

# Практический анализ данных и машинное обучение: искусственные нейронные сети

Ульянкин Филипп

4 февраля 2019 г.

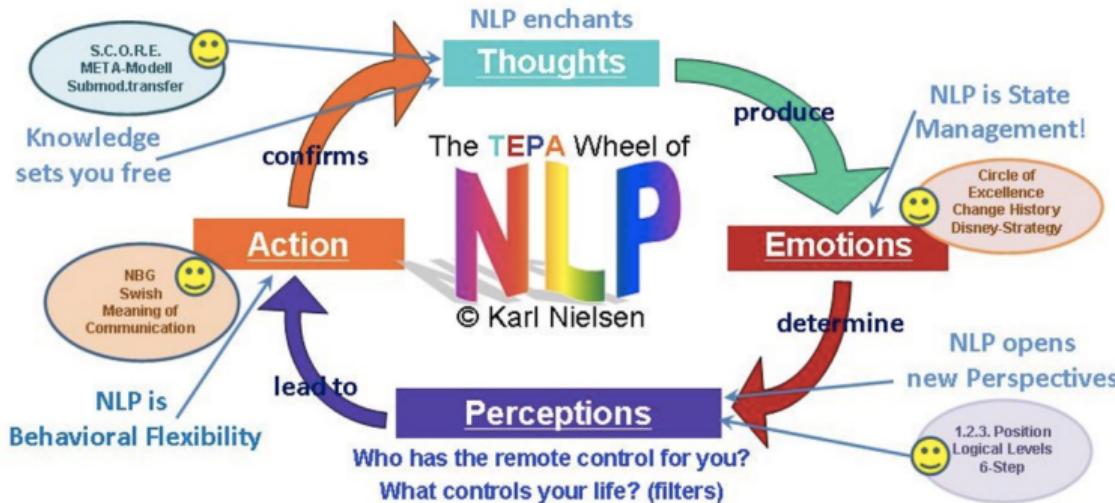
Как научить компьютер читать, w2v

# Agenda

- Анализ текстов и логистическая регрессия
- Что такое embeddings, модель word2vec
- Учим свой собственный w2v на википедии, сравниваем с чужим
- Сентимент-анализ недавних событий по твиттеру

# NLP (Natural Language Processing)

# NLP, the freedom in Thinking, Feeling, Perceiving and Behavior





# freedom in thinking, receiving, behavior

Feelings

S.C.O.R.E.  
META-Modell  
Submod.transfer

Knowledge  
sets you free

confirms

NBG  
Swish  
Meaning of  
Communication

NLP is  
Behavioral F

Action

produce

NLP is State  
Management!

Circle of  
Excellence  
Change History  
Disney-Strategy

Emotions

determine

NLP opens  
new Perspectives

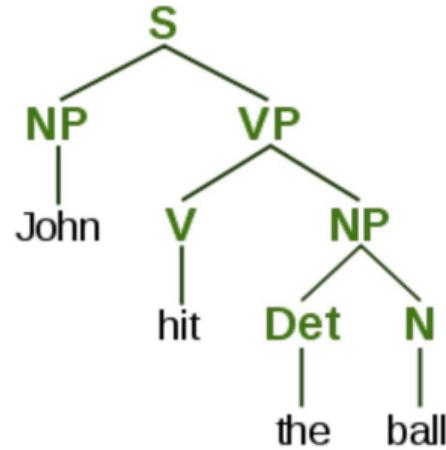
1,2,3, Position  
Metaphorical Levels  
One-Step

Perceptions

Who has the remote control for you?  
What controls your life? (filters)



