



# СОВРЕМЕННЫЕ ЗАДАЧИ И МЕТОДЫ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА

Василий Шампоров

Июль 2020

Internet of Things Group

# Естественный язык

— ... язык, используемый для общения **людей** ... и не созданный целенаправленно (в отличие от искусственных языков).

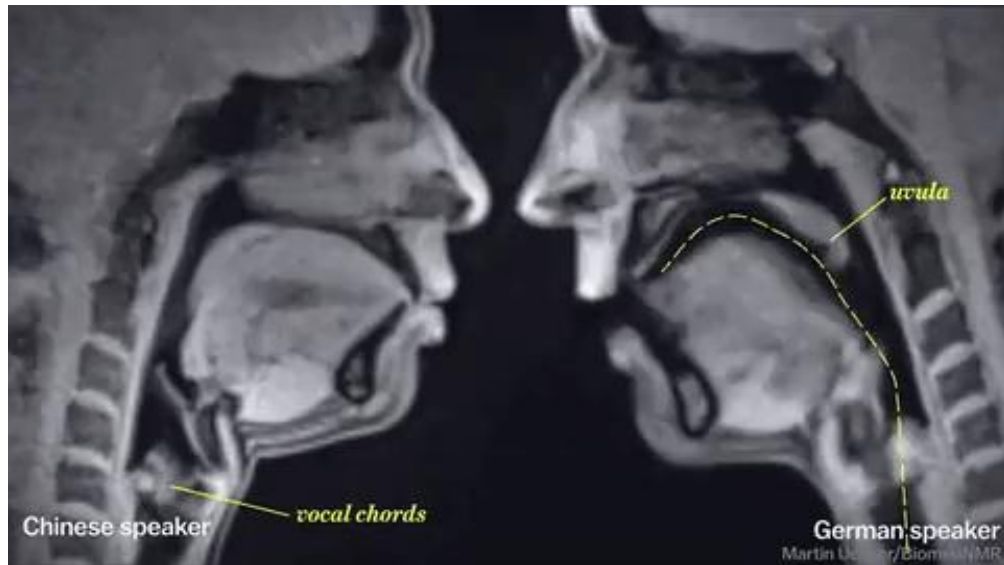
# Естественный язык - письменность



- Последовательность определенных символов (букв, идеографов, пиктограмм, иероглифов, слов, предложений), воспринимаемая глазом

# Естественный язык - устная речь

Image credit: Vox, "The Simpsons"



- Последовательность определенных механических колебаний (звуков, фонем), воспринимаемая ухом

# Естественный язык - жесты

Image credit: <https://giphy.com/signwithrobert>



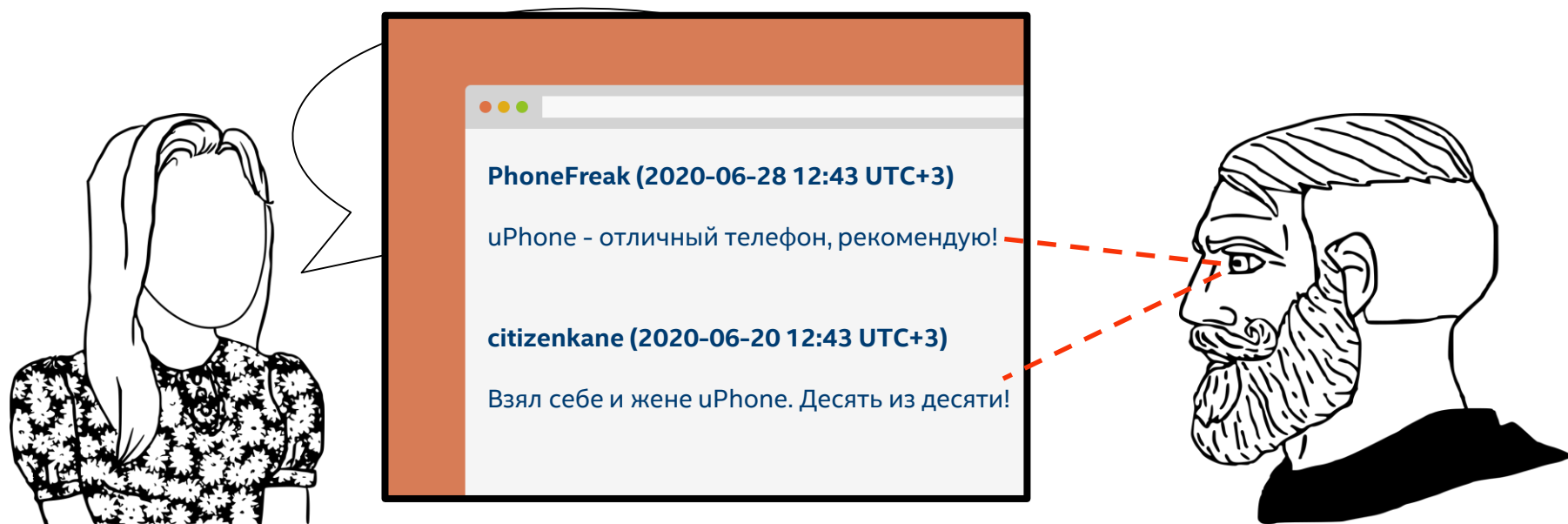
- Последовательность определенных движений (жестов, мимики, формы или движения рта и губ, положения корпуса тела), воспринимаемая глазом

# Задачи обработки естественного языка

# Задачи обработки естественного языка

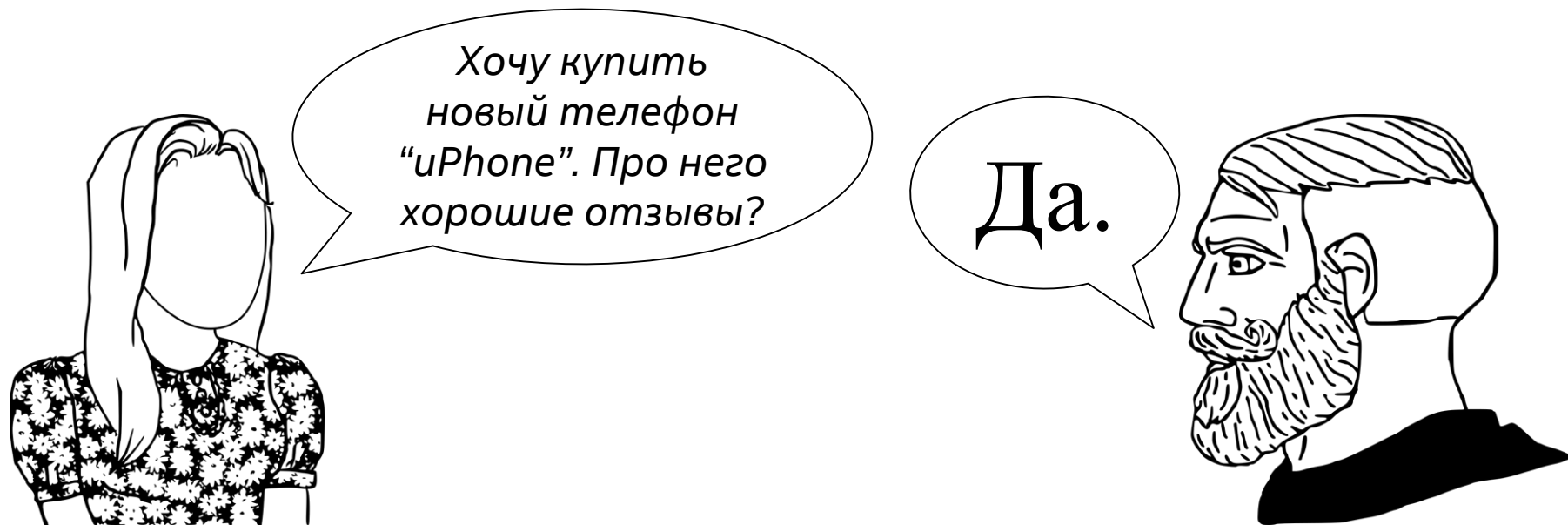


# Задачи обработки естественного языка





# Задачи обработки естественного языка



# Задачи обработки естественного языка

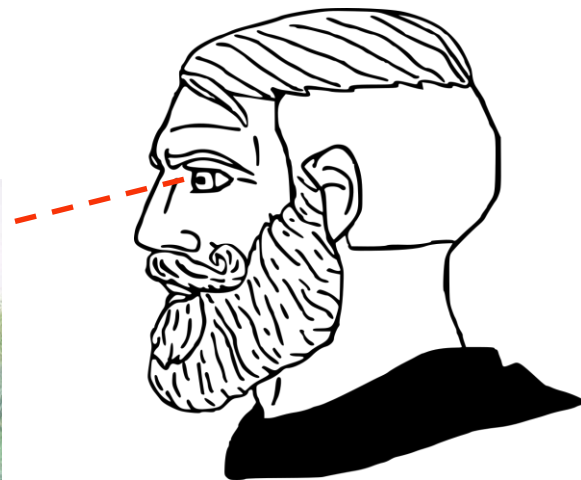
- **Sentiment analysis** - оценка эмоциональной окраски текста



# Задачи обработки естественного языка



# Задачи обработки естественного языка



# Задачи обработки естественного языка



# Задачи обработки естественного языка

- **Text classification** - классификация текстов



# Задачи обработки естественного языка



# Задачи обработки естественного языка



## Метагалактика

Материал из Википедии — свободной энциклопедии  
(перенаправлено с «[Наблюдаемая Вселенная](#)»)

**Наблюда́емая Вселе́нная** — понятие в [космологии](#) [Большого взрыва](#), описывающее часть [Вселенной](#), являющуюся абсолютным [прошлым](#). Число [галактик](#) оценивается более чем в 500 млрд<sup>[2]</sup>.





# Задачи обработки естественного языка



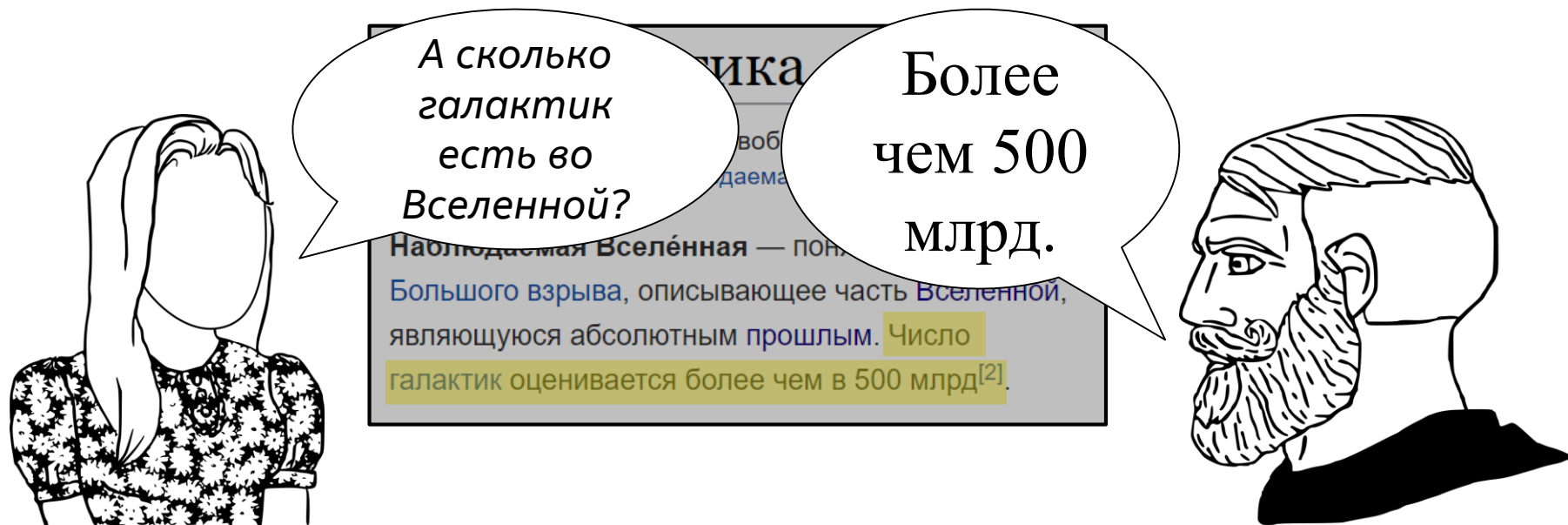
## Метагалактика

Материал из Википедии — свободной энциклопедии  
(перенаправлено с «[Наблюдаемая Вселенная](#)»)

**Наблюда́емая Вселе́нная** — понятие в [космологии](#) [Большого взрыва](#), описывающее часть [Вселенной](#), являющуюся абсолютным [прошлым](#). [Число галактик оценивается более чем в 500 млрд<sup>\[2\]</sup>](#).

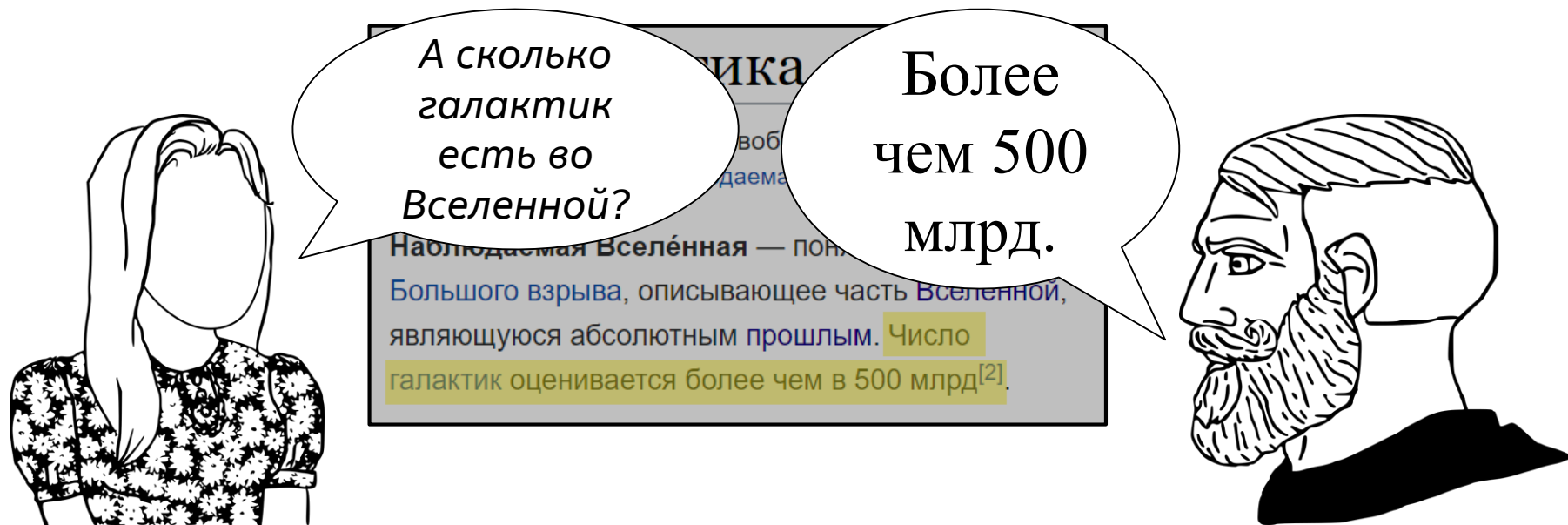


# Задачи обработки естественного языка



# Задачи обработки естественного языка

- **Question answering** - ответ на вопрос по тексту



# Задачи обработки естественного языка



# Задачи обработки естественного языка



# Задачи обработки естественного языка

- **(Machine) Translation** - перевод с одного языка на другой



# Задачи обработки естественного языка



# Задачи обработки естественного языка



*Ты такой  
умный :3*

Спасибо.





# Задачи обработки естественного языка

- **Text/Dialogue generation** - генерация текста/ведение диалога



# Задачи обработки естественного языка

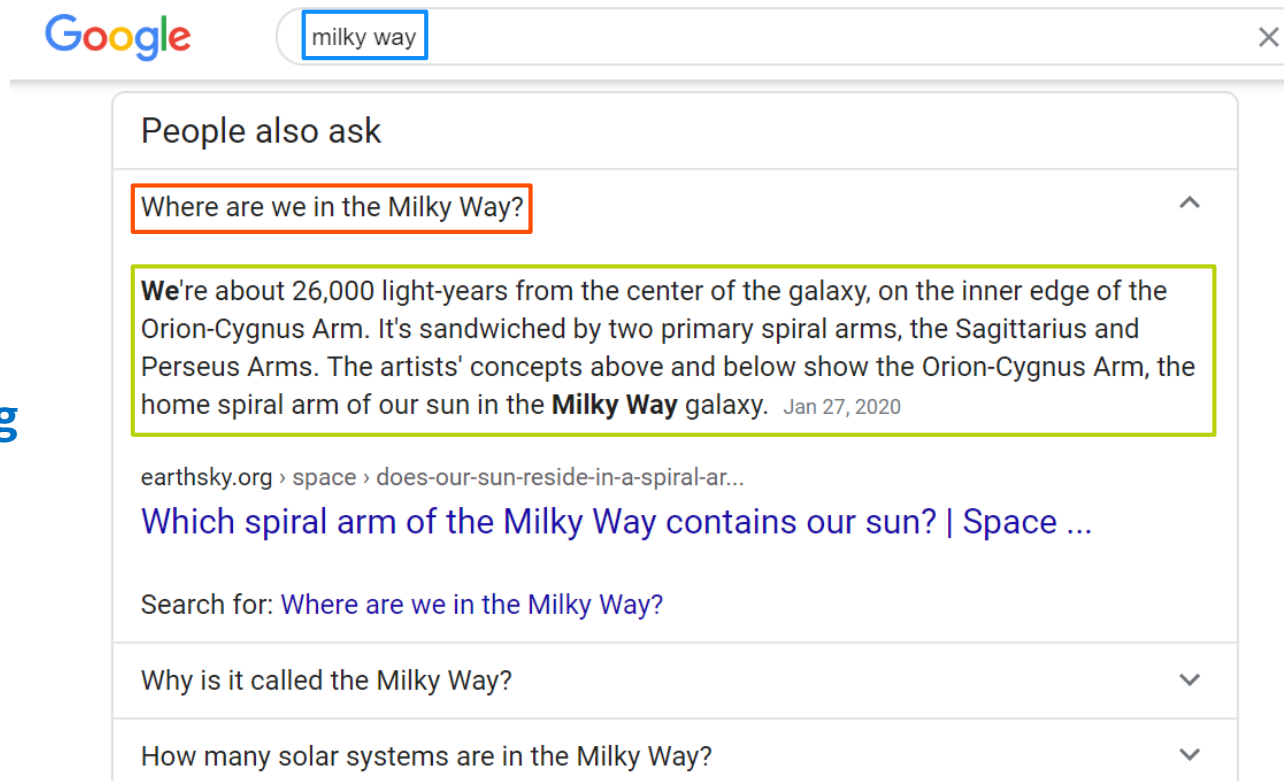
- **Text to speech (TTS)** - озвучивание текста
  - **Speech recognition** - распознавание речи/перевод в текст
  - **Sign language recognition** - распознавание жестовых языков
  - ...
- 

*Англоязычное название сферы задач обработки естественного языка -*

**NLP (Natural Language Processing)**

# Машинный NLP - здесь и сейчас

- Question answering



The screenshot shows a Google search interface. The search bar contains the text "milky way". Below the search bar, under the heading "People also ask", there is a list of related questions. The first question, "Where are we in the Milky Way?", is highlighted with a red border. Below this question, a green-bordered box contains the answer: "We're about 26,000 light-years from the center of the galaxy, on the inner edge of the Orion-Cygnus Arm. It's sandwiched by two primary spiral arms, the Sagittarius and Perseus Arms. The artists' concepts above and below show the Orion-Cygnus Arm, the home spiral arm of our sun in the **Milky Way** galaxy. Jan 27, 2020". Below the answer, the source "earthsky.org" is listed. Further down, there is a link titled "Which spiral arm of the Milky Way contains our sun? | Space ...". At the bottom, there are two more questions: "Why is it called the Milky Way?" and "How many solar systems are in the Milky Way?", each with a downward arrow indicating they can be expanded.

Google milky way

People also ask

Where are we in the Milky Way?

**We're** about 26,000 light-years from the center of the galaxy, on the inner edge of the Orion-Cygnus Arm. It's sandwiched by two primary spiral arms, the Sagittarius and Perseus Arms. The artists' concepts above and below show the Orion-Cygnus Arm, the home spiral arm of our sun in the **Milky Way** galaxy. Jan 27, 2020

earthsky.org › space › does-our-sun-reside-in-a-spiral-ar...

Which spiral arm of the Milky Way contains our sun? | Space ...

Search for: Where are we in the Milky Way?

Why is it called the Milky Way?

How many solar systems are in the Milky Way?

