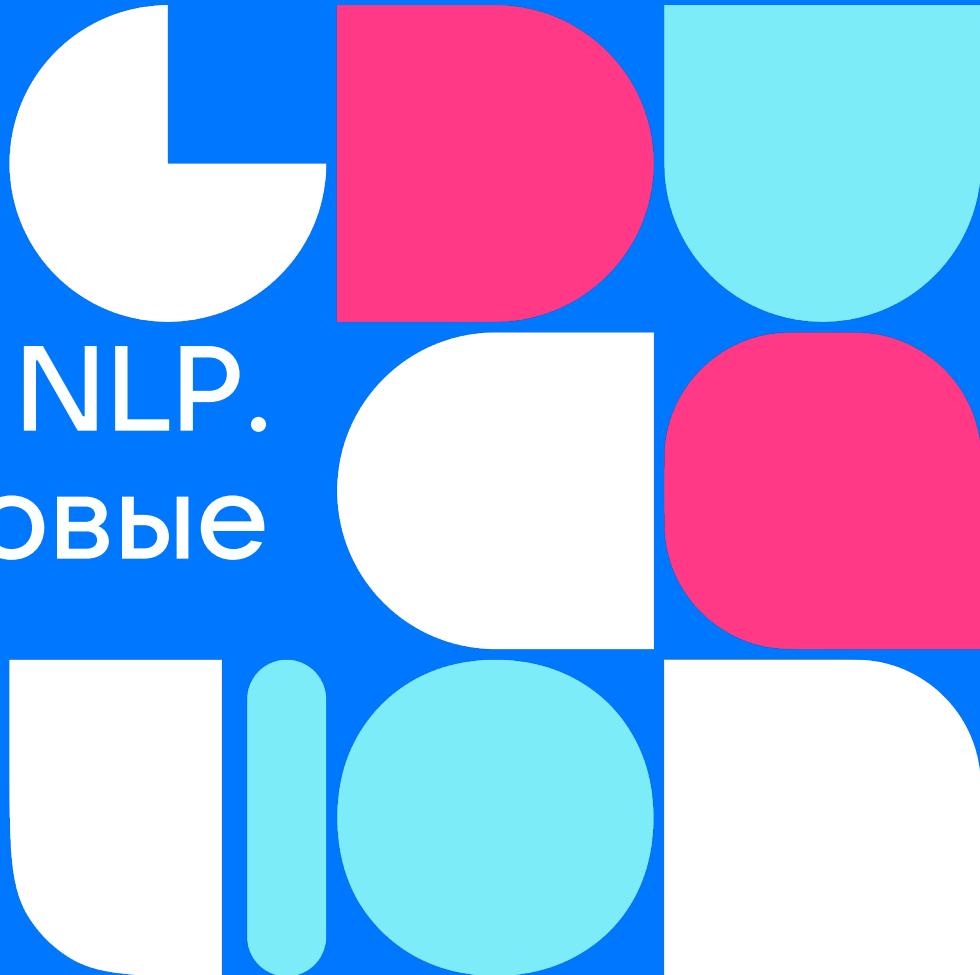




# Современный NLP. Большие языковые модели

ML Антиспам  
Почты Mail.ru



# Лекция 1

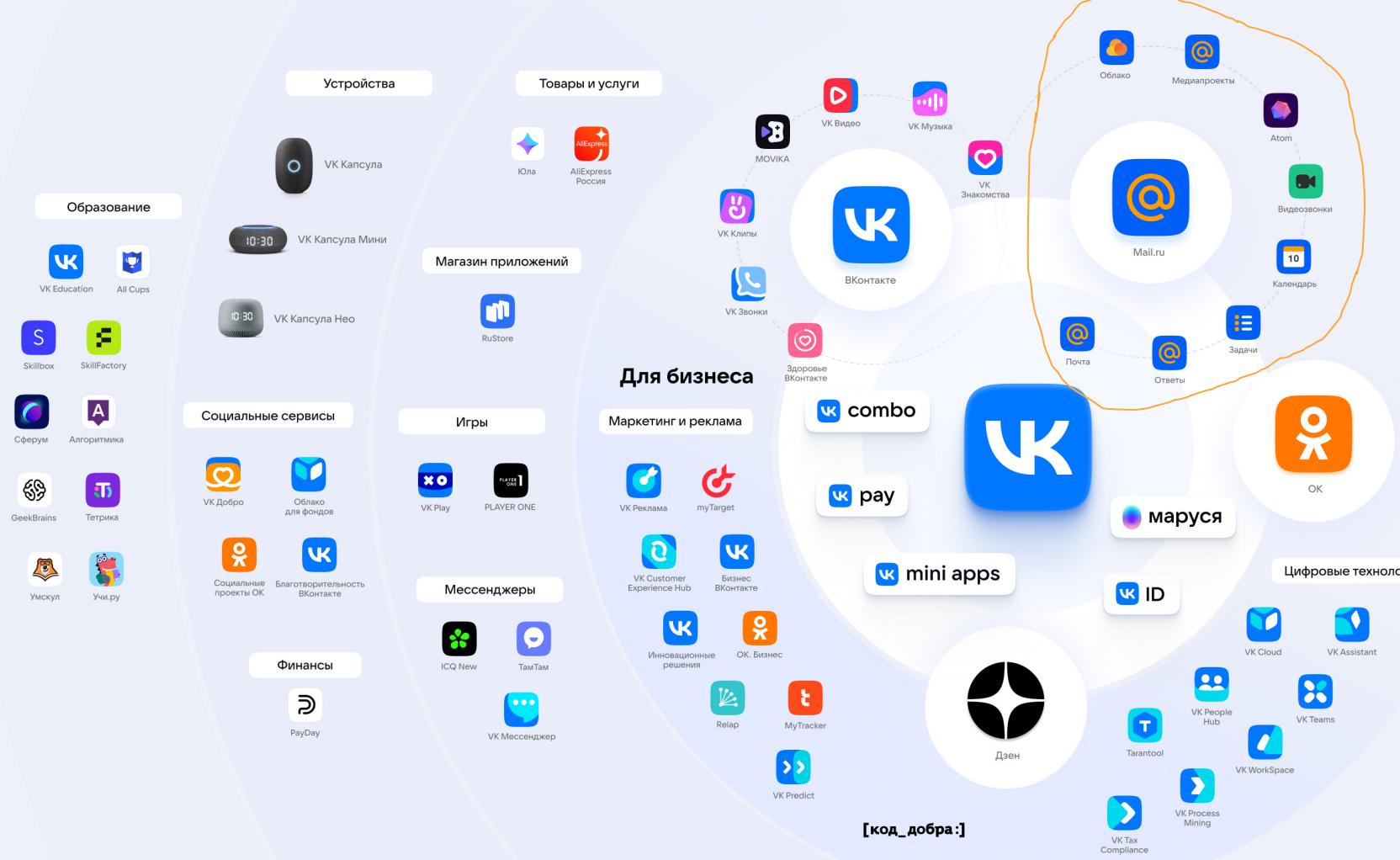
## Обзор области NLP

13.02.24, ВМК МГУ  
Дмитрий Калашников

# О чём сегодня поговорим

- Познакомимся
- Правила игры
- Область NLP
- Особенности работы с текстом
- Классические методы векторизации
- Эмбеддинги и нейронные сети
- Языковое моделирование
- Нейросетевые подходы к задаче LM

Давайте  
знакомиться



## Почта в цифрах

**700 000**

Количество обрабатываемых писем  
в минуту

**600 ПБ**

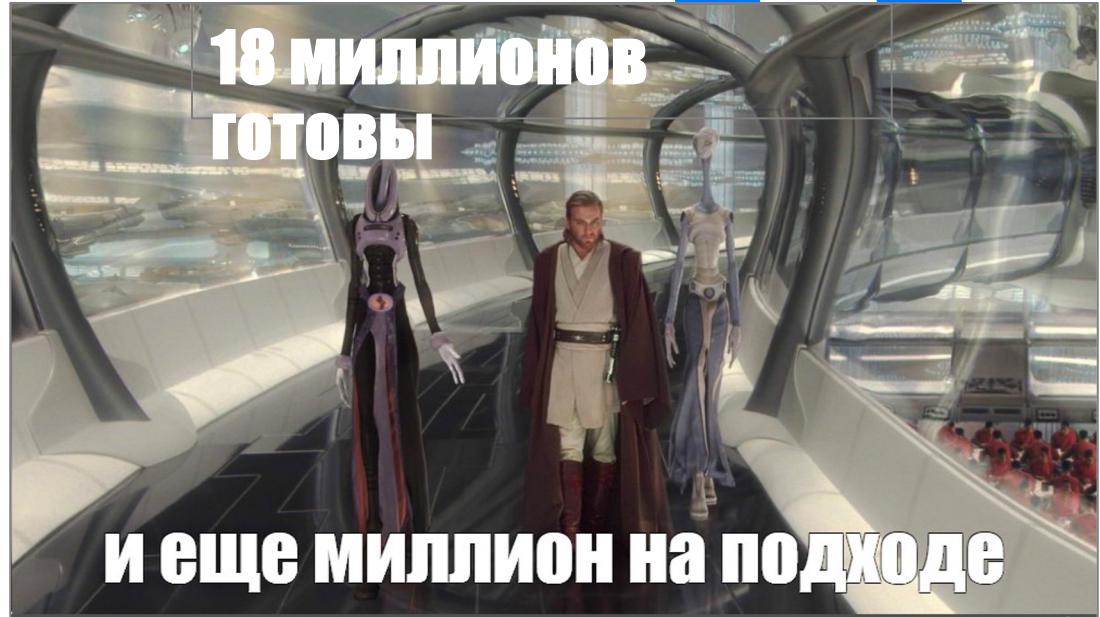
Объём хранилища

**24/7**

Обновления без даунтаймов

Почта  
Mail.ru —  
ЭТО...

~18M DAU



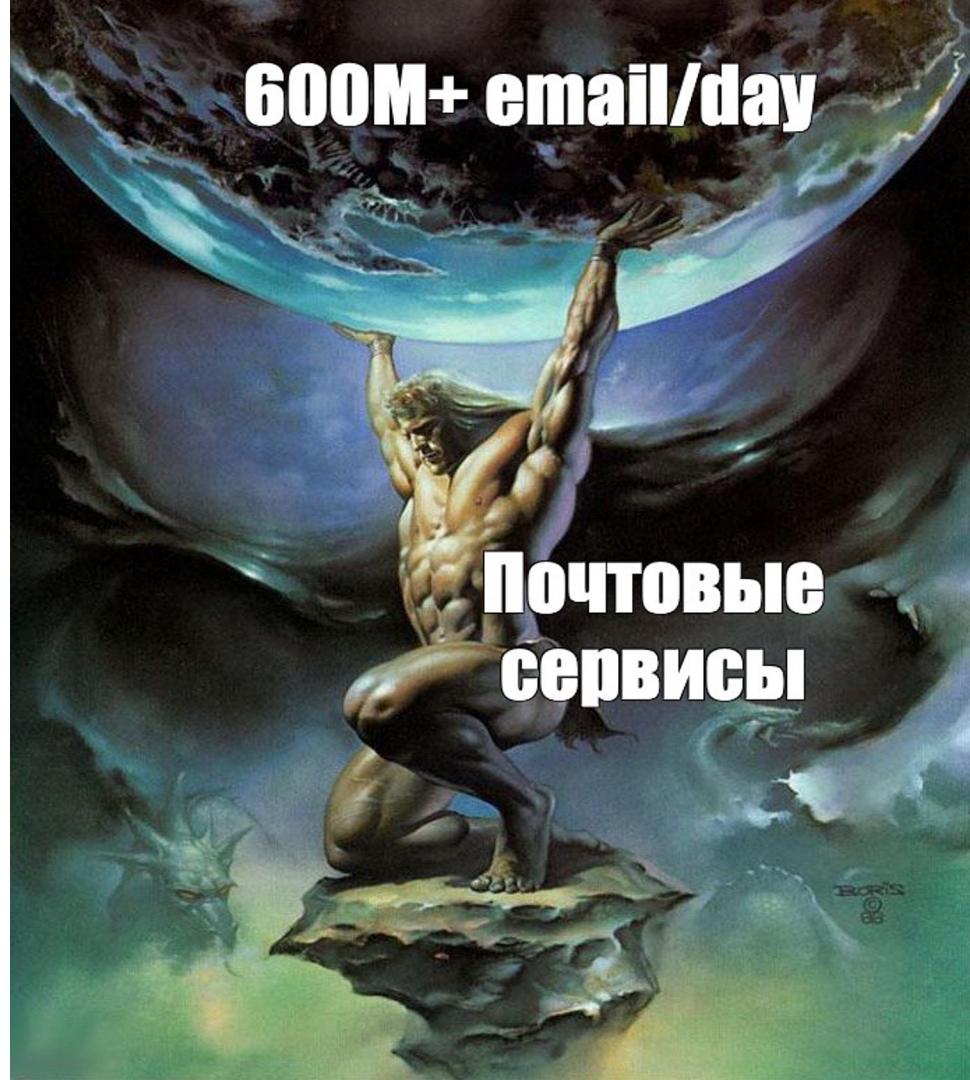
Почта Mail.ru — это...

