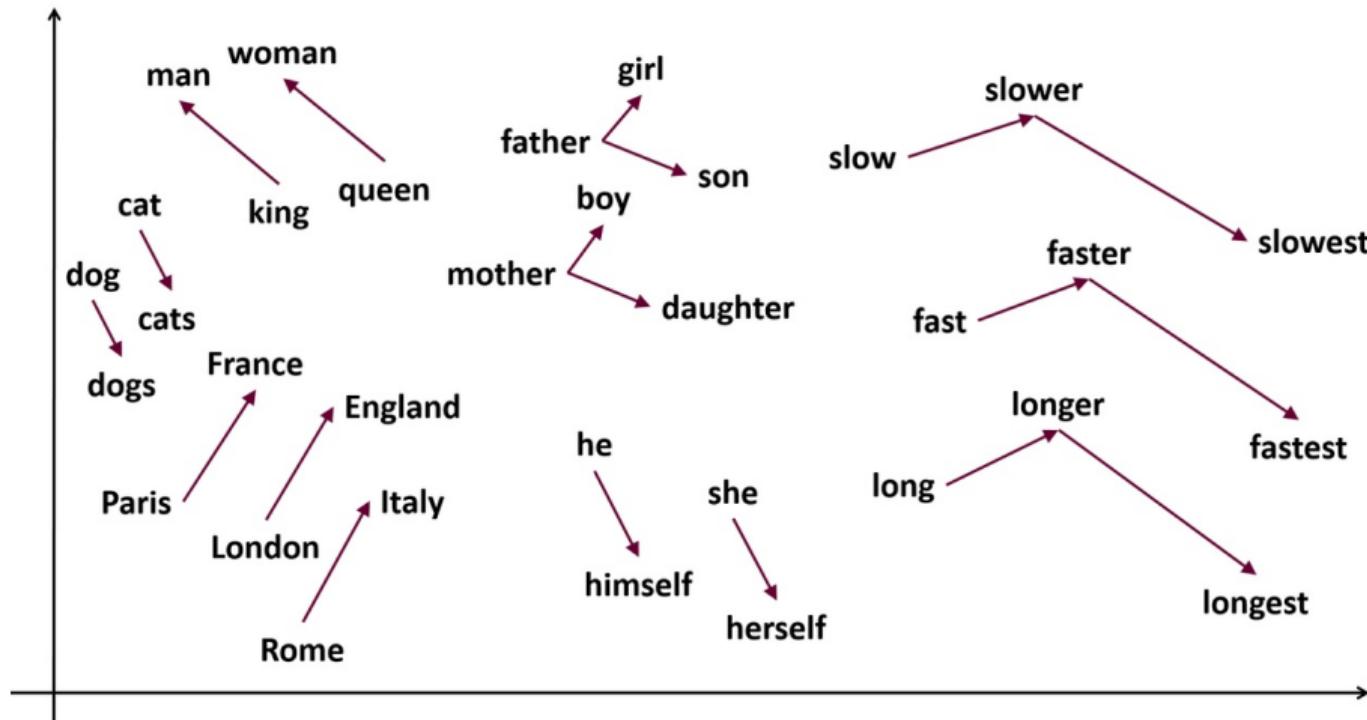
# Words embeddings

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



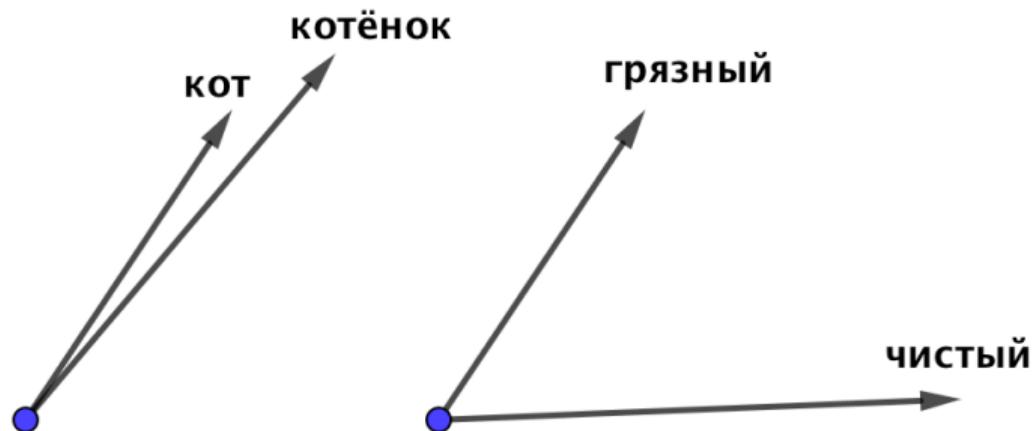
# Хотелка первая

- Хотим, чтобы модель улавливала семантические свойства слов



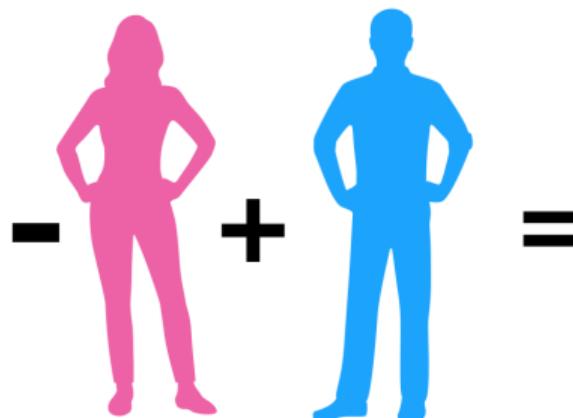
# Хотелка вторая

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



# Хотелка третья

- Арифметика!



# 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  $c$ :

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

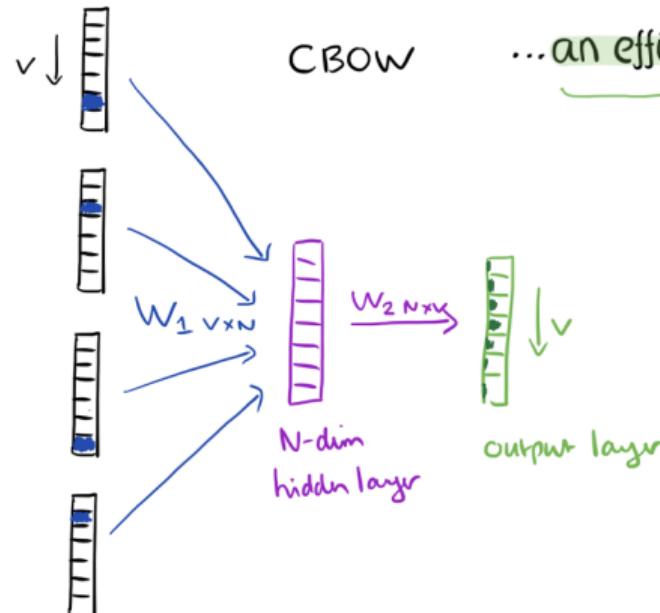