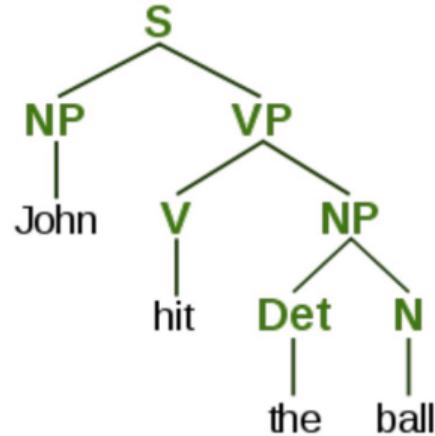
Neuro-linguistic programming



Natural language processing



NLP курильщика



NLP здорового человека

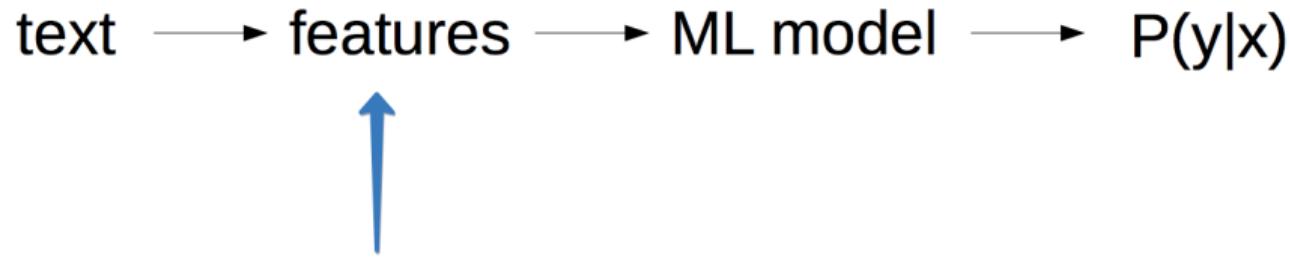
## Классификация и регрессия:

- Поиск спама
- Сентимент-анализ
- Модерация контента
- Предсказание популярности поста
- Многое многое другое ...

## Задачи посложнее:

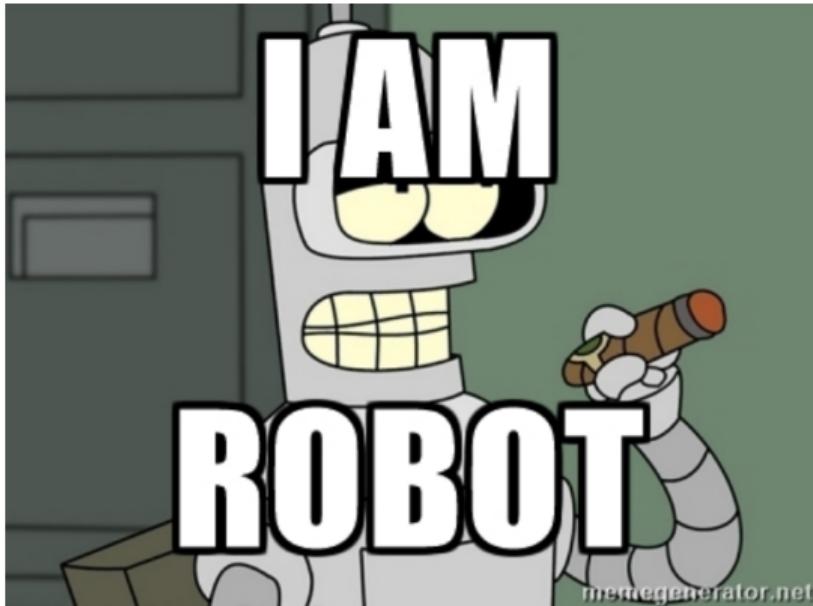
- Рекомендательные системы
- Диалоговые системы (чат боты)
- Машинный перевод
- Тематизация новостей, их сжатие в краткое саммари

