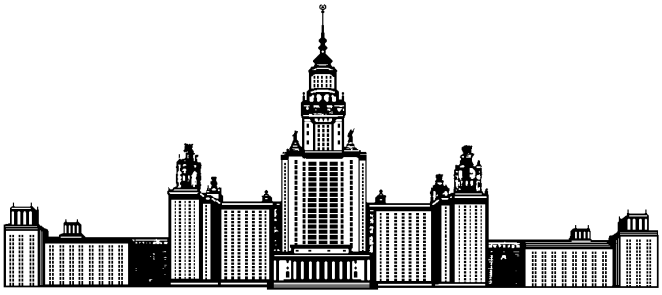


Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Васильев Руслан Леонидович

Языковые модели

Language Models

Эссе по курсу «Глубокое обучение»

Москва, 2021

Аннотация

В данном эссе рассматривается задача построения языковой модели. Приводятся стандартные алгоритмы, основанные на вычислении статистик, а также нейросетевые модели. Обозреваются современные архитектуры. Также обсуждаются проблемы, присущие нейросетевым языковым моделям.

Содержание

1 Введение

Языковые модели (**Language Models**) возникают практически во всех задачах *обработки естественного языка* (**Natural Language Processing, NLP**): они определяют, как именно представляется и генерируется текст. Основные области применения:

- Машинный перевод (выбор подходящего слова или сочетания слов);
- Распознавание речи (выбор подходящего варианта);
- Проверка текста (поиск ошибок);
- Набор текста (выдача подсказок).

Хотя построение языковой модели является крайне сложной задачей, на сегодняшний день существуют в достаточной мере универсальные [1] (т.е. пригодные для задач разной специфики) алгоритмы, основанные на нейронных сетях.

2 Задача моделирования языка и токенизация

Формально задача LM сводится к моделированию вероятности произвольного текста $p(x_1, \ldots, x_n)$, где $x_1 \ldots, x_n$ — последовательность *токенов* (**tokens**). Под токеном мы будем понимать последовательность символов, образующую неразделимую семантическую единицу в процессе обработки [2]. Причем во время *токенизации* (**tokenization**) некоторые символы могут исключаться (например, пробелы, табуляции), заменяться другими (ё $\mapsto$ е) или объединяться в новые (ааа $\mapsto$ А). Рассмотрим способы токенизации на примере пословицы «Семь раз отмерь, один раз отрежь», в качестве токенов могут браться:

- Слова: (Семь, раз, отмерь, один, раз, отрежь).
- Символы: (С, е, м, ь, < >, р, а, з, , ...).
- Подслова (**subwords**): (Семь, раз, от, мерь, один, раз, от, режь).
- Токены, получающиеся в результате работы более сложных алгоритмов: (7, раз, @мерь, 1, раз, @режь).

Отметим, что в токены могут быть добавляться служебные символы (например, символ конца слова), а в словарь — служебные токены (например, для границ предложений).

Современные алгоритмы токенизации в основном развивают идею подслов, поскольку такой подход позволяет бороться с проблемой редких и новых слов, но при этом не разделяет часто встречающиеся морфемы на отдельные символы. Так, в BPE [3] (**Byte Pair Encoding**) наиболее частые пары символов итеративно объединяются в один новый

(жадным образом), а затем текст рассматривается как последовательность полученных символов. Здесь возникают нюансы с введением служебных символов, запретом на объединение разных групп (например, букв и знаков препинания). В [4] BPE применялся к байтовому, а не символьному представлению.

Определившись с тем, что считается токеном, можем вернуться к задаче языкового моделирования — моделирования вероятности текста:

$$p(x_1, \ldots, x_n) = ? \tag{1}$$

Вероятность (1) формально дана в многомерном пространстве n токенов, но мы можем ее декомпозировать. Например, предсказывать следующий токен на основе всех предыдущих:

$$p(x_1, \ldots, x_n) = \prod_{t=1}^{n} p(x_t \mid x_1, \ldots, x_{t-1}),$$
$$p(x_t \mid x_1, \ldots, x_{t-1}) = p(x_t \mid \mathbf{x}_{<t}), \tag{2}$$

или на основе N предыдущих токенов (свойство Маркова):

$$p(x_t \mid x_1, \ldots, x_{t-1}) = p(x_t \mid x_{t-N}, \ldots, x_{t-1}). \tag{3}$$

Как мы увидим в следующих разделах, в языковых моделях часто используется как свойство Маркова, так и полный предшествующий контекст.

3 Статистический подход

Предположение, что каждый токен в тексте генерируется независимо от остальных, приводит к **Unigram Model**:

$$p(x_t|x_1, \ldots, x_{t-1}) = p(x_t),$$

тогда вероятность текста — произведение всех вероятностей токенов (частот токенов в корпусе), которые вошли в него. Однако предположение о полной независимости чересчур сильное, и с использованием свойства Маркова (3) мы можем построить **N-gram Model**, где для оценок в простейшем случае снова используются частоты:

$$p(x_t \mid x_{t-N}, \ldots, x_{t-1}) = \frac{\#\,(x_{t-N}, \ldots, x_{t-1}, x_t)}{\#\,(x_{t-N}, \ldots, x_{t-1})} \tag{4}$$

Для рассмотренной выше пословицы (в качестве токенов оставим слова) модель примет вид

$$p(\text{Семь, раз, отмерь, один, раз, отрежь}) =$$

в **Unigram Model**:

$$= p(\text{Семь}) \cdot p(\text{раз}) \cdot p(\text{отмерь}) \cdot p(\text{один}) \cdot p(\text{раз}) \cdot p(\text{отрежь});$$

в **Bigram Model** ($N = 2$):

$= p(\text{Семь}) \cdot p(\text{раз} \mid \text{Семь}) \cdot p(\text{отмерь} \mid \text{раз}) \cdot p(\text{один} \mid \text{отмерь}) \cdot p(\text{раз} \mid \text{один}) \cdot p(\text{отрежь} \mid \text{раз})$.

С увеличением N растет контекст, но некоторых сочетаний в корпусе может не оказаться, и тогда мы не сможем оценить (4). Для решения данной проблемы существуют различные техники.

Сглаживание по Лапласу

Можем искусственно добавить N-грамму в корпус, что эквивалентно следующему пересчету вероятностей:

$$p(x_t \mid x_{t-N}, \ldots, x_{t-1}) = \frac{\#(x_{t-N}, \ldots, x_{t-1}, x_t) + \alpha}{\#(x_{t-N}, \ldots, x_{t-1}) + \alpha \cdot |V|},$$

где $\alpha > 0$ — параметр сглаживания, $|V|$ — мощность словаря. При $\alpha = 0$ модель не изменится, а с ростом α распределение приближается к равномерному.