# Машинный NLP - здесь и сейчас

- Question answering

Google

how old is |

how old is **the milky way**

**13.51 billion years**

how old is **the universe**

how old is **the sun**

how old is **the solar system**

how old is **pluto**

how old is **the moon**

how old is **the earth**

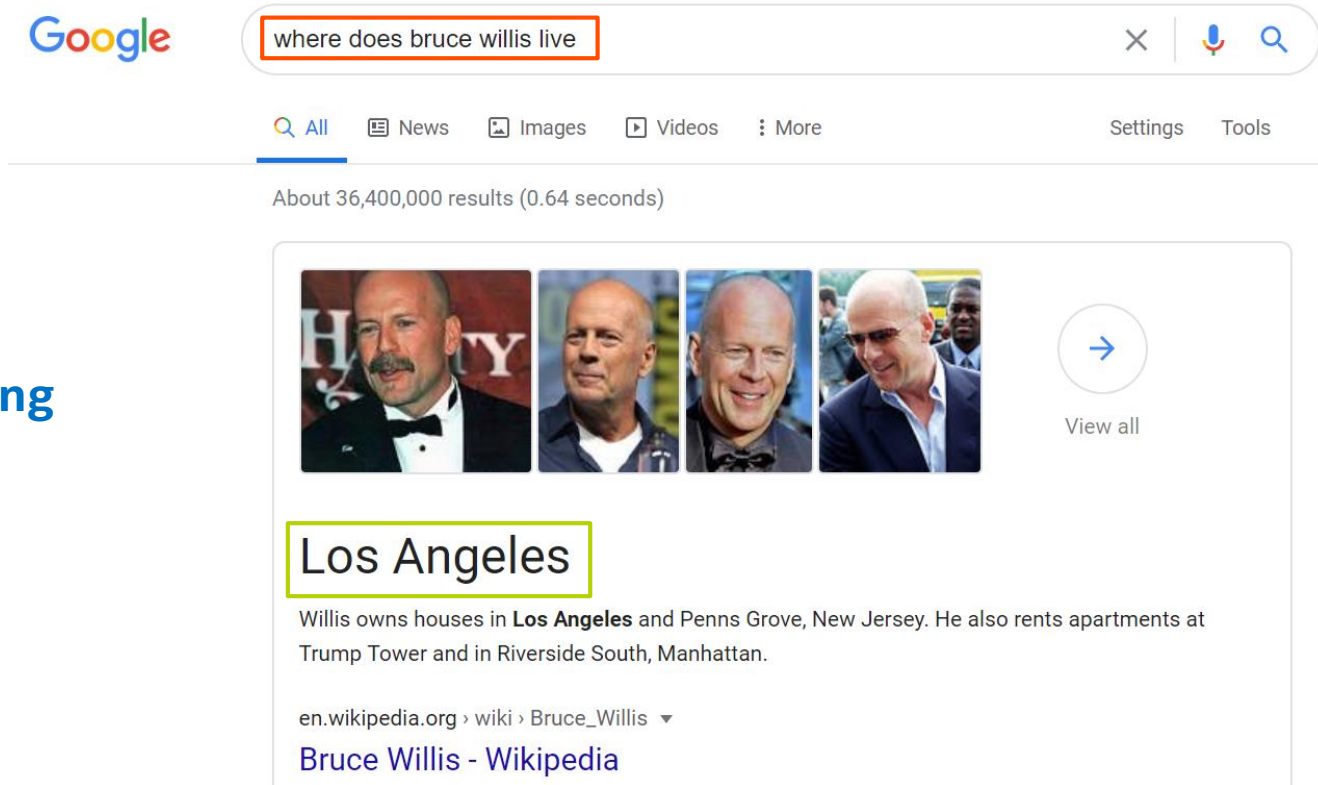
how old is **jupiter**

how old is **saturn**

how old is **our milky way galaxy**

# Машинный NLP - здесь и сейчас

- Question answering



The screenshot shows a Google search interface. The search bar contains the text "where does bruce willis live". Below the search bar, the results show "About 36,400,000 results (0.64 seconds)". A carousel of four images of Bruce Willis is displayed, followed by a "View all" link. Below the images, the text "Los Angeles" is highlighted in a yellow box. Underneath, a paragraph states: "Willis owns houses in **Los Angeles** and Penns Grove, New Jersey. He also rents apartments at Trump Tower and in Riverside South, Manhattan." At the bottom, a link to "en.wikipedia.org > wiki > Bruce\_Willis" is shown, followed by the text "Bruce Willis - Wikipedia".

Google

where does bruce willis live

All News Images Videos More

Settings Tools

About 36,400,000 results (0.64 seconds)

View all

**Los Angeles**

Willis owns houses in **Los Angeles** and Penns Grove, New Jersey. He also rents apartments at Trump Tower and in Riverside South, Manhattan.

en.wikipedia.org > wiki > Bruce\_Willis

Bruce Willis - Wikipedia

# Машинный NLP - здесь и сейчас

Source: <https://www.youtube.com/watch?v=aMcjxSThD54>

- **Speech to text**



# Машинный NLP - здесь и сейчас

Image credit: vk.com

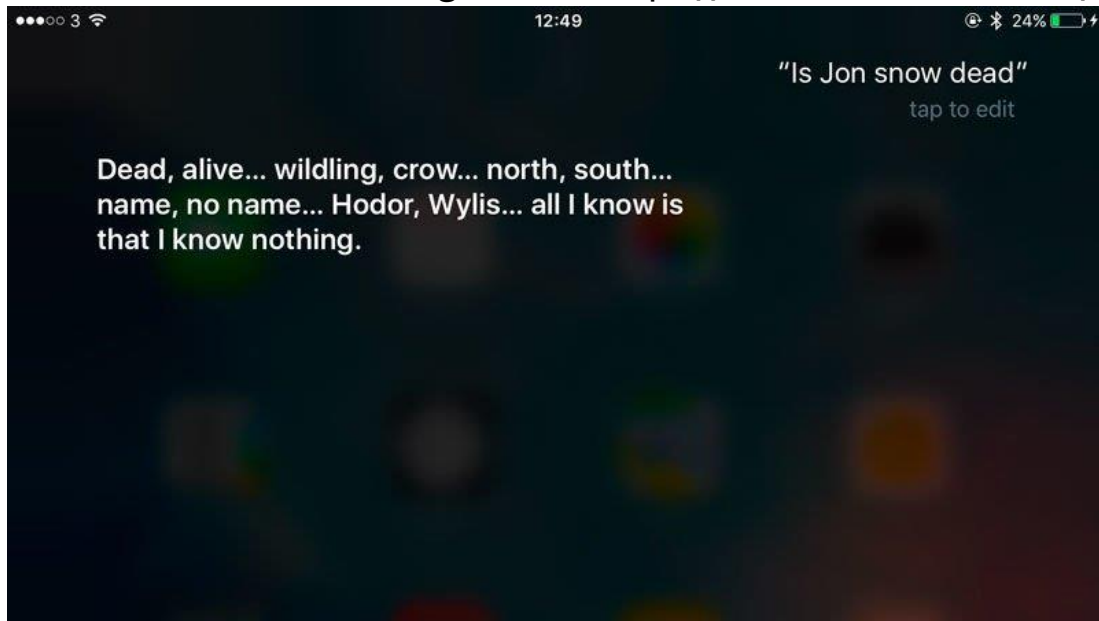
- **Speech to text**



# Машинный NLP - здесь и сейчас

- Dialogue generation
- Text to speech

Image credit: <https://www.macworld.co.uk/>





# Постановка задачи для машин

- Писменность - **последовательность ...**
- Устная речь - **последовательность ...**
- Язык жестов - **последовательность ...**

**Обработка естественного языка**  
**=**  
**обработка последовательностей**

# Последовательность имеет значение

- На уровне букв:  
*Она испугалась **тока**.*  
*Она испугалась **кота**.*
- На уровне слов:  
*Ему **серьёзно** нужно заниматься спортом.*  
*Ему нужно заниматься спортом **серьёзно**.*
- На уровне предложений:

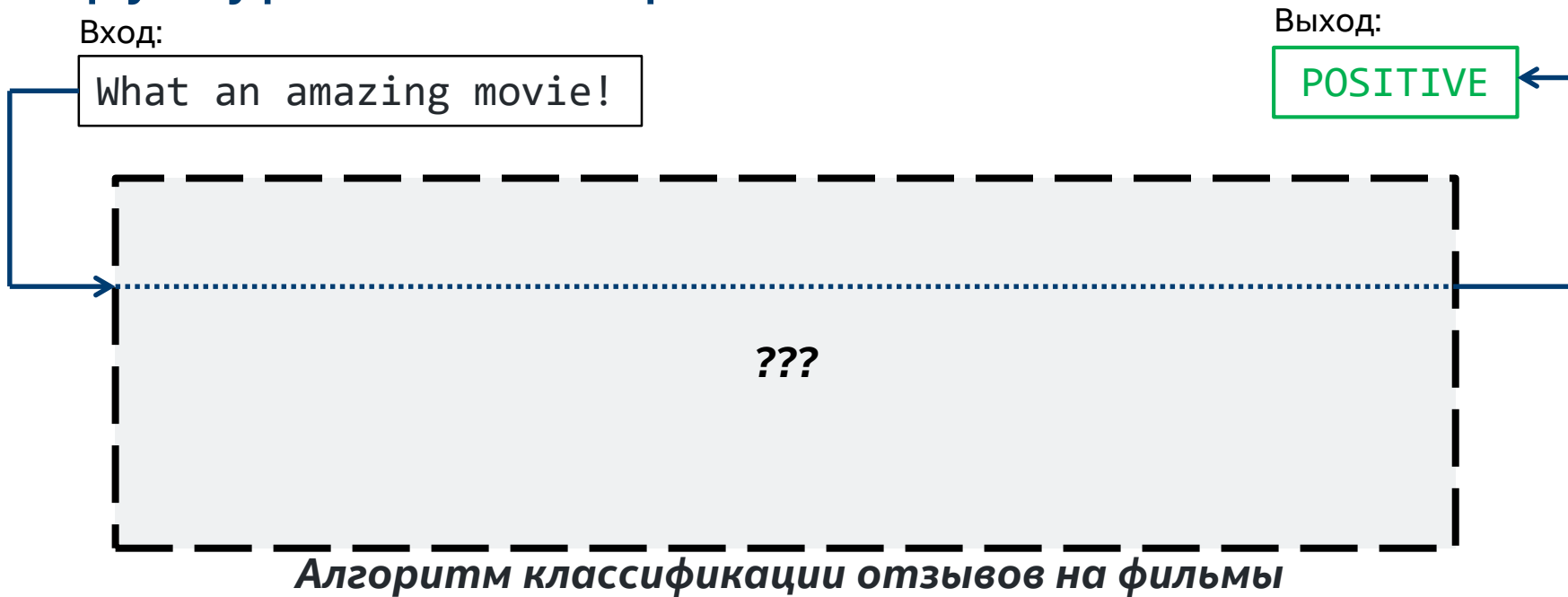
*В стране произошла вспышка гриппа. **Но обошлось лёгкой формой.***  
*Маша тоже заразилась. **Болезнь подкосила всю её жизнь.***

*В стране произошла вспышка гриппа. **Болезнь подкосила всю её жизнь.***  
*Маша тоже заразилась. **Но обошлось лёгкой формой.***

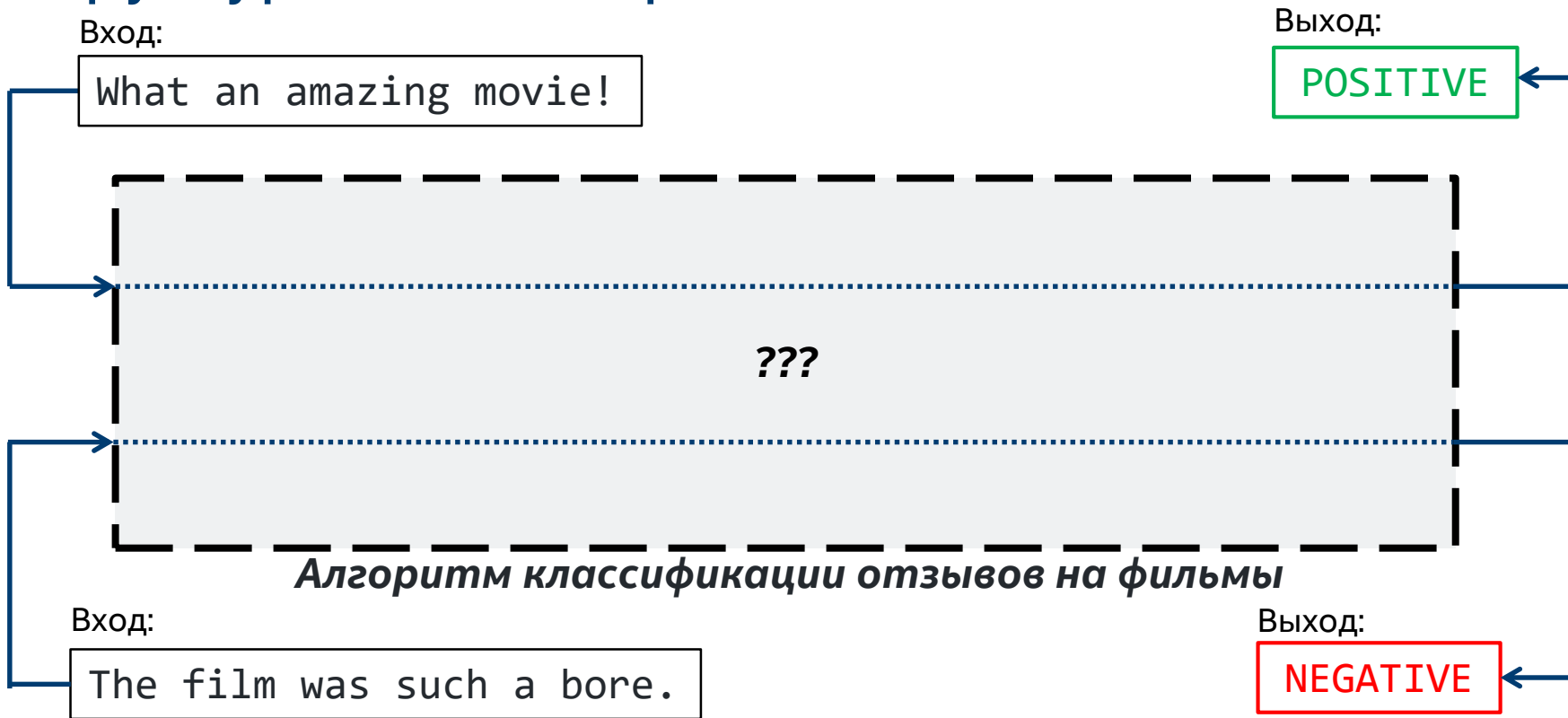
# Структура NLP-алгоритма



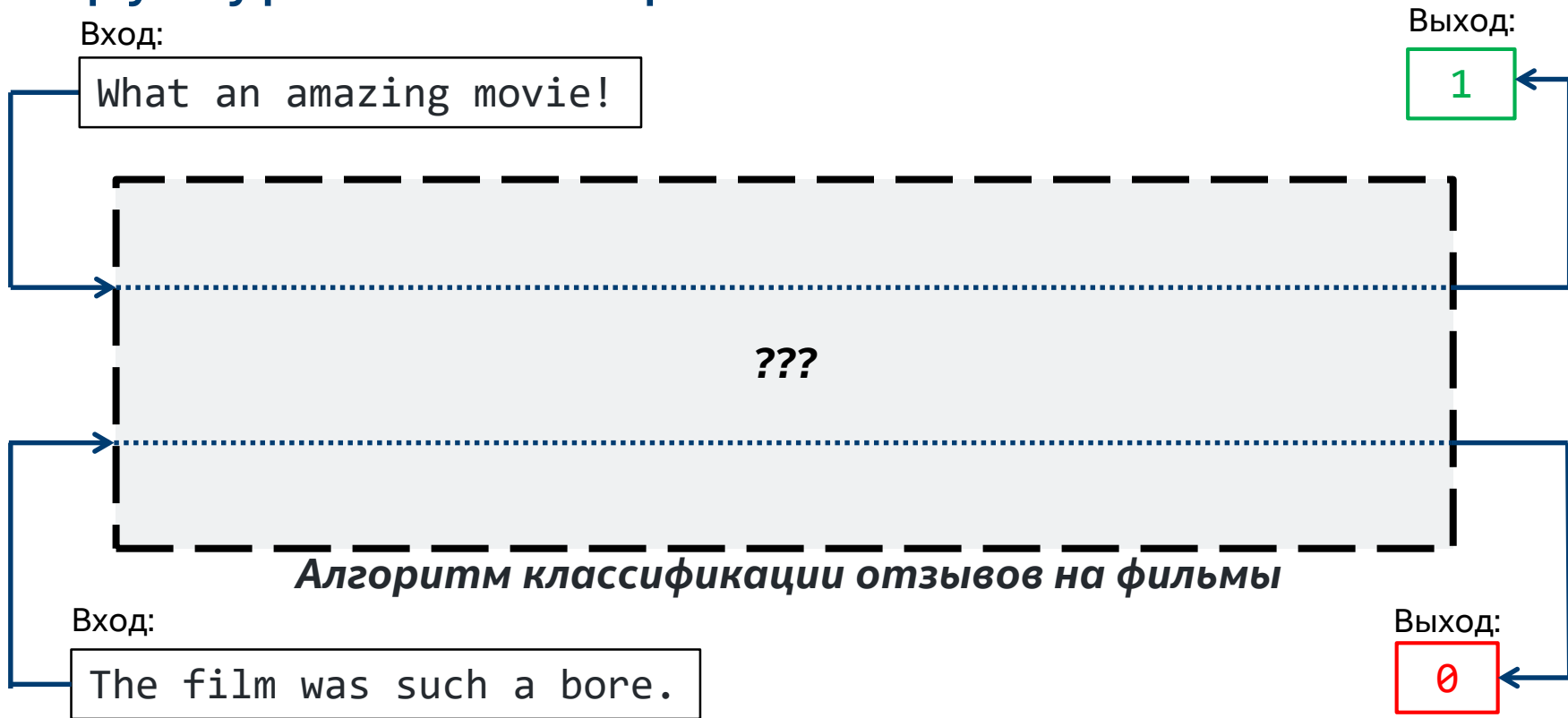
# Структура NLP-алгоритма



# Структура NLP-алгоритма




# Структура NLP-алгоритма



# Токенизация и словари

Вход:

What an amazing movie!

A blue line starts from the left side of the text box, goes down, and then turns right into an arrow pointing towards the right.

Как перевести текст в формат, пригодный для использования в алгоритме машинного/глубокого обучения?

## Токенизация:

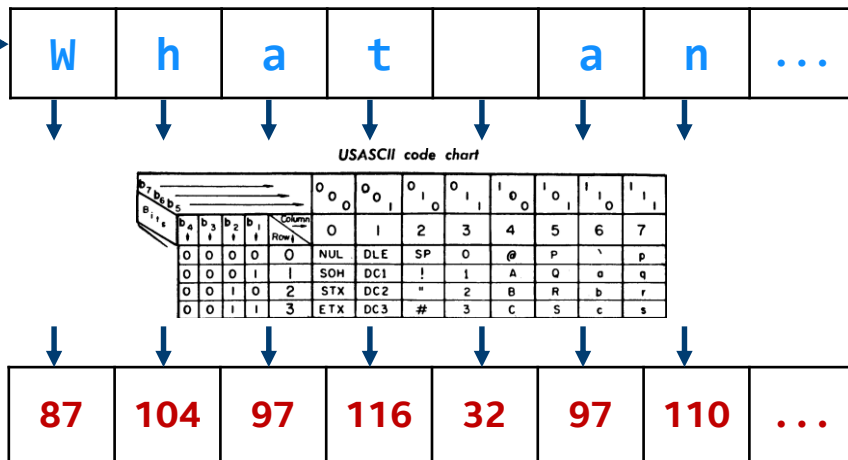
- Разбить текст на отдельные компоненты (токены)
- Каждому токenu присвоить уникальный идентификатор

# Токенизация и словари

Вход:

What an amazing movie!

**Вариант 1:** превращать текст в числа на уровне букв



Число идентификаторов: ~200 (ASCII)

Плюсы и минусы:

- + Сделано за нас кодировкой
- + Малый размер отдельного идентификатора (для алфавитных языков)
- Слишком большое число элементов в последовательности
- Отдельные буквы - локальный признак (нет смысла сравнивать буквы в разных словах)

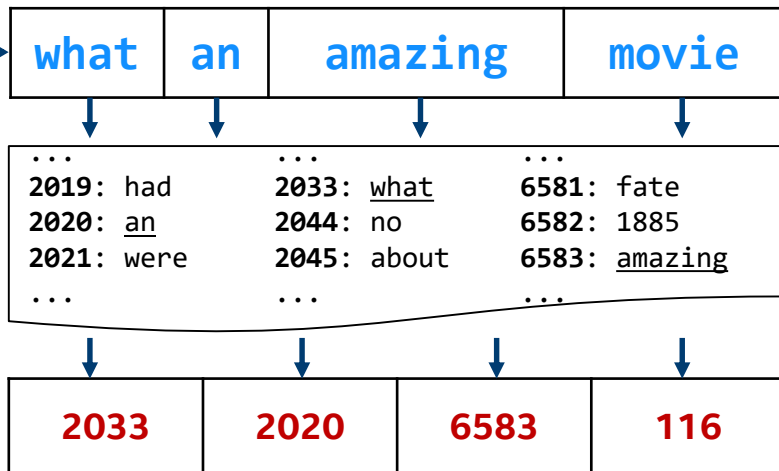


# Токенизация и словари

Вход:

What an amazing movie!

**Вариант 2:** превращать текст в числа на уровне слов - **словарь**



Число идентификаторов: ~**200000** (англ.)

Плюсы и минусы:

- + Длина последовательности для предложений меньше, чем при кодировке отдельных букв
- + Малый размер отдельного идентификатора
- Необходимо учитывать все словоформы (склонение, спряжение и т. д.)
- Пунктуацию необходимо учитывать наряду с обычными словами
- Нужно хранить словарь, в общем случае - свой для каждой задачи/языка

# Токенизация и словари

Вход:

What an amazing movie!

**Вариант 2а:** превращать текст в числа на уровне кусков слов

what an amaz ##ing movie

Словарь

...	...	...
2019: had	2033: <u>what</u>	6581: fate
2020: <u>an</u>	2044: no	6582: 1885
2021: were	2045: <u>##ing</u>	6583: <u>amaz</u>
...	...	...