# Классификация текстов



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

# Что такое текст?



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

# Мешок слов

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

# Предобработка текста



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

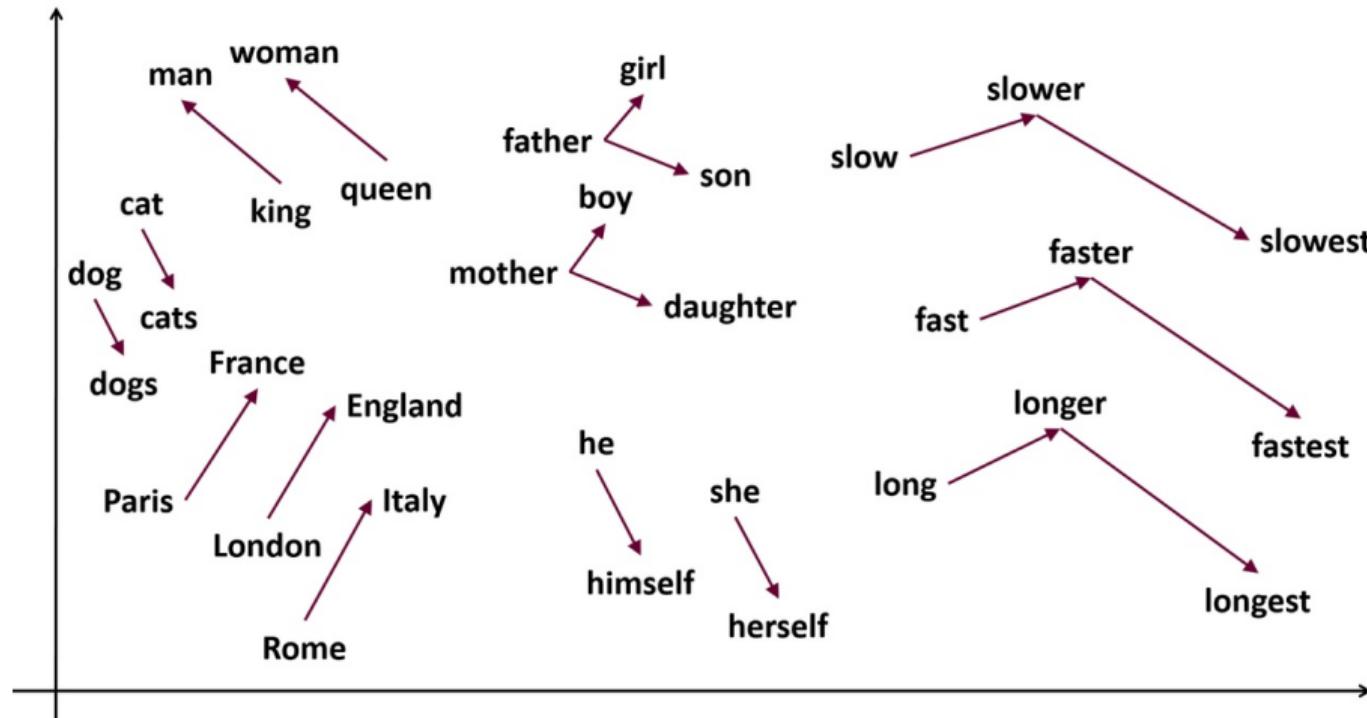
# Words embeddings

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



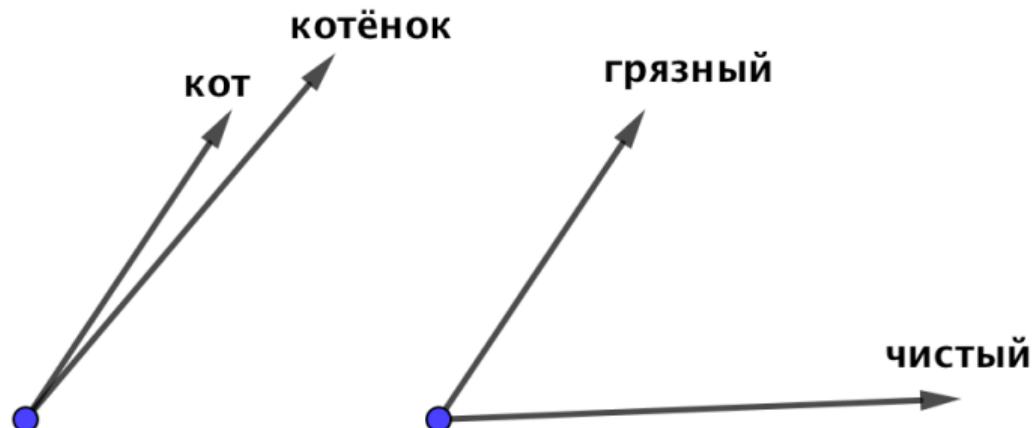
# Хотелка первая

- Хотим, чтобы модель улавливала семантические свойства слов



## Хотелка вторая

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



# Хотелка третья

- Арифметика!