~18M DAU

600M+ emails/day

600M+ email/day

Почтовые  
сервисы



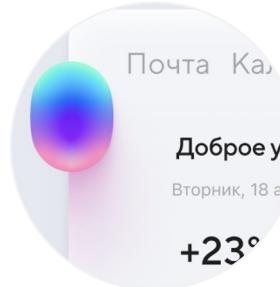
# ML-фичи в Почте



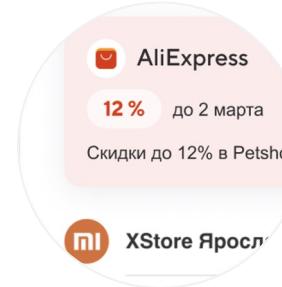
Антиспам &  
Integrity



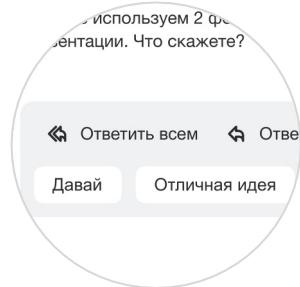
Умная  
сортировка  
писем



Чтение писем  
Марусей



Парсинг  
заказов и дат,  
СКИДКИ



Умные  
быстрые  
ответы

# О разнообразии задач

- В Почте работает более 50 ML-моделей
- Стремимся создавать новое, а не поддерживать старое
- 3 направления
  - **Продукт:** ML для улучшения пользовательского опыта
  - **Антиспам:** сложнейшие системы признаков спама
  - **Integrity:** быстрое детектирование ботов и взломов

# Без лишних слов



Мы тут не в первый раз!

Гостевая лекция на ВМК

от нашей команды:

<https://www.youtube.com/watch?v=QjYv7Gvra-4>

# Кто мы



Дмитрий  
Калашников



Владимир  
Макаренко



Мария  
Анисимова



Артём  
Степанов

# Кто я

- ВМК МГУ
- За моими плечами:
  - Категоризация писем в Почте
  - Антиспам-классификатор
  - Классификатор-взлома
  - Генеративные модели
- Интервенция VK на ВМК
- Инициатор этого курса
- Лекции и конференции



# Почему нас можно слушать



Давно и активно в NLP:  
- Классические модели;  
- Нейронка;  
- Трансформеры  
(включая BERT)



Знаем, какие навыки нужны  
бизнесу, и его ограничения:  
- Реальные пользователи  
- Высоконагруженные  
системы  
- Сухие метрики ->  
реальное восприятие  
пользователей



Много внутренних  
исследований, связанных с  
генеративными моделями

# Почему нас важно слушать

# А что случилось

...в области за последнее время

- Публикация BERT и GPT
- Релиз ChatGPT
- "Утечка" весов генеративной open-source модели Llama

Всё это привело к резкому развитию NLP и генеративного AI



# Наиболее активное влияние на NLP

**Bert: Pre-training of deep bidirectional transformers for language understanding**

J Devlin, MW Chang, K Lee, K Toutanova - arXiv preprint arXiv ..., 2018 - arxiv.org

... We introduce **BERT** and its detailed implementation in this ... For finetuning, the **BERT** model is first initialized with the pre-... A distinctive feature of **BERT** is its unified architecture across ...

☆ Сохранить 99 Цитировать Цитируется: 90165 Похожие статьи Все версии статьи (46) »

[PDF] **Improving language understanding by generative pre-training**

A Radford, K Narasimhan, T Salimans, I Sutskever - 2018 - mikecaptain.com

... to **better understand** why **language** model **pre-training** of transformers is effective. A hypothesis is that the underlying **generative** ... in order to **improve** its **language** modeling capability and ...

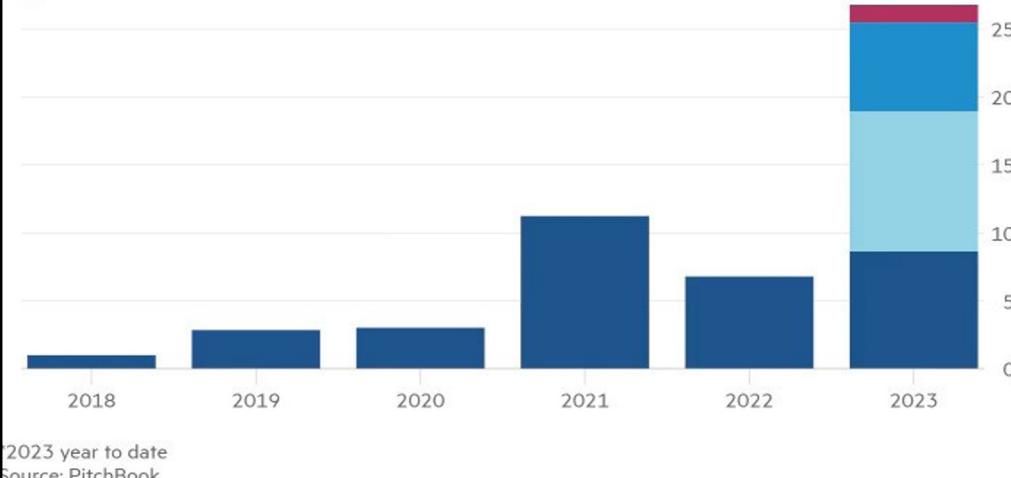
☆ Сохранить 99 Цитировать Цитируется: 7931 Похожие статьи Все версии статьи (15) »

# Резкий рост инвестиций

Big tech investors push generative AI fundraising to record highs

Investment into generative AI (\$bn)

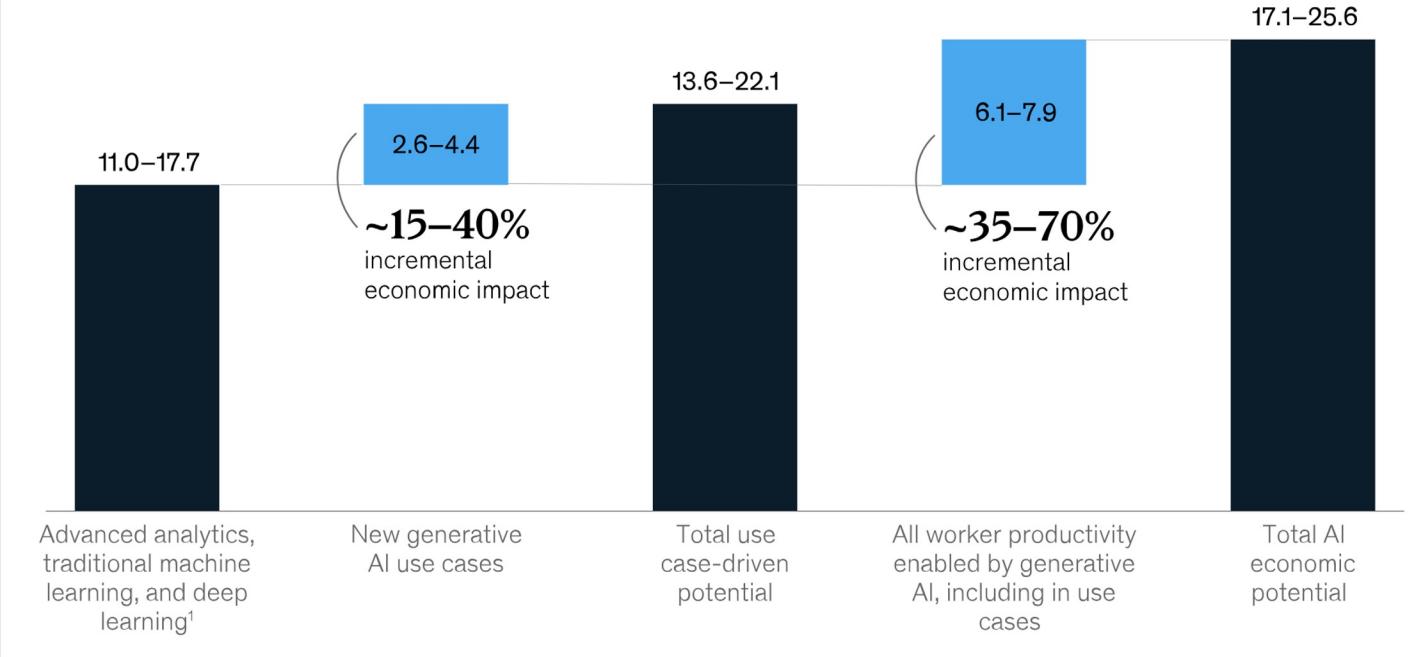
Conventional VC      OpenAI - Microsoft      Anthropic - Amazon  
Inflection - Microsoft



# Огромный вклад в экономику

- Вклад генеративного ИИ в глобальную экономику может эквивалентен \$2.6-\$4.4 триллионов
- Для сравнения: ВВП Великобритании в 2021 был \$3.1 триллионов
- У текущих технологий есть потенциал автоматизировать 60-70% работы сотрудников

## AI's potential impact on the global economy, \$ trillion



Источник: <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier>

# Скорость прогресса

- Это всё за полгода:
  - ChatGPT – Open AI
  - GPT-4 – Open AI
  - Claude – Anthropic
  - PaLM 2 – Google
  - Llama – Meta
- Продукты:
  - ChatGPT и армия аналогов
  - Copilot и др.
- Мультимодальность: текст, изображения, видео, аудио, код



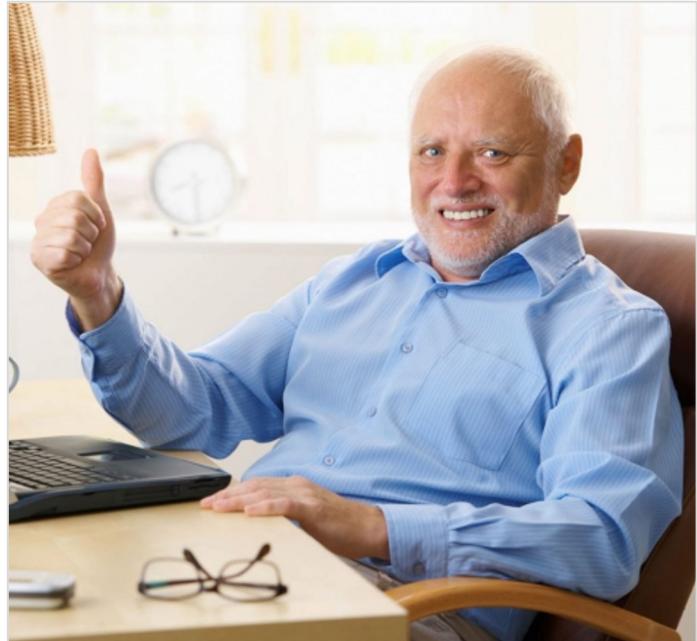
# Почему это полезно лично вам

- LLM важно, круто, актуально
- Полезно не только в рамках LLM: под особым углом раскроем NLP в целом
- Дадим возможность пощупать современные подходы и технологии
- Всем нам нужно закрывать спецкурсы: интересно, полезно и достаточно несложно
- Предусмотрели вариант прокачаться углубленно
- Самых активных студентов – заметим :)



# Как выглядит курс

- Каждую неделю: лекция + семинар
- 8 лекций + 1 бонусная
- После каждой лекции:
  - Теория: теоретический тест по содержанию лекции
  - Практика: тест, для которых надо будет немного поработать с кодом
  - Для всех
- После некоторых лекций:
  - Более умная модель для LLM-ассистента
  - Формат: внедряем в чат-бота в Telegram
  - Для желающих прокачаться