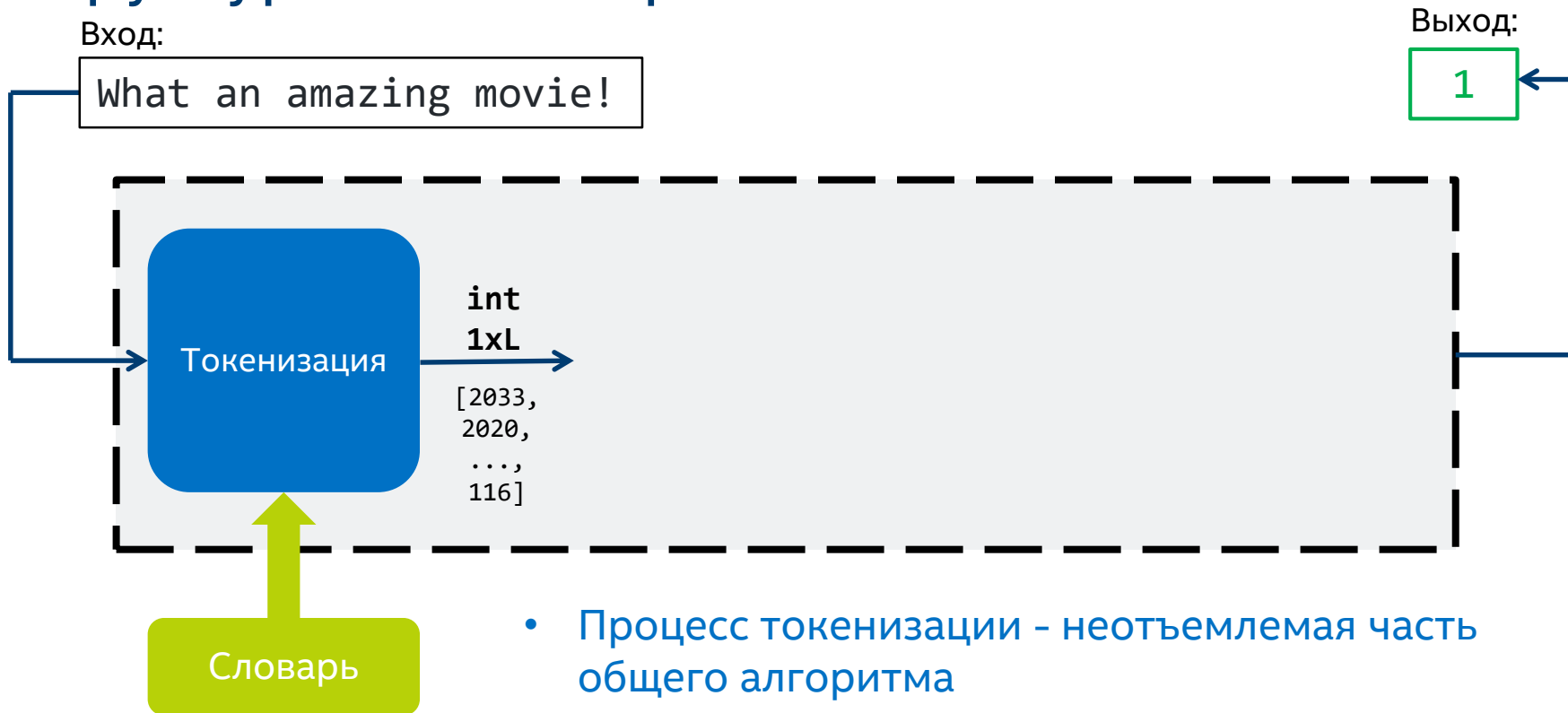
2033 2020 6583 2045 116

Число идентификаторов: ~30000 (англ.)

Плюсы и минусы:

- + Меньший размер словаря
- + Явное выделение морфем
- Большой эффективный размер последовательности
- Чуть более сложная предобработка

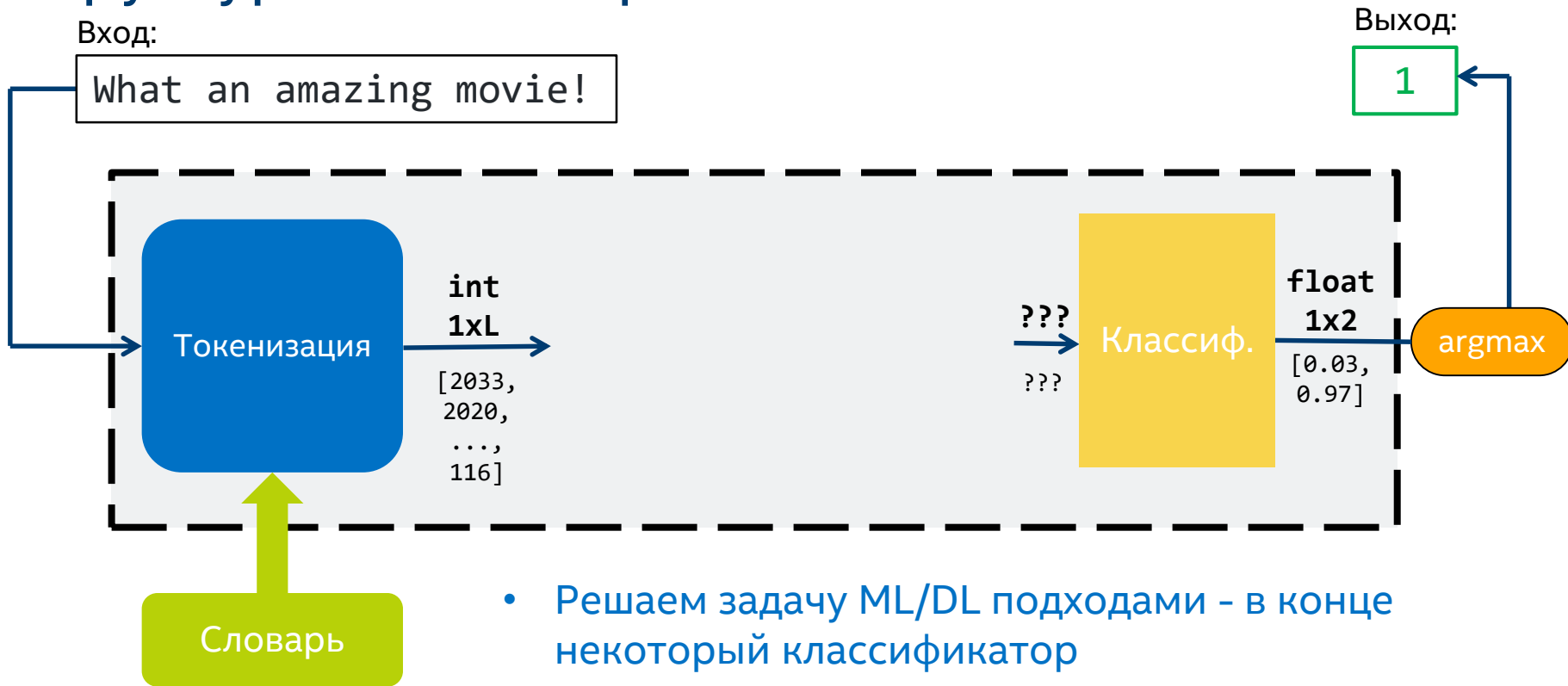
# Структура NLP-алгоритма



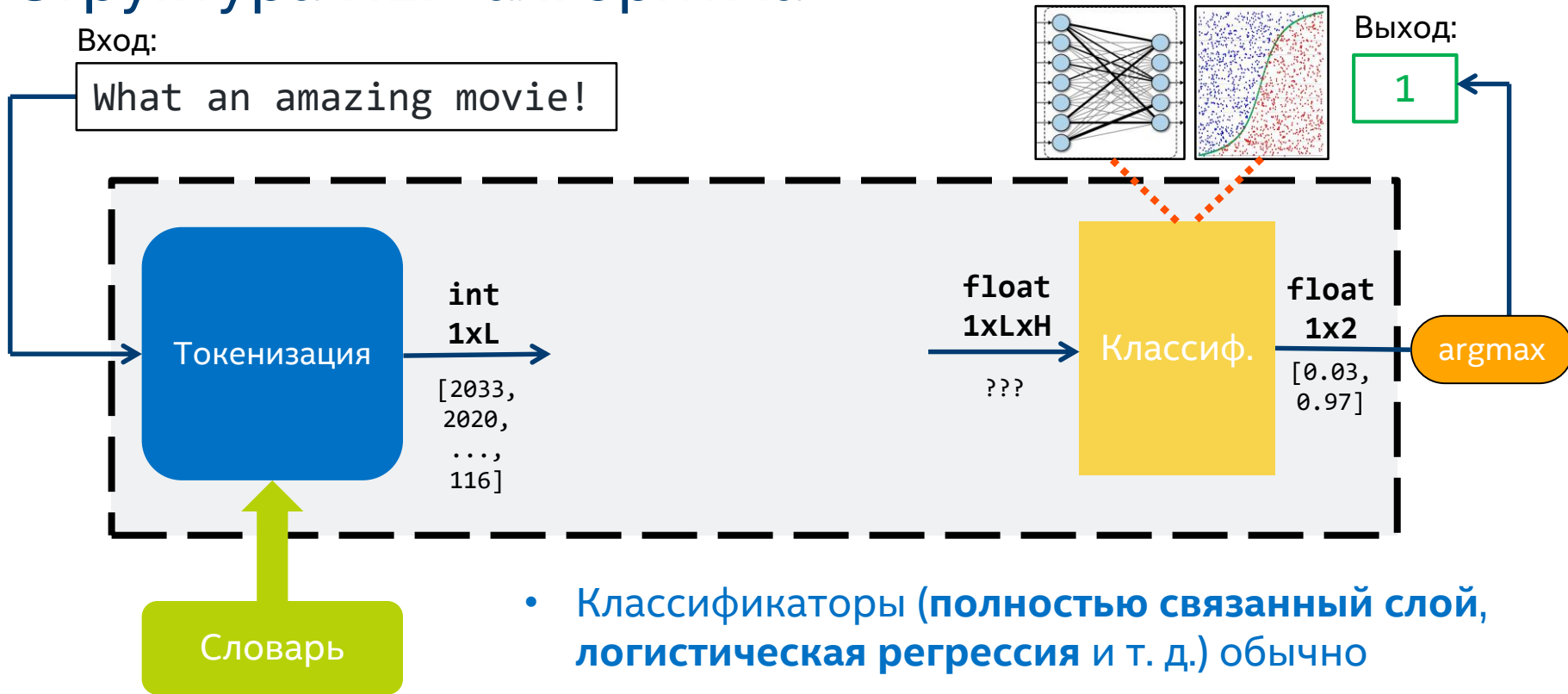
- Процесс токенизации - неотъемлемая часть общего алгоритма

*NB: L - эффективная длина последовательности после токенизации*

# Структура NLP-алгоритма

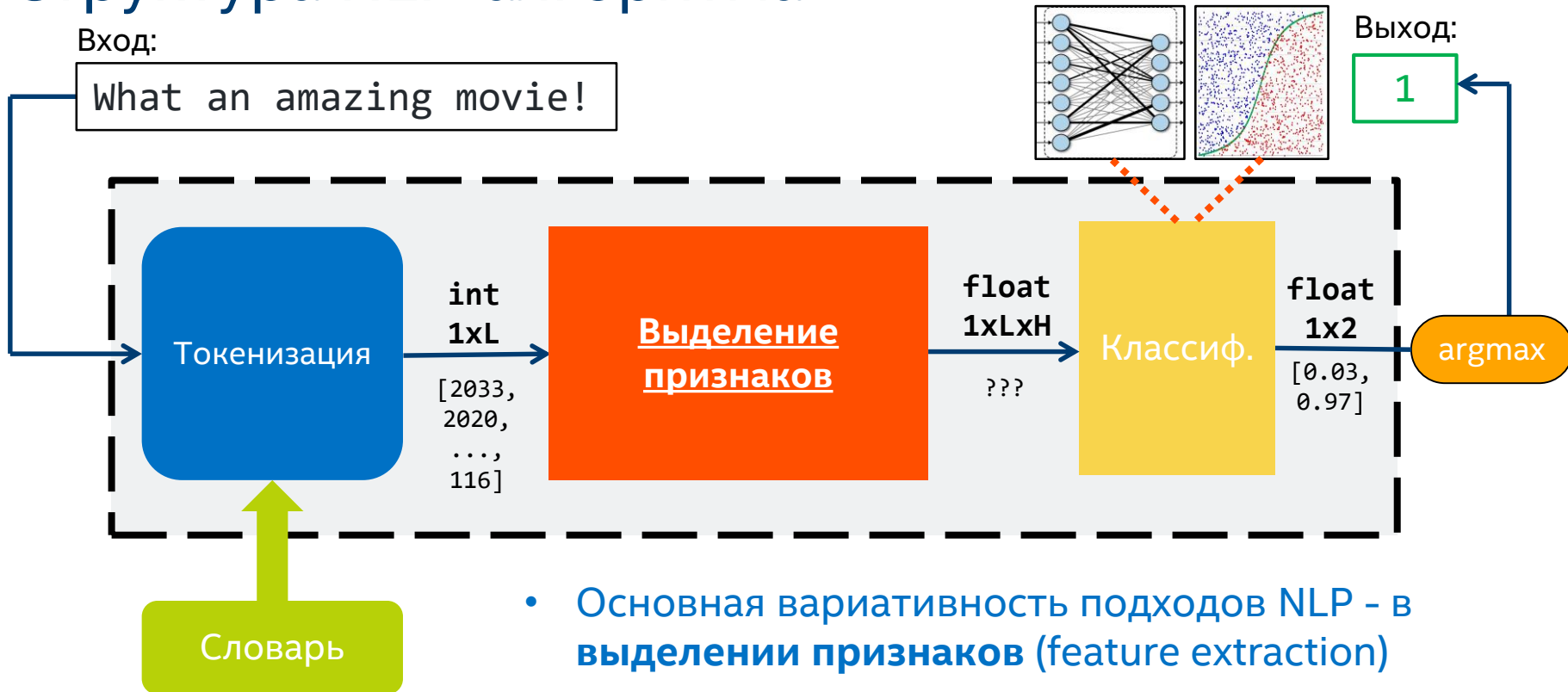


# Структура NLP-алгоритма



- Классификаторы (полностью связанный слой, логистическая регрессия и т. д.) обычно имеют вещественные входы - признаки


# Структура NLP-алгоритма



NB: На схеме прямоугольниками указаны обучаемые компоненты.

# Входные признаки - унитарный код

1xL int (L - длина последовательности)



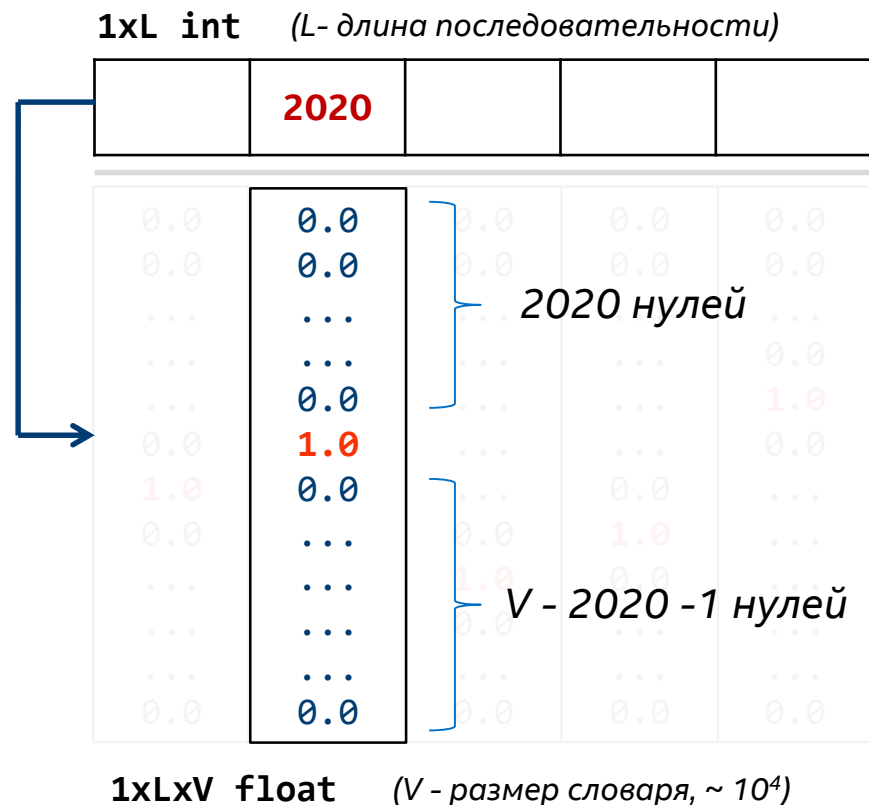
The diagram illustrates the process of converting a sequence of input features into a one-hot encoding matrix. A blue line starts from the left, passes above the first table, and then turns into an arrow pointing to the second table, indicating the transformation process.

2033	2020	6583	2045	116
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
...	...	...	...	...
...	...	...	...	0.0
...	0.0	...	...	1.0
0.0	1.0	...	...	0.0
1.0	0.0	...	0.0	...
0.0	...	0.0	1.0	...
...	...	1.0	0.0	...
...	...	0.0	...	...
...	...	...	...	...
0.0	0.0	0.0	0.0	0.0

1xLxV float (V - размер словаря,  $\sim 10^4$ )

- Входные признаки - категориальные (каждый идентификатор в словаре - отдельный класс) = целые числа
- ML/DL существенно использует операции с вещественными числами, а не целыми
- Стандартное решение - преобразование каждого целого числа в вектор вещественных чисел с помощью **унитарного кода** (one-hot encoding)

# Входные признаки - унитарный код

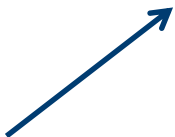


- Каждому  $i$ -му идентификатору ставится в соответствие  $V$ -мерный вектор, где  $V$  - число категорий (в нашем случае  $V$  = размер словаря)
- В  $V$ -мерном векторе  $i$ -ый элемент равен 1.0, а остальные равны нулю
- Унитарный код хорошо работает при не связанных между собой входных категориях



# Похожесть векторов

- Какой из векторов внизу наиболее похож на вектор справа?



# Похожесть векторов

- Какой из векторов внизу наиболее похож на вектор справа?



# Похожесть векторов

- Какой из векторов внизу наиболее похож на вектор справа?

$[7.3, 8.1]$

$[6.2, -4.3]$     $[-0.5, -0.7]$     $[6.7, 7.9]$     $[0.0, 1.3]$     $[-0.7, 0.0]$

# Похожесть векторов

- Какой из векторов внизу наиболее похож на вектор справа?

$[7.3, 8.1]$

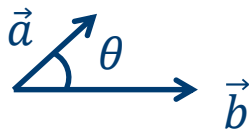
$[6.2, -4.3]$   $[-0.5, -0.7]$   $[6.7, 7.9]$   $[0.0, 1.3]$   $[-0.7, 0.0]$

# Похожесть векторов

- Эффективная метрика для векторов порождается скалярным произведением

$$(\vec{a} \cdot \vec{b}) = ab \cos \theta$$

$$(\vec{a} \cdot \vec{b}) = \sum a_i b_j$$



$$\vec{x}_0 = [7.3, 8.1]$$

$$\vec{x}_1$$

$$\vec{x}_2$$

$$\vec{x}_3$$

$$\vec{x}_4$$

$$\vec{x}_5$$

$$[6.2, -4.3]$$

$$[-0.5, -0.7]$$

$$[6.7, 7.9]$$

$$[0.0, 1.3]$$

$$[-0.7, 0.0]$$

$$(\vec{x}_0 \cdot \vec{x}_1) = 10.43$$

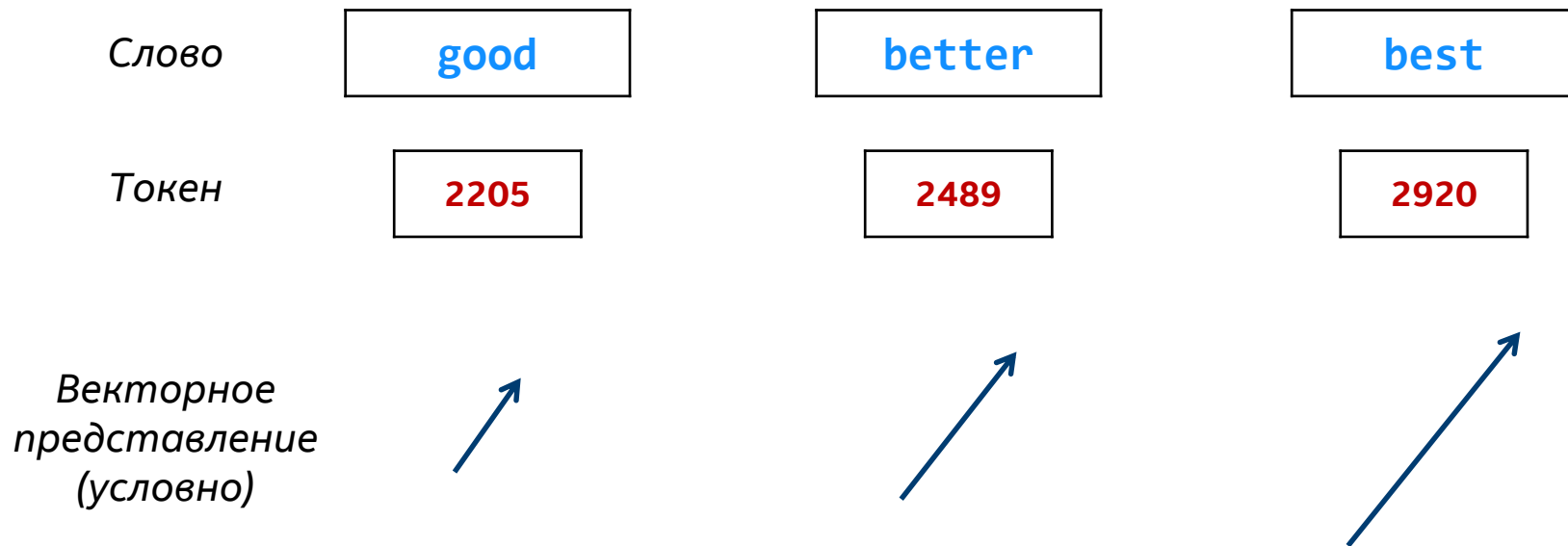
$$(\vec{x}_0 \cdot \vec{x}_3) = 112.90$$

