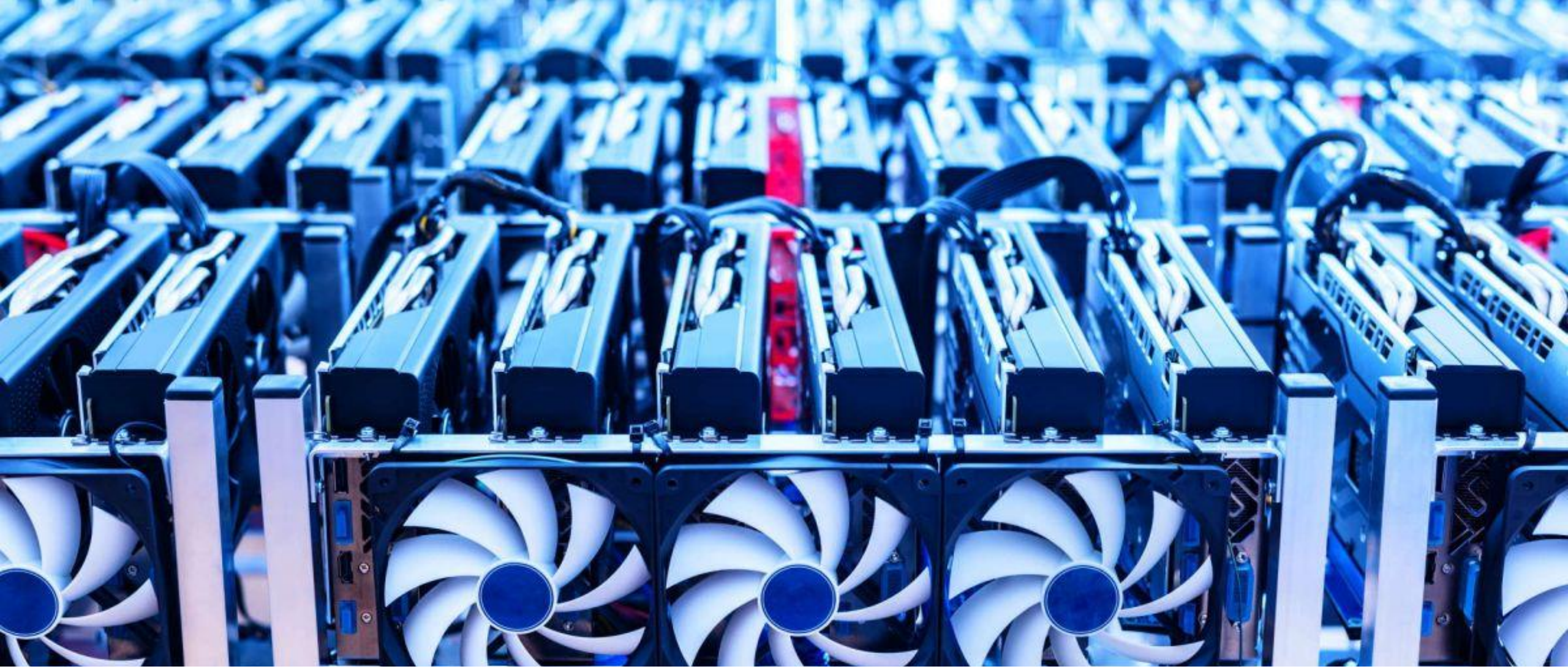
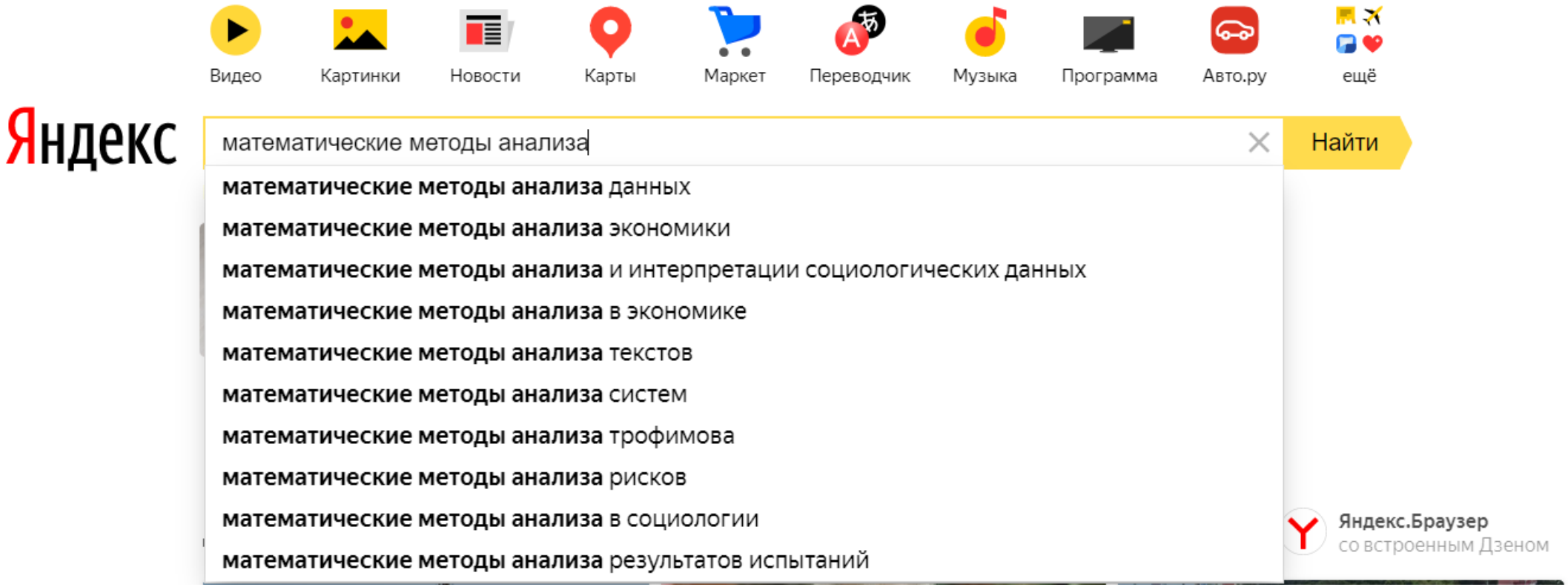


Языковое моделирование

Попов Артём, осень 2022

Natural Language Processing

Поисковые подсказки



Можно ли создать такую систему на основе энкодера-декодера?

Постановка задачи языкового моделирования

Языковая модель – модель, решающая одну из двух задач:

- оценивание вероятности появления произвольной последовательности слов $(w_1, \dots, w_n)$ в тексте, $w_i \in W, n \in \mathbb{N}$
- оценивание вероятности появления произвольного слова w_n после произвольной последовательности слов $(w_1, \dots, w_{n-1})$, $w_i \in W, n \in \mathbb{N}$

Задачи являются взаимозаменяемыми в силу цепного правила (chain rule) и формулы условной вероятности:

$$p(w_1, \dots, w_n) = p(w_n | w_1, \dots, w_{n-1}) \dots p(w_2 | w_1) p(w_1)$$

$$p(w_n | w_1, \dots, w_{n-1}) = \frac{p(w_1, \dots, w_n)}{p(w_1, \dots, w_{n-1})}$$

Марковское правило

Проблема: чем больше n, тем меньше будет обучающих данных для оценивания вероятности $p(w_n|w_1, \dots, w_{n-1})$

Решение: распределение w_n зависит не от всех предыдущих слов, а только от k

$$p(w_n|w_1, \dots, w_{n-1}) \approx p(w_n|w_{n-k}, \dots, w_{n-1})$$

$$p(w_1, \dots, w_n) = p(w_1) \prod_{i=2}^{n} p(w_i|w_{i-k}, \dots, w_{i-1})$$

Обучающая выборка и критерии качества

Для обучения модели используется неразмеченный корпус D.