Backoff

Метод предложен в [5]. Идея заключается в сокращении контекста, если N-грамма требуемой длины не встречается в корпусе:

$$\hat{p}(x_t \mid x_{t-N}, x_{t-N+1} \ldots, x_{t-1}) = \begin{cases} \alpha \cdot p(x_t \mid x_{t-N+1} \ldots, x_{t-1}), & \#(x_{t-N}, \ldots, x_{t-1}) = 0 \\ p(x_t \mid x_{t-N} \ldots, x_{t-1}), & \#(x_{t-N}, \ldots, x_{t-1}) > 0. \end{cases}$$

$\alpha > 0$ — «понижающий множитель», который зависит от контекста $x_{t-N}, \ldots, x_{t-1}$ и может выбираться по-разному. Необходимость в нем по меньшей мере нужна для нормировки вероятностей. Обычно α настраивается с привлечением **Good-Turing** оценки — чтобы оценить вероятности новых последовательностей, мы можем посчитать N-граммы, которые встречались в корпусе единожды [6].

Интерполяция

Вместо перехода к узкому контексту только при необходимости можно всегда использовать линейную комбинацию частотных оценок для всех N-грамм [6] меньшего порядка:

$$\hat{p}(x_t \mid x_{t-N}, \ldots, x_{t-1}) = \lambda_0 p(x_t) + \lambda_1 p(x_t \mid x_{t-1}) + \cdots + \lambda_N p(x_t \mid x_{t-N}, \ldots, x_{t-1}),$$

где параметры $\lambda_0, \ldots, \lambda_N$ могут быть получены с помощью EM-алгоритма на отложенной выборке [7].

Даже с введением дополнительных параметров, рассмотренные ранее статистические модели обладают слабой обобщающей способностью (**Lack of Generalization**). Во-первых, число всевозможных N-грамм слишком велико, и частотные оценки плохо моделируют настоящие предложения. Во-вторых, Марковская модель не всегда оправдана: например, заключение в тексте может коррелировать со вступлением. Решить данные проблемы позволяет нейросетевой подход.

4 Базовые нейросетевые модели

Прежде чем перейти к современным нейросетевым языковым моделям, основанным на трансформерах, рассмотрим две более простых парадигмы.

4.1 Параметрическое оценивание

Сохраняя Марковское свойство, будем моделировать (3). Далее при использовании токена x_j в формулах считается, что он записан в one-hot encoding.

- $s_j = W_{d\times|V|}x_j$ — проекция на d-мерное пространство, то есть получение плотных представлений слов (**embeddings**);
- $h = \tanh(W_{d'\times Nd}[s_{t-N}, \dots, s_{t-1}] + b)$ — нелинейная проекция;
- $y = U_{|V|\times d'}h + c$ — еще один линейный слой;
- $p(x_t \mid x_{t-N}, \dots, x_{t-1}) = \text{softmax}(y)$ — оценки вероятностей.

Отметим, что $W_{d\times|V|}$ может быть обучена предварительно — например, с помощью модели **word2vec** [8].

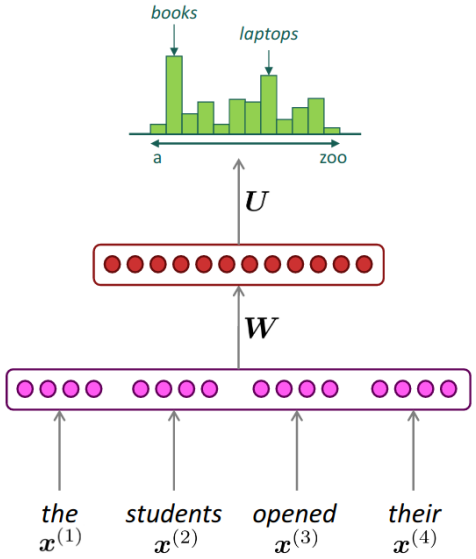


Рис. 1: Параметрическое оценивание [9]

Нейросеть проиллюстрирована на рис. 1: синим обозначена конкатенация представлений $[s_{t-N}, \dots, s_{t-1}]$, красным — нелинейная проекция h, зеленым — распределение на выходе softmax.

4.2 RNN-моделирование

В предыдущем подходе мы уже решили часть проблем статистического подхода, однако в такой параметризации по-прежнему используется свойство Маркова. Рекуррентные нейронные сети позволяют отказаться от него, принимая на вход последовательности произвольной длины.

Для моделирования (2) введем скрытое состояние h_t (память), определим *функцию перехода* (**transition function**) f и *функцию вывода* (**output/readout function**). Положим также $h_0 = 0$, тогда процедура моделирования вероятностей для t-го токена будет иметь вид:

1. $h_t = f(x_{t-1}, h_{t-1})$,
2. $p(x_t \mid x_1, \ldots, x_{t-1}) = g(h_t)$.

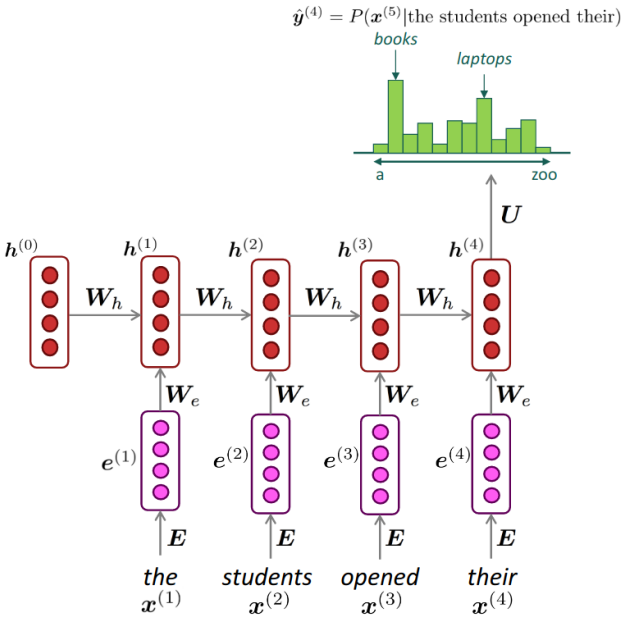


Рис. 2: Языковая модель на основе RNN [9]

Приведем пример, как можно определить f и g для простейшей рекуррентной сети. Введем матрицы $W_e \in \mathbb{R}^{d \times |V|}$, $W_h \in \mathbb{R}^{d \times d}$, $U \in \mathbb{R}^{|V| \times d}$, b и c — векторы смещения соответствующих размерностей. Модель принимает вид:

$$h_t = f(x_{t-1}, h_{t-1}) = \tanh(W_e \cdot x_{t-1} + W_h \cdot h_{t-1} + b),$$
$$p(x_t \mid x_1, \ldots, x_{t-1}) = g(h_t) = \text{softmax}\,(U \times h_t + c),$$

формулы проиллюстрированы на рис. 2 (на рисунке также добавлено промежуточное преобразование one-hot векторов в плотные представления e_j).

Каждый раз на выходе функции g мы получаем распределение на множестве всех токенов. В качестве ошибки используется кросс-энтропия:

$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T(i)} \log p(x_t^{(i)} \mid x_1^{(i)}, \ldots, x_{t-1}^{(i)}) \to \min,$$

где m — число предложений в выборке (батче), $T(i)$ — длина i-го предложения. Таким образом, сеть должна верно прогнозоровать каждое следующее слово: ошибка суммируется.