$$(\vec{x}_0 \cdot \vec{x}_5) = -5.11$$

$$(\vec{x}_0 \cdot \vec{x}_2) = -9.32$$

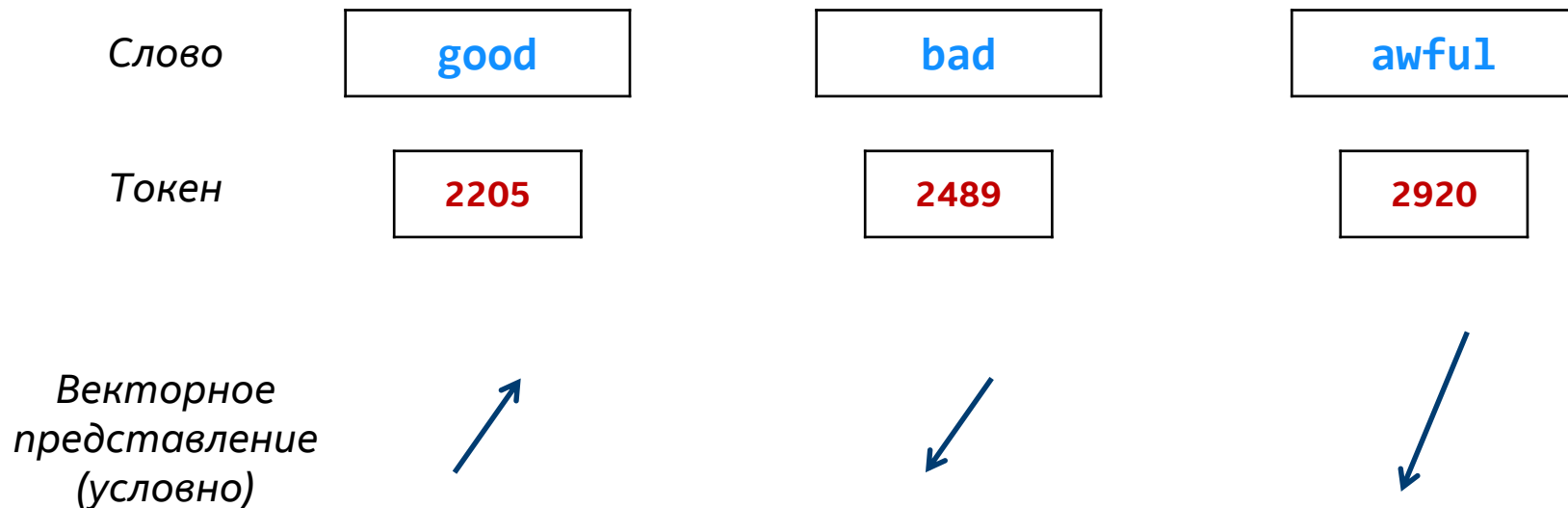
$$(\vec{x}_0 \cdot \vec{x}_4) = 10.53$$

# Похожесть векторов



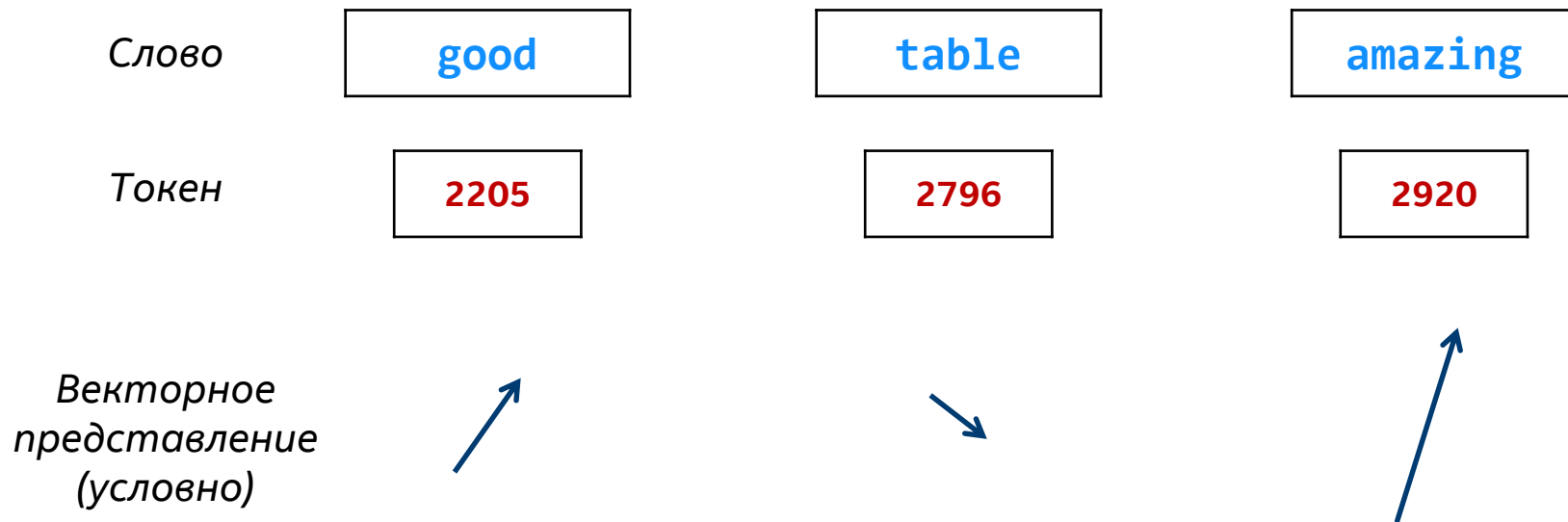
- «Близкие» (в некотором смысле) слова должны иметь большое положительное скалярное произведение друг с другом

# Похожесть векторов



- «Противоположные» (в некотором смысле) слова должны иметь большое отрицательное скалярное произведение

# Похожесть векторов




- «Не связанные» (в некотором смысле) слова должны иметь скалярное произведение, близкое к нулю



# Входные признаки - унитарный код

1xL int (L - длина последовательности)

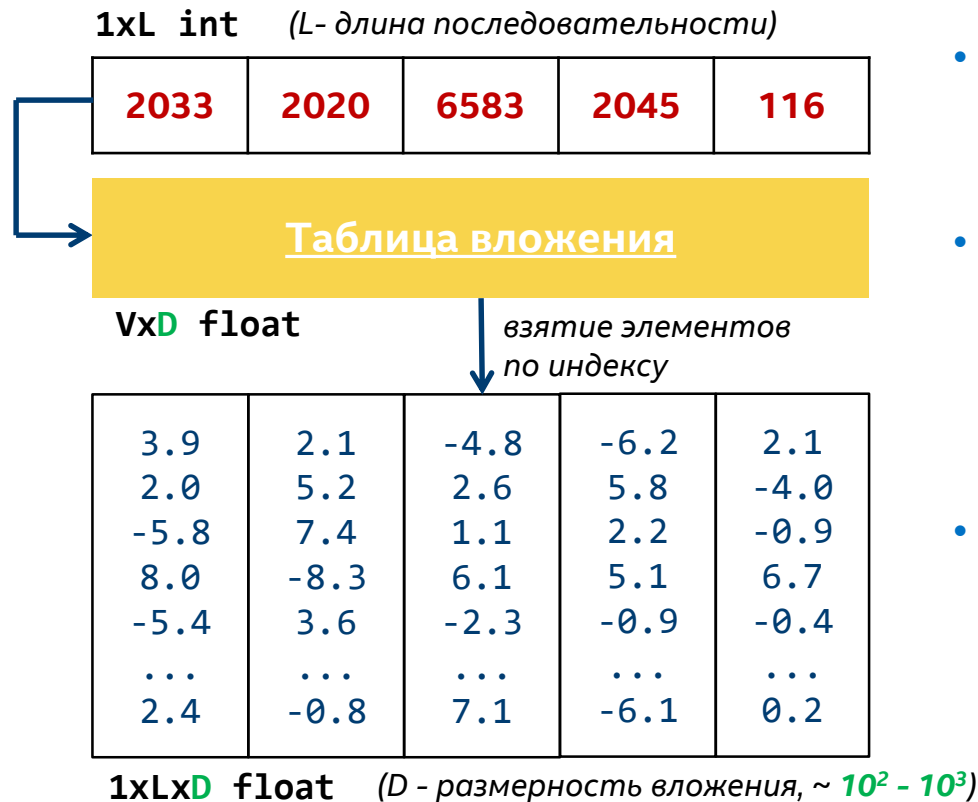


2033	2020	6583	2045	116
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
...	...	...	...	...
...	...	...	...	0.0
...	0.0	...	...	1.0
0.0	1.0	...	...	0.0
1.0	0.0	...	0.0	...
0.0	...	0.0	1.0	...
...	...	1.0	0.0	...
...	...	0.0	...	...
...	...	...	...	...
0.0	0.0	0.0	0.0	0.0

1xLxV float (V - размер словаря, ~ 10<sup>4</sup>)

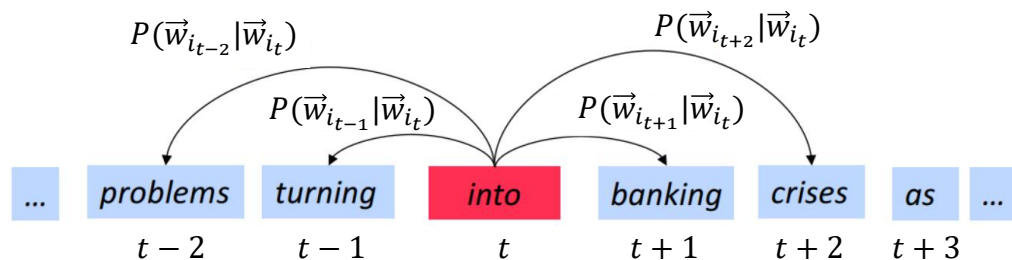
- Унитарный код совершенно не подходит для использования с метрикой в виде скалярного произведения
- Размерность векторов слишком велика и избыточна (десятки тысяч)
- Нет никакой возможности установить близость слов

# Входные признаки - вложения



- **Вложения** (embedding) - более общая процедура соотнесения вектора со словом
- В общем случае, если входные идентификаторы принимают значения  $[0, 1, \dots, V - 1]$  - это просто таблица из V векторов произвольного вида (но заданной размерности D)
- Являются обучаемым параметром - нужно обучать вместе с остальной моделью, либо в качестве отдельной задачи

# Обучение вложений - Word2Vec



$$\rho(V) = \prod_{t=0}^{T-1} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(\vec{w}_{t+j}|\vec{w}_t; V) \rightarrow \max$$

$$L'(V) = -\ln L(V) = -\frac{1}{T} \sum_{t=0}^{T-1} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \ln P(\vec{w}_{t+j}|\vec{w}_t; V) \rightarrow \min$$

- Обучение - обычно без учителя
- Векторы вложения подбираются так, чтобы максимизировать вероятность появления окружающих слов возле некоторого «центрального»

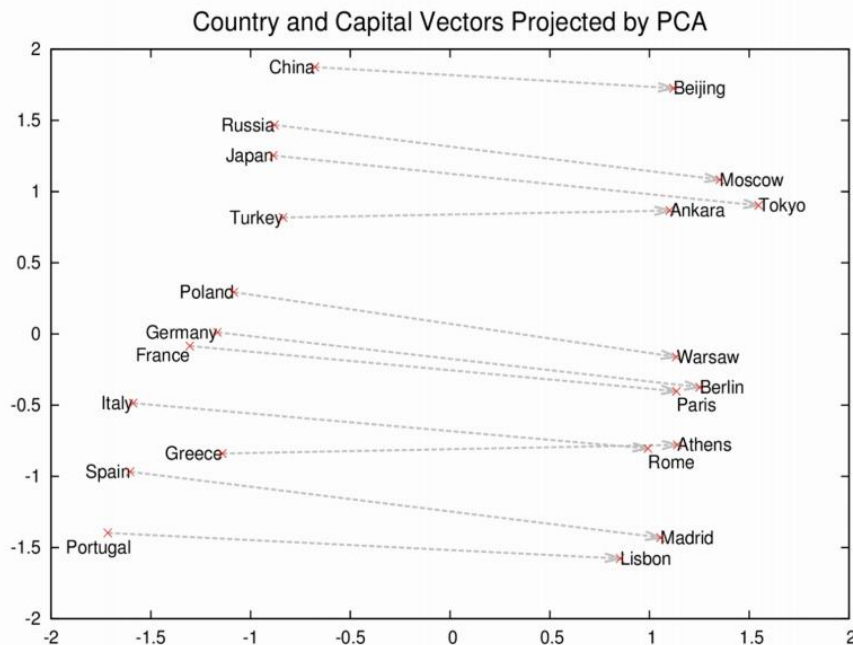
$$P(\vec{w}_{t+j}|\vec{w}_t; V) = \frac{e^{\vec{w}_{t+j} \cdot \vec{w}_t}}{\sum_{k \in V} e^{\vec{w}_k \cdot \vec{w}_t}}$$

Image credit: <http://web.stanford.edu/class/cs224n/>

$w_i$  - вектор вложения

$V$  - таблица вложений (= все векторы вложений для словаря)

# Обучение вложений - Word2Vec



$$\vec{w}_{Moscow} - \vec{w}_{Russia} = \vec{X},$$

$$\vec{w}_{China} + \vec{X} = \vec{w}_{Beijing}?$$

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Таблица: Пары слов с одинаковой векторной разностью в обученном вложении Word2Vec

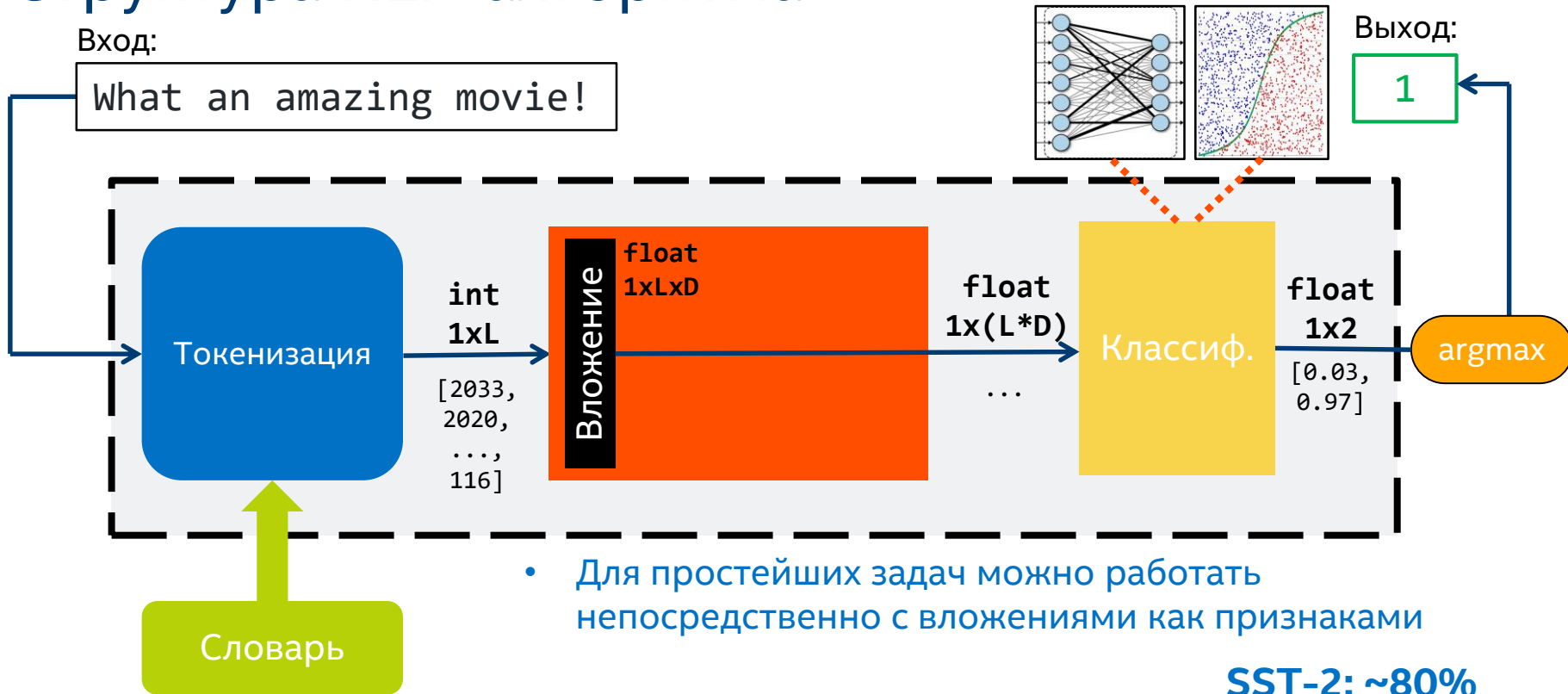
Sources:

<https://pathmind.com/wiki/word2vec>

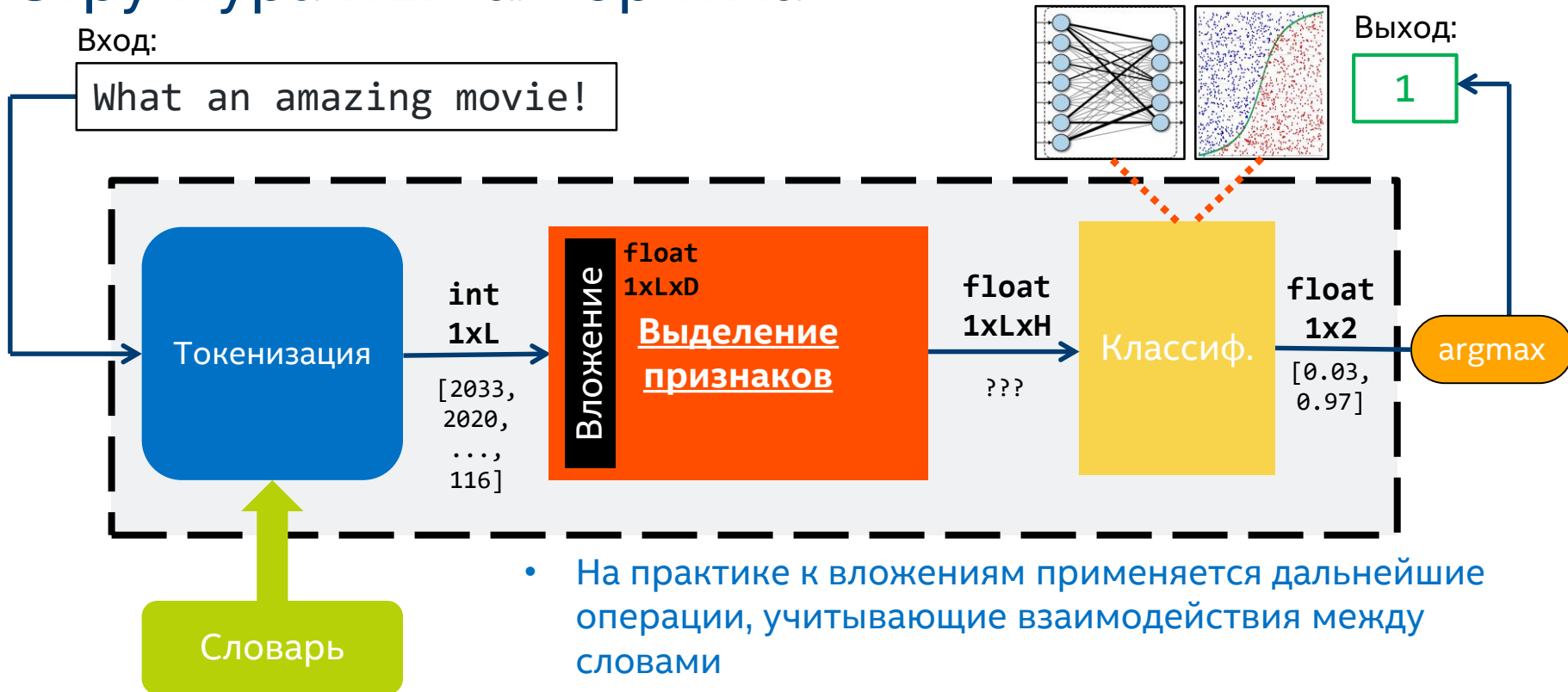
<https://arxiv.org/pdf/1301.3781.pdf>

- Отслеживает закономерности между словами и их векторами вложений на уровне линейной алгебры!

# Структура NLP-алгоритма



# Структура NLP-алгоритма



NB: На схеме прямоугольниками указаны обучаемые компоненты.

# Свёрточные сети для NLP

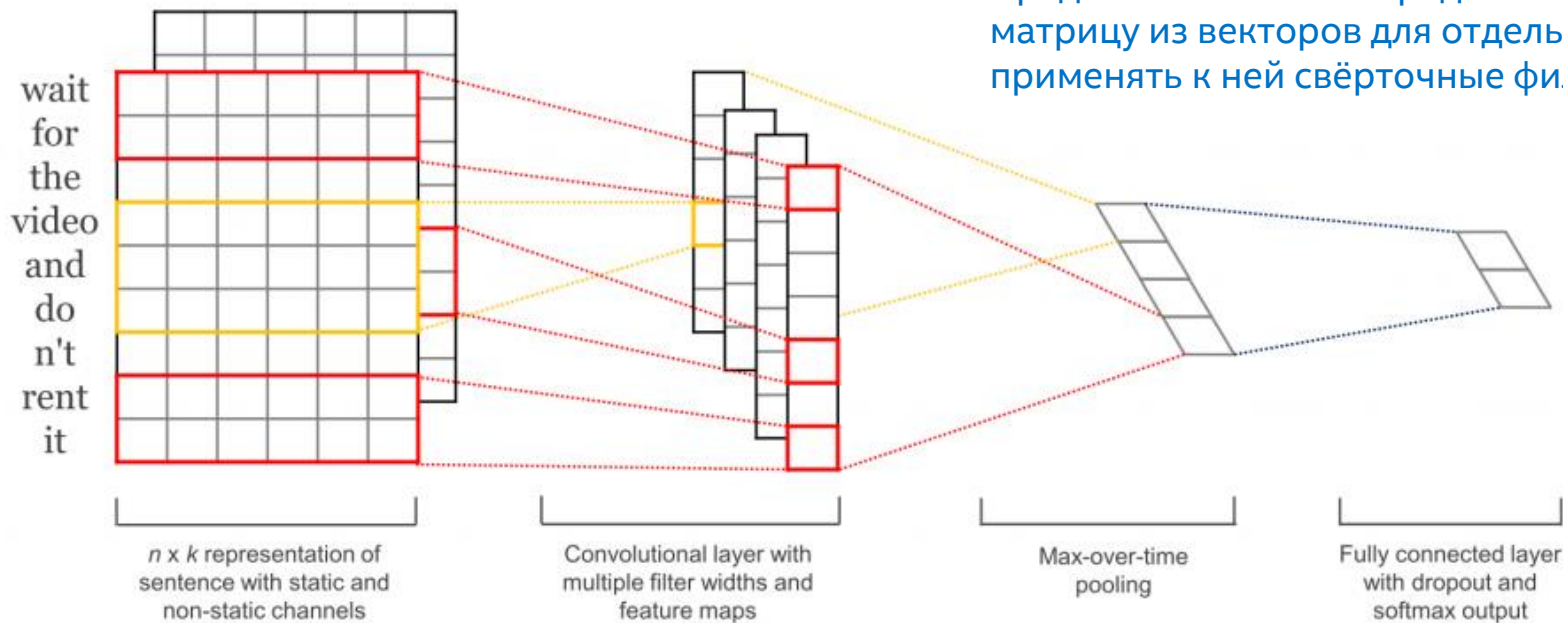
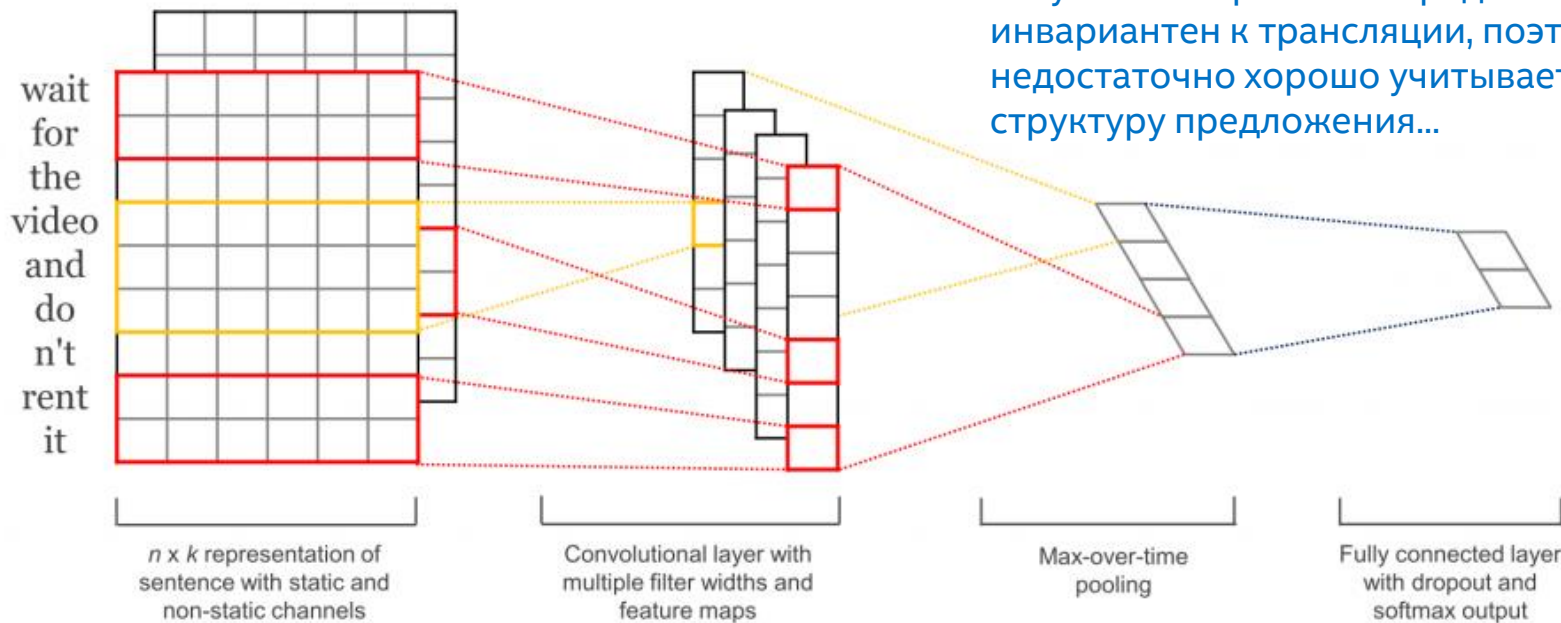


Image credit: Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification

# Свёрточные сети для NLP



- Результат свёртки по определению инвариантен к трансляции, поэтому также недостаточно хорошо учитывает глобальную структуру предложения...