Критерии качества:

- Метрики качества из классификации или ранжирования для задачи предсказания следующего слова по предыдущим
- Правдоподобие (может оцениваться по отложенной выборке)

$$\mathcal{L}(D) = \sum_{d \in D} \log p(d)$$

- Перплексия – аналог правдоподобия

$$P(D) = \exp\left(-\frac{1}{|D|}\mathcal{L}(D)\right)$$

Приложения языкового моделирования

Задачи, связанные с генерацией данных:

- Поисковые подсказки
- Автодополнение кода
- Генерация текста
- Генерация данных, имеющих последовательную структуру

Постобработка результатов для задач с сложным выходом:

- Исправление опечаток
- Распознавание символов
- Распознавание речи
- Машинный перевод

N-граммные (статистические) языковые модели

Построим численную оценку вероятности для фиксированного k:

$$p(w_n|w_{n-k}, \dots, w_{n-1}) = \frac{c(w_{n-k}, \dots, w_{n-1}, w_n)}{c(w_{n-k}, \dots, w_{n-1})},$$

где $c(w_i, \dots, w_{i+j})$ – число последовательностей $(w_i, \dots, w_{i+j})$ в D

Обучение модели – запоминание статистик последовательностей.

N-граммные модели небольшой граммности

Униграммная модель, $k = 0$

$$p(w_1, \dots, w_n) = p(w_n) \dots p(w_1), \qquad p(w_i) = \frac{c(w_i)}{\sum_{d \in \mathrm{D}} |d|}$$

Биграммная модель, $k = 1$

$$p(w_1, \dots, w_n) = p(w_n|w_{n-1}) \dots p(w_2|w_1)p(w_1)$$

Триграммная модель, $k = 2$

$$p(w_1, \dots, w_n) = p(w_n|w_{n-1}, w_{n-2}) \dots p(w_3|w_2, w_1)p(w_2|w_1)p(w_1)$$

Проблема первого токена

Как в $p(w_1, \dots, w_n) = p(w_n|w_{n-1}, w_{n-2}) \dots p(w_3|w_2, w_1)p(w_2|w_1)p(w_1)$ оценивать $p(w_2|w_1)$ и $p(w_1)$?

1. Использовать для оценки $p(w_2|w_1)$ и $p(w_1)$ модели меньшей граммности
2. При обучении дополнять каждую последовательность в начале k специальными токенами <START>

$$p(w_2|w_1) \equiv p(w_2|w_1, < S >)$$
$$p(w_1) \equiv p(w_1| < S >, < S >)$$

Сглаживание Лапласа (add-k smoothing)

В n-граммной модели большое количество нулевых вероятностей!

Если в последовательности любой длины хотя бы одна n-грамма отсутствует в обучении, её вероятность будет нулевой.

Для уменьшения числа нулевых вероятностей можно использовать сглаживание Лапласа:

$$p(w_n|w_{n-k},\dots,w_{n-1}) = \frac{c(w_{n-k},\dots,w_{n-1},w_n)+\alpha}{c(w_{n-k},\dots,w_{n-1})+\alpha|W|}$$

Сглаживание Лапласа может повлиять на порядок вероятностей и форму распределения.

Интерполяционное сглаживание (Jelinek-Mercer smoothing)

Вероятность оценивается смесью из нескольких n-граммных моделей разного порядка:

$$p(w_n|w_{n-k},\dots,w_{n-1}) = \sum_{c=1}^{k} \lambda_c \hat{p}(w_n|w_{n-c},\dots,w_{n-1})$$

$$\sum_{c=1}^{k} \lambda_c = 1, \qquad \lambda_c \geq 0$$

Самый частый вариант при использовании статистический моделей на практике!

Откат последовательности (Katz smoothing)

Если модель нужного порядка даёт нулевую вероятность, обращаемся к модели меньшего порядка с понижающим множителем:

$$p(w_n|w_{n-k}, \dots, w_{n-1}) =$$

$$= \begin{cases} \hat{p}(w_n|w_{n-k}, \dots, w_{n-1})\ \alpha(w_{n-k}, \dots w_n), & \hat{p}(w_n|w_{n-k}, \dots, w_{n-1}) > 0 \\ p(w_n|w_{n-k+1}, \dots, w_{n-1})\ \beta(w_{n-k}, \dots, w_n), & \hat{p}(w_n|w_{n-k}, \dots, w_{n-1}) = 0 \end{cases}$$

$\alpha(\dots)$ и $\beta(\dots)$ выбираются исходя из условий нормировки или полагаются константами (в этом случае выход модели не является вероятностью)

Влияние сглаживания: модельный пример

Построение биграммной модели без учёта сглаживания:

```
counts = np.random.zipf(a=3, size=(100, 100)) - 1
probs = counts / counts.sum(axis=1)[:, None]
```

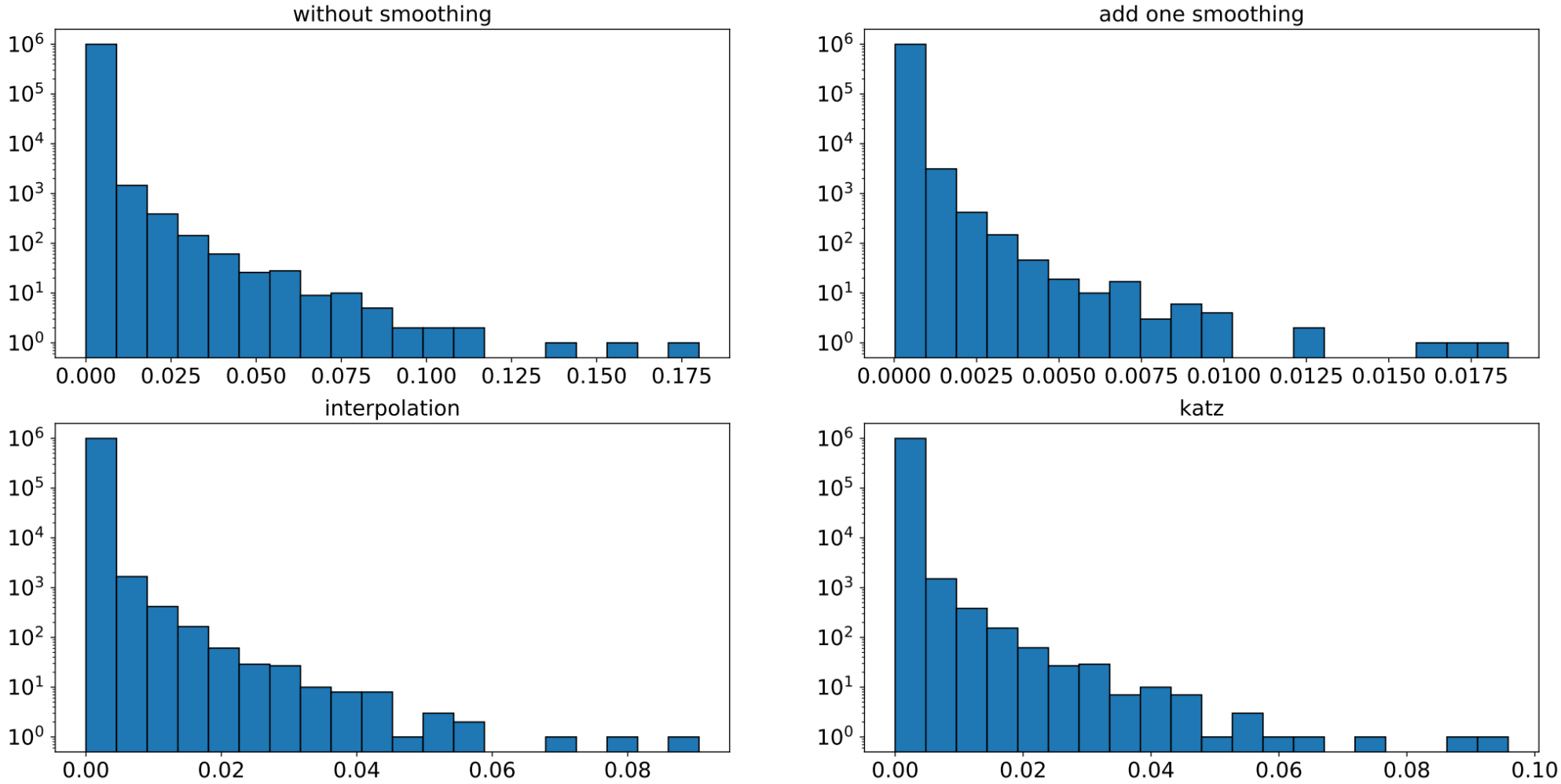
Функция для вычисления вероятностей триграмм по биграммной модели:

```
def get_trigram_probs(probs):
    return (probs[:, :, None] * probs[None, :, :]).ravel()
```