5 Стратегии генерации

На выходе нейронной сети на каждом шаге мы получаем вектор вероятностей. Тем не менее в тех задачах, где применяются языковые модели, обычно требуется получить на выходе предложение, то есть последовательность токенов.

Стратегии генерации могут быть основаны на максимизации правдоподобия:

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \to \max \tag{5}$$

$$\Longleftrightarrow \tag{6}$$

$$\sum_{t=1}^{T} \log p(x_t \mid x_1, \ldots, x_{t-1}) \to \max. \tag{7}$$

Но при максимизации правдоподобия модели часто генерируют короткие предложения. Для борьбы с данной проблемой максимизируют среднеарифметическое:

$$\frac{1}{T} \sum_{t=1}^{T} \log p(x_t \mid x_1, \ldots, x_{t-1}) \to \max,$$

причем с введением нормировки выражение эквивалентно минимизации *перплексии* (**perplexion**) — показателю качества языковой модели:

$$\mathrm{PP}(x_1, \ldots, x_T) = \left(\frac{1}{p(x_1, \ldots, x_T)} \right)^{\frac{1}{T}} \to \min$$

Рассмотрим основные подходы к генерации.

5.1 Детерменированные

В данной группе методов декодирование производится однозначным образом при фиксированном векторе вероятностей токенов.

Полный перебор (Exhaustive Search)

Полный перебор подразумевает вычисление правдоподобия для всех возможных последовательностей (с ограниченным T). Этот способ слишком дорогой и обычно не пригоден для задач.

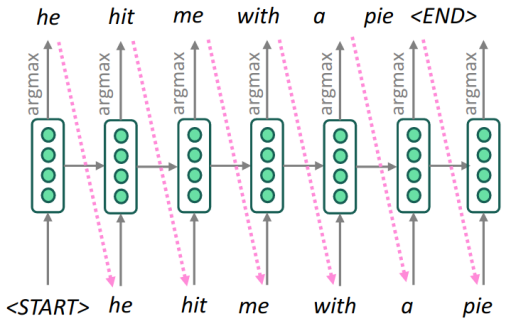


Рис. 3: Жадное декодирование (Greedy Decoding) [9]

Жадное декодирование (Greedy Decoding)

В данной стратегии на каждом шаге t генерируется наиболее вероятный токен: $p(x_t \mid x_1, \ldots, x_{t-1}) \to \max$, что проиллюстрировано на рис. 3. Жадное декодирование позволяет эффективно генерировать предложения, но даже с точки зрения приведенных функционалов неоптимальные, поскольку игнорируется слишком много вариантов.

Метод луча (Beam Search)

В методе луча мы храним k наиболее вероятных последовательности на каждом шаге. Метод проиллюстрирован на рис. 4. Метод луча является одним из самых часто ис-

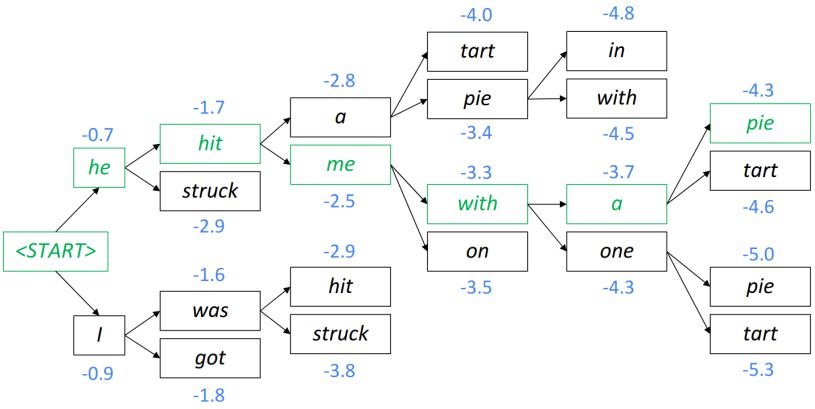


Рис. 4: Метод луча (Beam Search) [9]

пользуемых стратегий генерации. Увеличивая k, мы можем повысить качество выходных последовательностей (ценой больших расходов памяти и времени). Считается, что малые значения k соответствуют релевантным фразам, но нередко приводят к неязыковым фразам, большие — к грамматически верным, но слишком общим.

5.2 Стохастические

Оказывается, что детерменированные стратегии генерации в случае декодирования в нейронных сетях нередко уступают *семплированию* (**(sampling)**) — генерации токенов

в соответствии с полученными оценками вероятностей.

Pure Sampling

Данный метод похож на жадное декодирование, но вместо выбора наиболее вероятного токена производится семплирование на основе предсказанного распределения.

Семплирование с температурой (Sampling with Temperature)

В нейросетевых моделях перед генерацией текста мы можем отшкалировать логиты:

$$\text{softmax}\left(z_1, \ldots, z_{|V|}\right) \mapsto \text{softmax}\left(\frac{z_1}{\tau}, \ldots, \frac{z_{|V|}}{\tau}\right),$$

где параметр $\tau > 0$ — *температура*. С ростом τ итоговое распределение приближается к равномерному (становится более *широким*, **wide/flat**), при уменьшении температуры, напротив, распределение стремится к вырожденному (становится *узким/островершинным*, **narrow/peaked**). Иллюстрация широкого и узкого распределения токенов приведена на рис. 5

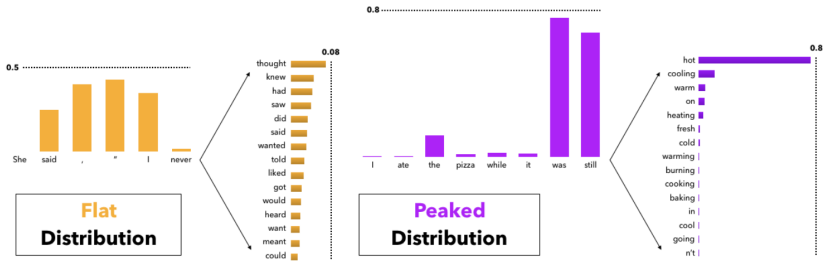


Рис. 5: Широкое и узкое распределение [10]

Top-k Sampling

Стратегия отличается от Pure Sampling тем, что вместо семплирования по всему словарю оставляют только k наиболее вероятных токенов. Далее вероятности оставшихся нормируются и семплирование производится только по ним.

Ядерное семплирование (Nucleus Sampling)

Метод был предложен в [10] и отличается от предыдущего тем, что вместо k задается минимальный порог p, который должны иметь токены для семплирования (после соответствующего выбора производится перенормировка).