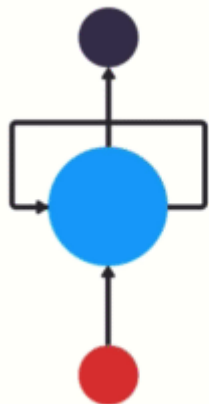
Image credit: Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification

**SST-2: до 88.1%**



# Рекуррентные сети для NLP

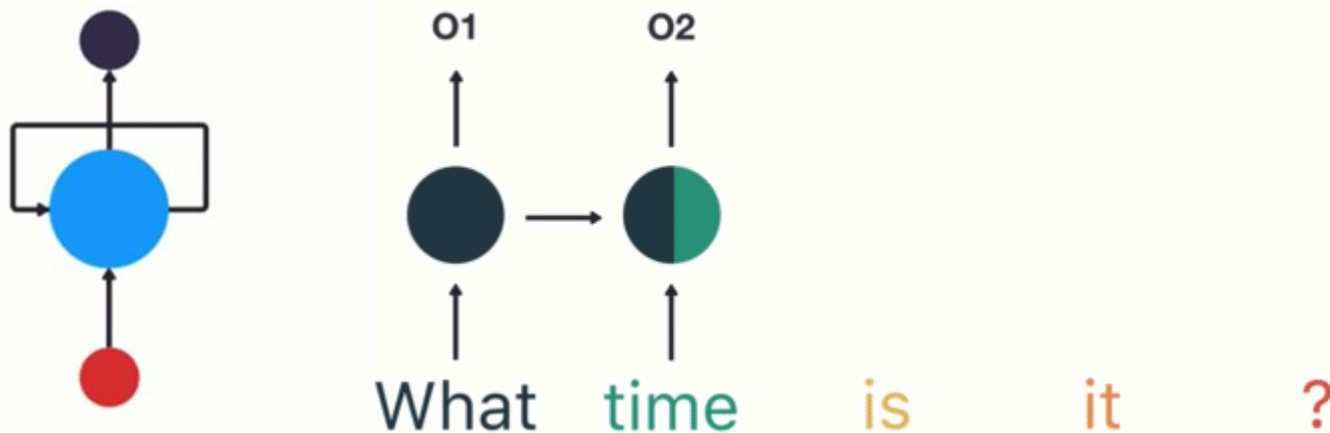
Image credit:  
*Illustrated Guide to Recurrent Neural Networks, M. Phi*



- Рекуррентные сети используют один и тот же набор весов, но для разных входов
- У рекуррентного слоя обычно два выхода - основной и «скрытое состояние» -, и два входа - основной вход и «скрытое состояние», полученное во время обработки предыдущего входа

# Рекуррентные сети для NLP

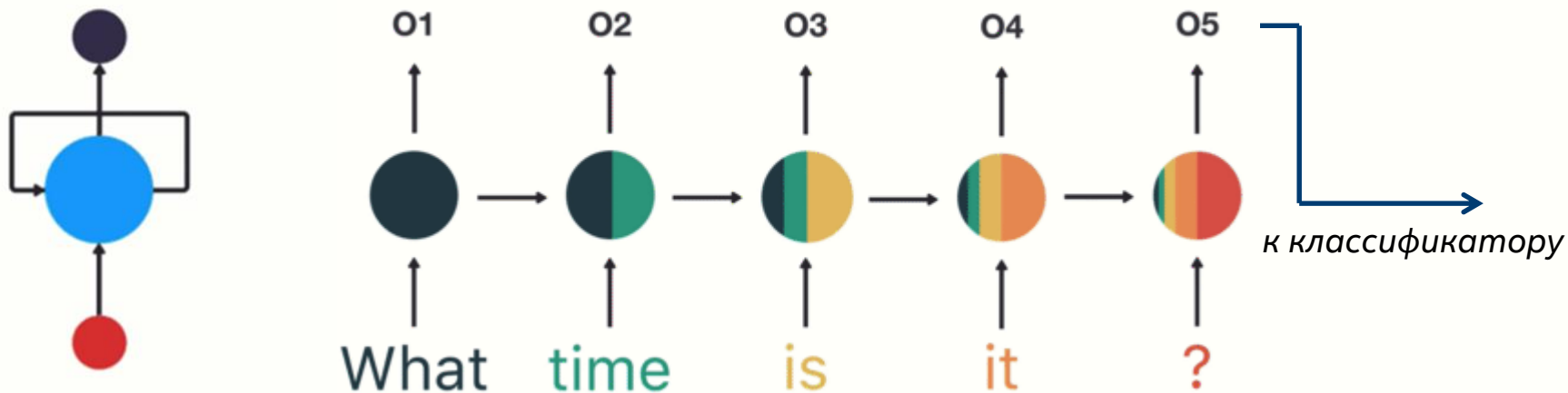
Image credit:  
*Illustrated Guide to Recurrent Neural Networks, M. Phi*



- Выход каждой итерации строится с учетом текущего «скрытого состояния», а значит, учитывает все предыдущие входы

# Рекуррентные сети для NLP

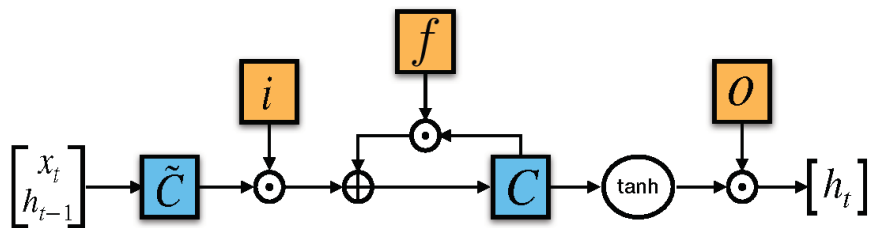
Image credit:  
*Illustrated Guide to Recurrent Neural Networks, M. Phi*



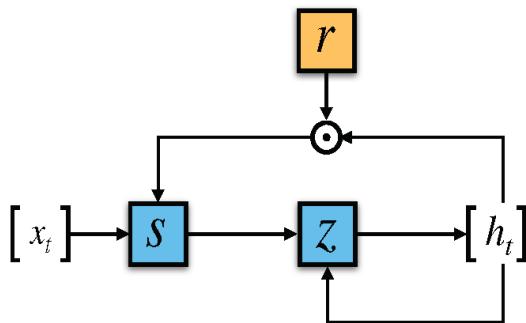
- Выход последней итерации можно использовать в качестве признака всей последовательности

# Рекуррентные сети для NLP

Image credit:  
<https://arxiv.org/abs/1412.3555>



(1) Long Short-Term Memory



(2) Gated Recurrent Unit

- LSTM и GRU - наиболее популярные архитектуры сверточных слоев
- Помогают избавиться от эффектов «затухающего» или «расходящегося» градиентов
- Теоретически могут обрабатывать сколь угодно большие последовательности без увеличения количества обучаемых параметров

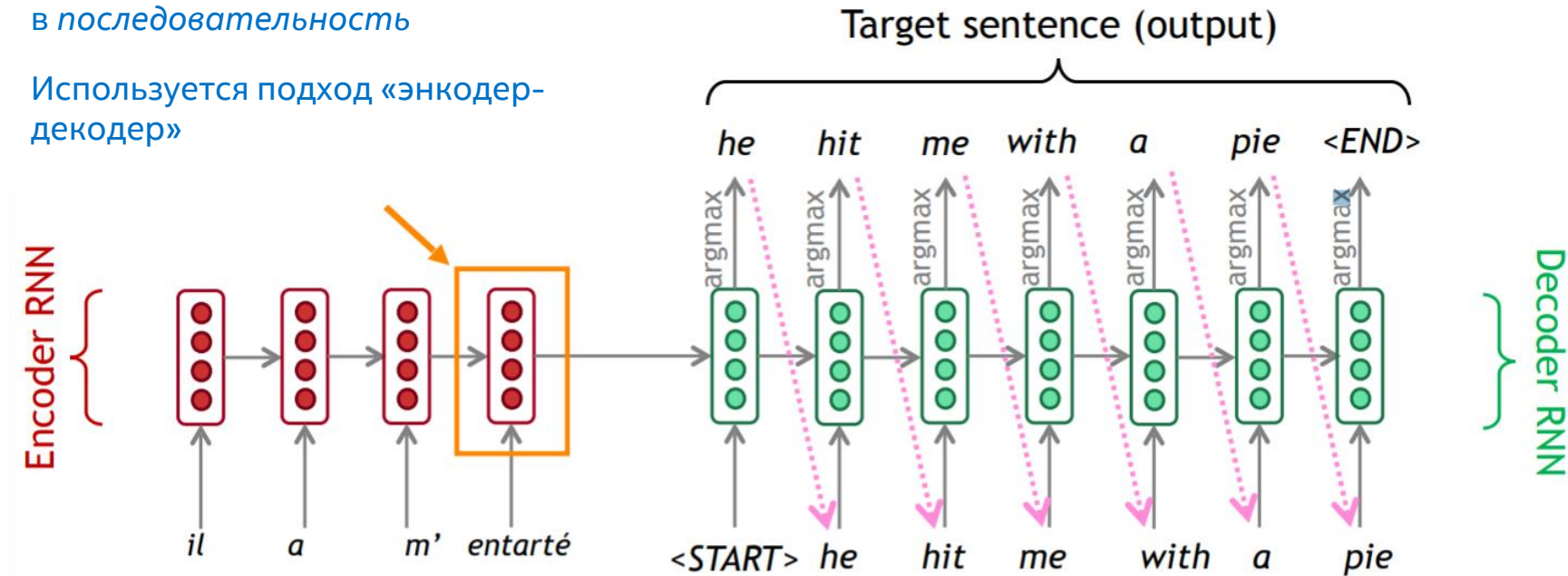
**SST-2: до 91.3%**

# Машинный перевод с помощью RNN - NMT

- Перевод с одного языка на другой - задача перевода *последовательности в последовательность*
- Используется подход «энкодер-декодер»

Image credit:

<http://web.stanford.edu/class/cs224n>

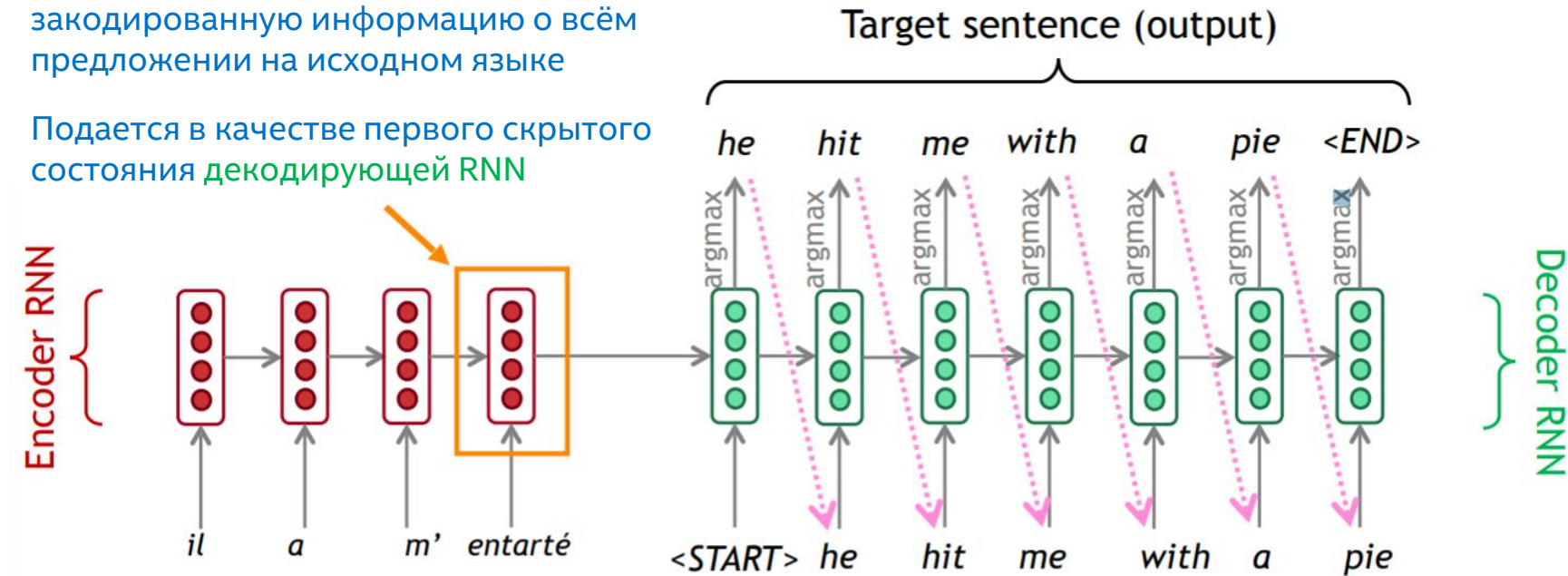


# Машинный перевод с помощью RNN - NMT

Image credit:

<http://web.stanford.edu/class/cs224n>

- Последнее скрытое состояние кодирующей RNN содержит в себе закодированную информацию о всём предложении на исходном языке
- Подается в качестве первого скрытого состояния декодирующей RNN

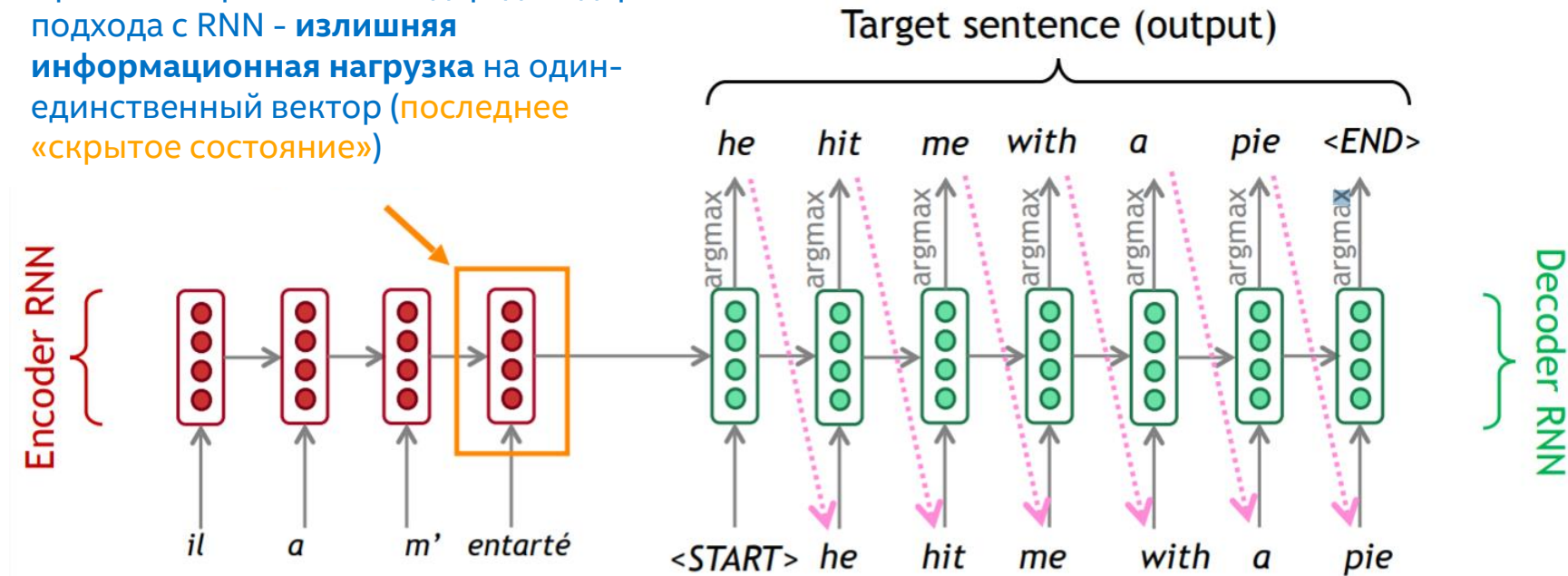


# Механизм внимания (attention)

Image credit:

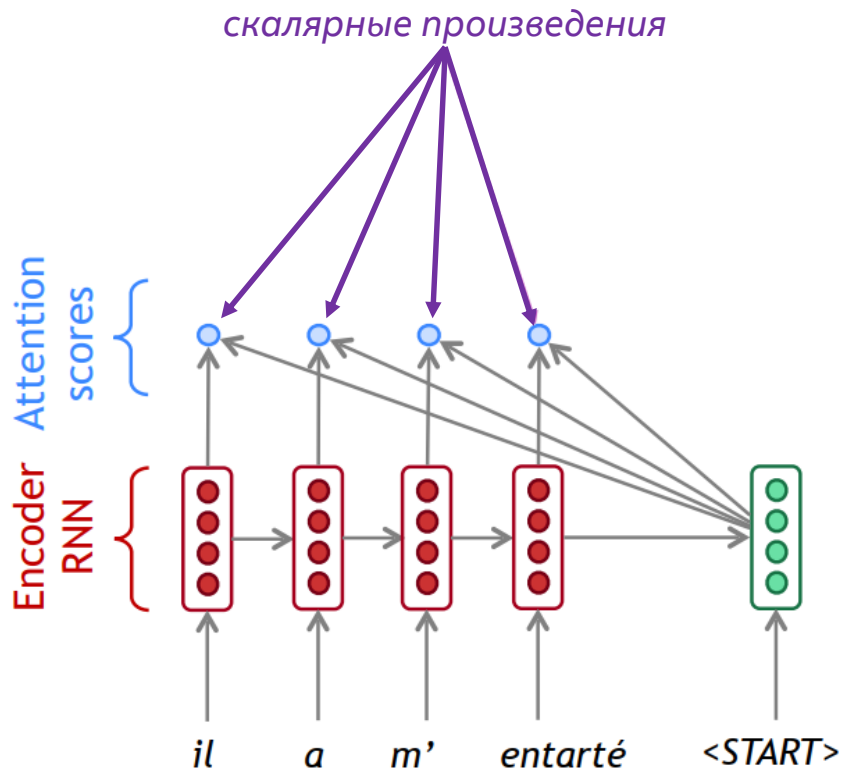
<http://web.stanford.edu/class/cs224n>

- Проблема простого энкодер-декодер подхода с RNN - **излишняя информационная нагрузка** на единственный вектор (последнее «скрытое состояние»)