- Мы хотим настроить вектора для слов так, чтобы эти вероятности были большими, если слово встречается в контексте и маленькими, если не встречаются.
- Воспользуемся методом максимального правдоподобия:

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

# Проблемы

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

# CBOW



one-hot  
content word  
input vectors

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

context

↑  
focus  
word

context

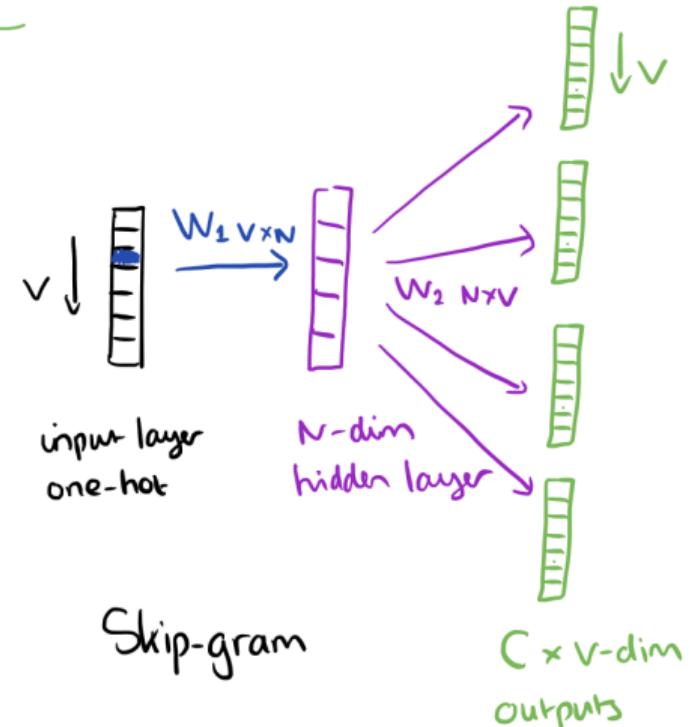
Пробуем предсказать  
целевое слово по контексту