Penalized sampling[11]

Для борьбы с повторами можно понизить вероятности тех слов, которые уже присутствовали. Для, как и в семплировании с температурой, масштабируются логиты:

$$\text{softmax}\left(z_1, \ldots, z_{|V|}\right) \mapsto \text{softmax}\left(\frac{z_1}{\tau\theta_1}, \ldots, \frac{z_{|V|}}{\tau\theta_{|V|}}\right),$$

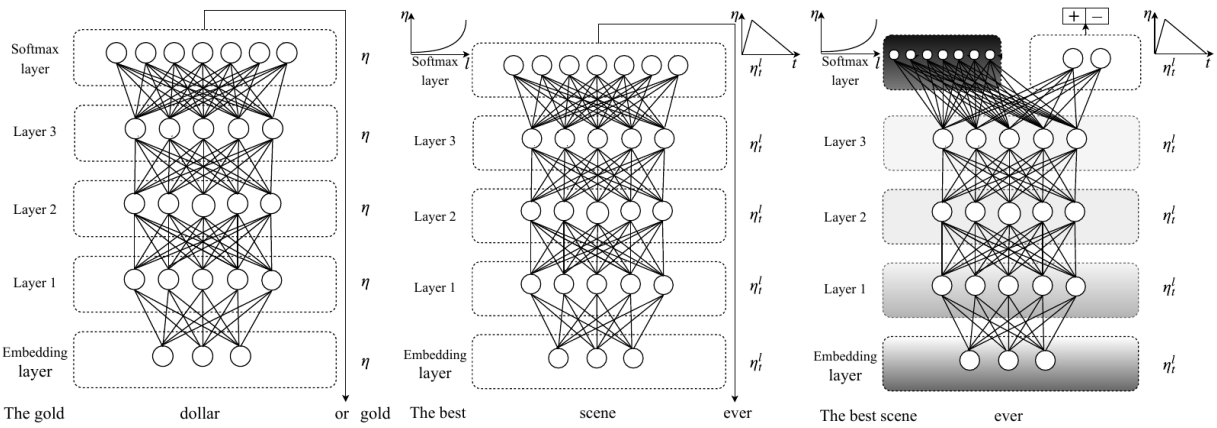
где

$$\theta_i = \begin{cases} 1, & \text{i не было в контексте,} \\ \theta, & \text{иначе.} \end{cases}$$

6 Современные архитектуры

6.1 ULMfit

ULMfit (**Universal Language Model Fine-tuning for Text Classification**) [12] стала известной как одна из первых моделей, успешно применивших *трансферное обучение* (**Transfer Learning**) в NLP. Трансферное обучение — популярная на сегодняшний день парадигма машинного обучения, в которой сначала модель настраивается на достаточно общую задачу (обычно при этом используется очень большой датасет), а затем уже может *дообучаться* (**fine-tuning**) на релевантных данных и использоваться для специфичных задач.



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Рис. 6: Стадии обучения ULMfit [12]

Рассмотрим конкретнее, как идея трансфера применялась в ULMfit (три ключевых этапа проиллюстрированы на рис. 6):

1. Сначала общая языковая модель предобучалась на огромном датасете, состоящем из > 28 тысяч статей Википедии. В качестве архитектуры использовалась AWD-LSTM [13] — рекуррентная архитектура.

2. Fine-tuning языковой модели на более специфичном датасете. На данном этапе обучение короче, так как предобученная модель уже имеет хорошие веса. Из особенностей, использованных конкретно ULMfit, можно выделить разделение темпов обучения по слоям (**discriminative fine-tuning**), причем они были не константными, а изменялись «треугольным» образом рис. 7. На начальных итерациях темпы обучения линейно возрастают, увеличивая скорость сходимости, а затем убывают, позволяя модели настроиться на оптимум.

3. Fine-tuning на задачу классификации: поверх языковой модели добавлялось два линейных слоя Softmax. Из особенностей можно выделить *постепенную разморозку* (**gradual unfreezing**) слоев: сначала настраивался только последний слой, потом он обучался вместе с предпоследним и т.д.

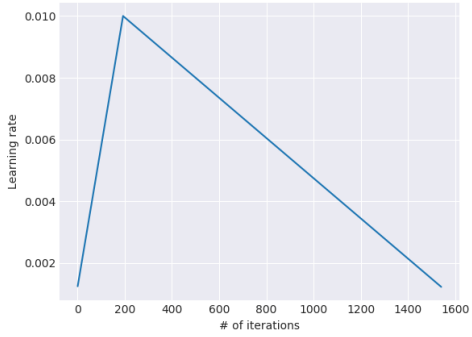


Рис. 7: Изменение темпов обучения в ULMfit — ***Slanted triangular learning rates*** *[12]*

6.2 ERNIE

Модель ERNIE [14] (**Enhanced Representation through Knowledge Integration**) основана на архитектуре трансформера. Авторы ERNIE развили идею маскирования, которая легла в основу обучения BERT [15]. Если в моделях BERT выбиралось 15% токенов случайно по корпусу рис. 8, то в ERNIE были предложены более осмысленные стратегии.

Кроме маскирования случайных токенов (**Basic-level Masking**), было использовано маскирование целых сущностей — **Entity-level masking**, например, инициалов и фамилии. Также маскировались целые фразы — устойчивые сочетания (**Phrase-level masking**) рис. 9. Использование всех стратегий маскирования дало наилучшие результаты.

В сравнении с BERT модель ERNIE оказалась лучше, но результаты приводятся только на датасетах с китайским языком рис. 10.

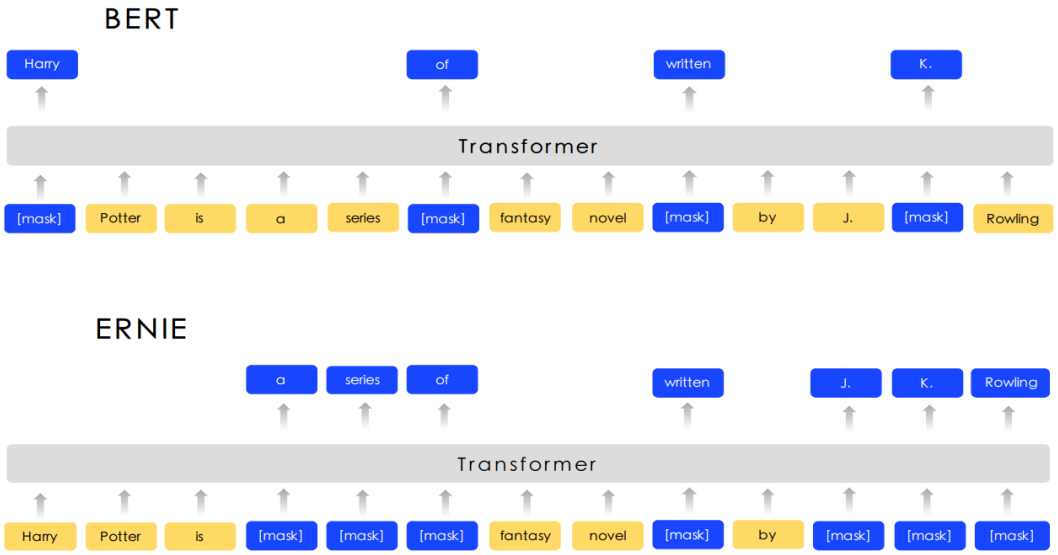


Рис. 8: Отличие стратегий маскирования ERNIE и BERT [14]

Sentence	Harry	Potter	is	a	series	of	fantasy	novels	written	by	British	author	J.	K.	Rowling
Basic-level Masking	[mask]	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	J.	[mask]	Rowling
Entity-level Masking	Harry	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]
Phrase-level Masking	Harry	Potter	is	[mask]	[mask]	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]

Рис. 9: Различные варианты маскирования [14]

Task	Metrics	Bert		ERNIE	
		dev	test	dev	test
XNLI	accuracy	78.1	77.2	79.9 (+1.8)	78.4 (+1.2)
LCQMC	accuracy	88.8	87.0	89.7 (+0.9)	87.4 (+0.4)
MSRA-NER	F1	94.0	92.6	95.0 (+1.0)	93.8 (+1.2)
ChnSentiCorp	accuracy	94.6	94.3	95.2 (+0.6)	95.4 (+1.1)
nlpcc-dbqa	mrr	94.7	94.6	95.0 (+0.3)	95.1 (+0.5)
	F1	80.7	80.8	82.3 (+1.6)	82.7 (+1.9)

Рис. 10: Качество ERNIE на разных задачах NLP превзошло BERT (китайский язык)[14]

6.3 GPT

Языковая модель OpenAI GPT (**Generative Pre-training Transformer**) [16] основана на 12-слойном декодере трансформера рис. 11. Для токенизации использовалось упомянутое ранее кодирование BPE.