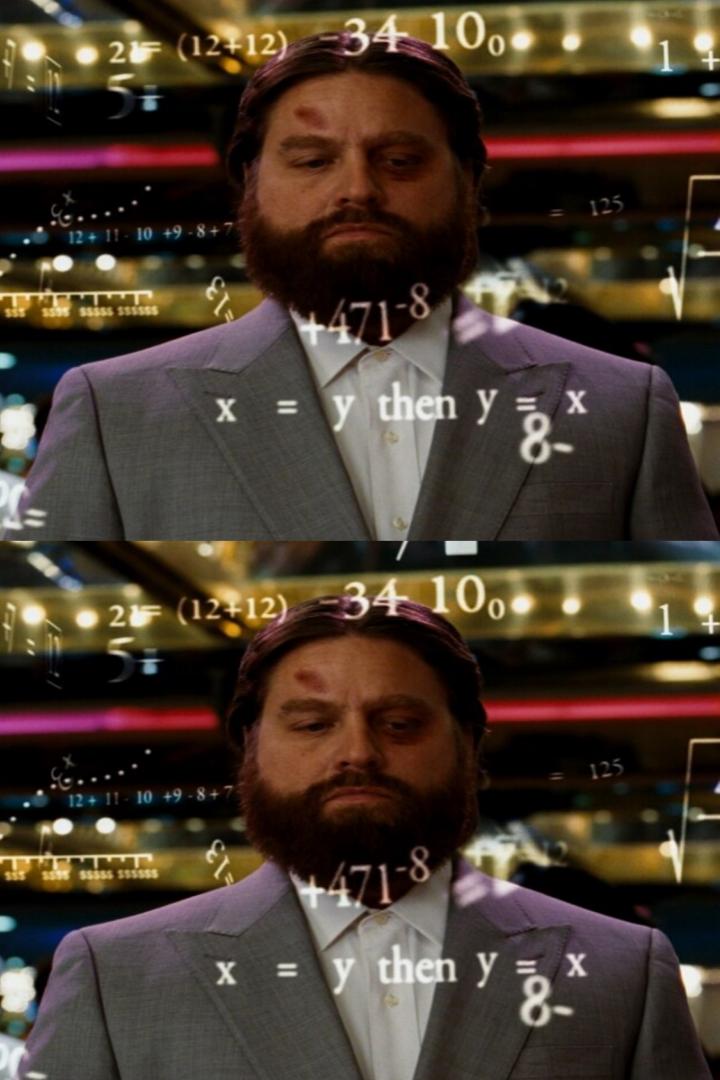
# Критерии оценки

- Накопительная система:
  - После каждой лекции – тесты
  - Баллы набираются за каждый тест
  - Тесты открыты ограниченное время  
(1-2 недели)
- Ориентировочные критерии:
  - 40% – удовлетворительно
  - 60% – хорошо / зачет
  - 80% – отлично



# Как сдать курс

- Курс можно сдать на любую оценку, только проходя оба теста после каждой лекции:
  - теоретический
  - практический
- 60% вопросов заведомо несложные :)
- Зачем нужен ассистент:
  - возможность попрактиковаться с LLM
  - возможность добрать баллы



# Как нас найти

- Связь, ссылки, записи лекций и домашки – в чате
- Вопросы стоит задавать очно, либо через чат
- Все лекторы есть в чате и стараются отвечать оперативно :)



ОТКРОЙТЕ КАМЕРОЙ VK

# Что ждём от студентов на старте

- Если минимально прослушали курс ML/DL на ВМК — вы нам подходите!
  - Знакомы с ML/DL
  - Поверхностное знание NLP
  - Пишете на Python 3, умеет работаете с pytorch либо готовы этому учиться
- Что почитать: в конце презентации



# План курса

## Прогрев

1. Обзор области NLP. Большие языковые модели. Эволюция методов обработки естественного языка
2. Архитектура трансформера.
3. Эволюция моделей семейства BERT
4. Эволюция моделей семейства GPT

## Прожарка

5. Общие принципы работы ChatGPT.
6. Работа с данными для больших языковых моделей
7. Подходы к оптимизации обучения и инференса LLM
8. Нюансы дообучения моделей семейства GPT. Alignment и SFT. Безопасность модели.
9. Бонусная: технические нюансы обучения GPT-моделей

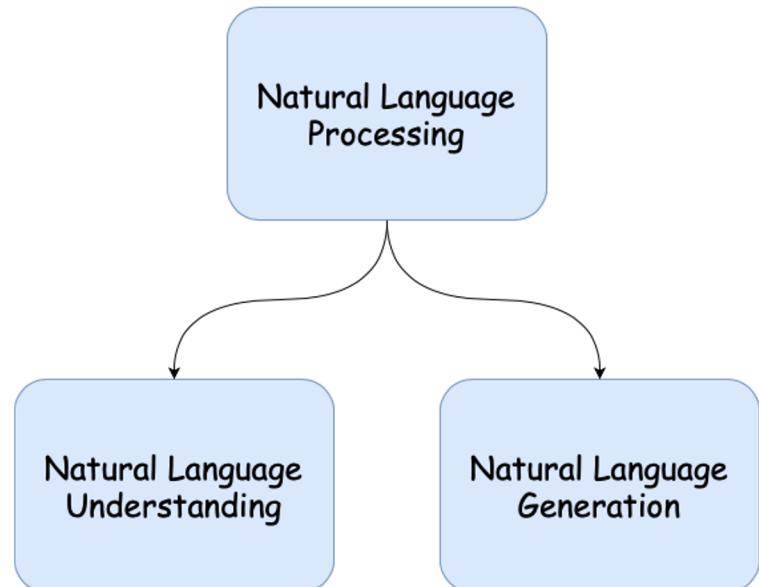
Так много  
ответов, но так  
мало вопросов...



В какой мы  
области

# NLP

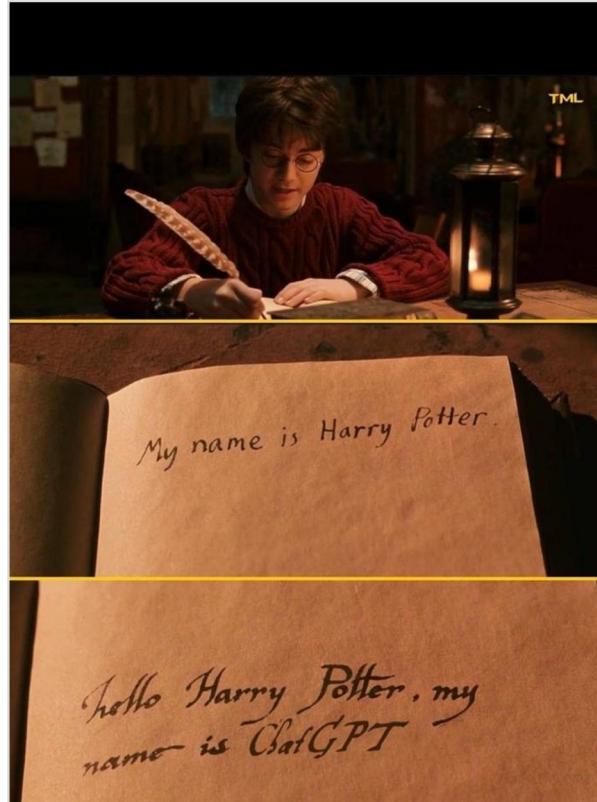
- NLP – позволяет компьютеру понимать человеческий язык в письменной и устной форме, с помощью ML/DL
- NLU – фокусируется на машинном понимании прочитанного с помощью грамматики и контекста, позволяя ему определять предполагаемое значение предложения
  - Анализ настроений
  - BERT
- NLG – фокусируется на генерации текста на основе заданного набора данных.
  - Суммаризация
  - GPT



Источник: <https://www.ibm.com/blog/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/>

# Языковое моделирование

- Языковые модели (LM) оценивают вероятность появления различных языковых единиц: символов, токенов, последовательностей токенов
- Применение языковых моделей:
  - Суммаризация
  - Генерация продолжения текста
  - Машинный перевод
- Применение "не языковых" моделей:
  - Анализ настроений
  - Поиск похожих текстов
  - NER



# Языковое моделирование

- Текст – дискретный сигнал
- Текст – последовательность символов, слов, токенов?
- Текст необходимо представить в виде числовых признаков

Как это сделать?

- Всегда можем разбить текст на части (слова)
- Составляем словарь из тех слов, которые могут встретиться в интересующих нас текстах
- Пронумеруем все слова в словаре
- Теперь текст – последовательность номеров слов в словаре

LLM - смысл моей жизни

Токенизация

|     |    |       |      |       |
|-----|----|-------|------|-------|
| LLM | -  | смысл | моей | жизни |
| 101 | 32 | 476   | 491  | 244   |

Представления

|         |         |        |         |         |
|---------|---------|--------|---------|---------|
| -0,0235 | 0,0542  | 0,1503 | 0,452   | -0,0011 |
| 0,5253  | -0,0999 | 0,3145 | -0,0000 | 0,0010  |
| ...     | ...     | ...    | ...     | ...     |
| 0,0424  | -0,0013 | 0,0145 | 0,2535  | -0,0146 |

# Языковое моделирование

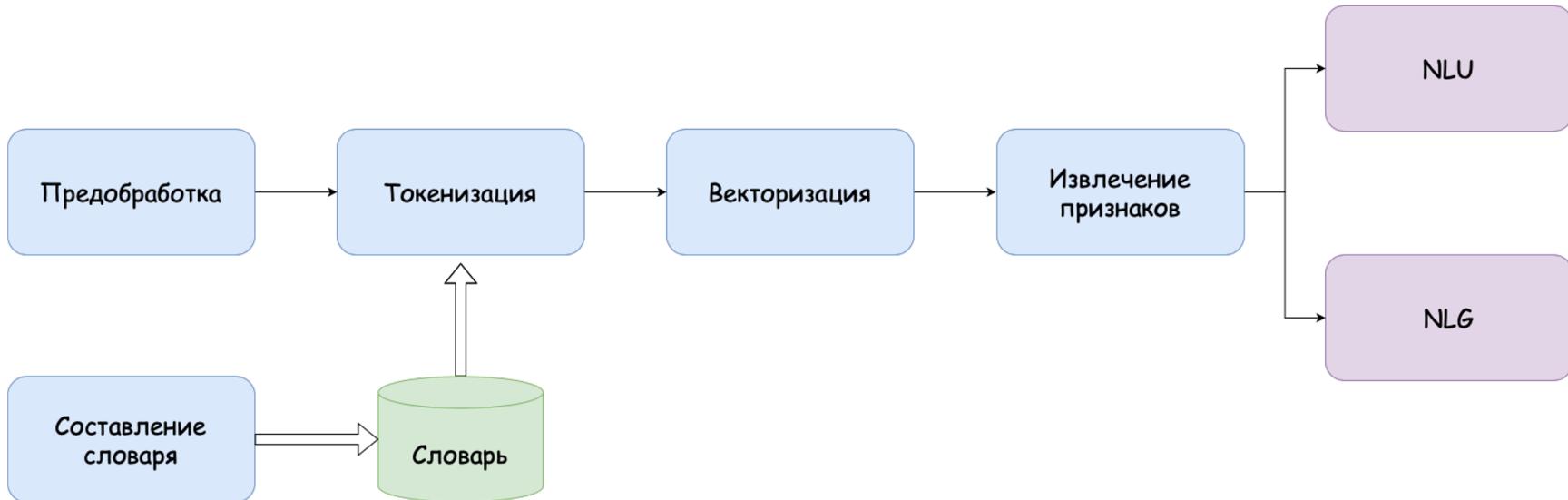
$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^n P(y_t|y_{<t}).$$

- Задача: научиться генерировать следующий токен по предыдущим
- Глобально решение задачи выглядит так
  - Представить текст в векторном виде
    - разобъем текст на токены
    - сформируем словарь
    - как-нибудь закодируем каждый токен из словаря
  - Модель => внутренние свойства и взаимосвязи токенов в последовательности
  - Внутреннее представление => для каждого токена из словаря оценить вероятность появления этого токена на следующей позиции
  - Вероятностное распределение для следующего токена => сэмплируем токен

# Работа с текстом

# Этапы работы с текстом

- Предобработка текста – удаление шума и нормализация
- Токенизация текста – разбиение текста на части для последующего кодирования
- Векторизация токенов
- Извлечение признаков
- Конечная задача: LM, классификация и другие задачи



# Предобработка

Цель: избавиться от шумного сигнала в тексте

- Приведение к нижнему регистру
  - Во многих задачах – лишний сигнал
  - Чтобы не раздувать словарь
- Если токенизируем на уровне слов
  - Удаление стоп-слов и спецсимволов
  - Унифицирование юникод-символов и разделителей
  - Нормализация **слов** — лемматизация / стемминг
  - Почему на уровне слов это важно?
- В современных системах — для каждого домена свои подходы

а LLM - это Смысл моей Жизни!