# Механизм внимания (attention)

Image credit:  
<http://web.stanford.edu/class/cs224n>

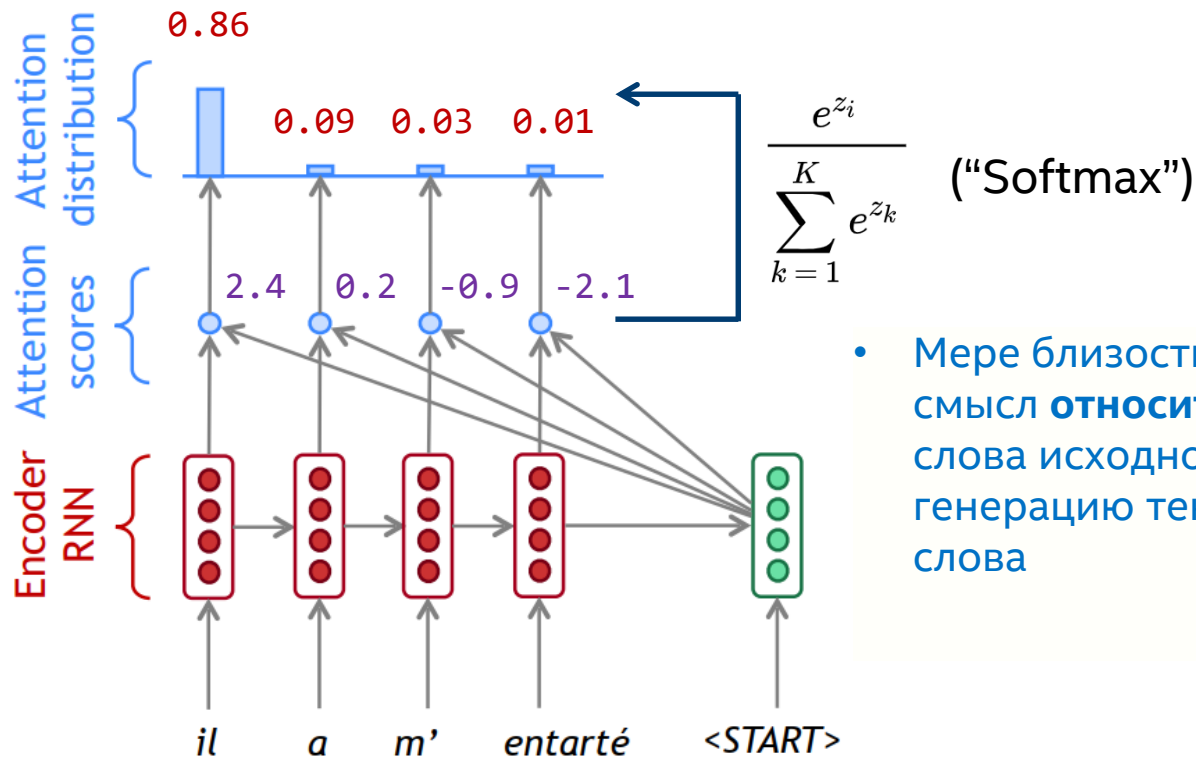


- На каждом шаге декодирования будем сравнивать текущее скрытое состояние **декодера** с каждым из выходных состояний **энкодера**
- Мера «близости» - **скалярное произведение**



# Механизм внимания (attention)

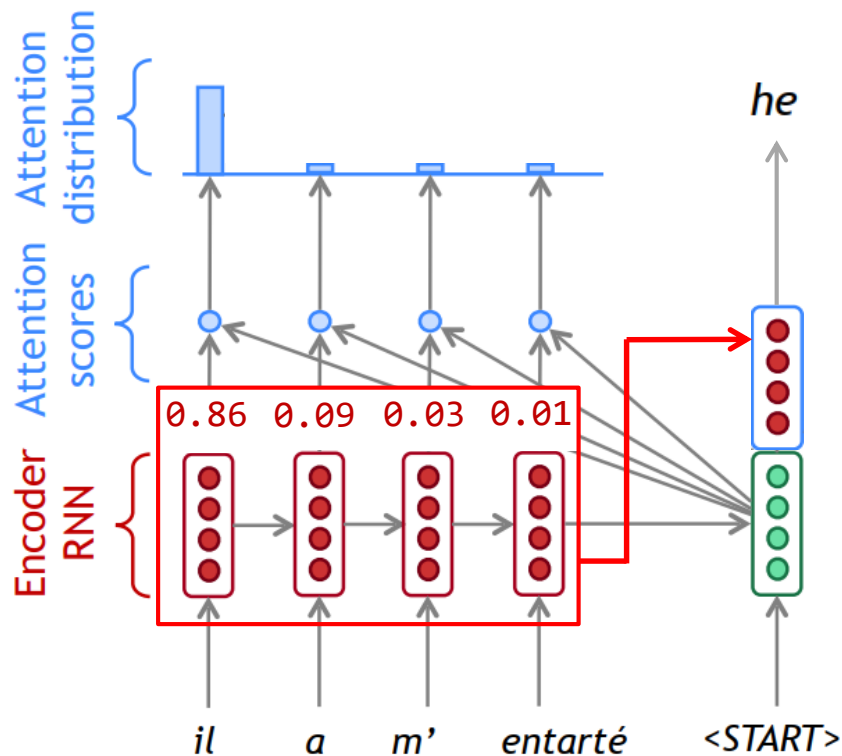
Image credit:  
<http://web.stanford.edu/class/cs224n>



- Мере близости можно приписать смысл **относительного вклада** каждого слова исходного предложения в генерацию текущего переведенного слова

# Механизм внимания (attention)

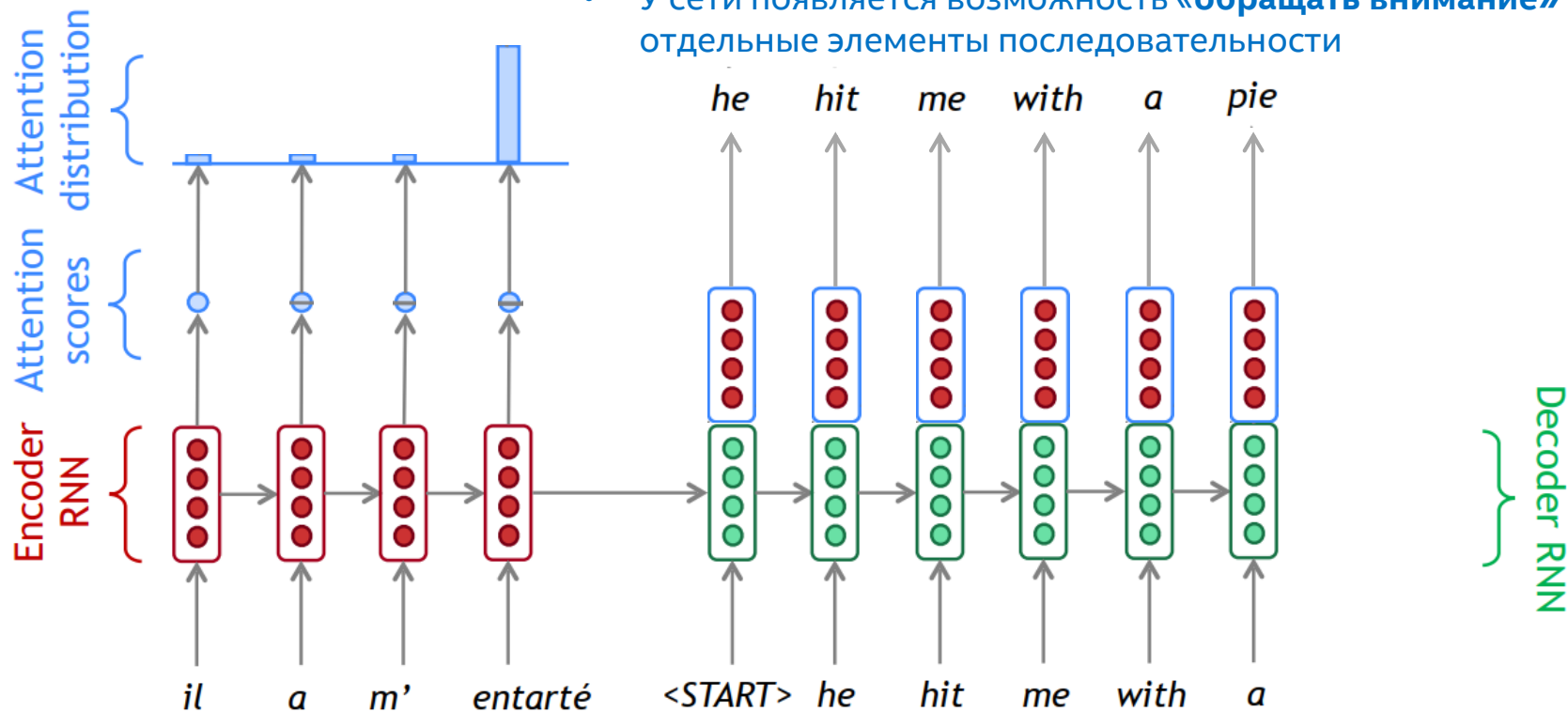
Image credit:  
<http://web.stanford.edu/class/cs224n>



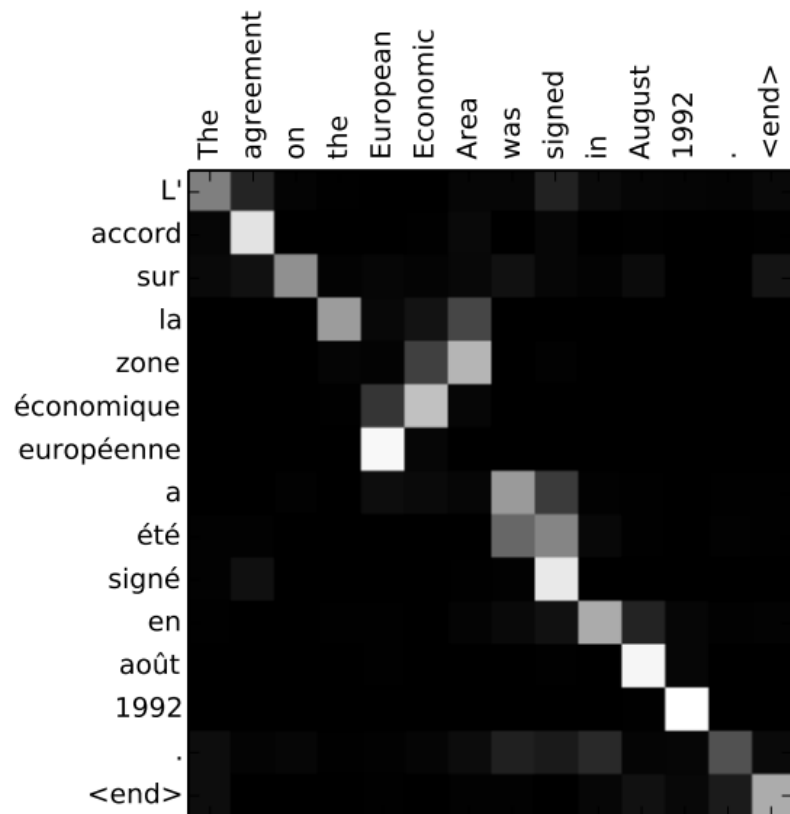
- Взвешенная (согласно мере близости) сумма всех «скрытых состояний» энкодера добавляется или конкатенируется с текущим «скрытым состоянием» декодера
- Большой вклад в перевод текущего слова внесет то исходное слово, чей вклад согласно мере близости оказался больше других

# Механизм внимания (attention)

- У сети появляется возможность «**обращать внимание**» на отдельные элементы последовательности



# Механизм внимания (attention)



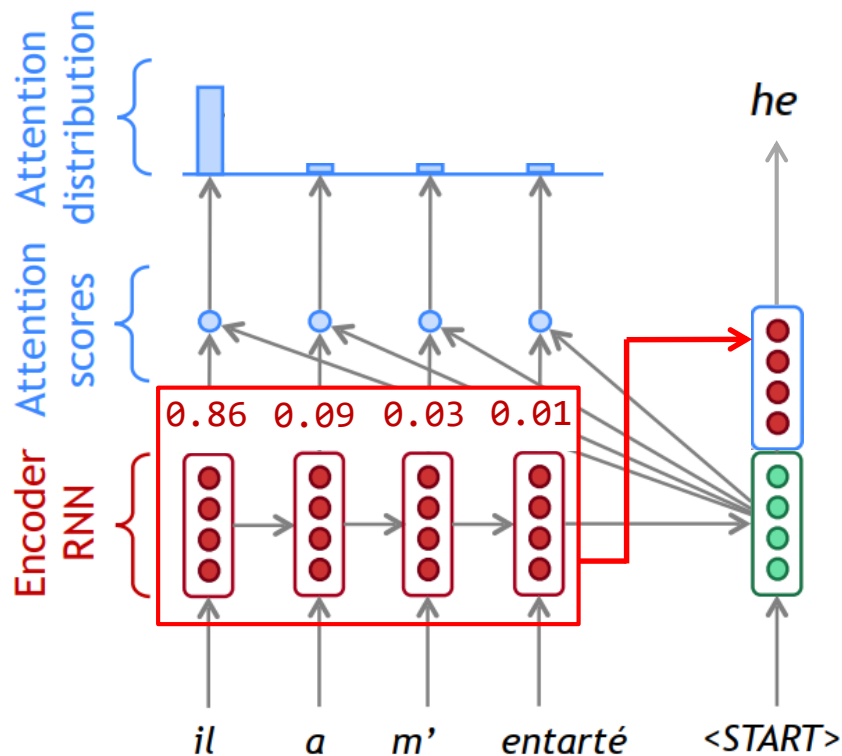
- Механизм внимания добавляет интерпретируемости в процесс перевода
- Сеть автоматически устанавливает соответствия между входными и выходными словами, где бы они ни находились в предложении
- **Механизм внимания оказывается гораздо более общим, чем задача перевода или даже NLP в целом**

Image credit:  
<https://arxiv.org/pdf/1409.0473.pdf>

# Недостатки RNN + attention

Image credit:

<http://web.stanford.edu/class/cs224n>



- Использование механизма внимания приводит к необходимости хранения всех промежуточных состояний энкодера
- При этом теряется достоинство чистых RNN - небольшое количество параметров, хранимых в памяти
- Вычисления RNN существенно непараллелизуемы
- В промежуточных состояниях энкодера сохраняется информационная асимметрия относительно начала и конца предложения

# «Нужно лишь внимание»

---

## Attention Is All You Need

---

**Ashish Vaswani\***  
Google Brain  
avaswani@google.com

**Noam Shazeer\***  
Google Brain  
noam@google.com

**Niki Parmar\***  
Google Research  
nikip@google.com

**Jakob Uszkoreit\***  
Google Research  
usz@google.com

**Llion Jones\***  
Google Research  
llion@google.com

**Aidan N. Gomez\* †**  
University of Toronto  
aidan@cs.toronto.edu

**Łukasz Kaiser\***  
Google Brain  
lukaszkaizer@google.com

**Illia Polosukhin\* †**  
illia.polosukhin@gmail.com

# Transformer-сети

Image credit:

<https://arxiv.org/pdf/1706.03762.pdf>

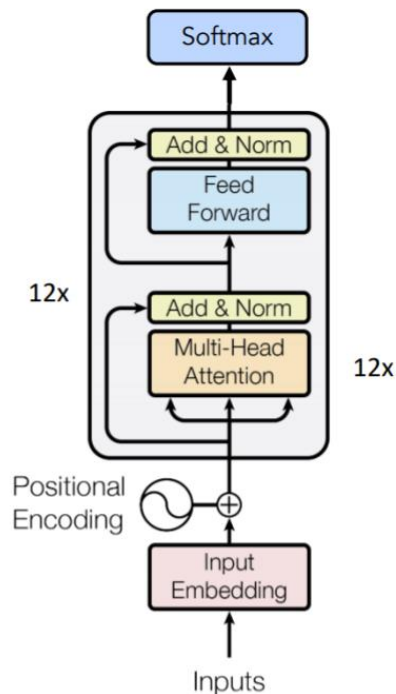


Рис.: Типичная структура энкодера в transformer-сетях

- В transformer-сетях происходит полный отказ от рекуррентности
- Все элементы последовательности обрабатываются одновременно
- Механизм внимания применяется не только между декодером и энкодером, но также и внутри энкодера - «**self-attention**»
- Применение механизма внимания сводится к матричным умножениям:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Transformer-сети

Scaled Dot-Product Attention

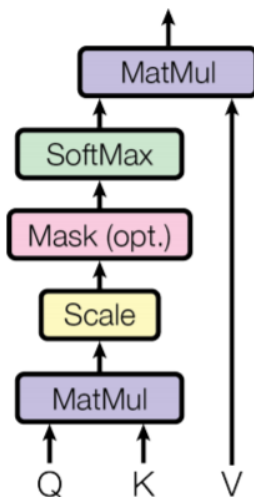


Image credit:  
<https://arxiv.org/pdf/1706.03762.pdf>

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

## Пример self-attention:

- **Q** – 2D-матрица  $L \times D$  из векторов вложений для каждого слова из предложения
- **K** – та же самая матрица

Тогда:

- **$QK^T$**  – матрица из скалярных произведений каждого вектора вложения с каждым другим;  $(i,j)$  элемент соответствует близости  $i$ -го вектора вложения к  $j$ -му вектору



# Transformer-сети

Scaled Dot-Product Attention

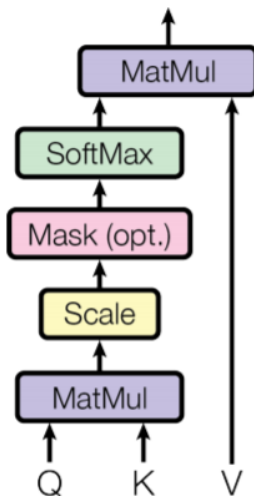


Image credit:  
<https://arxiv.org/pdf/1706.03762.pdf>

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- **$\text{softmax}(QK^T)V$**  = матрица V, в которой каждый столбец состоит из по-своему взвешенной смеси всех исходных столбцов матрицы V
- **V** – может быть той же самой матрицей LxD из векторов вложений, что и **Q, K**,

Тогда:

- **$\text{softmax}(QK^T)V$**  – векторы вложения, трансформированные (смешанные друг с другом) согласно их исходной «близости»

# Transformer-сети

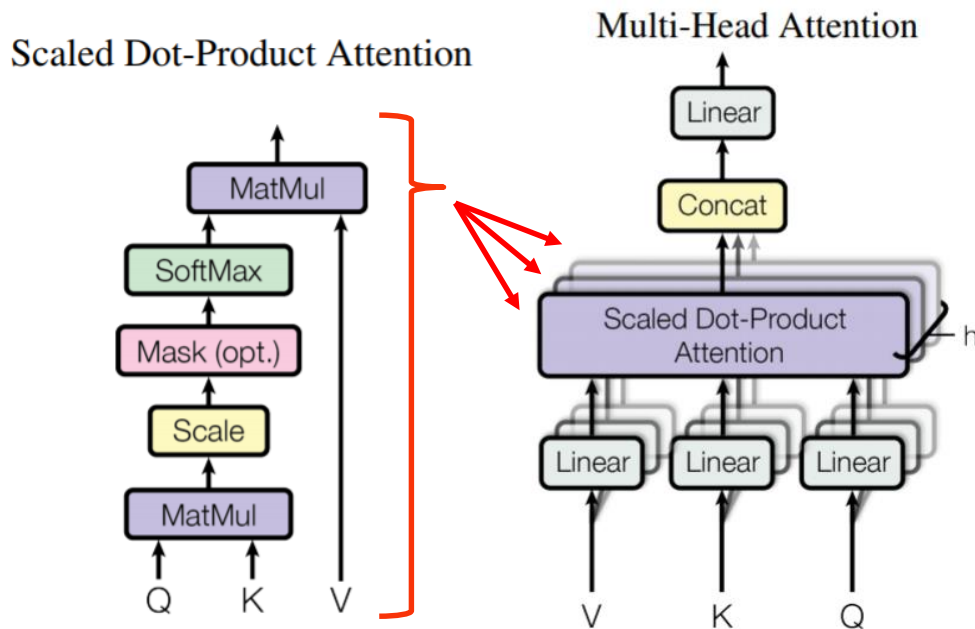


Image credit:  
<https://arxiv.org/pdf/1706.03762.pdf>

- На практике к одному и тому же входу применяется несколько «трансформаций внимания», различающиеся только входным полносвязным слоем
- Позволяет на одном и том же шаге устанавливать близости слов друг к другу в «разных смыслах»

# Позиционные вложения

Image credit:

<https://arxiv.org/pdf/1706.03762.pdf>

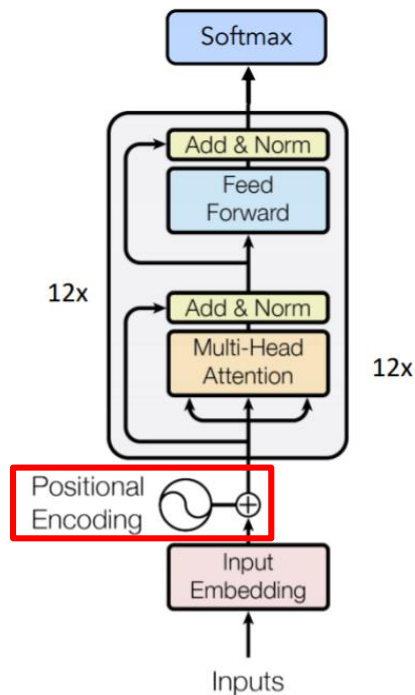
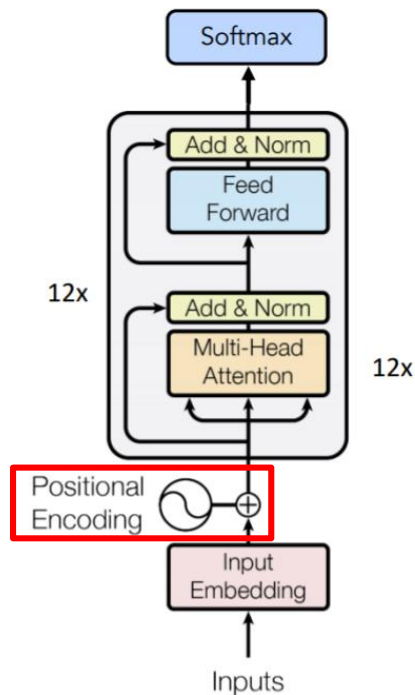


Рис.: Типичная структура энкодера в transformer-сетях

- Все элементы последовательности обрабатываются одновременно, а не последовательно друг за другом, как в RNN => теряется информация об относительном положении слов в предложении и их порядке
- Решается с помощью добавления **позиционного вложения/кодирования** (positional encoding) - к каждому вектору слова в последовательности добавляется сторонний вектор, зависящий только от позиции слова в последовательности

# Позиционные вложения

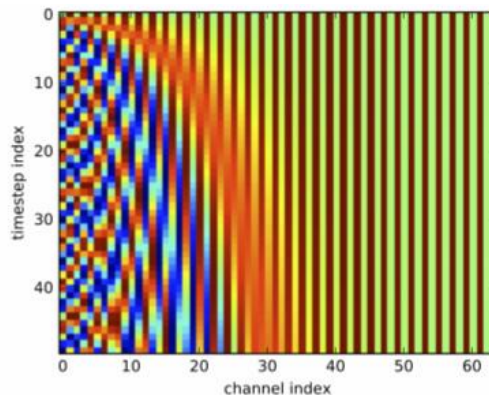
Image credit:  
<https://arxiv.org/pdf/1706.03762.pdf>  
<http://web.stanford.edu/class/cs224n>



- Позиционные вложения могут быть обучаемыми или фиксированными (ниже):