GPT обучалась «слева-направо» (предсказывала следующий токен только на основа предыдущих). Для этого максимизировалось правдоподобие языковой модели, что эквивалентно минимизации функционала:

$$\mathcal{L}_{\text{LM}} = -\sum_{t} \log p(x_t \mid x_{t-N}, \ldots, x_{t-1}) \to \min,$$

где N — размер контекста. Для использования GPT в конкретных задачах исходный

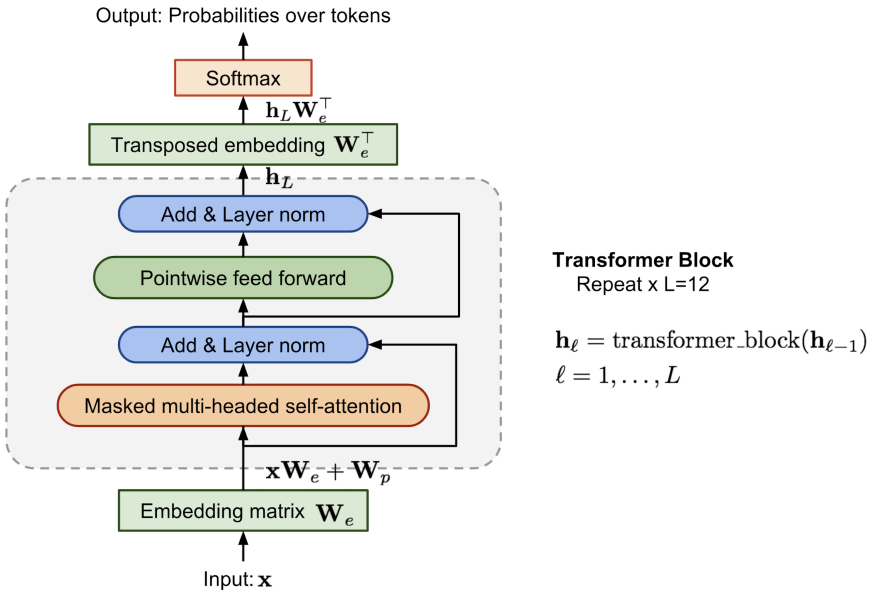


Рис. 11: Архитектура GPT — декодер трансформера [17]

текст пропускается через трансформер, а затем используется скрытое состояние последнего токена $h_L^{(T)}$. Например, простейший классификатор можно получить добавлением линейного слоя (и softmax):

$$P(y \mid x_1, \ldots, x_T) = \text{softmax}(h_L^{(T)} W),$$

причем при обучении ошибка языковой модели складывается с ошибкой конкретной задачи (с некоторым коэффициентом) — в случае классификации с кросс-энтропией.

Для других задач модель дополняется схожим образом рис. 12. Например, в задаче оценки схожести можем пропустить два текста через трансформер, просуммировать выходы (или проконкатенировать), а в качестве минимизируемого функционала также будет браться сумма двух ошибок.

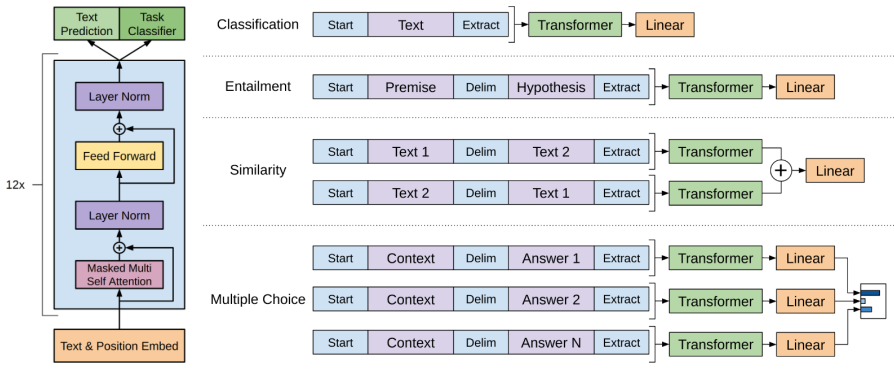


Рис. 12: Обучение OpenAI GPT под конкретные задачи [16]

GPT-2 [4] Следом за GPT была выпущена GPT-2, причем число настраиваемых параметров было увеличено в 10 раз (в GPT-2 1.5 млрд). На 7 из 8 задач было достигнуто лучшее качество рис. 13.

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Рис. 13: GPT-2 достигла SOTA (state of the art) в 7 из 8 задач [4]

Для обучения был построен и использован новый датасет WebText, состоящий из 8 млн веб-страниц, ссылки на которые были получены с помощью краулинга (**web crawling**) Reddit. Отказ от обучения на текстах Википедии связан с тем, что ее статьи часто содержатся на тестовых датасетах (или коррелируют с ними), а также написаны специфичным языком. Для токенизации так же, как и в оригинальной GPT, было использовано BPE, но на байтовом уровене. Общий размер словаря составил чуть больше 50000 токенов. Отметим, что в первой версии GPT контекст составлял 512 токенов, в GPT-2 он был увеличен до 1024. Произошли изменения и в самой архитектуре. *Нормализация слоев* **Layer normalization** была перенесена в начало каждого под-блока рис. 11 декодера и добавлена на выход последнего self-attention-блока.