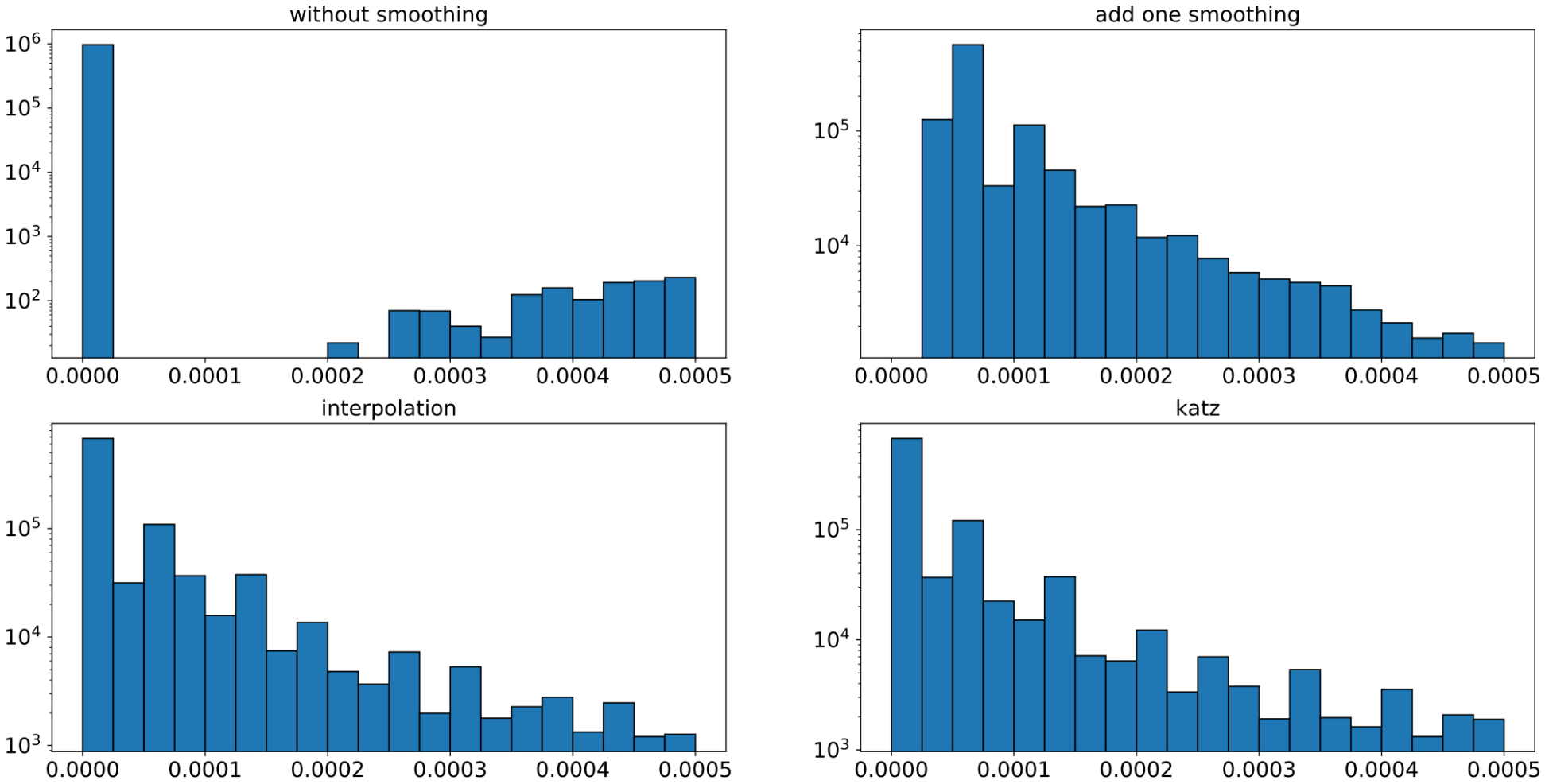
Влияние сглаживания: модельный пример

```
# сглаживание лапласа
addk_counts = counts + 1
addk_probs = addk_counts / addk_counts.sum(axis=1))[:, None]

# интерполяционное сглаживание
uni_probs = counts.sum(axis=1) / counts.sum()
interpolation_probs = 0.7 * probs + 0.3 * uni_probs

# сглаживание через откат
katz_counts = 0.7 * probs + 0.3 * uni_probs * np.int8(probs == 0)
katz_probs = katz_counts / katz_counts.sum(axis=1)[:, None]
```

Влияние сглаживания: общая картина



Влияние сглаживания: вблизи нуля



Другие виды сглаживания

Мы рассмотрели далеко не все виды сглаживания:

- Good-Turing estimate
- Witten-Bell smoothing
- Absolute discounting
- Kneser-Ney smoothing — один из самых популярных способов,

сглаживание зависит от вариативности контекстов

Многие типы сглаживания придумывались для задачи статистического машинного перевода и сейчас не так актуальны.

Задача исправления опечаток (spellchecking)

Необходимо по входной строке получить её исправленный вариант.

Задача сложная и разнообразная:

- ошибки пропуска/вставки символов
- ошибки склейки/расклейки слов
- ошибки неправильного написания слов
- неверная раскладка
- написание транслитерацией

Есть мнение, что лучше тренировать модели, устойчивые к опечаткам, чем пытаться исправить опечатки на этапе предобработки…

Модель шумного канала для исправления опечаток

Исходное слово w подаётся в канал и искажается в слово $\hat{w}$

Хотим восстановить исходное слово w по $\hat{w}$

Пусть $p(u)$ – языковая модель, $p(\hat{w}|u)$ – модель канала:

$$w = \arg\max_{u \in W} p(u|\hat{w}) = \arg\max_{u \in W} \frac{p(u, \hat{w})}{p(\hat{w})} = \arg\max_{u \in W} p(\hat{w}|u)p(u)$$

Зачем это нужно? Можем использовать невероятностную модель канала.

Модель канала на основе расстояния Левенштейна

Простейшая модель шумного канала – ненулевые вероятности имеют слова, которые находятся на расстоянии Левенштейна $\leq t$ от заданного слова.

Расстояние Левенштейна – минимальное количество операций вставки, удаления и замены символа, необходимых для превращения одной последовательности символов в другую.

Расстояние между двумя последовательностями размером n и m можно посчитать за $O(mn)$.