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



# 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  $c$ :

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

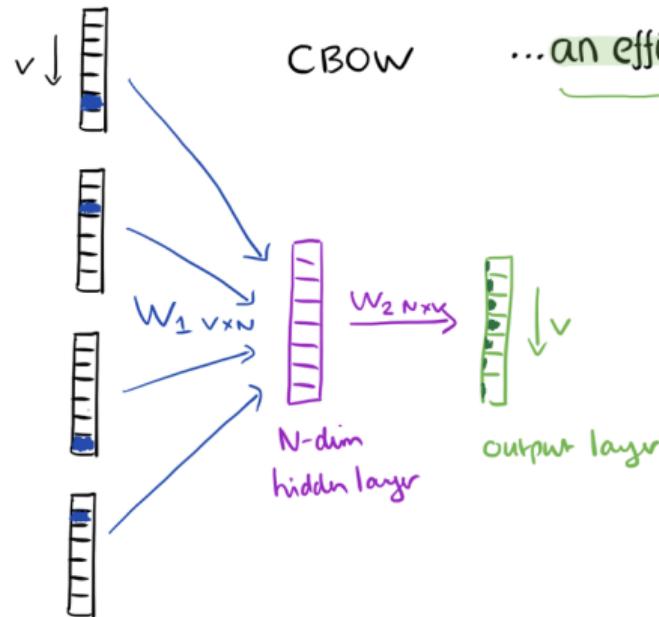
- Мы хотим настроить вектора для слов так, чтобы эти вероятности были большими, если слово встречается в контексте и маленькими, если не встречаются.
- Воспользуемся методом максимального правдоподобия:

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

# Проблемы

- Очень много параметров  $W$ . По каждому брать производную?!
- Хочется побыстрее. Любая взвешенная сумма - нейросеть. Давайте запишем правдоподобие в виде нейросетки.
- Есть два подхода: 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 ...

content

focus  
word

content

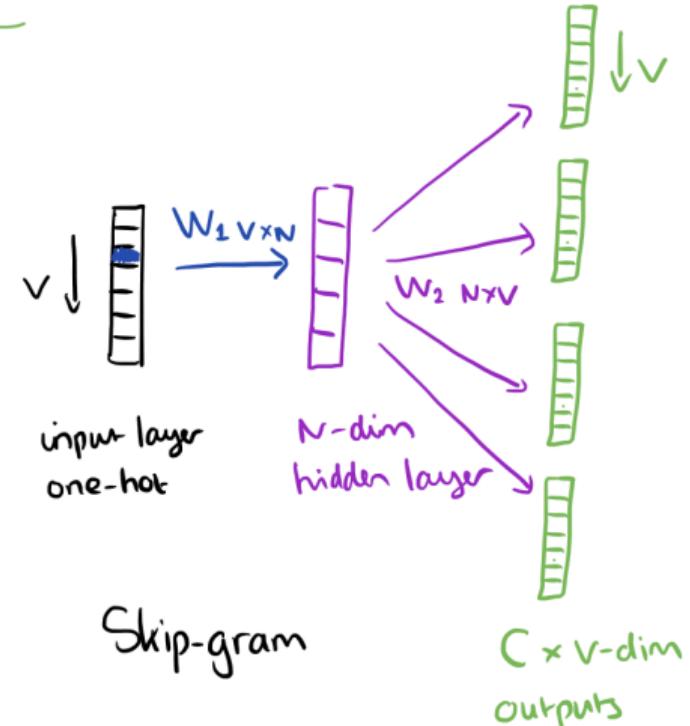
Пробуем предсказать  
целевое слово по контексту

# Skip-gram

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



Пробуем предсказать  
Контекст по целевому слову



# Word2Vec

- Трансформирует пространство текстов в  $d$ -мерное пространство векторов
- В матрице  $W_1$  будут записаны наши итоговые вектора
- Между словами на выходе выполняются интересные арифметические свойства
- Для обучения требует много данных

# 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)



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!

Стоянный идентификатор ▾	Скачать ▾	Корпус ▾	Размер корпуса ▾	Объём словаря	Частотный порог ▾	Таргет ▾	Алгоритм ▾	Размерность вектора ▾	Размер окна ▾
<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>корpora_upos_skipgram_300_5_2018</b>	<b>191 Мбайт</b>	НКРЯ	260 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 Мбайта</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 (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/>