# Skip-gram

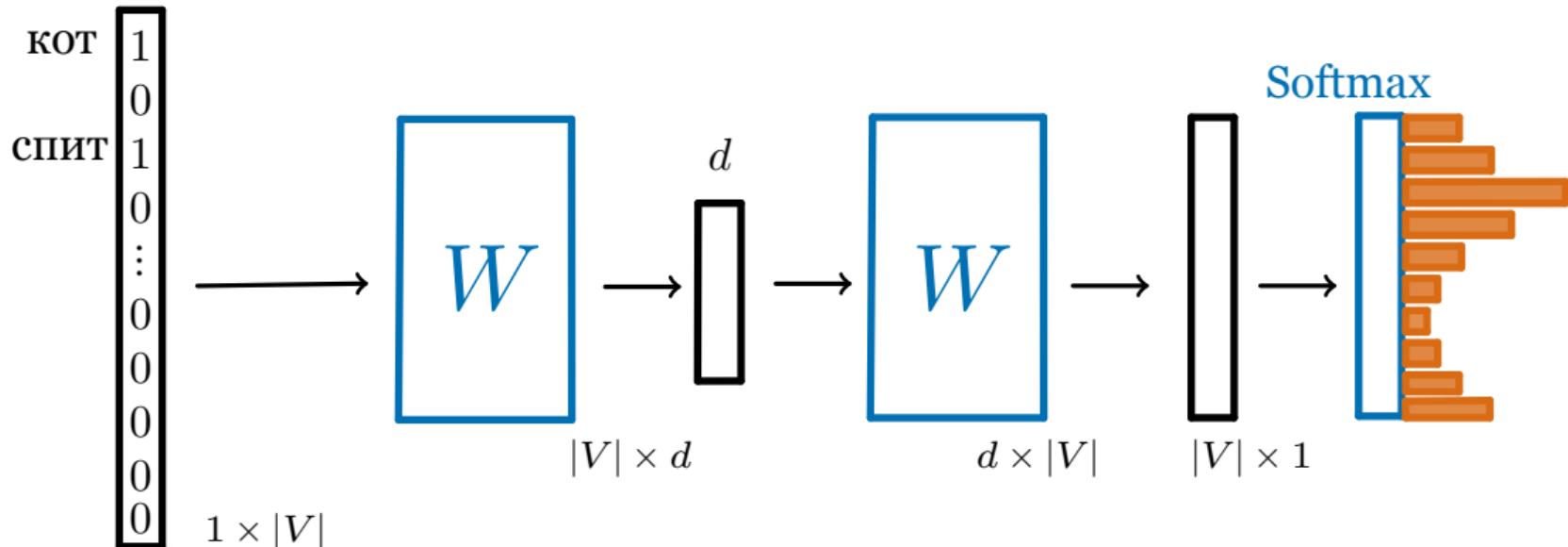
...an efficient method for learning high quality distributed vector ...