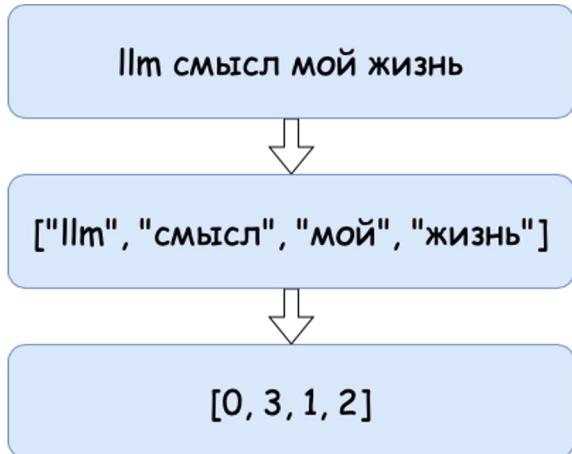


Ит смысл мой жизнь

# Токенизация

Цель: разделить текст (дискретный сигнал) на единицы (токены) для последующего кодирования

- За единицу токенизации можно брать слова либо символы
- Промежуточный вариант – «подслова» – позже рассмотрим алгоритмы
- Для всех кейсов в общем случае – словарь
- Вход: текст, выход: список токенов (либо их порядковые номера в словаре)



| Токен | Номер |
|-------|-------|
| Им    | 0     |
| мой   | 1     |
| жизнь | 2     |
| смысл | 3     |
| вмк   | 4     |

# Векторизация

Цель: перевод токенов из списка в признаковое пространство для дальнейшего извлечения признаков моделями

Как кодировать токен:

- Аналогия с категориальными фичами: есть словарь => one-hot encoding
- Можно сопоставить вектор в некотором признаковом пространстве и обучать его

Тогда вектор текста может быть функцией от векторов токенов (например, усреднение)

| Токен | Номер | Вектор          |
|-------|-------|-----------------|
| Им    | 0     | [1, 0, 0, 0, 0] |
| мой   | 1     | [0, 1, 0, 0, 0] |
| жизнь | 2     | [0, 0, 1, 0, 0] |
| смысл | 3     | [0, 0, 0, 1, 0] |
| вмк   | 4     | [0, 0, 0, 0, 1] |

# Классическая векторизация



# Мешок слов

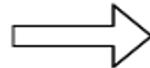
- Отдельный текст – документ
- Множество документов – корпус
- Мешок слов – рассматриваем каждый токен вне контекста и без учета порядка слов в предложении
- Вектор токена – статистика вхождений в документ (и корпус):
  - факт вхождения токена в документ
  - количество вхождений в документ
  - и другие
- По таким статистикам можно составить вектор документа
  - Сумма / объединение векторов частот
  - TF-IDF: количество вхождений токена в документ, нормированное на частоту встречаемости токена в документах корпуса



# One-hot encoding

- Вектор токена – бинарный вектор размерности словаря, единица на месте вхождения токена в словарь
- ОНЕ – факт вхождения
- Частотный подход: вместо факта вхождения можно считать количество вхождений
- Слишком частые слова – мб шум (как правило, стоп-слова)
- Слишком редкие слова – мб шум (часто опечатки)

| Словарь |
|---------|
| Им      |
| смысл   |
| моей    |
| жизни   |



|       | Им | смысл | моей | жизни |
|-------|----|-------|------|-------|
| Им    | 1  | 0     | 0    | 0     |
| смысл | 0  | 1     | 0    | 0     |
| моей  | 0  | 0     | 1    | 0     |
| жизни | 0  | 0     | 0    | 1     |

# TF-IDF

- Гипотеза: документ определяют слова, которые часто встречаются в одном документе, но в других документах этих слов нет
- TF — частота слова
- IDF — обратная доля документов, в которых встречается данное слово
- TF\*IDF — вектор документа
- Можно понизить размерность: PCA, LDA, NMF и др
- Можно использовать как инициализацию более сложных эмбеддингов

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

Term  $x$  within document  $y$

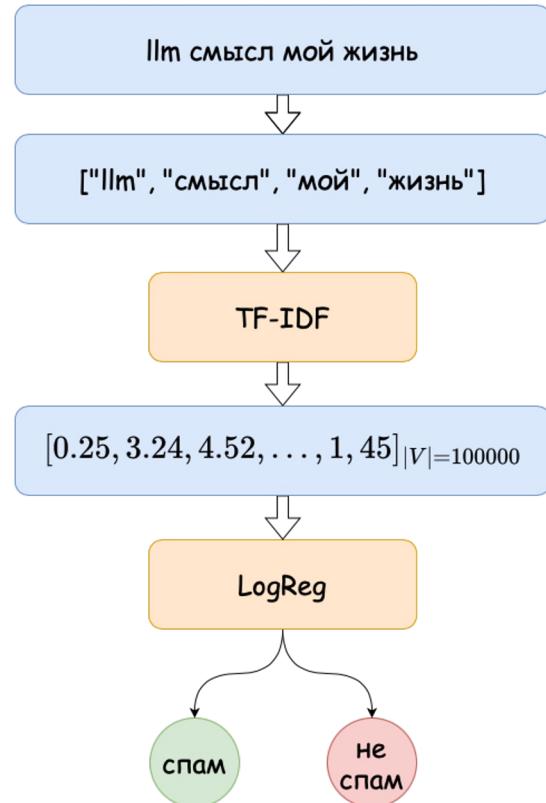
$tf_{x,y}$  = frequency of  $x$  in  $y$

$df_x$  = number of documents containing  $x$

$N$  = total number of documents

# Как использовать

- Задача: классификация текста (спам / не спам)
- Вход: корпус текстов, метки, текст для теста
  - тест не содержится в трейне
- По корпусу составили словарь
- Для каждого текста: через ОНЕ или TF-IDF получили вектор текста
  - размерность – размер словаря
- Выбрали и обучили классификатор
- Проверили на тестовом тексте – какие могут быть проблемы?



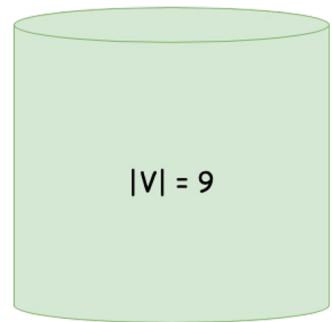
# Контекст

- Дистрибутивная гипотеза: слово определяется контекстом
- Решение: учитываем совместную встречаемость слов
- Простое решение: N-граммы
- Проблемы:
  - еще сильнее увеличивается словарь
  - зависимость может быть через несколько слов

LLM смысл моей жизни

| N | N-граммы                              |
|---|---------------------------------------|
| 1 | LLM<br>смысл<br>моей<br>жизни         |
| 2 | LLM смысл<br>смысл моей<br>моей жизни |
| 3 | LLM смысл моей<br>смысл моей жизни    |

Словарь N-грамм, N = 3



# Контекст

- Дистрибутивная гипотеза: слово определяется контекстом
- Будем учить модель предсказывать слово по контексту
- Word2Vec, Fasttext
  - Варианты: CBOW и Skip-gram

|     |       |     |   |       |      |       |
|-----|-------|-----|---|-------|------|-------|
| все | знают | что | m | смысл | моей | жизни |
|-----|-------|-----|---|-------|------|-------|

|     |       |     |   |       |      |       |
|-----|-------|-----|---|-------|------|-------|
| все | знают | что | m | смысл | моей | жизни |
|-----|-------|-----|---|-------|------|-------|

|     |       |     |   |       |      |       |
|-----|-------|-----|---|-------|------|-------|
| все | знают | что | m | смысл | моей | жизни |
|-----|-------|-----|---|-------|------|-------|

# Word2Vec

- Составляем по корпусу и фиксируем
- Единица словаря – привычное нам слово
- CBOW – восстанавливаем слово по небольшому контексту
- Skip-gram – учим, какие пары слов употребляются вместе, а какие нет

CBOW

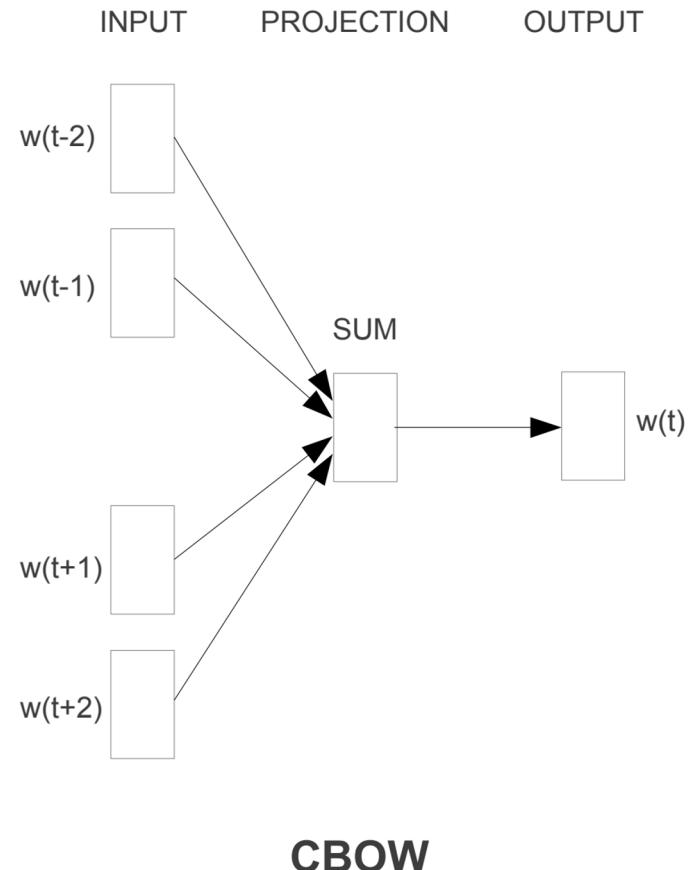
| feat 1 | feat 2 | feat 3 | feat 4 | target |
|--------|--------|--------|--------|--------|
| все    | знают  | m      | смысл  | что    |
| знают  | что    | смысл  | моей   | m      |
| что    | m      | моей   | жизни  | смысл  |

Skip-gram

| feat     | target |
|----------|--------|
| знают    | m      |
| m        | m      |
| мемы     | m      |
| старость | m      |

# Word2Vec CBOW

- Обучение:
  - Скользящим окном проходимся по тексту
  - Середина окна – таргет ( $y$ )
  - Остальные слова в окне (контекст) – “признаки” ( $X$ )
- Использование:
  - Проекцию  $h_t$  используем как векторное представление слова (эмбеддинг слова)



# Word2Vec CBOW

- Единица словаря – привычное нам слово
  - А если встретим новое слово на инференсе?
- В обучении видим задачу LM – предсказываем вероятность слова (центр контекста)
- Softmax по всему словарю - дорого
  - Используют иерархический Softmax

$$h_t = W_{d*|V|} \sum_{w_i \in context(w)} OneHot(w_i)_{|V|}$$

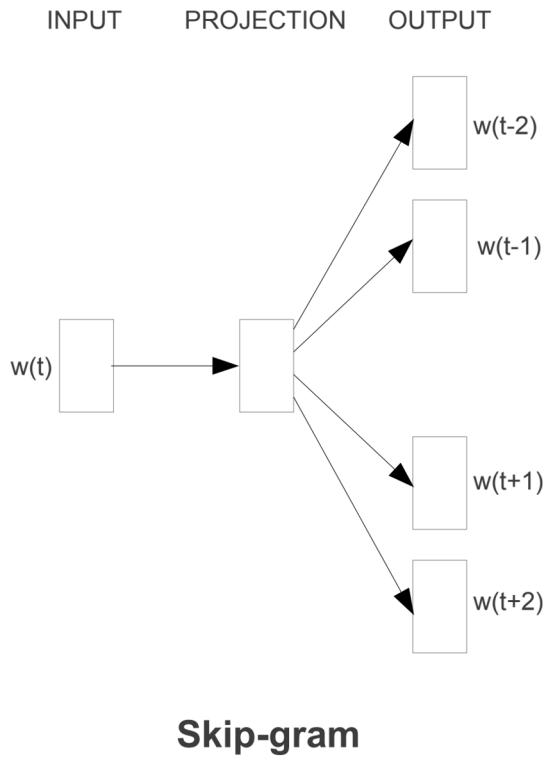
$$P(w|context(w)) = SoftMax(U_{|V|*d} h_t)$$

$$L = -\log P(w|context(w))$$

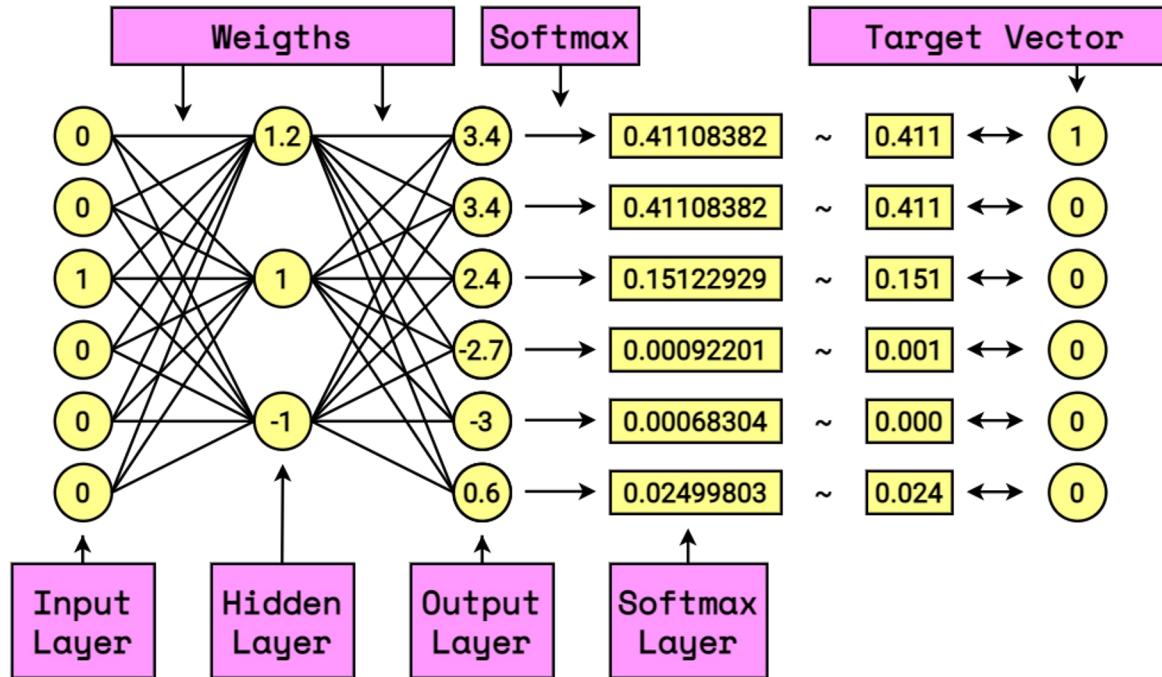
# Word2Vec Skip-gram

- По слову предсказываем одно слово из контекста
- По одному контексту составляется несколько обучающих пар – позитивные и негативные
- Умный Negative Sampling
  - Все негативные примеры из словаря считать дорого и некорректно => семплируем
  - У частых слов вероятность семплирования меньше (почему?)