Алгоритм исправления опечаток

1. Разбиваем строку на слова, обрабатываем каждое слово отдельно
2. Если слова нет в словаре, генерируем для него кандидатов на исправление согласно модели канала – используем заданное число элементарных операций
3. Удаляем кандидатов, не входящих в словарь
4. При помощи языковой модели выбираем лучшего кандидата

На последнем шаге можно использовать языковую модель любого порядка или специально обученный классификатор

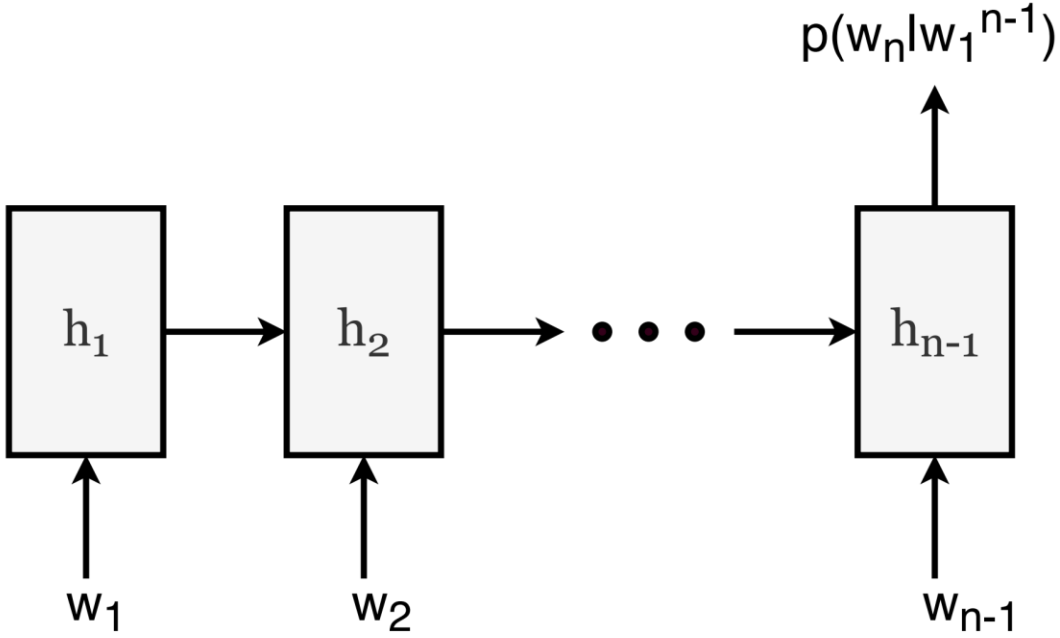
[Статья на Хабре про исправление опечаток]

N-граммные модели: что нужно запомнить

- Быстро обучаются
- Требуют много памяти для хранения
- Нет смысла использовать большие k
- Можно использовать для построения автодополнения, генерации признаков или решения задач в формате шумного канала
- Не нужно использовать для генерации текста
- Есть готовые реализации в библиотеке nltk
- Есть готовые реализации спеллчекеров с поддержкой языковых моделей: библиотека Jamspell, levenshtein_corrector в библиотеке deeppavlov

Нейросетевые языковые модели

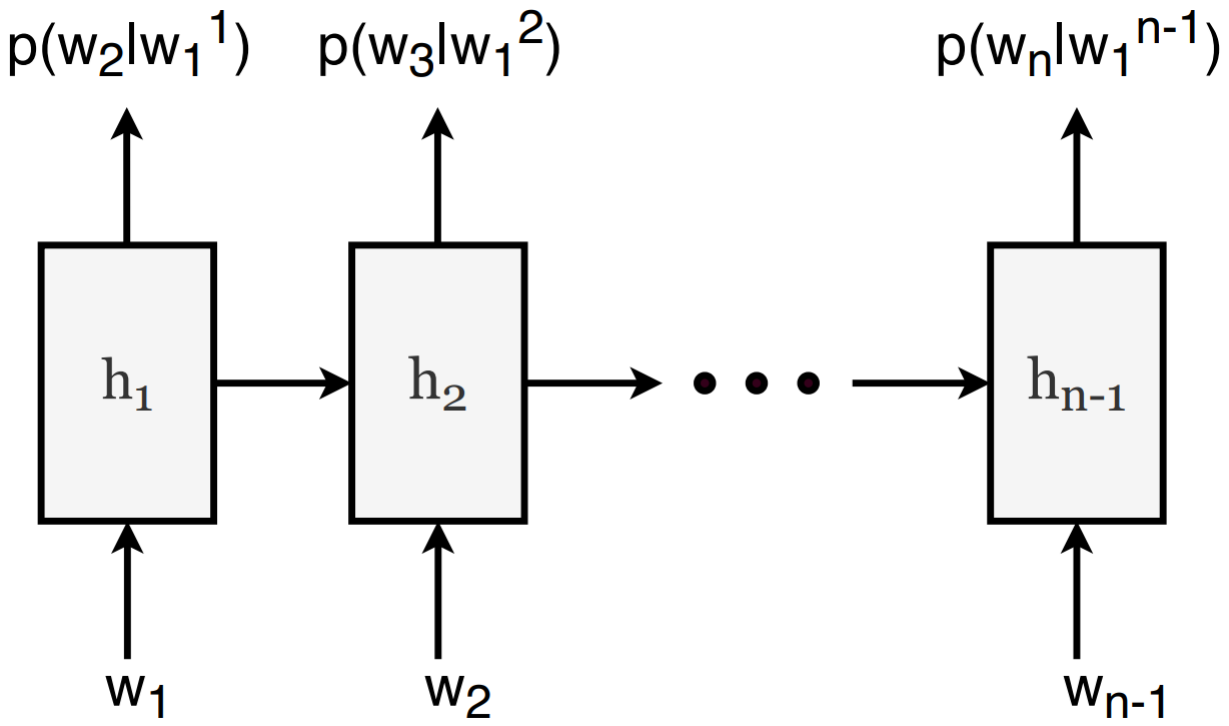
Идея: моделировать $p(w_n|w_1, \ldots, w_{n-1})$ при помощи нейросети для обработки последовательности (RNN, CNN, Transformer)



Можем ли мы сделать лучше?

Нейросетевые языковые модели

Идея: использовать “однонаправленные” модели, на i-ом выходе моделируется $p(w_{i+1}|w_1, \ldots, w_i)$

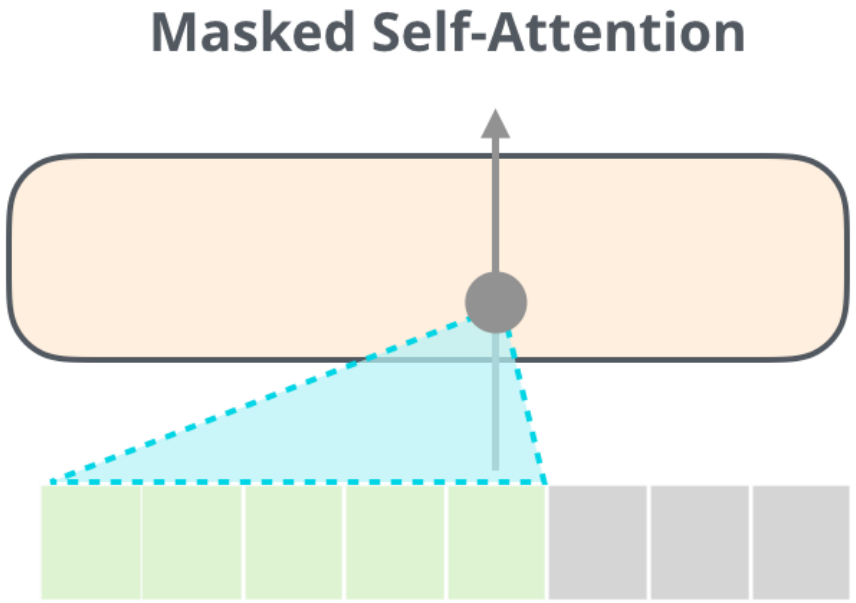


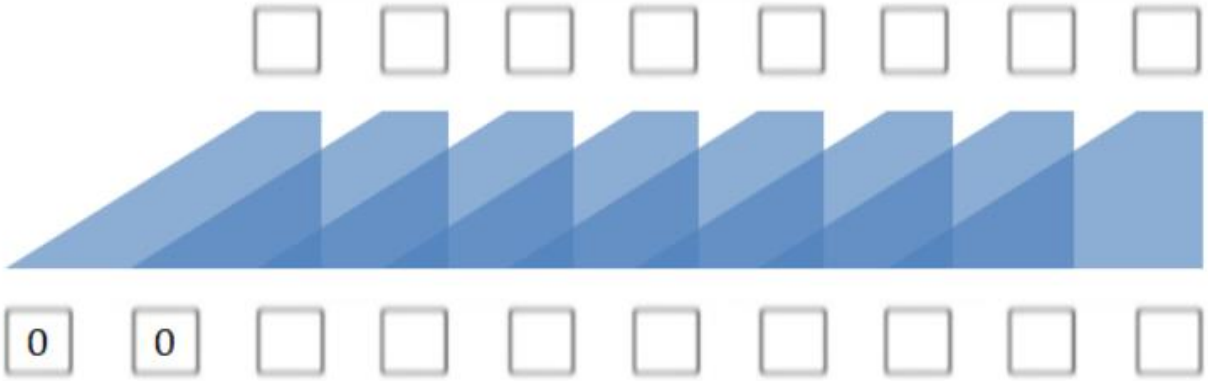
Однонаправленные нейросети

RNN: нельзя использовать двунаправленные сети (bidirectional)