Пробуем предсказать  
Контекст по целевому слову



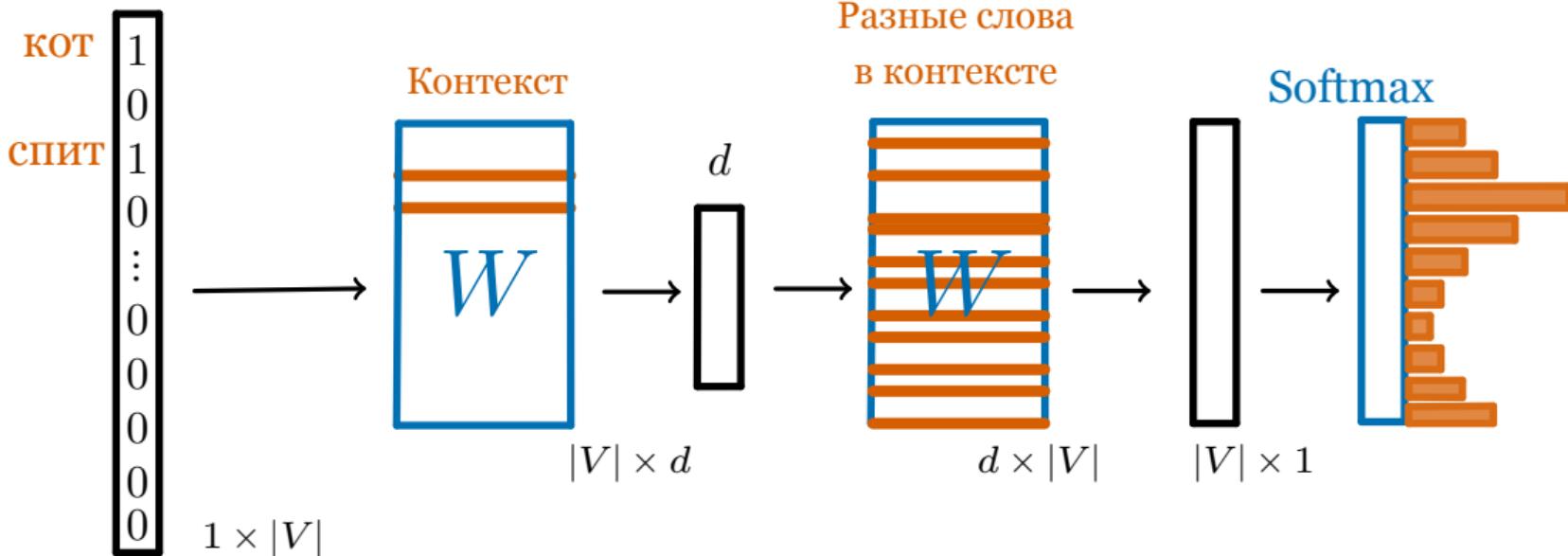
# CBOW



# Word2Vec

- Трансформирует пространство текстов в  $d$ -мерное пространство векторов
- В матрице  $W$  будут записаны наши итоговые вектора
- Между словами на выходе выполняются интересные арифметические свойства
- Для обучения требует много данных
- В текущей процедуре обучения есть проблемка