$$L = - \sum_{j=-m, j \neq 0}^m [\log(\hat{y}_{c+j}) + \sum_{k \in Neg_{c+j}} \log(1 - \hat{y}_k)]$$



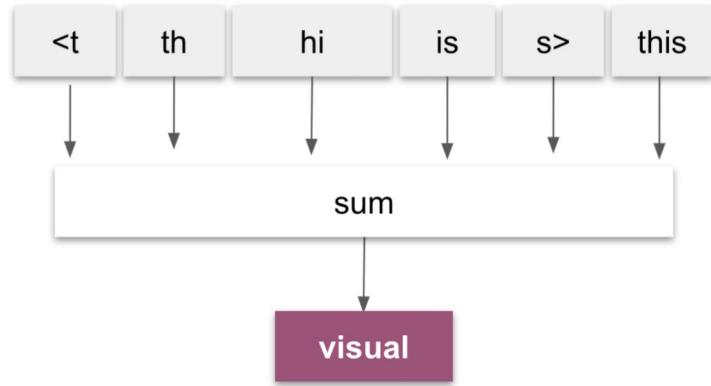
# Word2Vec внутри



[Источник](#)

# Fasttext

- Word2Vec + N-граммы символов + их хеширование
- Код слова = сумма кодов N-грамм
- Решается проблема несловарных слов
- На практике: сильный бейзлайн



# GloVe

- Составляем словарь
- Составляем матрицу совместной встречаемости слов – A
  - рассматриваем слово  $i$  в контексте слова  $j$
  - кол-во совместных употреблений  $#ij$  на расстоянии  $k$  слов
- Матрица со-встречаемости - высокой размерности (какой?)
- Приближаем произведением 2х низкоранговых матриц (+ сдвиги)
- По-разному взвешиваем редкие и частые пары

$$J = \sum_{i, j=1}^V F(x)(w_i^T \tilde{w}_k + b_i + \tilde{b}_j - \log X_{ij})^2$$

$$F(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^\alpha & x < x_{max} \\ 1 & x \geq x_{max} \end{cases}$$

# Маленькое резюме

- Рассмотрели классические подходы:
  - Частотные подходы (TF-IDF)
  - Word2Vec
  - Fasttext
  - GloVe
- Обсудим?

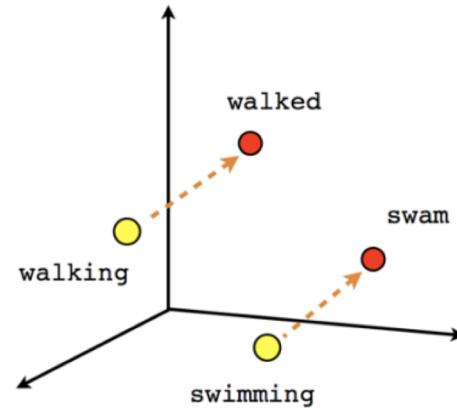


# Под капотом эмбеддингов



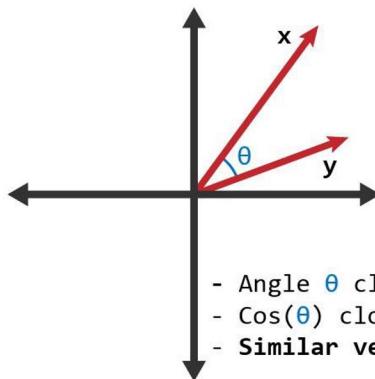
# Эмбеддинги в пространстве

- Похожие слова должны иметь близкие эмбеддинги
- Непохожие - далекие
- у W2V, Fasttext, GloVe работает векторная арифметика на эмбеддингах
- Выучиваем некоторую семантику слова

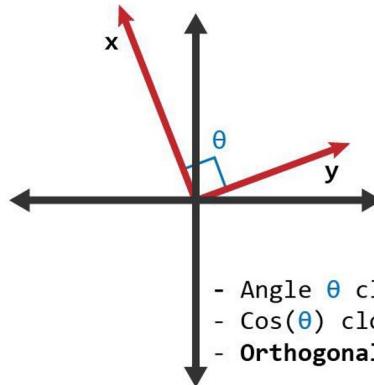


# Как измерять качество эмбеддингов

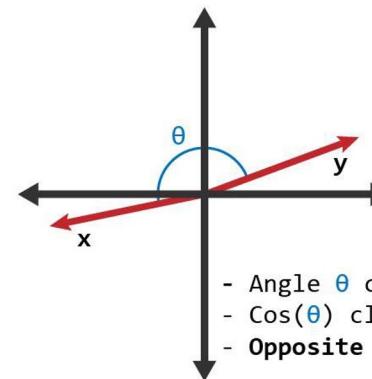
- Качество на downstream-задачах
- Равномерность эмбеддингов и отдельных компонент в пространстве
- Качество поиска по тексту, косинусная близость представлений похожих текстов



- Angle  $\theta$  close to 0
- $\text{Cos}(\theta)$  close to 1
- **Similar vectors**



- Angle  $\theta$  close to 90
- $\text{Cos}(\theta)$  close to 0
- **Orthogonal vectors**



- Angle  $\theta$  close to 180
- $\text{Cos}(\theta)$  close to -1
- **Opposite vectors**

# Как измерять качество эмбеддингов

- Бенчмарки:
  - STS – semantic textual similarity
    - Насколько 2 текста похожи
  - MIRACL – Multilingual Information Retrieval Across a Continuum of Languages
    - измерение качества поиска по многоязычному корпусу
  - MTEB – Massive Text Embedding Benchmark
    - крупнейший бенчмарк для текстовых эмбеддингов
    - 8 задач, 58 датасетов, 112 языков

# MTEB

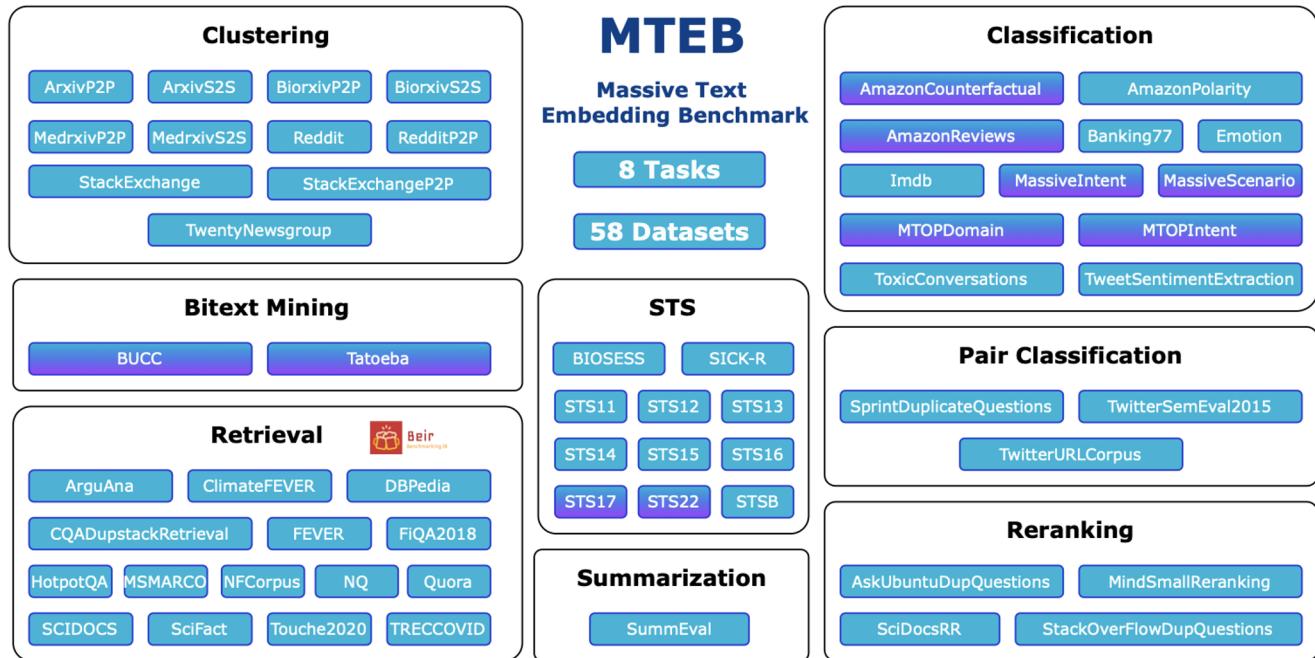


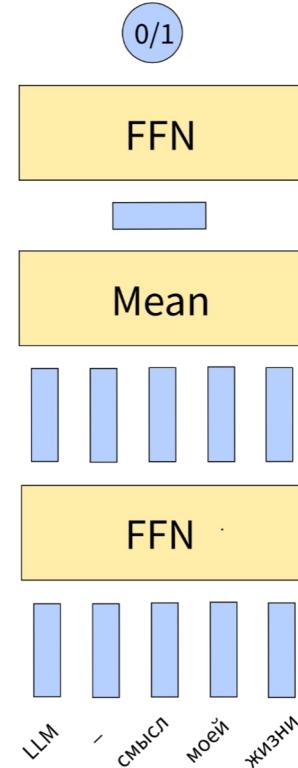
Figure 1: An overview of tasks and datasets in MTEB. Multilingual datasets are marked with a purple shade.

# Эмбеддинги и нейронные сети



# Эмбеддинги и нейронные сети

- Классификация текста
  - Текст в токены
  - Токены в эмбеддинги
  - Эмбеддинги в любую нейронную сеть
  - Получаем скрытое состояние текста
  - Классифицируем по скрытому состоянию
- Как получить эмбеддинги?