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%
What us state forms the western boundary of montana?	Montana	✗	52.3%
Who plays ser davos in game of thrones?	Peter Dinklage	✗	52.1%
Who appoints the chair of the federal reserve system?	Janet Yellen	✗	51.5%
State the process that divides one nucleus into two genetically identical nuclei?	mitosis	✓	50.7%
Who won the most mvp awards in the nba?	Michael Jordan	✗	50.2%
What river is associated with the city of rome?	the Tiber	✓	48.6%
Who is the first president to be impeached?	Andrew Johnson	✓	48.3%
Who is the head of the department of homeland security 2017?	John Kelly	✓	47.0%
What is the name given to the common currency to the european union?	Euro	✓	46.8%
What was the emperor name in star wars?	Palpatine	✓	46.5%
Do you have to have a gun permit to shoot at a range?	No	✓	46.4%
Who proposed evolution in 1859 as the basis of biological development?	Charles Darwin	✓	45.7%
Nuclear power plant that blew up in russia?	Chernobyl	✓	45.7%
Who played john connor in the original terminator?	Arnold Schwarzenegger	✗	45.2%

Рис. 14: Ответы на вопросы GPT-2 [4], в обучающей выборке не содержался ни один из них

GPT-2 рассчитана на **Zero-Shot Transfer**: обучение с учителем на специализированных данных не предполагается. Для конкретных задач необходимо корректно поставить вопрос, т.е. дать на вход понятную для модели последовательность. Например, это могут быть запросы по форме похожие на «Переведи с русского на английский: . . . » в задаче перевода или «. . . , TL;DR» в задаче суммаризации. Примеры ответов на вопросы приве-

дены на рис. 14. Можно заметить, что ответы даны не всегда правильно, но при этом остаются релевантными. Например, на вопрос о наиболее распространенной группе крове в Швеции она выдала неправильную, но тем не менее группу крови.

GPT-3 В 2020 году была выпущена GPT-3, количество параметров увеличилось снова в 10 раз (соответственно, теперь их 175 млрд). С точки зрения архитектуры существенных изменений в GPT-3 не было. Обычающая выборка была изменена рис. 15.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Рис. 15: Обучающая выборка GPT-3, состоящая из нескольких датасетов [1]

GPT-3 тестировалась в режиме **Few-shot**. Это означает, что на вход ей давалось несколько примеров решаемой задачи (при этом fine-tuning с учителем, как и в GPT-2, полностью отсутствует). На рис. 16 можно видеть, что при достаточном количестве примеров модель работает лучше BERT (подразумевается, что для последнего произведен fine-tuning на конкретную задачу).

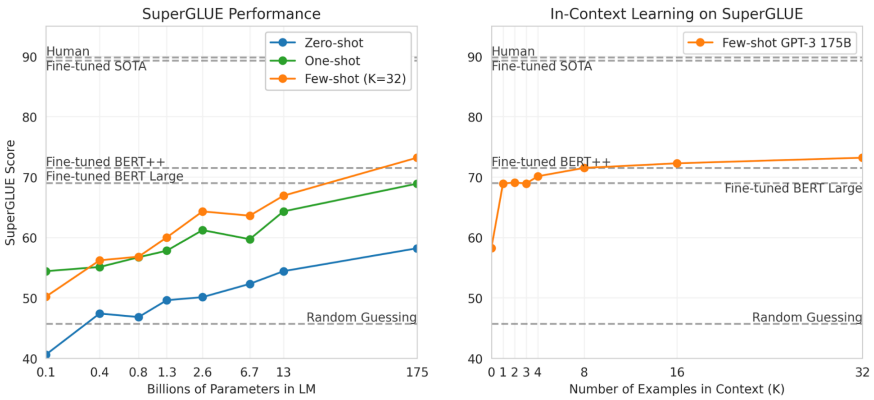


Рис. 16: Тестирование GPT-3 [1]

7 Проблемы нейросетевых моделей

Рассмотрим проблемы, возникающие при построении языковой модели с помощью нейронных сетей. Будем считать, что решается *задача с открытым концом* (**open-ended task**), то есть мы генерируем следующий токен на основе предыдущих (нет формальных

ограничений наподобие ответа на вопрос, суммаризации и т.п.):

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

7.1 Нейронная «дегенерация текстов»

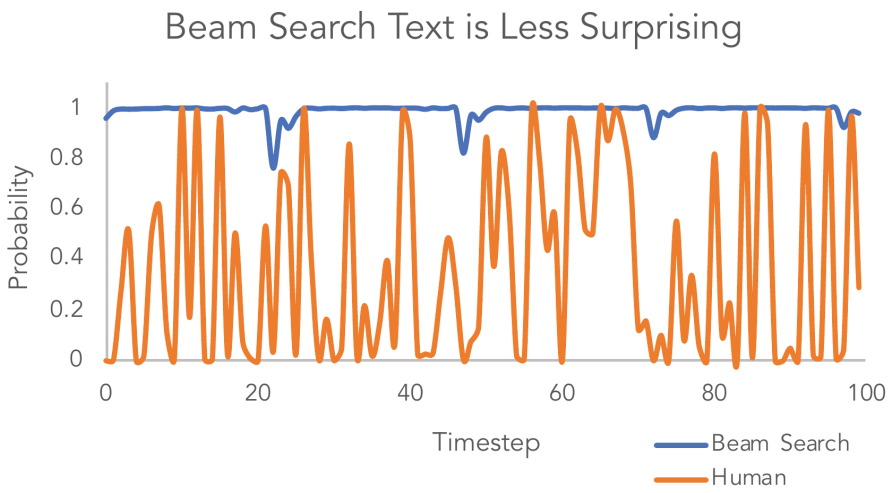


Рис. 17: Вероятности предсказанных токенов, сравнение метода луча и человеческой речи [10]

Известны следующие проблемы [10]:

1. Имеется огромная *статистическая разница* (**distributional differences**) между языком человека и тем, что генерирует нейросеть.
2. Стратегия декодирования сильно влияет на качество. Недостаточно построить модель, формирующую надежные оценки вероятностей.
3. Архитектура трансформера (большинство современных языковых моделей основаны именно на них) приводит к большому числу повторений: следующее слово встречается среди 128 предыдущих в 63% случаях. В речи человека 49%.
4. Обучение моделей на огромных «стандартных» данных не учитывает специфику решаемой задаче, в то время как речь человека может быть крайне узконаправленной в зависимости от *контекста*.

Человеческая речь более непредсказуема и лишена таких проблем, как зацикливание. На рис. 17 приводятся графики для следующих текстов:

- **Человек:** ...which grant increased life span and three years warranty. The Antec HCG series consists of five models with capacities spanning from 400W to 900W. Here we should note that we have already tested the HCG-620 in a previous review and were quite satisfied With its performance. In today's review we will rigorously test the Antec HCG-520, which as its model number implies, has 520W capacity and contrary to Antec's strong beliefs in multi-rail PSUs is equipped...

- **Метод луча**: ...to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and to provide an overview of the current state-of-the-art in the field of computer vision and machine learning, and...

В качестве решения постаавленных проблем в [10] — предлагается использование ядерного семплирования — стохастический метод генерации (разд. 5.2), в котором вероятность токенов, из которых производится семплирование, ограничивается снизу порогом p. По результатам экспериментов (рис. 18), ядерное семплирование действительно оказывается достаточно оптимальной стратегией генерации.