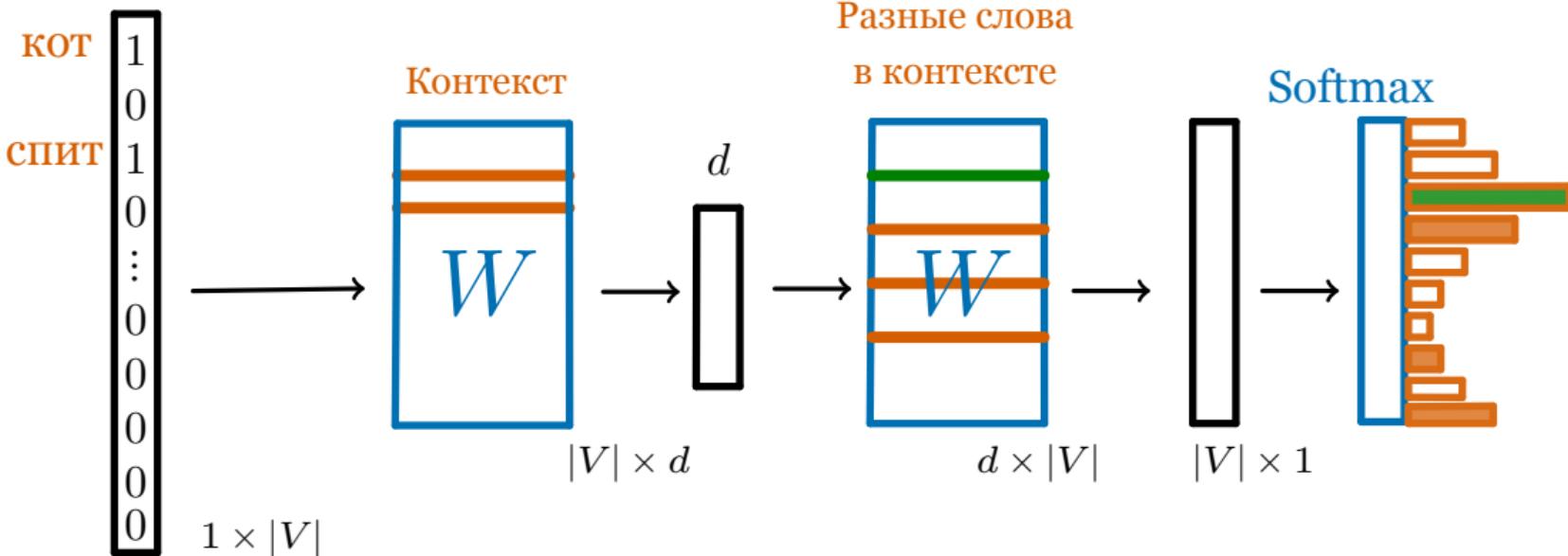
# Проблема



# Negative Sampling

- Контекст, который подходит под конкретное слово обычно мал. Получаем очень много избыточных вычислений
- Считать все около нулевые вероятности достаточно накладно и тяжело, хочется ускорить процесс.
- Давайте введём случайную величину, которая с некоторой вероятностью не будет учитывать контекстные слова.

# Negative Sampling



# Полезные мысли про обучение

- Вы можете довольно легко собрать свой собственный w2v и обучить его, но не стоит делать это. Ваша реализация не будет такой эффективной, как уже существующие специализированные реализации. За последние годы алгоритмы для обучения w2v претерпели существенную эволюцию.
- Реализация w2v из пакета gensim зачастую работает быстрее, чем модели, написанные в стандартных для нейросеток бэкэндах. Это происходит из-за многопоточности и разных умных оптимизаций тонких мест в обучении.

# Полезные мысли про обучение

- При достаточно большом корпусе текстов можно не делать лемматизацию. Сетка сама поймёт по контексту, что слова близки и присвоит им похожие вектора.
- Если у вас специфическая задача, в которой встречается специфическая лексика, возьмите предобученную на большом корпусе сетку и дообучите её под свои нужды.
- w2v может выдавать эмбединги только для слов из заданного при обучении словаря. Этот минус можно попытаться побороть и получить другую модель, fasttext

# 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 отранижировал сериалы и тп

# something2vec



-



+



=



# Где взять данные и предобученные модели

# 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			Алгоритм	Размерность вектора	Размер окна
				Частотный порог	Тэгсет	Алгоритм			
<b>a_upos_skipgram_300_2_2018</b>	<b>331 Мбайт</b>	Тайга	почти 5 миллиардов слов	237 255	200	Universal Tags	Continuous Skipgram	300	2
<b>corpora_upos_skipgram_300_5_2018</b>	<b>191 Мбайт</b>	НКРЯ	250 миллионов слов	195 071	20	Universal Tags	Continuous Skipgram	300	5
<b>ikiuruscorpora_upos_skipgram_300_2_2018</b>	<b>376 Мбайт</b>	НКРЯ и Википедия за декабрь 2017	600 миллионов слов	384 764	40	Universal Tags	Continuous Skipgram	300	2
<b>rs_upos_cbow_600_2_2018</b>	<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
<b>ieum_upos_skipgram_300_2_2018</b>	<b>192 Мбайт</b>	Araneum	около 10 миллиардов слов	196 620	400	Universal Tags	Continuous Skipgram	300	2
<b>ieum_none_fasttextcbow_300_5_2018</b>	<b>1 Гбайт</b>	Araneum	около 10 миллиардов слов	195 782	400	Нет	fastText CBOW (3.-5-граммы)	300	5
<b>ieum_none_fasttextskipgram_300_5_2018</b>	<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 (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: `J/demo-word.sh` and `J/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/> Risk

# Свойства word2vec



Частотность слова

- Высокая  Средняя  Низкая

## НКРЯ и Wikipedia

- чай noun 0.56
- пиво noun 0.56
- самогон noun 0.56
- лимонад noun 0.53
- напиток noun 0.53



## НКРЯ и Wikipedia

- преданность noun 0.38
- доброта noun 0.37
- нежность noun 0.37
- упование noun 0.35
- умиление noun 0.35

<https://rusvectores.org/ru/calculator>

# Свойства word2vec

Результат обучения векторных представлений сильно зависит от коллекции документов. Могут возникать неожиданные артефакты.