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

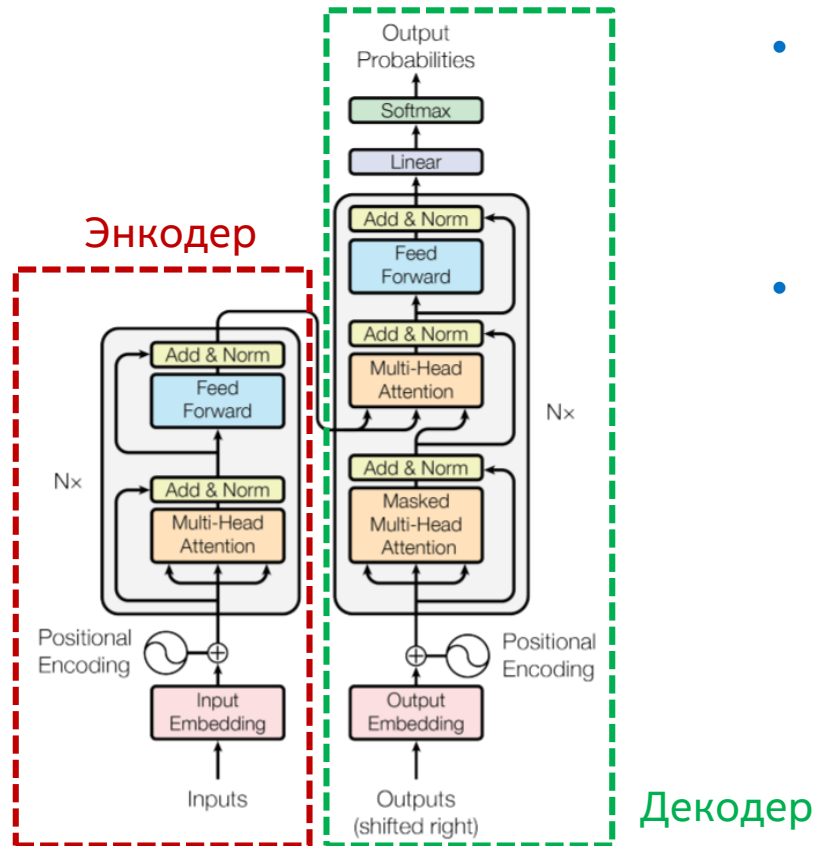
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



Слева: Синусоидальное позиционное вложение для последовательности из 50 слов (токенов) и размерности вложения 64

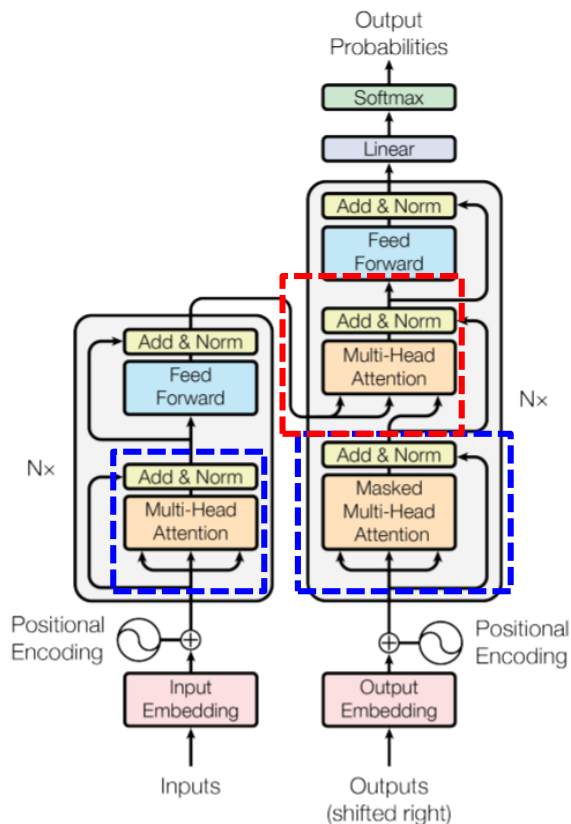
Рис.: Типичная структура энкодера в transformer-сетях

# Transformer-сети



- В общем случае в transformer-сети так же есть и энкодер, и декодер (например, при решении задачи машинного перевода)
- Декодер transformer-сетей - также состоит из attention-блоков

# Transformer-сети



- В декодере, помимо блоков типа «**self-attention**», также есть блоки типа «**encoder-decoder attention**» – аналогичны по смыслу тому, что использовалось в RNN:
- **Q** – матрица из векторов-признаков на некотором этапе **энкодера**,
- **K** := **Q** (например)
- **V** – матрица из векторов-признаков на соответствующем этапе декодера

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Примеры transformer-сетей: GPT, GPT-2

SYSTEM PROMPT (HUMAN-WRITTEN)

*Recycling is good for the world.*

*NO! YOU COULD NOT BE MORE WRONG!!*

MODEL COMPLETION (MACHINE-WRITTEN, 25 TRIES)

Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price



OpenAI

- GPT и GPT-2 - эффективно генерируют продолжения для текстов

Image credit:  
<https://openai.com/blog/better-language-models/>

## Примеры transformer-сетей: AlphaStar

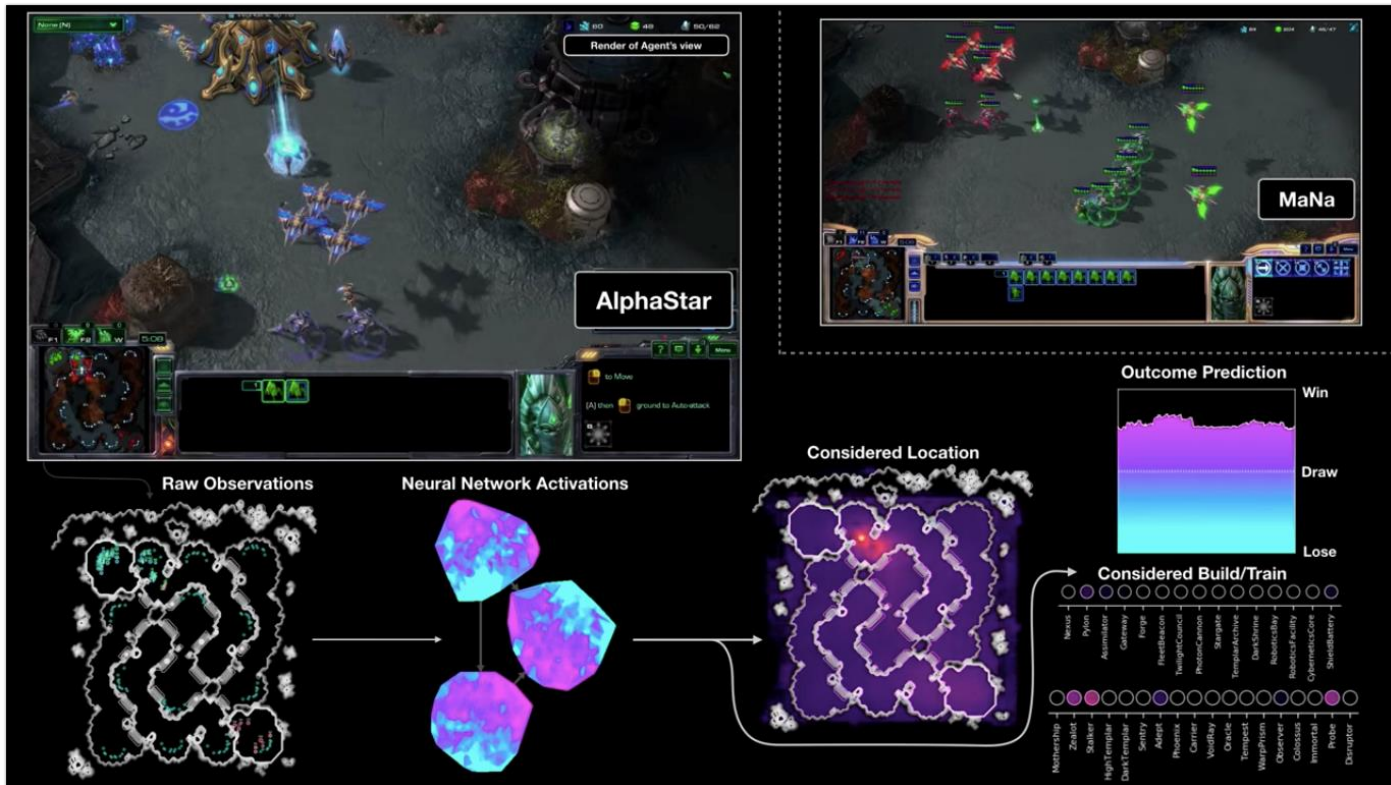


Image credit: <https://www.youtube.com/watch?v=HcZ48JDamyk>



# Примеры transformer-сетей: AlphaStar



## How AlphaStar is trained

AlphaStar's behaviour is generated by a deep neural network that receives input data from the raw game interface (a list of units and their properties), and outputs a sequence of instructions that constitute an action within the game. More specifically, the neural network architecture applies a transformer torso to the units (similar to relational deep reinforcement learning), combined with a deep LSTM core, an auto-regressive policy head with a pointer network, and a centralised value baseline. We believe that this advanced model will help with many other challenges in machine learning research that involve long-term sequence modelling and large output spaces such as translation, language modelling and visual representations.

# Примеры transformer-сетей: OpenAI MuseNet

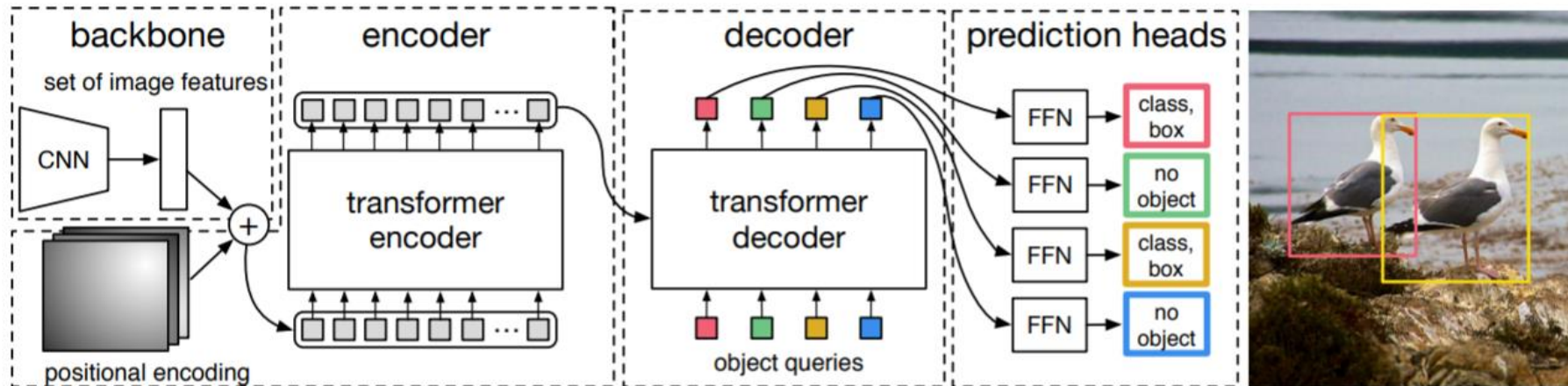
Compose in the style of Chopin -  
starting with Mozart's Rondo alla Turca -

SHOW ADVANCED SETTINGS



Image credit: <https://openai.com/blog/musenet/>

# Примеры transformer-сетей: DETR

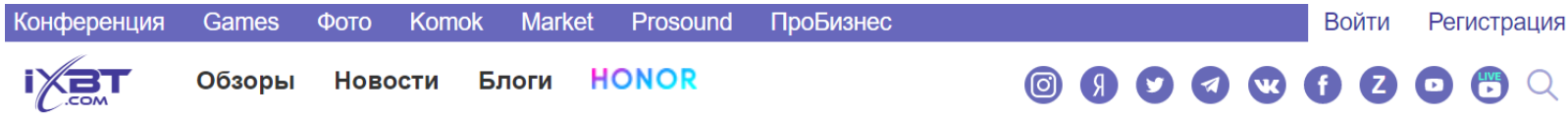


- Transformer-архитектуры можно применять даже в задачах компьютерного зрения!

Image credit: <https://arxiv.org/pdf/2005.12872.pdf>

# Примеры transformer-сетей: BERT

- **BERT** = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- Разновидности: BERT-base, BERT-large, RoBERTa, ALBERT, DistilBERT, CamemBERT, MobileBERT, ...



Главная / Новости / 09 декабря 2019 в 19:09 /

## Крупнейшее за пять лет обновление Google пришло в Россию

### Внедрён алгоритм BERT для русскоязычного поиска

Компания Google объявила о запуске в русскоязычном фирменном сервисе поиска новых алгоритмов. Благодаря технологиями машинного обучения поисковик стал намного лучше обрабатывать запросы.

Image credit: <https://www.ixbt.com/news/2019/12/09/krupnejshee-za-pjat-let-obnovlenie-google-prishlo-v-rossiju.html>

# BERT

- **BERT** = **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

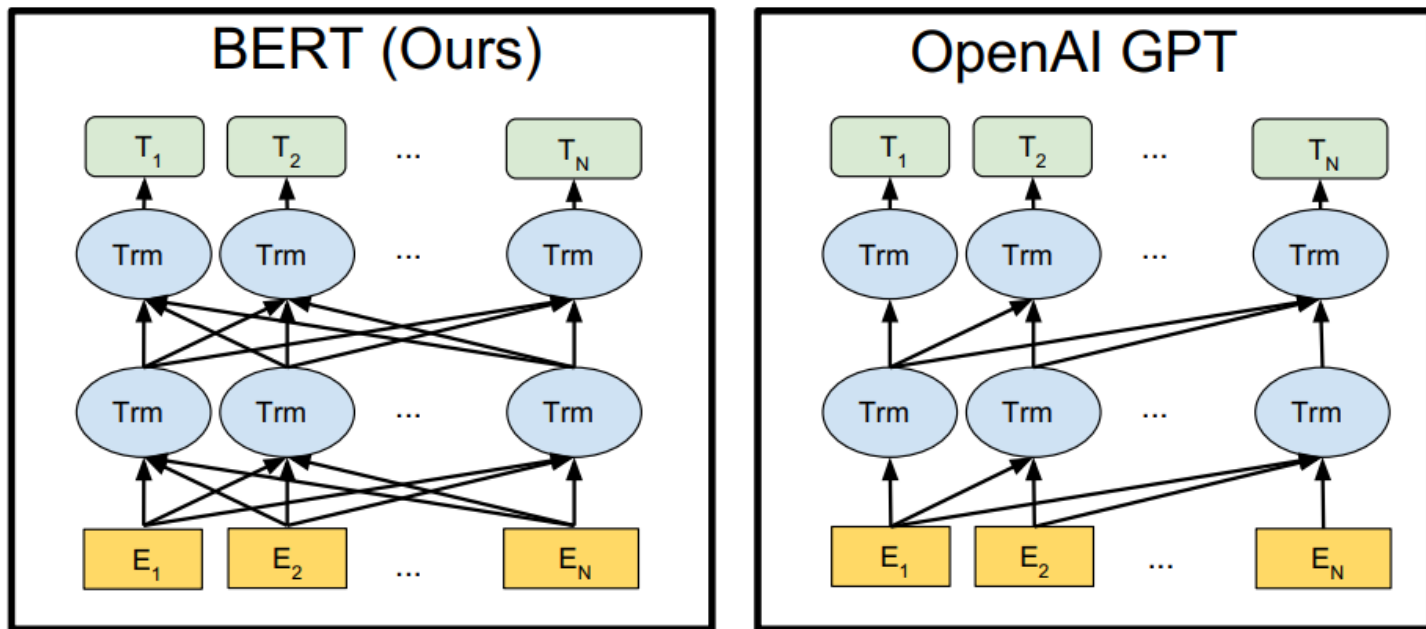


Image credit: <https://arxiv.org/pdf/1810.04805.pdf>

# BERT

**BERT** = Bidirectional **Encoder Representations** from Transformers

- Обычно используется лишь предтренированный энкодер, остальные задачи решаются путем добавления к выходам энкодера модулей, специфических для задачи (например, классификатор)
- Для каждого входного элемента последовательности BERT генерирует вектор, являющийся эффективным вектором-признаком элемента для многих дальнейших задач, т.е. для текста BERT генерирует эффективные **представления** слов

*Image credit: <https://arxiv.org/pdf/1810.04805.pdf>*

# BERT - предобучение

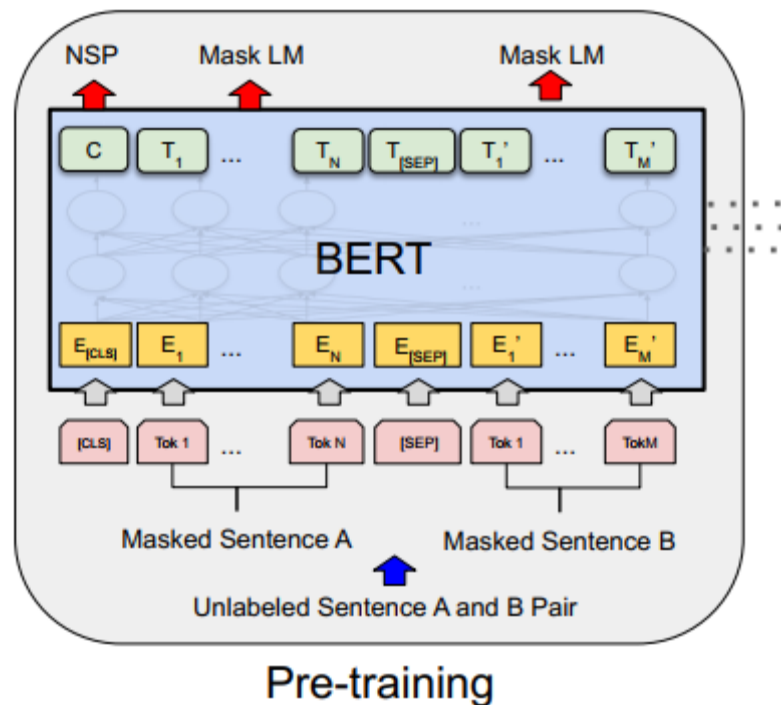


Image credit: <https://arxiv.org/pdf/1810.04805.pdf>

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]

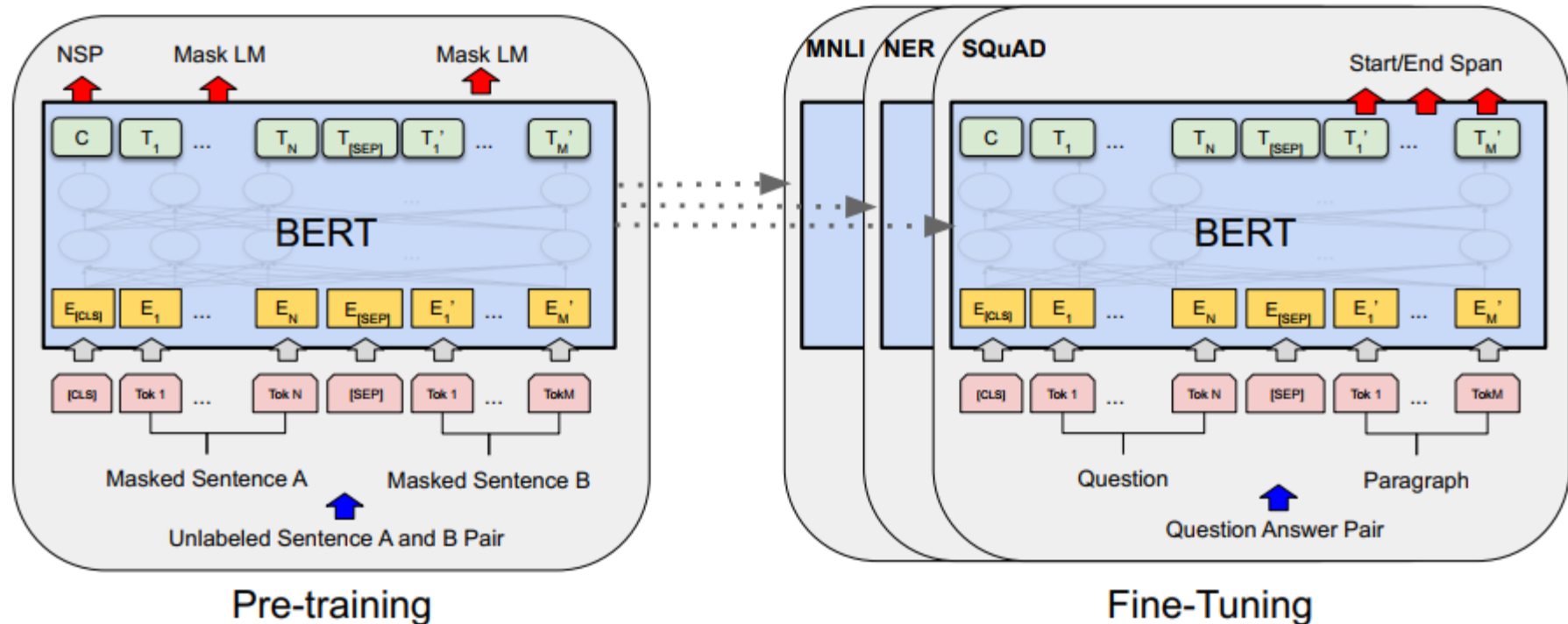
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

- Энкодер можно обучить без учителя - с помощью вспомогательных задач предсказания следующего предложения и одновременного восстановления замаскированных слов

# BERT - предобучение и дообучение

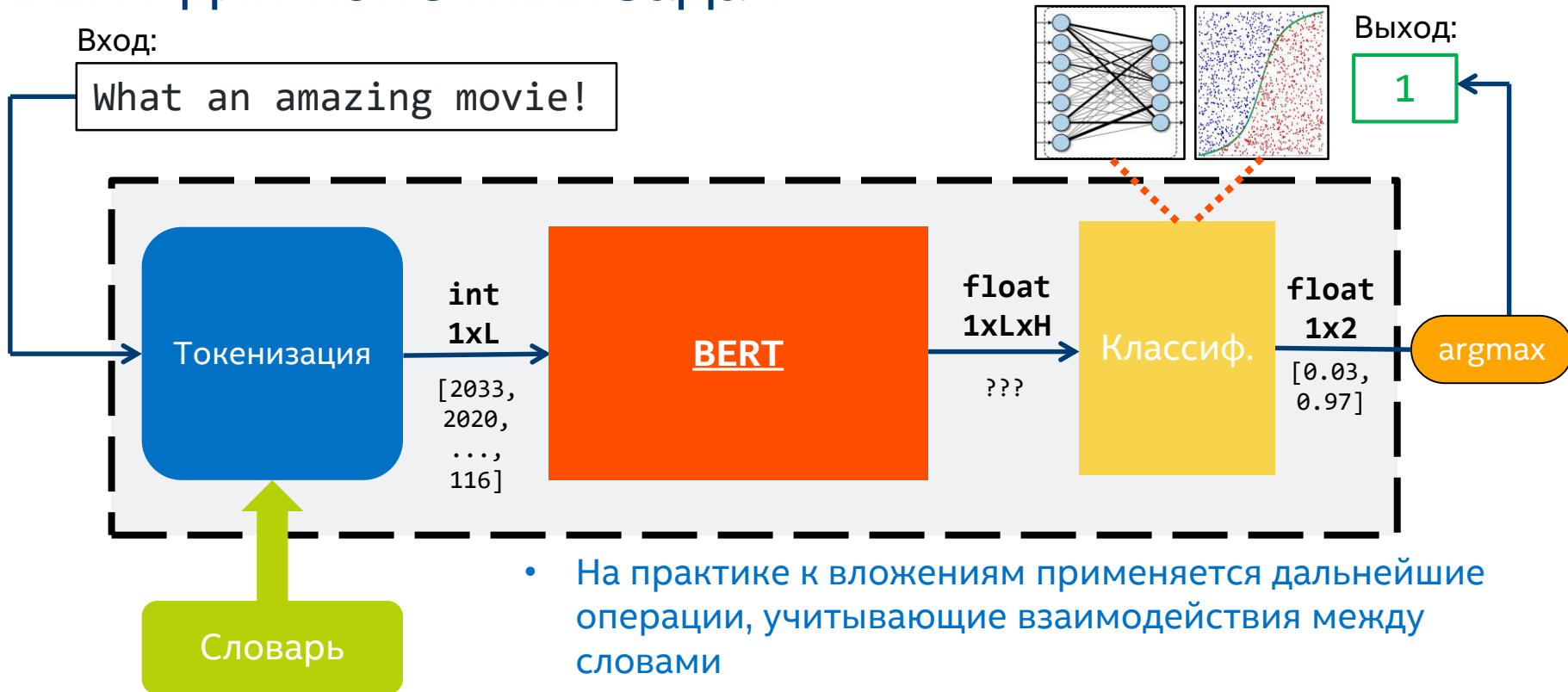
Image credit: <https://arxiv.org/pdf/1810.04805.pdf>



- Один и тот же предтренированный энкодер затем можно дообучить на множество конечных задач!