Transformer: маскированный self-attention

CNN: смещённые свёртки





Обучение нейросетевой языковой модели

1. На вход подаётся последовательность токенов:

$$w_1^n = \{w_1, \dots, w_n\}, \qquad w_i \in W$$

2. Каждый токен преобразуется в его эмбеддинг

$$v_1^n = Embedding(w_1^n) = \{v_1, \dots, v_n\}, \qquad v_i \in \mathbb{R}^m$$

3. Эмбеддинги подаются в слови для обработки последовательности

$$h_1^n = model(v_1^n) = \{h_1, \dots, h_n\}, \qquad h_i \in \mathbb{R}^{m_{encoder}}$$

4. К выходам на заданных позициях применяется линейный слой

$$o_i = Uh_i + b, \qquad o_i \in \mathbb{R}^{|W|}$$

5. Функционал для обучения – кросс-энтропия:

$$-\sum_{i=1}^{n} \log p(w_{i+1}|w_1, \dots, w_{i-1}) = -\sum_{i=1}^{n} \log softmax_{w \in W}(o_i) \to \min$$

Составление выборки для языковой модели

1. Выбираем максимальный размер контекста k (обычно 512)
2. Токенизируем документы (обычно используем BPE)
3. Выделяем в документах последовательности длины k
4. Если длина каких-то последовательностей оказалась меньше k, дополняем их паддингом или объединяем с другими последовательностями через специальный токен-разделитель
5. В начало каждой последовательности ставим <START> токен
6. На выходе получается выборка пар последовательностей
$$(< START >, w_1, \dots, w_{n-1}), (w_1, \dots, w_n)$$
7. Токен <END> может ставится в конце документов/предложений

Применение модели для генерации текста

Языковую модель можно использовать для генерации текста в авторегрессионном стиле аналогично энкодеру-декодеру.

Дано: $y_0 = < START >$

Необходимо сгенерировать последовательность $y_1, \dots, y_m$

Пока $y_i \neq < END >$ **или** $i \neq M$:

1. Получение из модели логитов $q(y|y_{<i})$
2. Получение из логитов распределения $p(y|y_{<i})$
3. Получение нового токена y_i из распределения $p(y|y_{<i})$

где M – максимальная длина выходной последовательности

Префикс и BPE: прооблемы

Иногда нам может быть доступен не только y_0, но и какой-то префикс, токенизированный BPE.

Пример. Задача автодополнения.

Система подсказала пользователю выход, но пользователю он не понравился. Он ввёл несколько дополнительных символов и ожидает увидеть более правильную подсказку.

OK: BPE(autocompletion) = [auto, ##compl, ##eti, ##on]

NOT OK: BPE(autoc) = [auto, ##c]

OK: BPE(autocompl) = [auto, ##compl]

Префикс и BPE: решение

Один из способов борьбы с неполным префиксом – откат.

1. Все BPE токены хранятся в префиксном дереве (боре)
2. Перед началом генерации мы проверяем, есть ли префикс, который нужно учесть
3. Если есть префикс $y_1, \dots y_m$ откатываемся назад на s токенов
4. Используя префикс $y_1, \dots, y_{m-s}$ генерируем следующий токен, но допускаем только те токены, которые соответствуют префиксу (благодаря бору можем делать это быстро)
5. После исчерпания префикса продолжаем генерацию в свободном режиме

Префикс и BPE: пример

Хотим автодополнить “x = max_cou”, ожидаем “max_count_df”

BPE(x = max_count_df) = [x, =, max, ##_count, ##df]

BPE(x = max_cou) = [x, =, max, ##_co, ##u]

Делаем откат до начала составного слова:

BPE(x =) = [x, =]

Генерируем токены, пока не исчерпаем префикс (3 итерации):

BPE(x = max_count) = [x, =, max, ##_count]

Генерируем в свободном режиме после исчерпания префикса.

Модели GPT (generative pre-training)

GPT – это популярные архитектуры для языкового моделирования и конкретные модели, обученные openAI (GPT1-3)

Устройство модели GPT:

- на входе и выходе – BPE токены
- архитектура всех моделей GPT – энкодер трансформера с небольшими изменениями + masked self-attention
- обучается по большому множеству неразмеченных текстов

Radford et al (2018); Improving Language Understanding by Generative Pre-Training

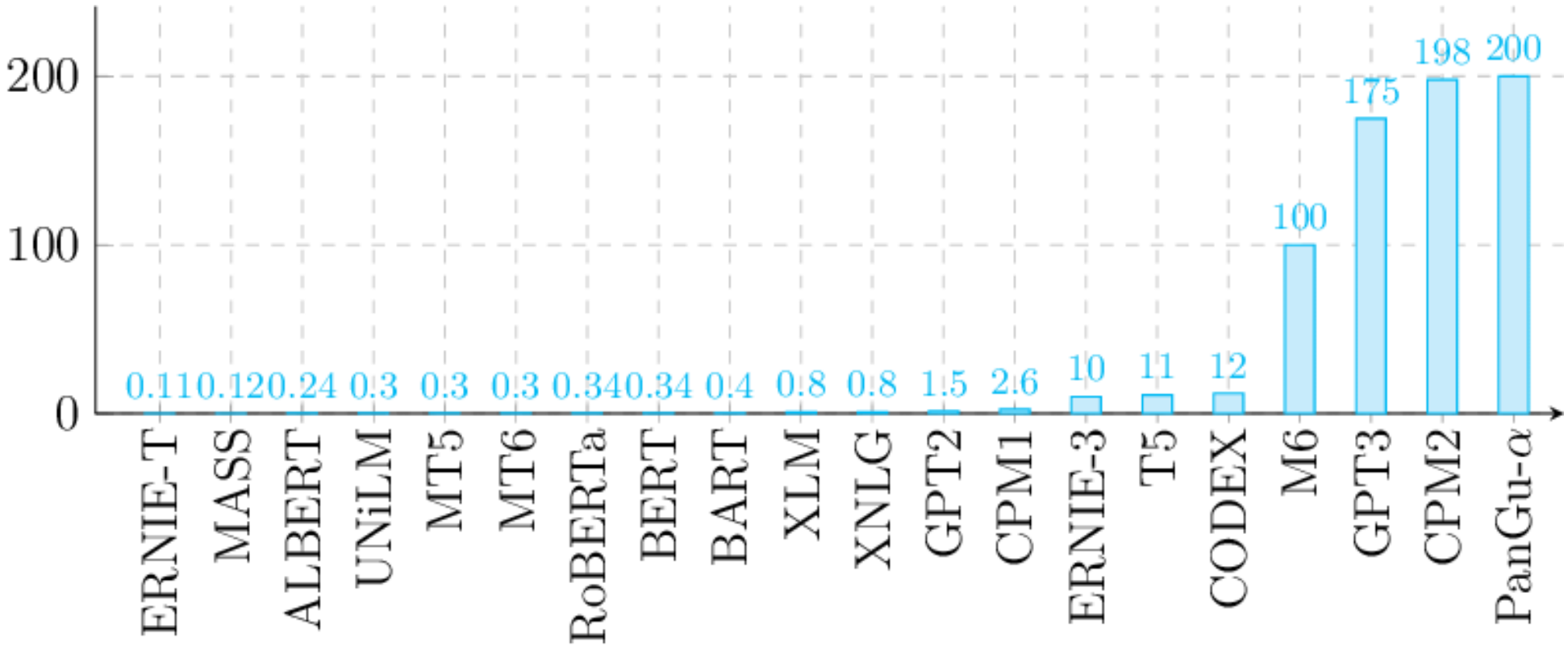
Radford et al (2019); Language Models are Unsupervised Multitask Learners