# Эмбеддинги и нейронные сети

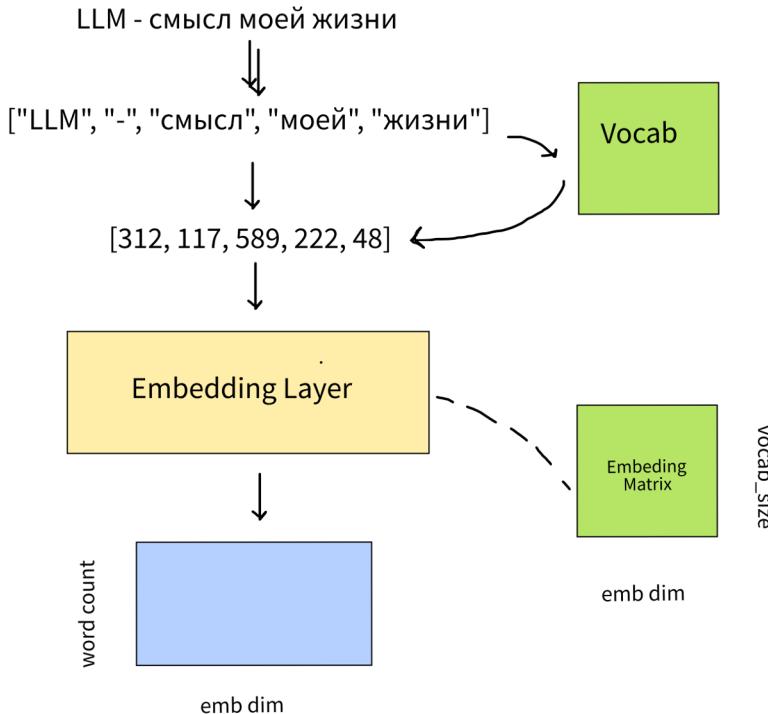
- Стандартные способы получить представления токенов:

Взять предобученные:

- Word2Vec
- Fasttex
- GloVe

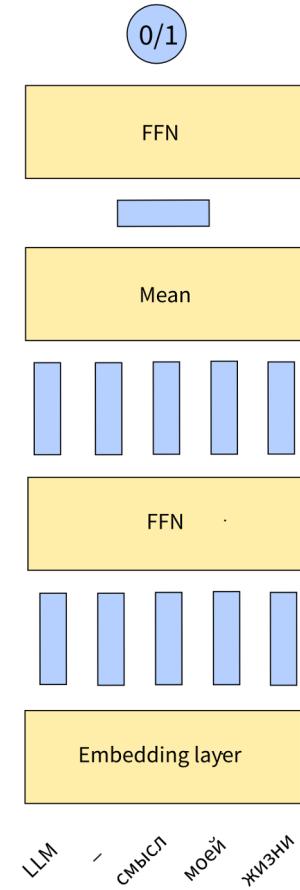
- Обучать эмбеддинги вместе с сетью

- Обучаемый слой эмбеддинга в нейронной сети
- nn.Embedding() в pytorch



# Embedding Layer

- Составляем словарь токенов
- Сопоставляем каждому токену из словаря случайный вектор заданного размера
- Обучаем эти вектора вместе с сетью
- Инициализация
  - $N(0, 1)$
  - Предобученными векторами
- В современных сетях обычно используют Embedding Layer



# Немного контекста

# Контекст

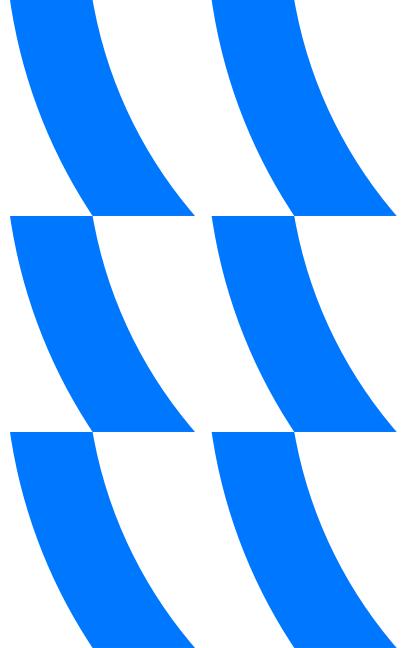
Представим, что используем уже предобученный Word2Vec

Различаются ли эмбеддинги у слова "среда" в фразах:

"Среда обитания"

"Среда - отличный день для похода в бар"

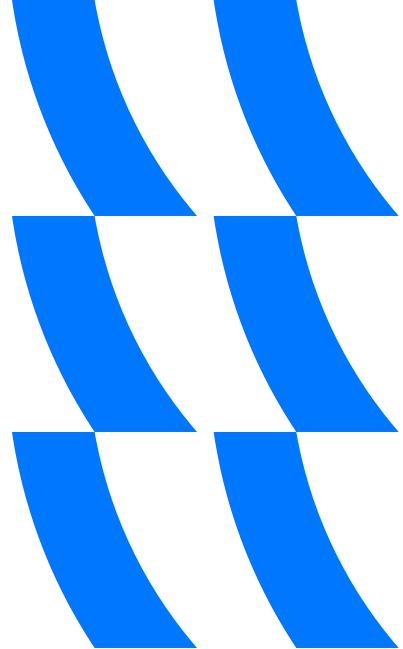
Должны ли?



# Контекст

Гипотеза: слово определяется контекстом.

Но: в разных контекстах слова означает разные вещи  
Контекст надо учитывает и во время тренировки, и во время  
инференса



# Проблемы классических моделей

## Частотные (TF-IDF)

большая размерность, разреженные векторы, не учитывается порядок, слова рассматриваются вне контекста, не учитывается контекст при инференсе

## Word2Vec

при инференсе не учитывается контекст, при тренировке контекст ограничен, OOV

## FastText

при инференсе не учитывается контекст, при тренировке контекст ограничен

## GloVe

при инференсе не учитывается контекст, при тренировке контекст ограничен

# А ЧТО С ЯЗЫКОВЫМ моделированием?

Как обучаются модели:

- Частотные подходы (TF-IDF) – подсчет статистик по корпусу
- GloVe – MSE-задача, аппроксимация совместной встречаемости
- Word2Vec – задача языкового моделирования
- Fasttext – задача языкового моделирования

Как используются эти модели:

- классификация текста / токенов
- поиск похожих текстов
- и др.

# А ЧТО С ЯЗЫКОВЫМ моделированием?

Зачем нужны рассмотренные подходы:

- Не используются для генерации текста
- Используются как (предобученные) эмбеддинги для конечных задач

Зачем рассматриваем:

- Могут использоваться как инициализация эмбеддингов в нейросетях
- Чтобы решать LM, нужно много чего делать с текстами (искать/удалять похожие)
- => Необходимо понимать, как получить хорошие эмбеддинги

# Моделируем язык



# Простейшая языковая модель

- По корпусу текстов считаем статистику совместных встречаемостей слов
- Предсказываем следующее слово на основе n-грамм
- Марковское свойство – вероятность появления слова зависит только от фиксированного количества предыдущих слов

Задача LM

$$p(w_1, \dots, w_n) = \prod_{i=1}^n p(w_i | w_1, \dots, w_{i-1})$$

Марковское свойство ( $k > 0$ ):

$$p(w_1, \dots, w_n) \approx \prod_{i=1}^n p(w_i | w_{i-k}, \dots, w_{i-1})$$

LM на основе n-грамм:

$$p(x_t | x_{t-n}, \dots, x_{t-1}) = \frac{\#(x_{t-n}, \dots, x_{t-1}, x_t) + \alpha}{\#(x_{t-n}, \dots, x_{t-1}) + \alpha |V|}$$

# LM на основе n-грамм: проблемы

- Проблема несловарных слов (OOV)
  - Решение: сглаживание по Лапласу, Backoff и др.
- Чем больше k, тем более переобученная модель
- Много памяти для хранения частот
- Ограниченнная длина контекста при k > 0

Задача LM

$$p(w_1, \dots, w_n) = \prod_{i=1}^n p(w_i | w_1, \dots, w_{i-1})$$

Марковское свойство ( $k > 0$ ):

$$p(w_1, \dots, w_n) \approx \prod_{i=1}^n p(w_i | w_{i-k}, \dots, w_{i-1})$$

LM на основе n-грамм:

$$p(x_t | x_{t-n}, \dots, x_{t-1}) = \frac{\#(x_{t-n}, \dots, x_{t-1}, x_t) + \alpha}{\#(x_{t-n}, \dots, x_{t-1}) + \alpha |V|}$$

# Включаем нейронные сети

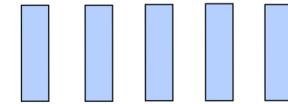
- Реализуем полносвязную нейронную сеть (FFN)
- Обучим на задачу LM: по тексту предсказать следующее "слово"
- Используем марковское свойство: зафиксируем длину контекста
- Текст -> токены -> эмбеддинги => последовательность векторов
- Агрегируем информацию из всех токенов для получения логитов
- На выходе – распределение вероятностей для следующего токена

[0.1, 0.05, ..., 0.0, 0.4]

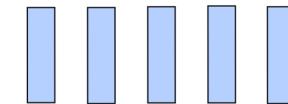
FFN + Softmax



Mean



FFN

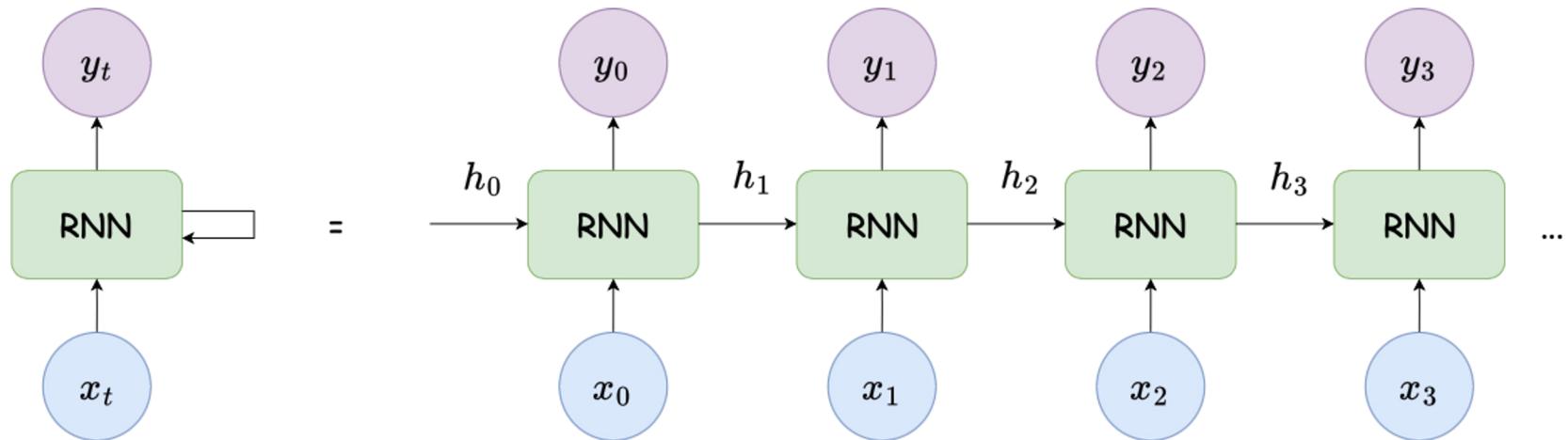


Embedding layer

LLM – смысл моей жизни

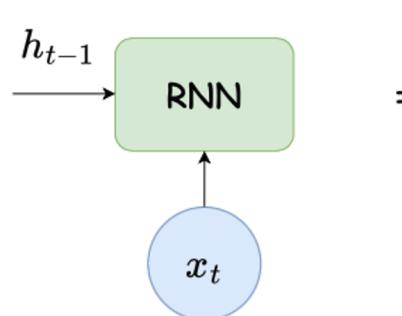
# Рекуррентные сети

- Немарковская модель (что это значит?)
- Модель - один RNN-блок, через который проходят все элементы последовательности
- В RNN всегда подается два вектора: эмбеддинг текущего слова и текущий hidden\_state



# Рекуррентные сети

- Проблема – затухающие и взрывающиеся градиенты
- LSTM и GRU – “прокаченные” RNN, решающие эту проблему (но суть та же)



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$

$$\theta = \{W_{hh}, W_{xh}, W_{hy}\}$$

не зависят от  $t$

$$\frac{\partial h_t}{\partial W_{hh}} = \sum_{k=1}^t \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W_{hh}}$$

# Рекуррентные сети

- Текст – последовательность токенов  
=> можем использовать RNN
- Можно использовать для LM
  - Во многих работах так и делают
- Информация о всей последовательности сохраняется в векторе “памяти”
  - Тот самый вектор  $h$ , который подаем на вход вместе с  $x$
- Длина последовательности формально неограничена
  - Фактически “память” у модели не очень большая
- Можем подавать на вход тексты произвольной длины
  - Но если обрабатываем тексты батчами, то придется выравнивать длины
  - Фиксируем длину, вводим спецтокен ( $<\text{pad}>$ ), дополняем / обрезаем

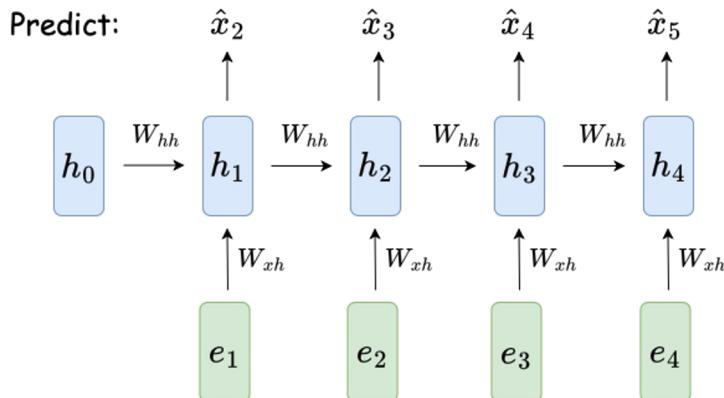
# Рекуррентные сети: обучение

Loss:  $L_1(\theta)$   $L_2(\theta)$   $L_3(\theta)$   $L_4(\theta) \longrightarrow \Sigma$