# BERT для конечных задач



NB: На схеме прямоугольниками указаны обучаемые компоненты.

# BERT - формат входа для задач со структурой текста

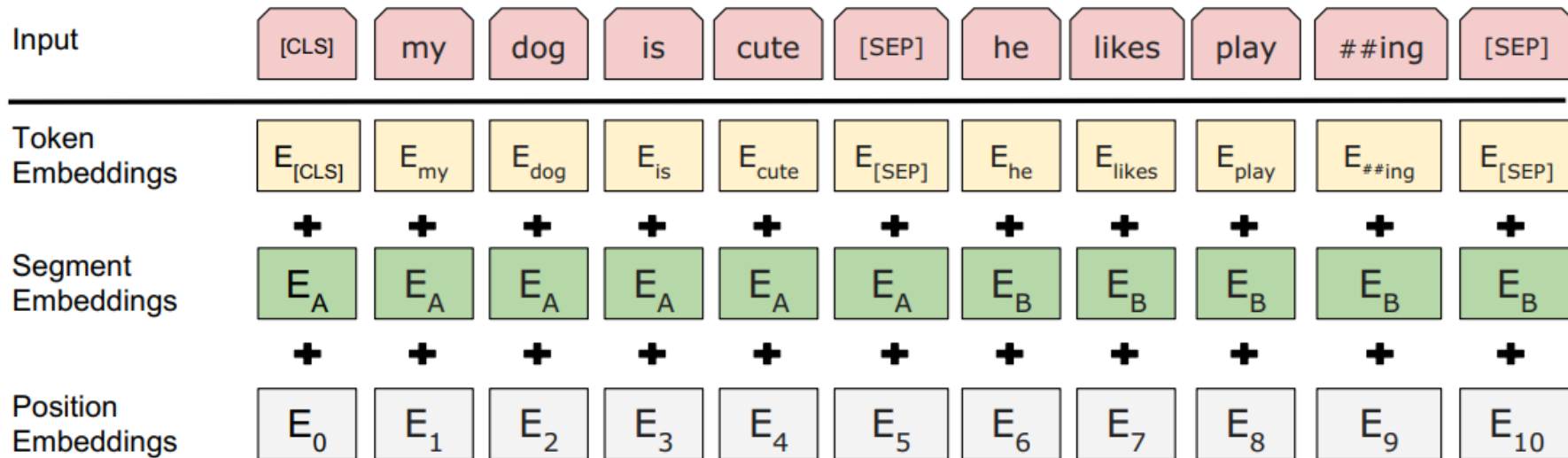


Image credit: <https://arxiv.org/pdf/1810.04805.pdf>

# Задача ответа на вопрос по тексту - SQuAD v1.1, v2.0

SQuAD

HomeExplore 2.0Explore 1.1

## Normans

### The Stanford Question Answering Dataset

The **Normans** (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the **10th and 11th centuries** gave their name to **Normandy**, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants would gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the **Normans** emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.

**In what country is Normandy located?**  
Ground Truth Answers: France France France France

**When were the Normans in Normandy?**  
Ground Truth Answers: 10th and 11th centuries in the 10th and 11th centuries **10th and 11th centuries** 10th and 11th centuries

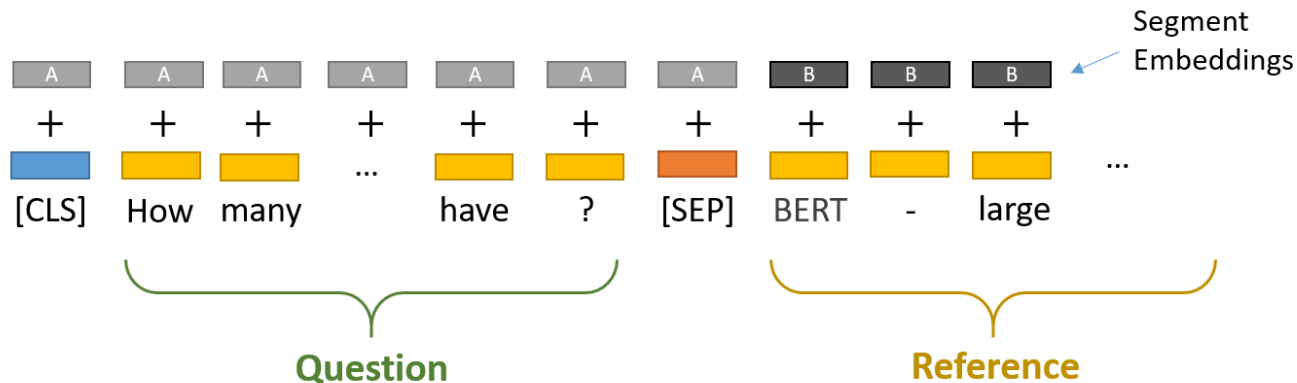
**From which countries did the Norse originate?**  
Ground Truth Answers: Denmark, Iceland and Norway Denmark, Iceland and Norway Denmark, Iceland and Norway Denmark, Iceland and Norway

**Who was the Norse leader?**  
Ground Truth Answers: Rollo Rollo Rollo Rollo

**What century did the Normans first gain their separate**

Image credit: <https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Normans.html>

# BERT - формат входа для ответов на вопросы по тексту



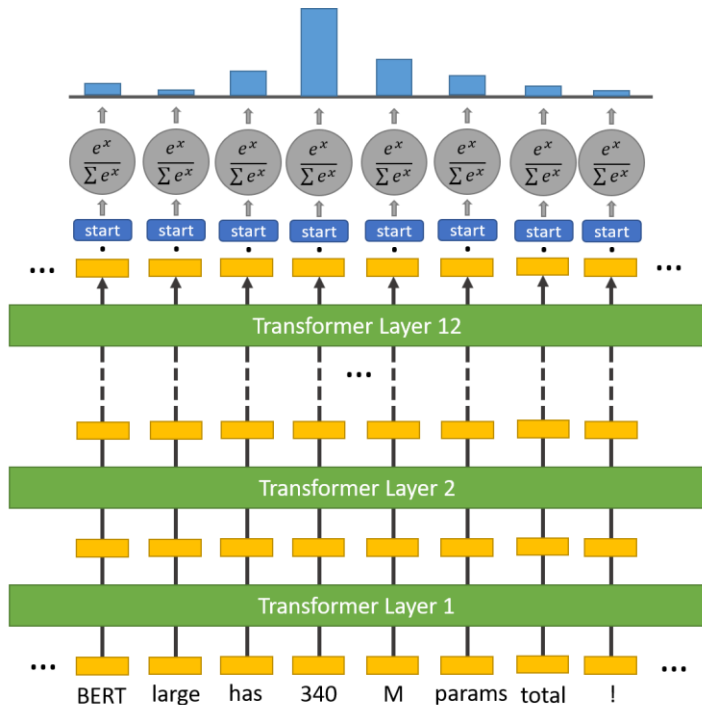
**Question:** How many parameters does BERT-large have?

**Reference Text:** BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

Image credit: <https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/>

# BERT - обработка выходов для получения ответов

Вероятность начала ответа в текущей позиции



Вероятность конца ответа в текущей позиции

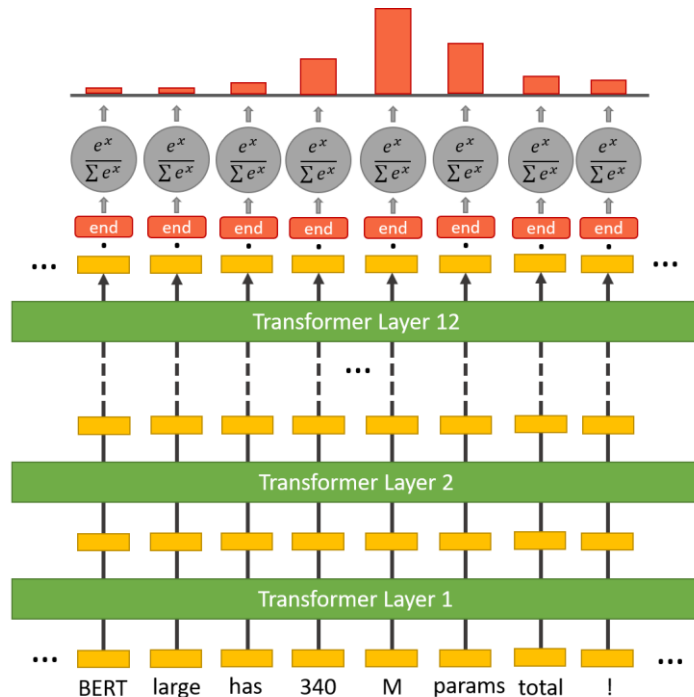


Image credit: <https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/>

# BERT - обработка выходов для получения ответов

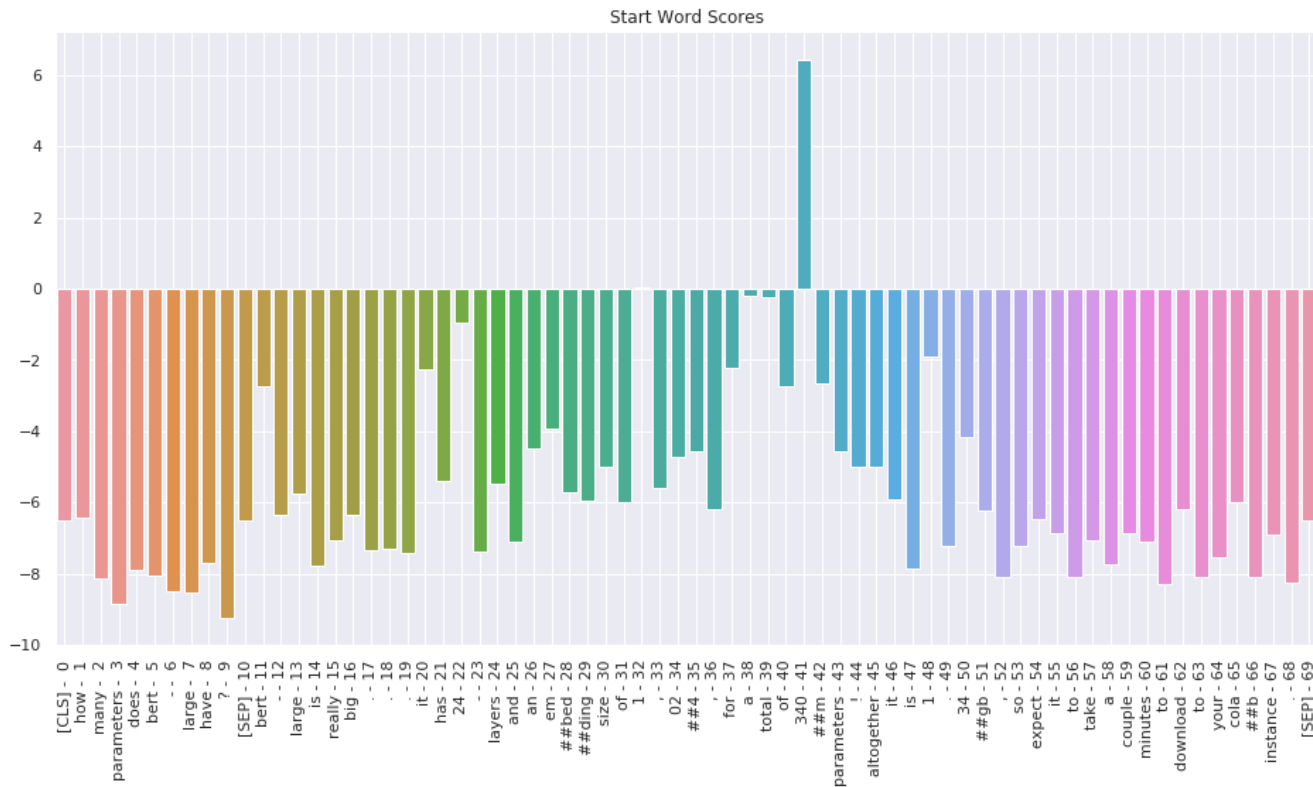


Image credit: <https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/>

# BERT - обработка выходов для получения ответов

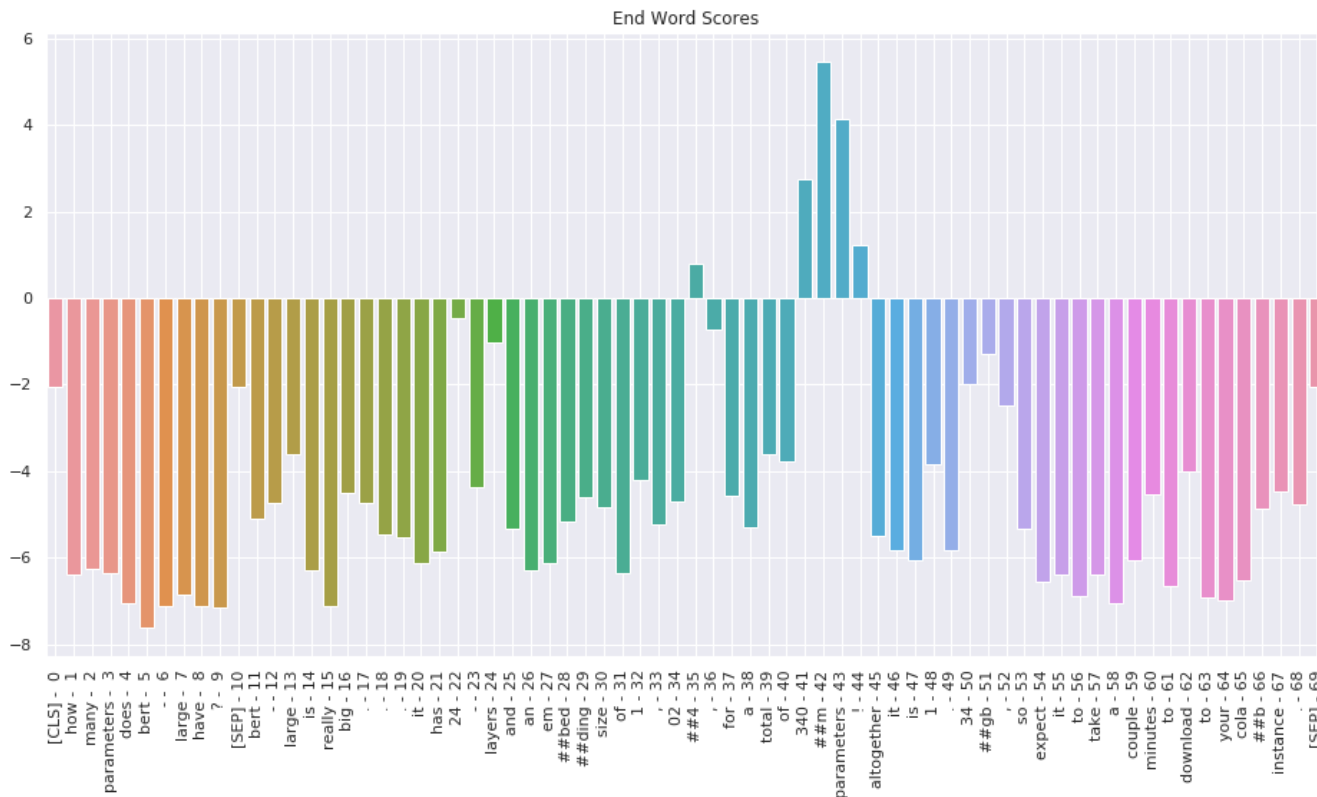
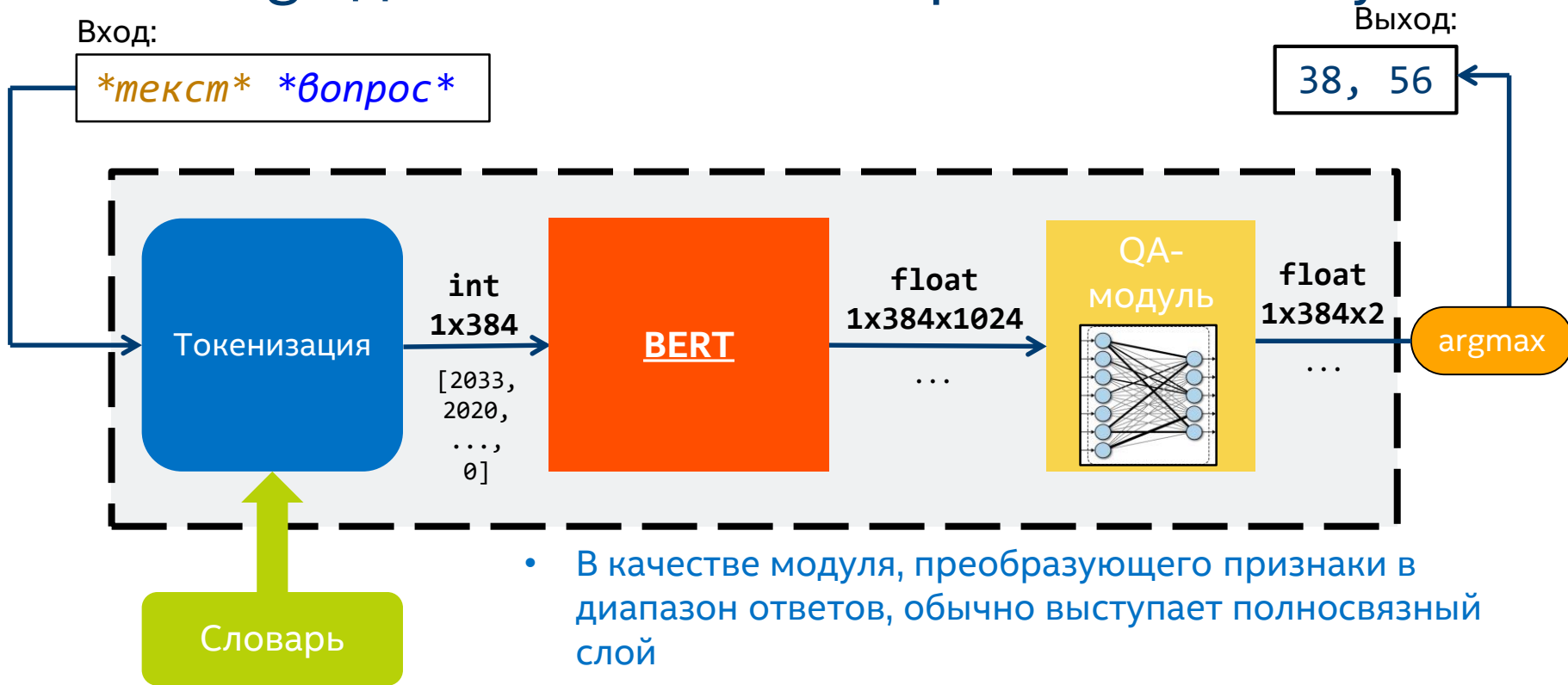


Image credit: <https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/>

# BERT-large для ответов на вопросы по тексту



NB: QA = question answering



# BERT-large для ответов на вопросы по тексту

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT <sub>BASE</sub> (Single)	80.8	88.5	-	-
BERT <sub>LARGE</sub> (Single)	84.1	90.9	-	-
BERT <sub>LARGE</sub> (Ensemble)	85.8	91.8	-	-
BERT <sub>LARGE</sub> (Sgl.+TriviaQA)	<b>84.2</b>	<b>91.1</b>	<b>85.1</b>	<b>91.8</b>
BERT <sub>LARGE</sub> (Ens.+TriviaQA)	<b>86.2</b>	<b>92.2</b>	<b>87.4</b>	<b>93.2</b>

- Возможности BERT-large по ответу на вопросы сравнима с человеческими, а в ансамблях - превосходит их

**EM** (exact match) - процентная метрика доли результатов нейросети, в которой ответ модели в точности совпадает с истинным

**F1** - метрика, аналогичная F1 в задачах классификации, но вычисляемая согласно числу перекрытия между токенами ответа модели и истинными токенами ответа