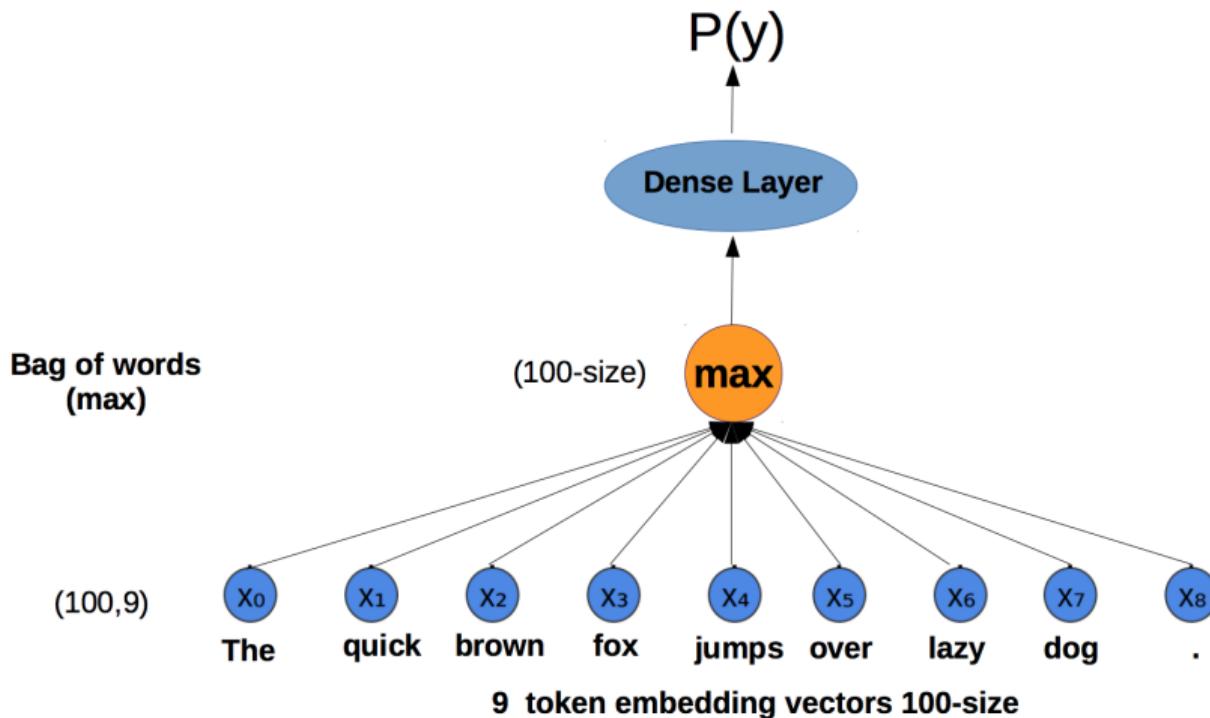
## Полезные мысли

- В tensorflow 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.
- 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.