Скрытое состояние на очередном шаге

Target:  $x_2^-$   $x_3^{\text{смысл}}$   $x_4^{\text{моей}}$   $x_5^{\text{жизни}}$

$$h_t = \tanh(W_{d*|V|}x_{t-1} + U_{d*d}h_{t-1} + b)$$



Языковое моделирование

$$g(h_t) = \text{softmax}(R_{|V|*d}h_{t-1} + c)$$

Обучение

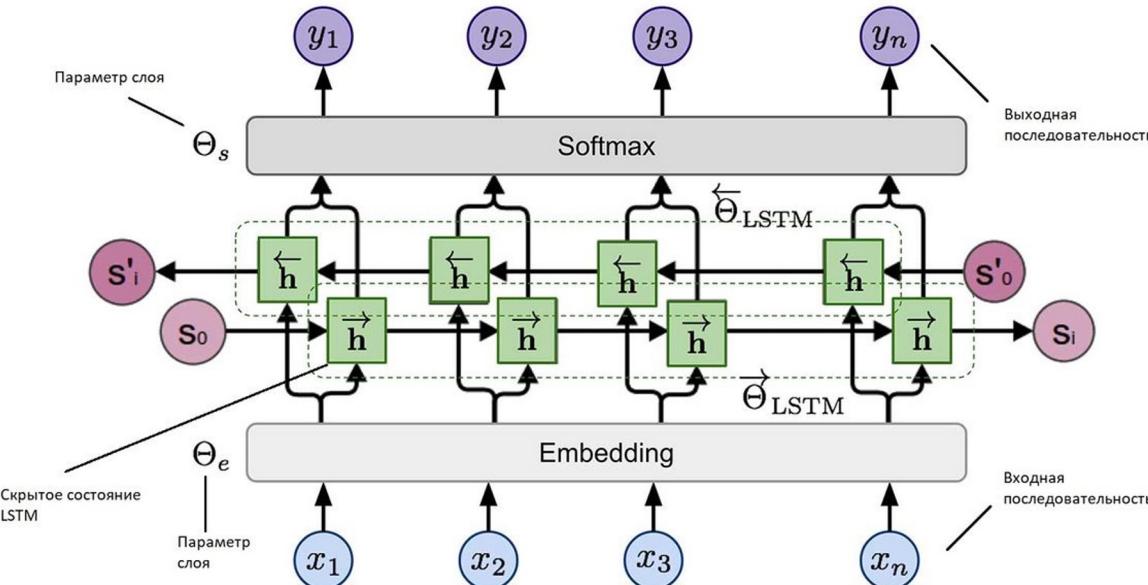
$$p(x_t = w | x_{<t}) = g(h_t), w = 1 \dots |V|$$

LLM - смысл моей  
 $x_1$   $x_2$   $x_3$   $x_4$

$$-\frac{1}{m} \sum_{i=1}^m \sum_{t=1}^{n\_tokens_i} \log p(x_t^{(i)} | x_1^{(i)}, \dots, x_{t-1}^{(i)}) \longrightarrow \min$$

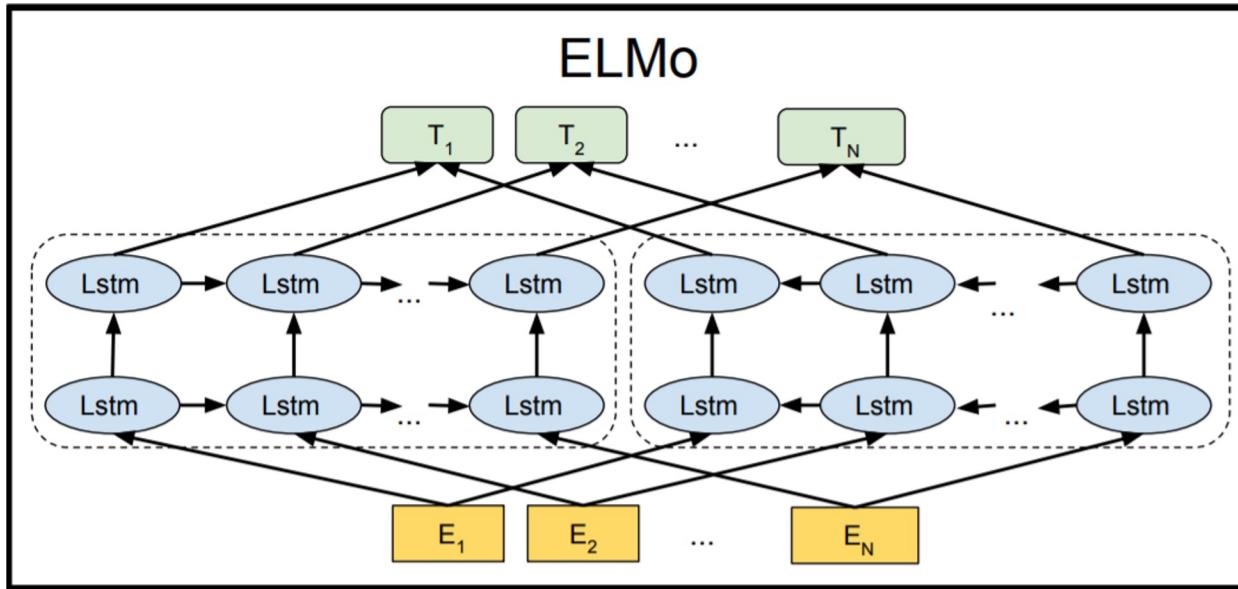
# ELMo

- Языковая модель
- Двунаправленная LSTM с несколькими слоями
- Выход - функция от всего входного предложения, а не ограниченного контекста



# ELMo

- Character-based
- 2 biLSTM
- Долгое время была одной из самых популярных для LM



# ELMo

- Учитывает контекст и на тренировке, и на teste
  - Контекст заключен в векторе `hidden_state`
- Контекст учитывается и левосторонний, и правосторонний
  - При этом, контекст не является двусторонним :)
  - Здесь двусторонний контекст = подглядывание в будущее
- На разных уровнях разный уровень абстракции информации из токенов
  - Поэтому берем выходы с нескольких слоев -> взвешенная сумма
- Pretrain + "fine-tuning"
  - biLM pretrain на большом корпусе
  - Обучаем линейную комбинацию выходов из всех слоев на конечную задачу

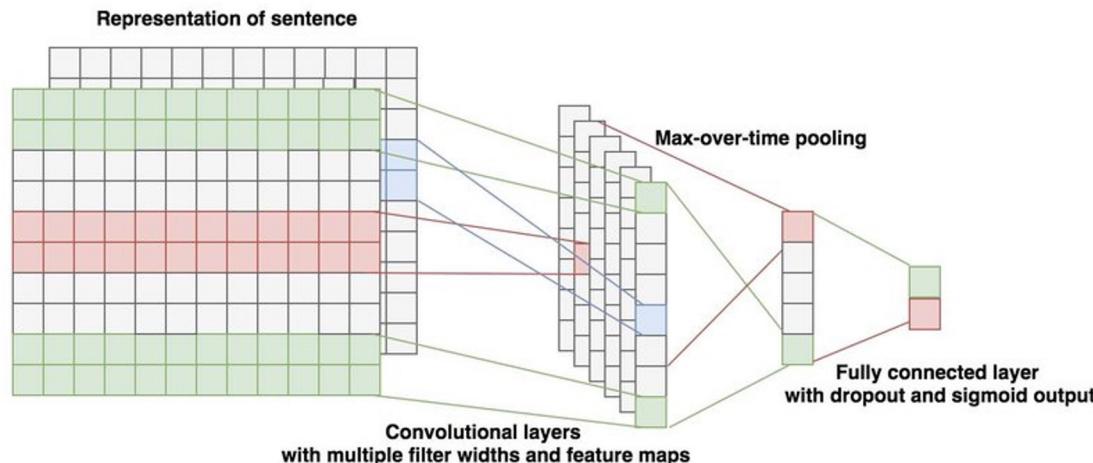
# Проблемы RNN

- Взрывы и затухания градиентов
  - Решаются использованием LSTM/GRU
- Важно смотреть на контекст и слева, и справа
  - Частично решается за счет двунаправленности
- "Короткая" память - проблемы для длинных текстов (забывание контекста)
- Последовательное вычисление => не можем распараллелить



# Сверточные сети

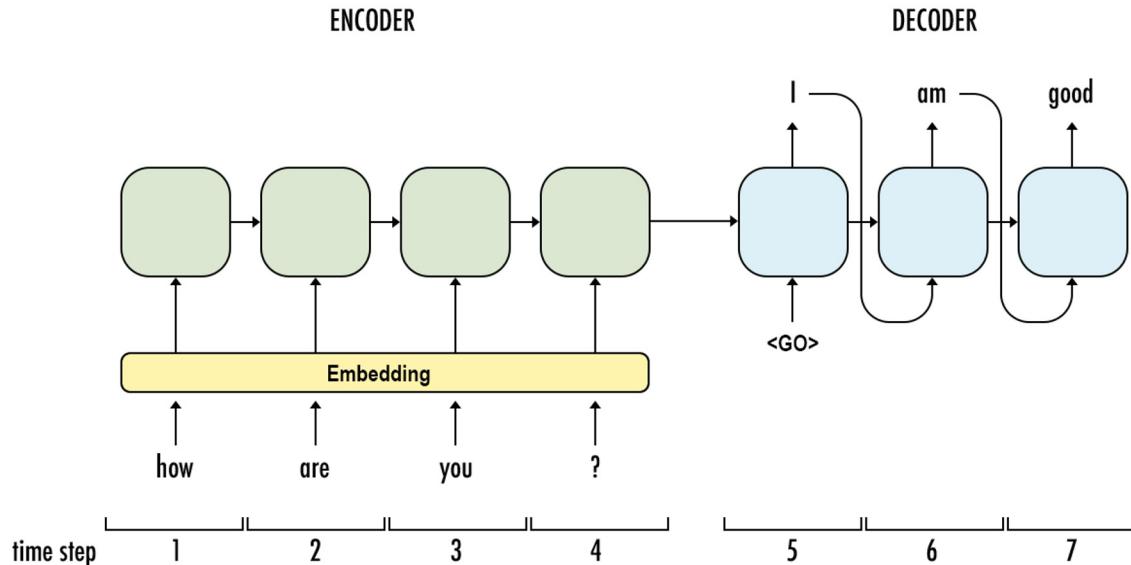
- Отличие от изображений: свертка по последовательности, каждым проходом захватывает **часть** элементов последовательности и **все** признаки
- Ряд работ по LM на свёртках
- Можно комбинировать RNN и свертки



# Seq2Seq и Attention

# Задача sequence-to-sequence

- Языковое моделирование напрямую используется в задаче seq2seq
- На входе: последовательность
- На выходе: новая последовательность при условии входной последовательности

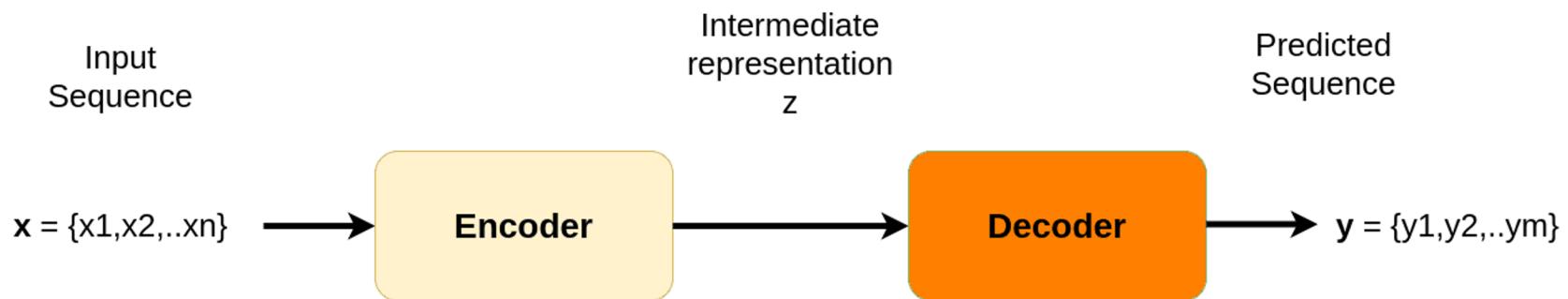


# Давайте не путать

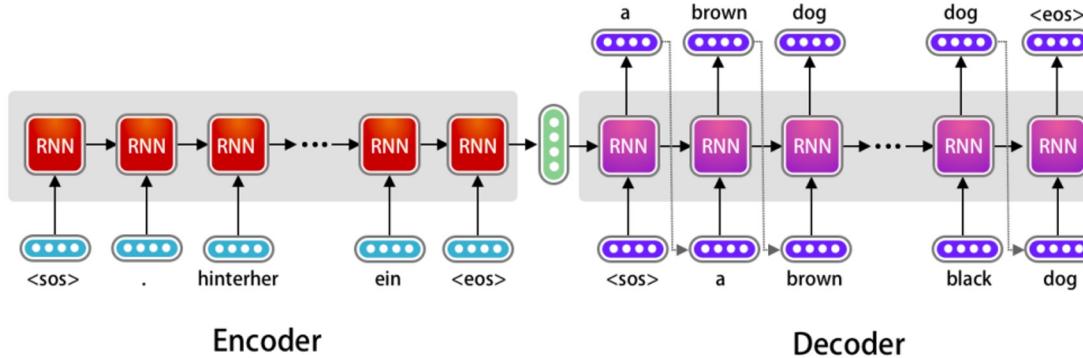
- Языковое моделирование – моделируем вероятностное распределение на множестве словаря (для одной языковой единицы)
- Seq2Seq – задача генерации выходной последовательности по некоторой входной
- Генерация текста – подзадача Seq2Seq, где обе последовательности – тексты