## Википедия

most_similar(россия)
российский 0.5653642416
рф 0.523694574833
украина 0.492026507854
ссср 0.473026573658
финляндия 0.464367419481
most_similar(тролль)
муметь 0.717674195766
гоблин 0.559770524502
великан 0.557757973671
злобный 0.55741250515
гном 0.554968833923

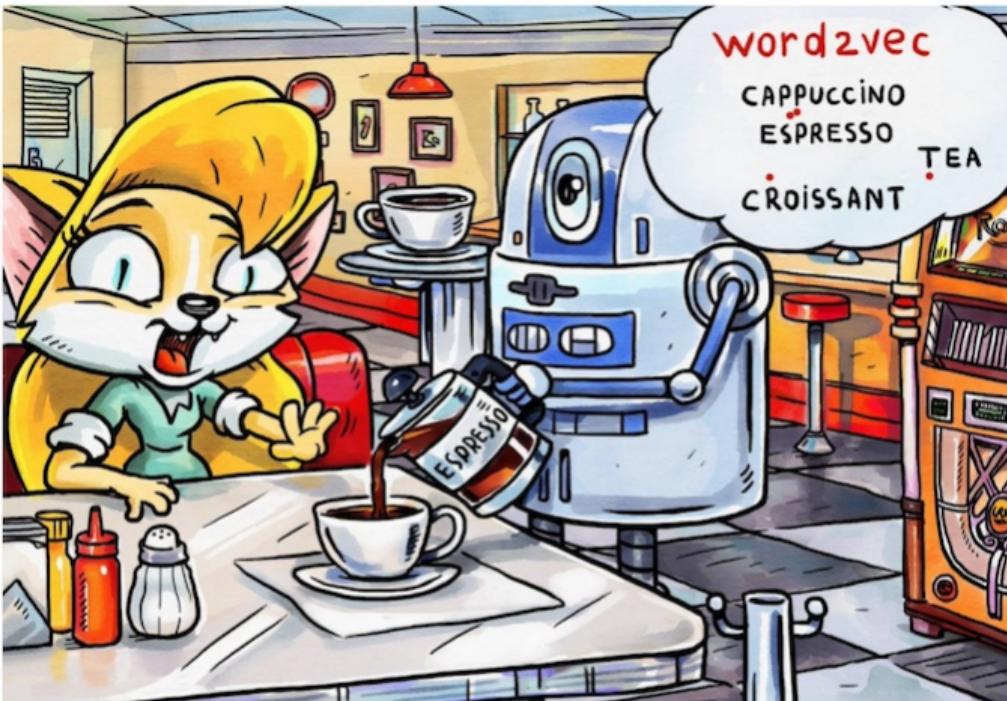
## Луркоморье

most_similar(россия)
беларусь 0.645048737526
европа 0.622894406319
украина 0.622316598892
рашка 0.619276404381
германия 0.609378278255
most_similar(тролль)
троллинг 0.725703835487
троль 0.660580933094
лжец 0.582996308804
проводокатор 0.57004237175
толстый 0.568691492081

# Модель - сексист

```
model.most_similar(u'интеллектуал')
[('моралист', 0.7139864563941956),
 ('теоретик', 0.6941959857940674),
 ('литератор', 0.6819325089454651)]  
  
model.most_similar(u'интеллектуалка')
[('бездельница', 0.6617184281349182),
 ('бунтарка', 0.6578608751296997),
 ('дилетантка', 0.6419748663902283)]  
  
  
model.most_similar(positive=[u'интеллектуал', u'женщина'], negative=[u'мужчина'])
[('знаменитость', 0.49774518609046936),
 ('поп-звезда', 0.4860984981060028),
 ('элита', 0.48200151324272156)]  
  
model.most_similar(positive=[u'ум', u'женщина'], negative=[u'мужчина'])
[('любовь', 0.43659064173698425),
 ('душа', 0.4330841302871704),
 ('внешность', 0.4280041456222534)]  
  
model.most_similar(positive=[u'гений', u'женщина'], negative=[u'мужчина'])
[('букашка', 0.4793989062309265),
 ('химера', 0.4589369595050812),
 ('душонка', 0.4547439217567444)]
```

# Word2vec всего лишь модель



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

# 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)



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>

# Учим свой w2v