Попробуем обучить свою модель и  
сравнить её с RusVectōrēs моделью

# Нейросетки для текстов

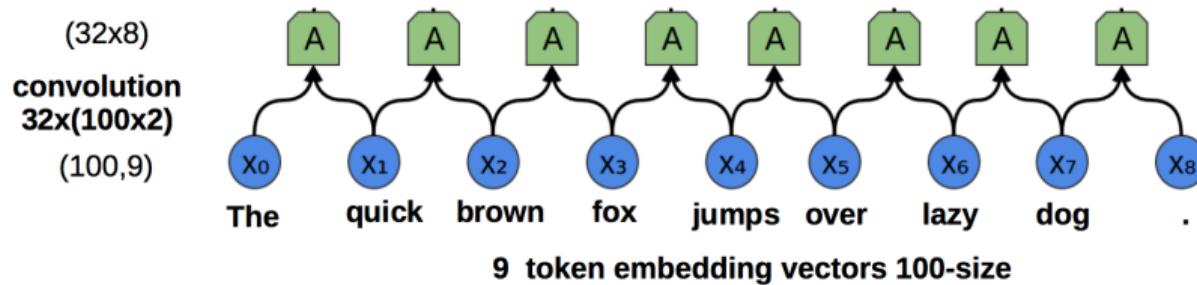
# Нейросеть для текстов



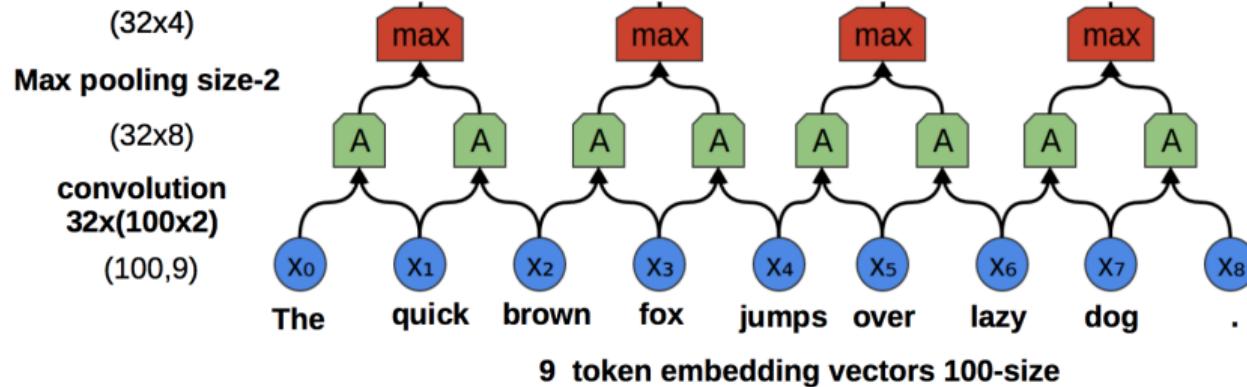
# Проблемы

- Теряем информацию о порядке слов
- Решить эту проблему можно обучив эмбединги для биграм, треграм и тд, но это расширит пространство признаков
- Можно решить эту проблему с помощью рекурентных нейросеток, о них мы будем говорить в следующий раз
- Можно решить эту проблему с помощью свёрточного слоя