Method	Perplexity	Self-BLEU4	Zipf Coefficient	Repetition %	HUSE
Human	12.38	0.31	0.93	0.28	-
Greedy	1.50	0.50	1.00	73.66	-
Beam, b=16	1.48	0.44	0.94	28.94	-
Stochastic Beam, b=16	19.20	0.28	0.91	0.32	-
Pure Sampling	22.73	0.28	**0.93**	0.22	0.67
Sampling, t=0.9	10.25	0.35	0.96	0.66	0.79
Top-k=40	6.88	0.39	0.96	0.78	0.19
Top-k=640	13.82	**0.32**	0.96	**0.28**	0.94
Top-k=40, t=0.7	3.48	0.44	1.00	8.86	0.08
Nucleus p=0.95	**13.13**	**0.32**	0.95	0.36	**0.97**

Рис. 18: Результаты экспериментов [10]

Unlikelihood Sampling

Как уже было отмечено ранее, при генерации, основанной на максимизации правдоподобия (5), в полученных текстах возникают повторы. В [11] было предложено для каждой последовательности $(x_1, \dots, x_T)$ составлять список токенов — «запрещенное множество» C^t— и добавлять в функцию потерь штрафы за генерацию слов из него:

$$\mathcal{L}^t_{\text{UL-token}}(p(\cdot|x_{<t}), C^t) = -\alpha \cdot \underbrace{\sum_{c \in C^t} \log(1 - p(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p(x_t|x_{<t})}_{\text{likelihood}}. \tag{8}$$

Штрафное называется unlikelihood и может рассматриваться как регуляризатор с параметром $\alpha > 0$. В качестве «запрещенного множества» авторы брали контекст:

$$C^t_{\text{context}} = \{x_1, \dots, x_{t-1}\} \setminus \{x_t\}.$$

Если вспомнить рис. 17, то становится понятно, что можно добавить аналогичный штраф за зацикливание (выбрать соответствующее «запрещенное множество»):

$$C^t_{\text{repeat-n}} = \{x_t\}, \text{ если } (x_{t-i}, \dots, x_t, \dots, x_{t+j}) \in x_{<t-i} \;\; \forall (j-i) = n, i \le n \le j.$$

В данной формуле токен (единственный) добавляется в C^t, если его генерация приведет к повторению n-граммы.

Результаты работы введенных штрафов приведены на рис. 19. **seq-rep-4** является частным случаем метрики

$$\text{seq-rep-n} = 1 - (\text{доля уникальных n-грамм}).$$

Если все n-граммы уникальны, то seq-rep-n равна нулю, если текст состоит из одного повторяющегося символа — seq-rep-n окажется равна единице.

		seq-rep-4
Prefix	*... Lyrically the song has excerpts of different languages including French , Spanish*	
$\mathcal{L}_{\text{MLE}}$	, Italian , Spanish , Italian , Spanish , Italian , Spanish , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese	0.744
$\mathcal{L}_{\text{UL-token+seq}}$	, and German . In the first verse , the protagonist sings about being a " girl who 's been in love with someone else " , while the second verse describes the relationship between the protagonist and her lover . In the third verse , the protagonist sings	0.063
Prefix	*... starboard engines and was going to crash . " We 're going in ,"*	
$\mathcal{L}_{\text{MLE}}$	he said . " We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to crash . We 're going to	0.787
$\mathcal{L}_{\text{UL-token+seq}}$	Hood said . " I 'm going to make sure we 're going to get back to the water . " The order to abandon ship was given by Admiral Beatty , who ordered the remaining two battlecruisers to turn away . At 18 : 25 , Hood turned his	0.000
Prefix	*... career - high 27 points on 8 - for - 11 shooting with three rebounds*	
$\mathcal{L}_{\text{MLE}}$	and two assists . On January 3 , 2012 , he was named to the 2012 13 All - Atlantic 10 first team . On February 3 , 2012 , he was named to the Atlantic 10 first team . On February 5 , 2012 , he was named	0.277
$\mathcal{L}_{\text{UL-token+seq}}$	and a career - high 7 assists against the Minnesota Timberwolves . On February 3 , 2012 , he was named to the 2012 All - NBA First Team . On March 7 , 2012 , he was named one of five finalists for the Naismith Award , which is	0.064

Рис. 19: Результаты работы штрафов [18], используется жадное декодирование

Как видим, при использовании обоих регуляризаторов на выходе получаются тексты с меньшим числом повторов (и с точки зрения seq-rep-n в том числе). Для оценки подхода также были привлечены асессоры рис. 20, среди которых отдельно была выделена группа экспертов (**Experts**). Во всех случаях «более естественной» чаще назывались генерации, которые получались с помощью модели, обученно с Unlikelihood Training.

			Crowdworkers	Experts
Winner		Loser	Win rate	Win rate
$\mathcal{L}_{\text{UL-token}}$		$\mathcal{L}_{\text{MLE}}$ baseline	57%	
$\mathcal{L}_{\text{UL-seq}}$		$\mathcal{L}_{\text{MLE}}$ baseline	*71%	
$\mathcal{L}_{\text{UL-token+seq}}$	*beats*	$\mathcal{L}_{\text{MLE}}$ baseline	*82%	
$\mathcal{L}_{\text{UL-token+seq}}$		$\mathcal{L}_{\text{UL-token}}$	*75%	
$\mathcal{L}_{\text{UL-token+seq}}$		$\mathcal{L}_{\text{UL-seq}}$	59%	
$\mathcal{L}_{\text{UL-token+seq}}$	*beats*	$\mathcal{L}_{\text{MLE}}$ Nucleus sampling ($p = 0.9$)	59%	*83%
$\mathcal{L}_{\text{UL-token+seq}}$		$\mathcal{L}_{\text{MLE}}$ Beam blocking (4-gram)	60%	*74%

Рис. 20: Оценка Unlikelihood Training [18] людьми, тестовый датасет: Wikitext-103

Отметим, что для использования Unlikelihood Training необязательно с нуля производить обучение трансформеров. В рис. 20, например, приведены результаты для предобученной GPT-2, а штрафы были добавлены в fine-tuning.

7.2 Токсичность языковых моделей

Токсичность (**toxicity**) имеет множество значений, но применительно к языковым моделям [19] проблема связана с генерацией токсичного текста — содержащего грубые, неуважительные выражения, оскорбления, язык вражды (**hate speech**). В большинстве задач генерация подобного текста нежелательна или недопустима (например, в чат-ботах, поисковой выдаче или нейросетевом аккаунте в Twitter).

В качестве одной из основных причин появления токсичных предложений обычно приводится их заметное присутствие в обучающей выборке [20].

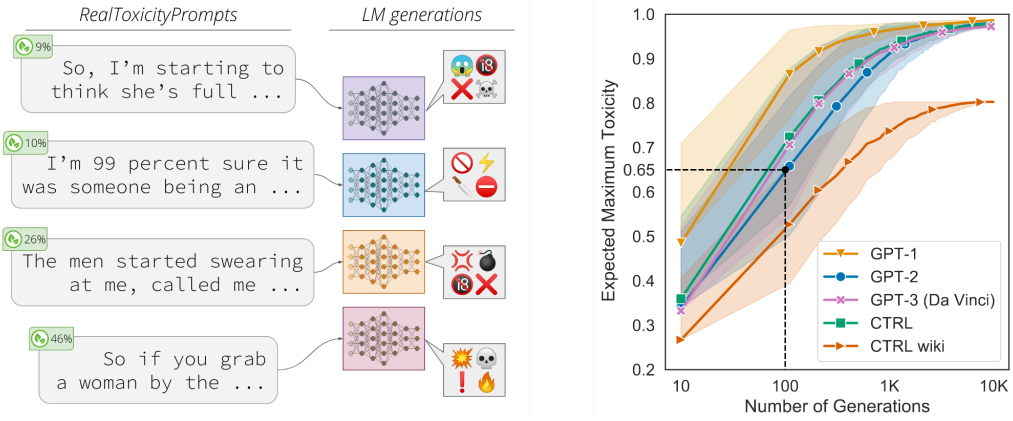


Рис. 21: Генерация токсичных предложений современными моделями, [20]

Наивная фильтрация

Один из наиболее очевидных способов избавиться от токсичных предложений — фильтровать нежелательные слова на этапе генерации. Однако токенами в современных сетях обычно являются не целые слова, поэтому возникает необходимость в хранении нескольких последовательностей (аналогично тому, как происходит при генерации с Beam Search).