Brown et al (2020); Language Models are Few-Shot Learners

Размеры моделей GPT

Название	Размер корпуса	Словарь	Число слоёв	Размер слоя	Число параметров
GPT	7k books	40k	12	768	≤117M
GPT2 small	40 GB	50k	12	768	117M
GPT2 medium	40 GB	50k	24	1024	345M
GPT2 large	40 GB	50k	36	1280	762M
GPT2 extra large	40 GB	50k	48	1600	1542M
GPT-3 small	≈ 1TB	??	12	768	125M
GPT-3 medium	≈ 1TB	??	24	1024	350M
GPT-3 large	≈ 1TB	??	24	1536	760M
GPT-3 13B	≈ 1TB	??	40	5140	13.0B

Размеры языковых моделей



Что умеет GPT: AI Dungeon

AI Dungeon – текстовая игра, в которой созданием мира, реакцией на любые действия игрока занимается GPT3.

One night you have a dream. In the dream, the earth begins to shake and you see an opening in the ground.

You see a huge train underground.

It is an engine made of old iron with a small cab on the front. Smoke comes out of the stack and you think to yourself "I must be near a great big machine that creates a lot of smoke.“

The train stops. People start to come out of it.

They are all heavily dressed in thick clothing and carry weapons. "That’s strange,"you think to yourself, "People shouldn’t be arriving at the train when it is still underground."

Что умеет GPT: AI Dungeon

AI Dungeon – текстовая игра, в которой созданием мира, реакцией на любые действия игрока занимается GPT3.

All people are in a hurry somewhere. They knock you off your feet.

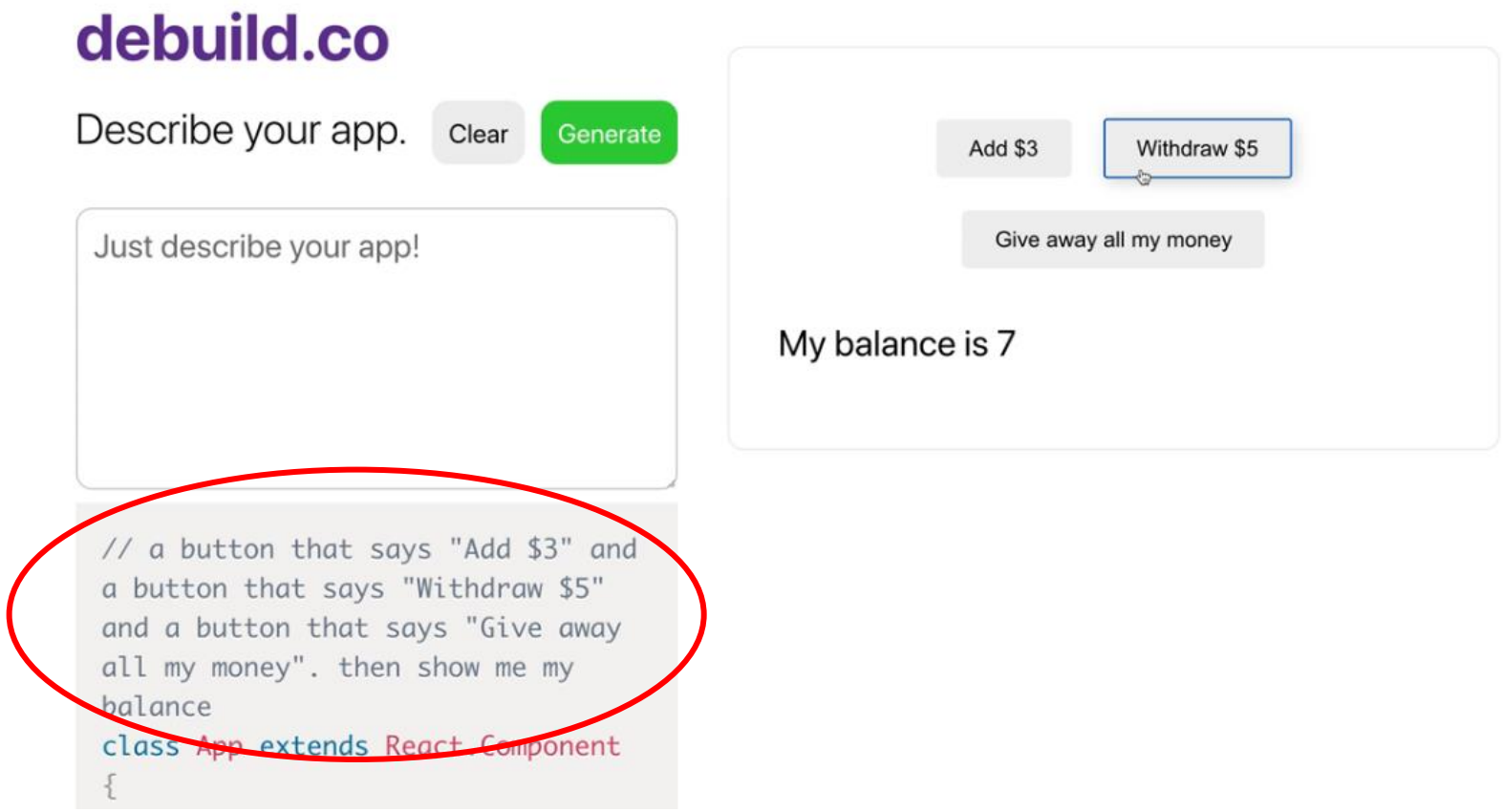
You try to get up, but a very large man nearly breaks your neck with one blow from his fist. You see more arriving everyday. You begin to wonder: "What is going on".

Suddenly, you understand everything. This is the metro at rush hour.

This is the busiest time of day and night. Every day hundreds of people come through here. They have their own agenda and own goals. However, at this moment you see something burning in the distance.

Что умеет GPT: debuild

Построение веб-интерфейса по описанию:



Что умеет GPT: copy.AI

По короткому описанию проекта генерирует маркетинговый текст.

OzonMasters NLP: an educational course in Russia about natural language processing

OzonMasters is a unique and the only intensive course in Russia on Natural Language Processing with tasks, exercises, practice and workshops for all those who want to learn NLP. You will definitely benefit from our unique approach: smart examples of real language problems are used, material is provided and all of this is supported by serious scientific research.

Что умеет GPT: copy.AI

По короткому описанию проекта генерирует маркетинговый текст.

OzonMasters NLP: an educational course in Russia about natural language processing

Have you ever wanted to be good at recognizing and processing any kind of texts in any language? Then this course is for you! You may not have heard about the Natural Language Processing (NLP) training in OzonMaster, but it is in fact one of the best prestigious courses on the topic. It covers a variety of aspects and applications of NLP in practice. It was written by well-known experts in the field and is taught by them too! At this comprehensive, full-day event, you will get to know more about the past, present and future of NLP, as well as gain unprecedented insights into its application to practice.

Что можно делать с нейросетевыми языковыми моделями?