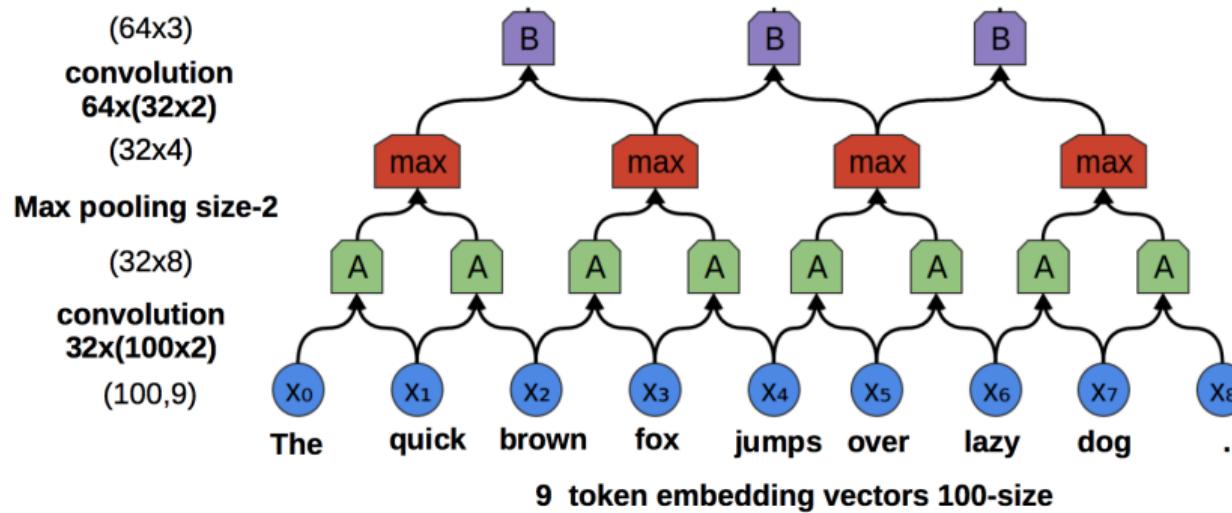
# Свёрточная сеть



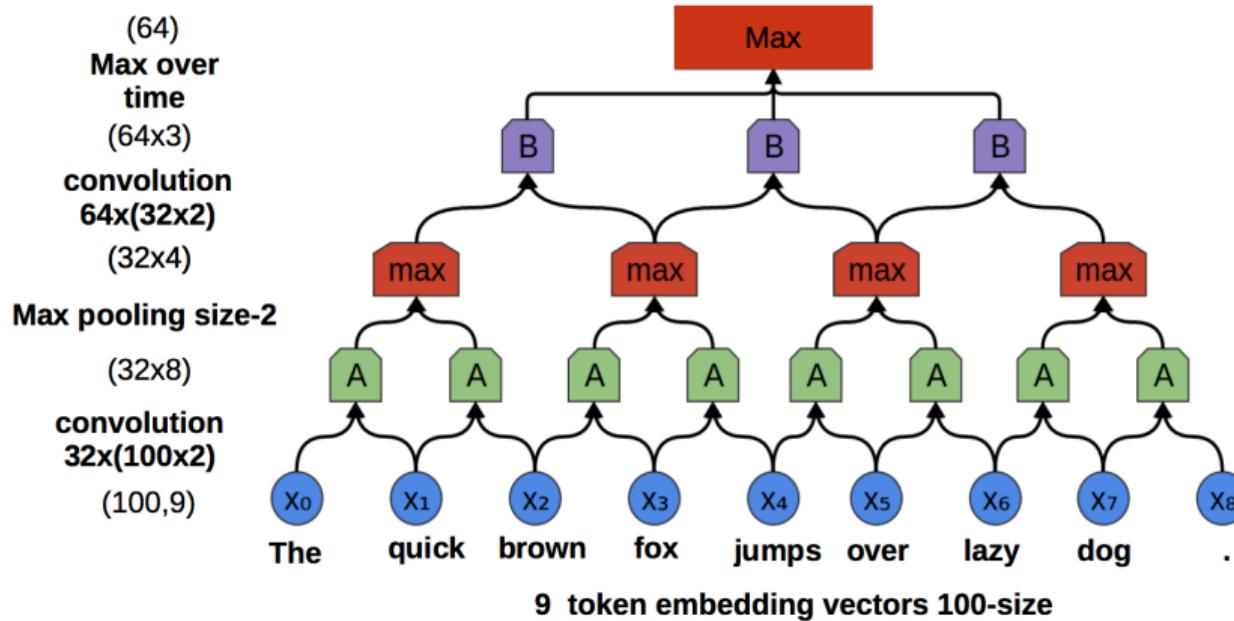
# Свёрточная сеть



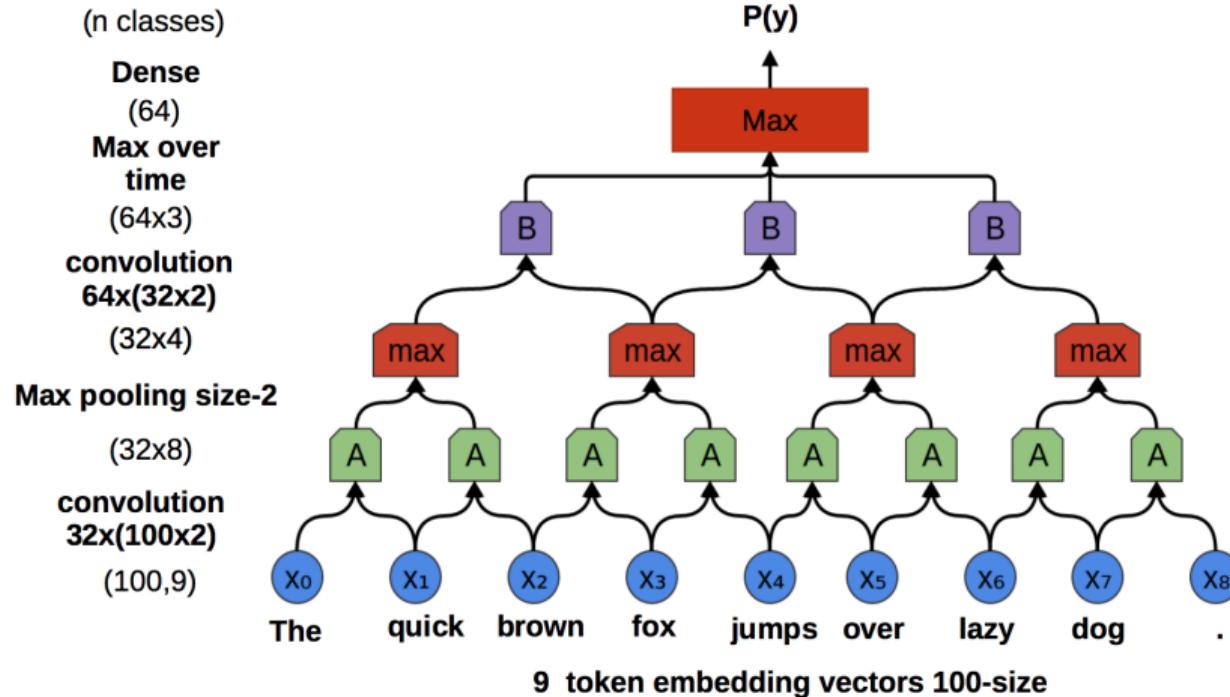
# Свёрточная сеть



# Свёрточная сеть

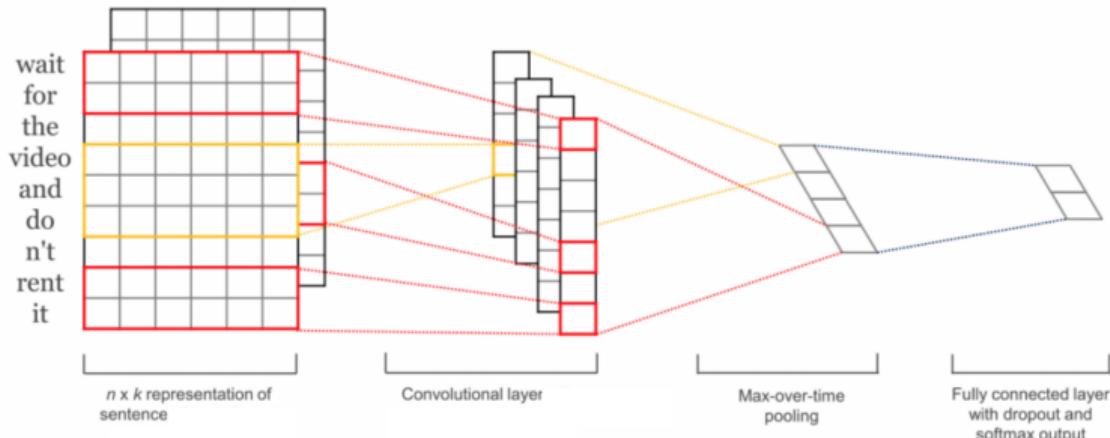