# Задача sequence-to-sequence

Как правило решается с помощью нейронных сетей, архитектурой encoder-decoder



# Модель sequence-to-sequence на основе RNN

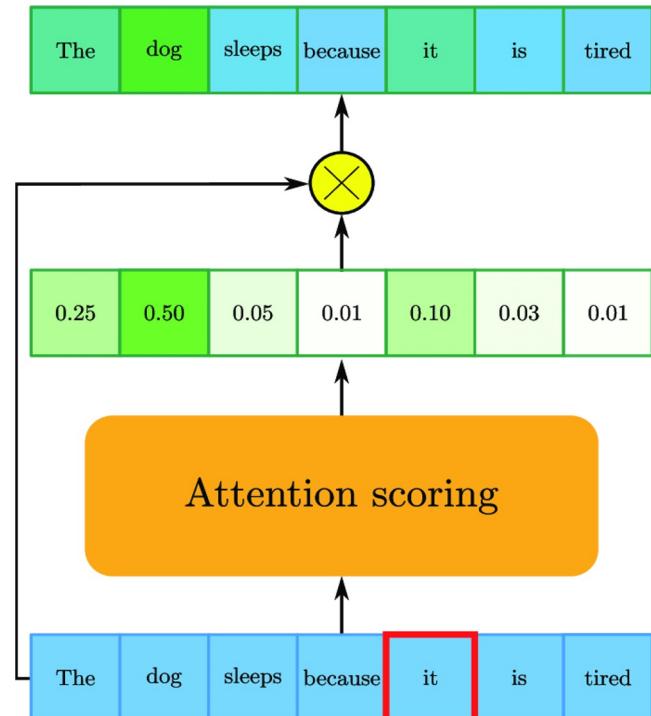


Проблемы:

1. RNN - вектор контекста обращается лишь к последнему элементу, рекурсивно собирающему в себе информацию о всей последовательности
2. С каждым новым шагом более старая информация забывается

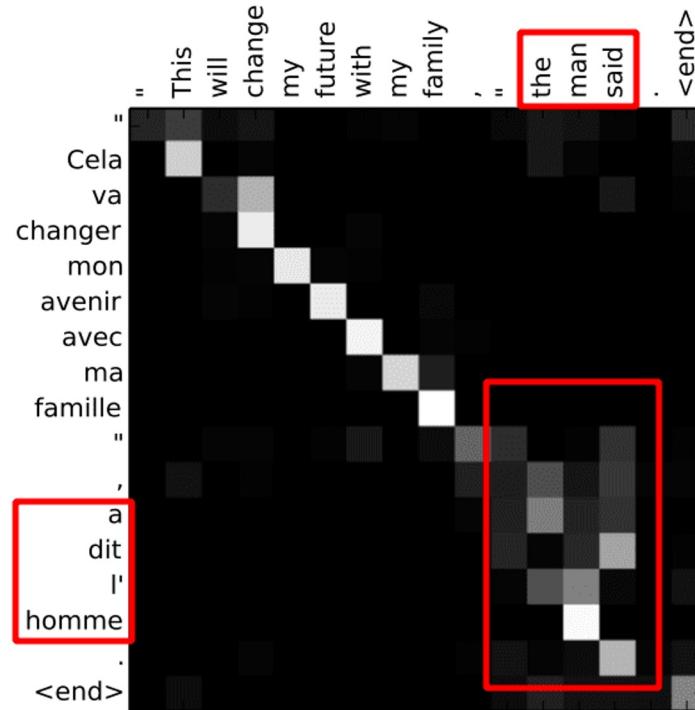
# Механизм внимания (attention)

- Идея: вектор контекста должен обращаться ко всем токенам входной последовательности в равной степени
- Attention – “умное усреднение”, взвешенное среднее скрытых состояний
- Веса для этого среднего – динамические, вычисляются по паре скрытых представлений
- Не стоит путать attention и self-attention

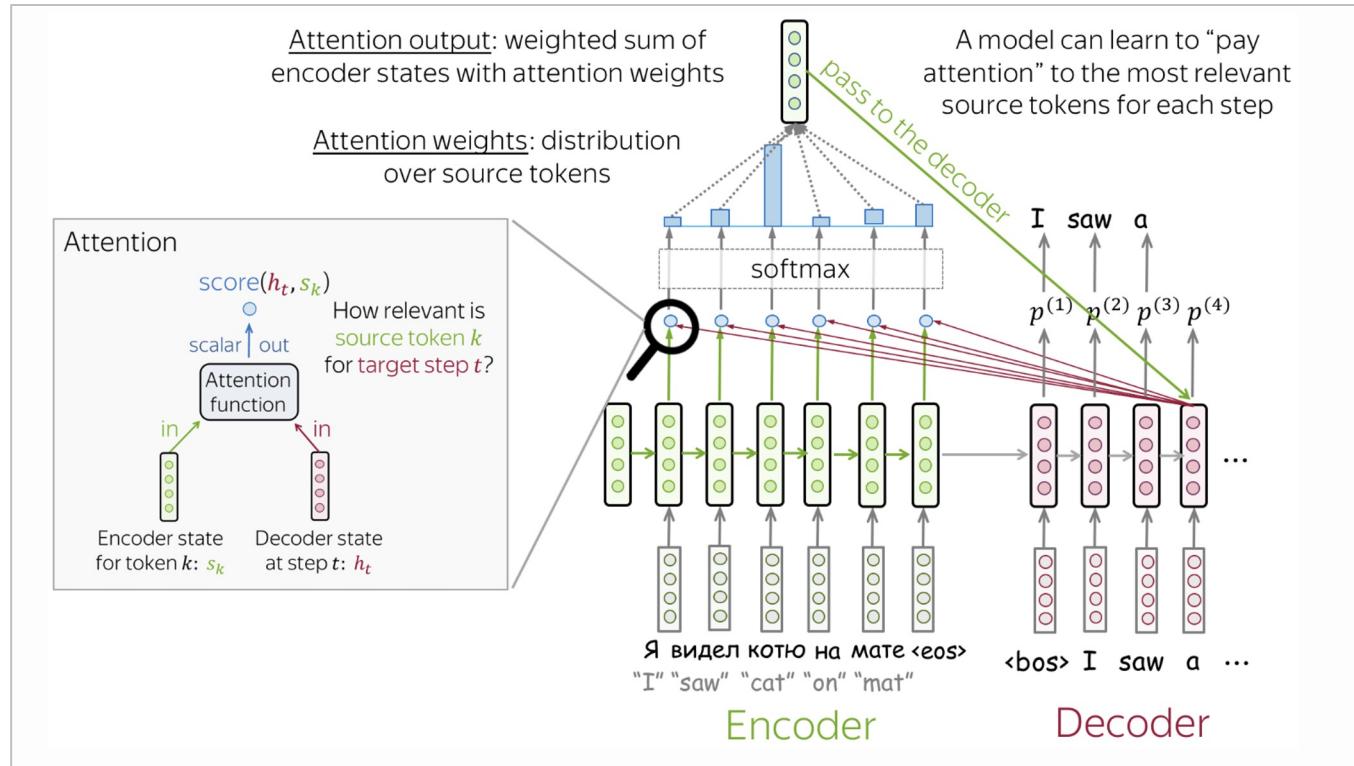


# Механизм внимания (attention)

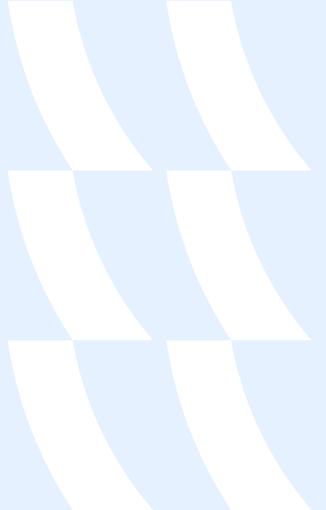
Выходные токены смотрят на каждый токен входной последовательности с разным весом (разная степень внимания)



# Модель seq2seq на основе RNN и Attention



# Как моделируем язык



1

Готовим словарь,  
токенизируем текст

2

Любым способом  
получаем эмбеддинги  
токенов

3

Обрабатываем  
последовательность  
эмбеддингов одной из  
следующих архитектур:  
- Полносвязные  
- Рекуррентные  
- Сверточные

4

Получаем внутреннее  
представление  
последовательности из  
сети

# Как моделируем язык



5

Проектируем на  
пространство  
размерности словаря

6

Получаем вероятностное  
распределение на  
следующий токен,  
выбираем токен

# Роль архитектуры

- Архитектура определяет лишь то, как из последовательности эмбеддингов будет извлекаться информация – готовим внутреннее представление
- Как правило не зависят от архитектуры сети:
  - токенизация
  - перевод токенов в векторное пространство (`nn.Embedding`)
  - выбор итогового токена из распределения (`softmax + sampling`)

# Трансформеры

- Суть – другая архитектура для получения представлений токенов
- Лучше учитывается контекст для каждого токена
- Подробнее – в следующей лекции

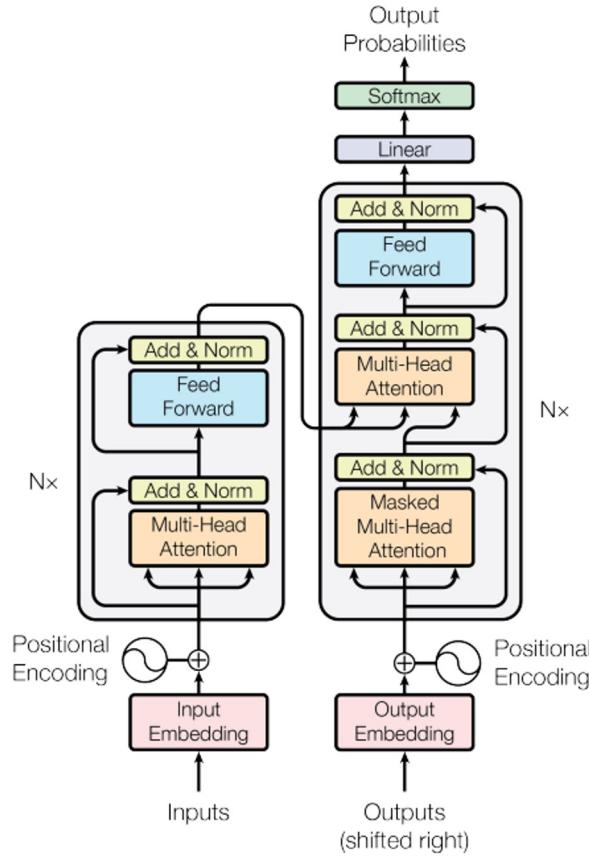


Figure 1: The Transformer - model architecture.

# Сегодня мы узнали

1. Задача языкового моделирования
2. Предобработка, токенизация, векторизация текста
3. Как несложно получить эмбеддинги: мешок слов, Word2Vec, Fasttext, GloVe
4. Как несложно промоделировать язык: n-граммы, MLP, RNN (ELMo), CNN, Attention
5. В предвкушении трансформера

# Что почитать

К курсу:

- А. Дьяконов, DL: <https://youtu.be/7eQJ2WiKTIA?si=OzAY3tDfikxiXEIj>
- Е. Войта: [https://lena-voita.github.io/nlp\\_course.html](https://lena-voita.github.io/nlp_course.html)
- LLM: <https://docs.cohere.com/docs/lmu>

К текущей лекции:

- Ссылки на слайдах
- RNN LM: <https://docs.chainer.org/en/stable/examples/ptb.html>
- Seq2Seq: [https://lena-voita.github.io/nlp\\_course/seq2seq\\_and\\_attention.html](https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html)

К следующей лекции:

- Attention: <https://theaisummer.com/attention/>
- Illustrated BERT: <http://jalammar.github.io/illustrated-bert/>

# Спасибо за внимание!

Дмитрий Калашников, разработчик-исследователь