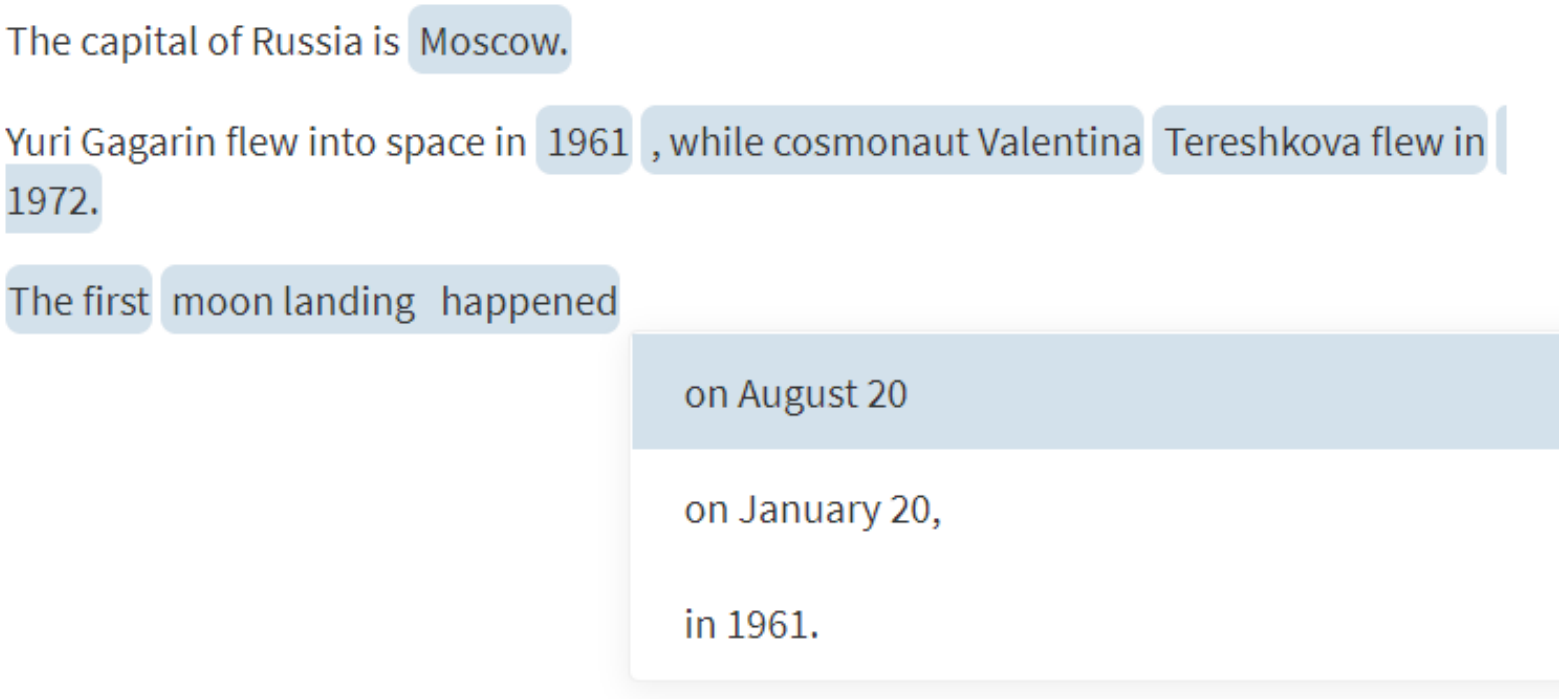
Можно использовать напрямую для задач генерации текста и автодополнения.

Можно использовать для любых других задач связанных с генерацией текста, применяя подходы:

- Обучение без подготовки (zero-shot learning)
- Обучение с небольшой подготовкой (few-shot learning)
- Использование выходов с промежуточных слоёв для контекстных эмбеддингов (context embeddings)
- Дообучение модели (fine-tuning)

Zero-shot learning

За счёт обучения на большом объёме данных GPT можно использовать в режиме zero-shot learning, например, как вопросно-ответную систему:



Few-shot learning

Zero-shot learning

Translate English to Russian: support vector machine ->

Метод опорных векторов

Few-shot learning

Translate English to Russian:
support vector machine -> метод опорных векторов
machine learning -> машинное обучение
gradient descent ->

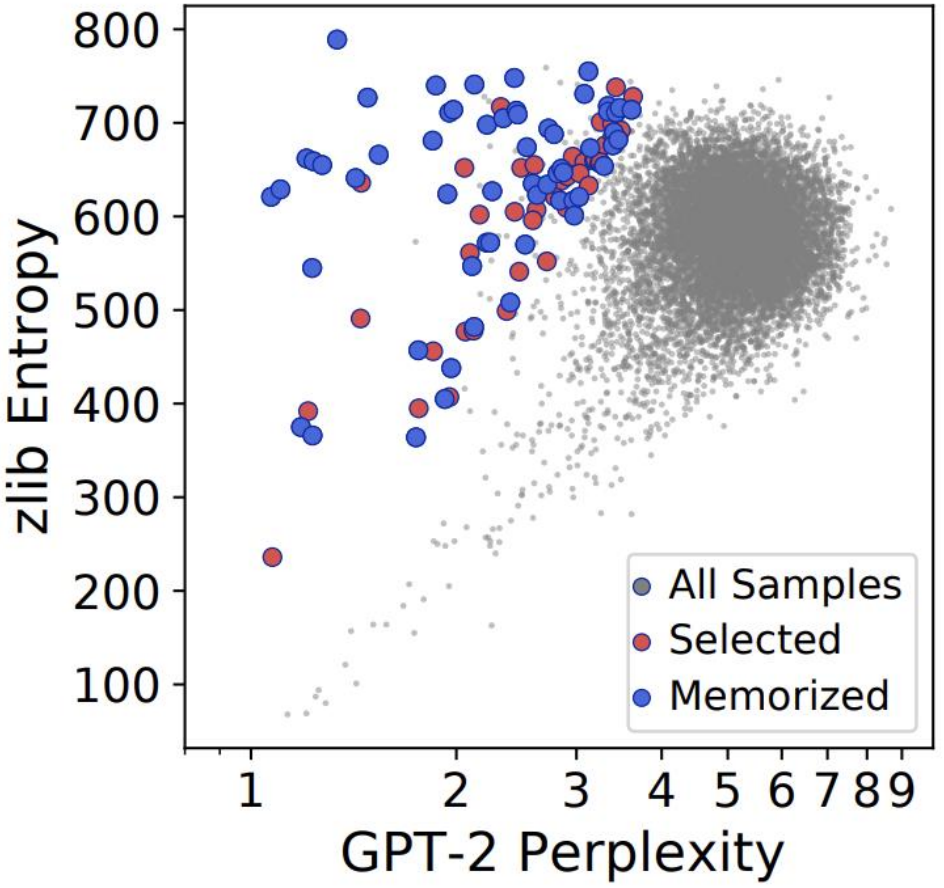
Градиентный спуск

Что следует помнить про GPT: запоминание

Что если GPT просто запоминает обучающую информацию в своих параметрах и выдаёт на тесте куски реальных данных?

Почему это плохо:

1. Выдача наружу личных/приватных данных
2. Копирайт



[Carlini et al (2021); Extracting Training Data from Large Language Models]

Что следует помнить про GPT: запоминание

Memorized String	Sequence Length	Occurrences in Data	
		Docs	Total
Y2...[redacted]...y5	87	1	10
7C...[redacted]...18	40	1	22
XM...[redacted]...WA	54	1	36
ab...[redacted]...2c	64	1	49
ff...[redacted]...af	32	1	64
C7...[redacted]...ow	43	1	83
0x...[redacted]...C0	10	1	96
76...[redacted]...84	17	1	122
a7...[redacted]...4b	40	1	311

Table 3: **Examples of $k = 1$ eidetic memorized, high-entropy content that we extract** from the training data. Each is contained in *just one* document. In the best case, we extract a 87-characters-long sequence that is contained in the training dataset just 10 times in total, all in the same document.