# Свёрточная сеть



# Свёрточная сеть

1-hot/emb

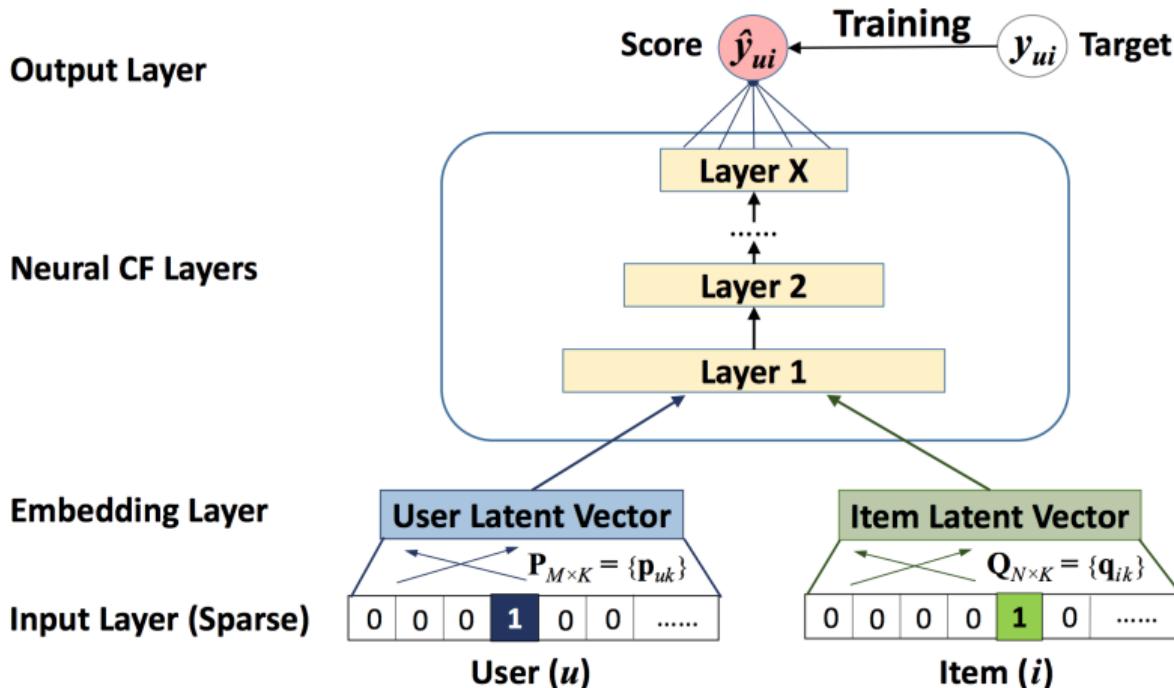


# Что такое эмбединги

## 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

# Рекомендательные системы



<https://arxiv.org/pdf/1708.05031.pdf>