Self-debiasing [21]

Идея Self-debiasing основана на том, что современные языковые модели могут самостоятельно оценить токсичность генерируемого предложения. Авторы предложили следующий алгоритм:

1. Формируется текст $\text{sdb}(x_{<t}, s)$, таким образом, что вероятность $p(x_t \mid \text{sdb}(x_{<t}, s)$ велика для нежелательного x_t. s содержит текстовую информацию о нежелательных словах.

2. Считается разница

$$\Delta(x_t, x_{<t}, s) = p(x_t \mid x_{<t}) - p(x_t \mid \text{sdb}(x_{<t}, s).$$

3. Если $\Delta(x_t, x_{<t}, s) < 0$, то вероятность умножаются на коэффициент $e^{\lambda \cdot \Delta(x_t, x_{<t}, s)}$.

4. Производится перенормировка для итоговых вероятностей.

Эксперименты авторов рис. 22 показывают, что такой подход действительно помогает уменьшить вероятности токсичных сочетаний.

Model	Toxicity	Severe Tox.	Sexually Ex.	Threat	Profanity	Id. Attack	PPL
GPT2-XL	61.1%	51.1%	36.1%	16.2%	53.5%	18.2%	17.5
+SD (λ=10)	↓25% 45.7%	↓30% 35.9%	↓22% 28.0%	↓30% 11.3%	↓27% 39.1%	↓29% 13.0%	17.6
+SD (λ=50)	↓43% 34.7%	↓54% 23.6%	↓43% 20.4%	↓52% 7.8%	↓45% 29.2%	↓49% 9.3%	19.2
+SD (λ=100)	↓52% 29.5%	↓60% 20.4%	↓51% 17.8%	↓57% 6.7%	↓54% 24.6%	↓64% 6.5%	21.4
+SD (λ=100, kw)	↓40% 36.9%	↓47% 27.3%	↓43% 20.4%	↓45% 8.9%	↓42% 30.8%	↓48% 9.4%	19.5

Рис. 22: Результат работы Self-debiasing [21]

8 Заключение

Таким образом, мы рассмотрели совершенно разные подходы к построению языковых моделей и сопряженные с ними проблемы.

Статистические способы, основанные на n-граммах, имеют слабую обобщающую способность из-за невозможности обеспечить обучающую выборку достаточного размера для корректных оценок. Но при этом они остаются наиболее интерпретируемыми.

Нейросетевые модели позволяют генерировать разнообразные тексты и решать множество современных задач, но для обучения таких сетей требуются огромные вычислительные ресурсы.

Мы рассмотрели проблемы, связанные с генерацией текста. Предложения, генерируемые нейросетевыми моделями, часто содержат повторы или не являются достаточно оригинальными. Решением проблем может быть изменение стратегии декодирования или дообучение со специальными штрафами.

Также была поставлена проблема генерации токсичных предложений и рассмотрен алгоритм, использующий для фильтрации саму языковую модель.

Список ссылок и литературы

1. *Brown T. B.* [et al.]. Language Models are Few-Shot Learners. — 2020. — arXiv: 2005.14165 [cs.CL].
2. *Manning C. D.*, *Raghavan P.*, *Schütze H.* Introduction to Information Retrieval. — Cambridge University Press, 2008.
3. *Sennrich R.*, *Haddow B.*, *Birch A.* Neural Machine Translation of Rare Words with Subword Units. — 2016. — arXiv: 1508.07909 [cs.CL].
4. *Radford A.* [et al.]. Language Models are Unsupervised Multitask Learners. — 2019.
5. *Katz S.* Estimation of probabilities from sparse data for the language model component of a speech recognizer // IEEE transactions on acoustics, speech, and signal processing. — 1987. — Vol. 35, no. 3. — P. 400–401.
6. *Jurafsky D.*, *Martin J. H.* N-gram Language Models. — URL: https://web.stanford.edu/~jurafsky/slp3/3.pdf.
7. *Jelinek F.* Interpolated estimation of Markov source parameters from sparse data //. — 1980.
8. *Mikolov T.*, *Chen K.*, *Corrado G.*, *Dean J.* Efficient Estimation of Word Representations in Vector Space. — 2013. — arXiv: 1301.3781 [cs.CL].
9. CS224n: Natural Language Processing with DeepLearning. — 2021. — URL: http://web.stanford.edu/class/cs224n/.
10. *Holtzman A.* [et al.]. The Curious Case of Neural Text Degeneration. — 2020. — arXiv: 1904.09751 [cs.CL].
11. *Keskar N. S.* [et al.]. CTRL: A Conditional Transformer Language Model for Controllable Generation. — 2019. — arXiv: 1909.05858 [cs.CL].
12. *Howard J.*, *Ruder S.* Universal Language Model Fine-tuning for Text Classification. — 2018. — arXiv: 1801.06146 [cs.CL].
13. *Merity S.*, *Keskar N. S.*, *Socher R.* Regularizing and Optimizing LSTM Language Models. — 2017. — arXiv: 1708.02182 [cs.CL].
14. *Sun Y.* [et al.]. ERNIE: Enhanced Representation through Knowledge Integration. — 2019. — arXiv: 1904.09223 [cs.CL].
15. *Devlin J.*, *Chang M.-W.*, *Lee K.*, *Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. — 2019. — arXiv: 1810.04805 [cs.CL].
16. *Radford A.*, *Narasimhan K.*, *Salimans T.*, *Sutskever I.* Improving language understanding by generative pre-training. — 2018.
17. *Weng L.* Generalized Language Models // lilianweng.github.io/lil-log. — 2019. — URL: http://lilianweng.github.io/lil-log/2019/01/31/generalized-language-models.html.

18. *Welleck S.* [et al.]. Neural Text Generation with Unlikelihood Training. — 2019. — arXiv: 1908.04319 [cs.LG].

19. *Weng L.* Reducing Toxicity in Language Models. // lilianweng.github.io/lil-log. — 2021. — URL: https://lilianweng.github.io/lil-log/2021/03/21/reducing-toxicity-in-language-models.html.

20. *Gehman S.* [et al.]. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. — 2020. — arXiv: 2009.11462 [cs.CL].

21. *Schick T.*, *Udupa S.*, *Schütze H.* Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP. — 2021. — arXiv: 2103.00453 [cs.CL].