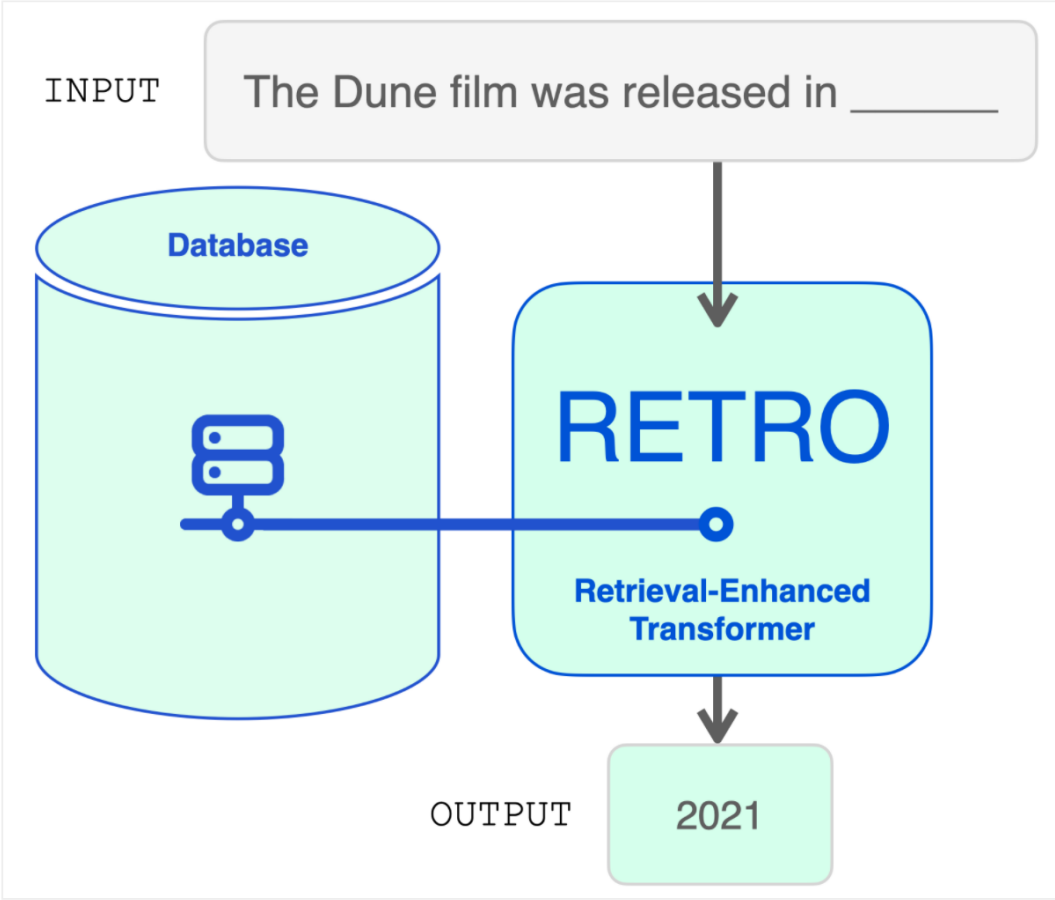
Чем больше размер модели, тем больше специфичный примеров она запоминает:

URL (trimmed)	Occurrences		Memorized?		
	Docs	Total	XL	M	S
/r/[redacted]51y/milo_evacua...	1	359	✓	✓	½
/r/[redacted]zin/hi_my_name...	1	113	✓	✓	
/r/[redacted]7ne/for_all_yo...	1	76	✓	½	
/r/[redacted]5mj/fake_news_...	1	72	✓		
/r/[redacted]5wn/reddit_admi...	1	64	✓	✓	
/r/[redacted]lp8/26_evening...	1	56	✓	✓	
/r/[redacted]jla/so_pizzagat...	1	51	✓	½	
/r/[redacted]ubf/late_night...	1	51	✓	½	
/r/[redacted]eta/make_christ...	1	35	✓	½	
/r/[redacted]6ev/its_officia...	1	33	✓		
/r/[redacted]3c7/scott_adams...	1	17			
/r/[redacted]k2o/because_his...	1	17			
/r/[redacted]tu3/armynavy_ga...	1	8			

Возможное развитие: RETRO

Идея 1. Модели огромных размеров работают лучше, потому что сохраняют больше информации о выборке в своих параметрах.

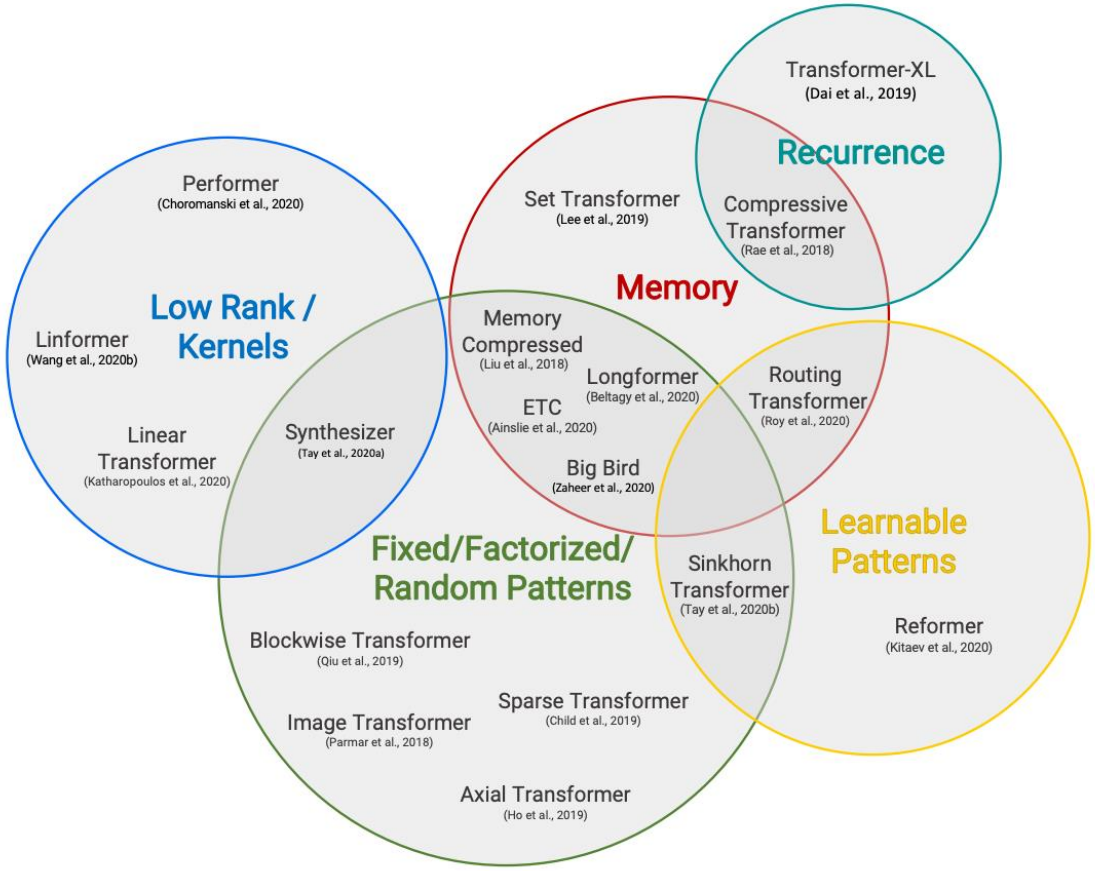
Идея 2. Дать сети доступ к большой базе данных текстов на этапе применения.



Borgeaud et al (2022); Improving language models by retrieving from trillions of tokens

Возможное развитие: длинные последовательности

Из-за квадратичной сложности по длине последовательности, мы не можем использовать очень длинные входы.



https://huggingface.co/blog/long-range-transformers

Резюме по занятию

- Языковые модели – модели оценивающие одну из двух вероятностей
- При помощи языковых моделей можно генерировать текст или делать постобработку результатов других моделей
- Простейшие языковые модели – статистические, подходят для простых и ненагруженных приложений
- Текущее развитие языковых моделей – большие модели на основе трансформеров (GPT3)
- Возможности больших моделей практически безграничны, но мало кто может позволить себе их использовать