Авторегресионная генерация текста

Попов Артём, OzonMasters, осень 2022

Natural Language Processing

Авторегрессионная генерация текста

Дано: $x = (x_1, \dots, x_n)$ – входная последовательность, $y_0 = < START >$

Необходимо сгенерировать последовательность $y_1, \dots, y_m$

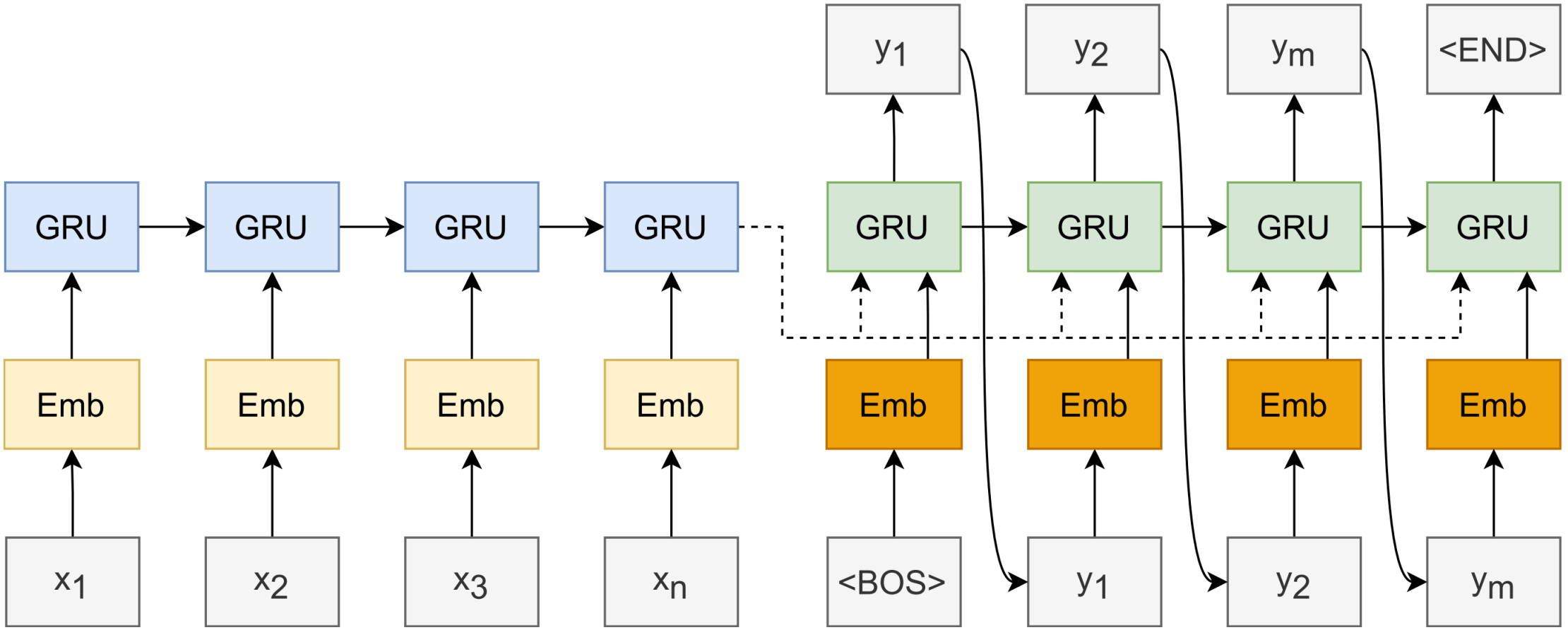
Используем авторегресионную генерацию!

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

1. Получение из модели логитов $q(y|y_{<i}, x) \equiv q(y|y_{<i})$
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

Применение кодировщика-декодировщика (RNN)



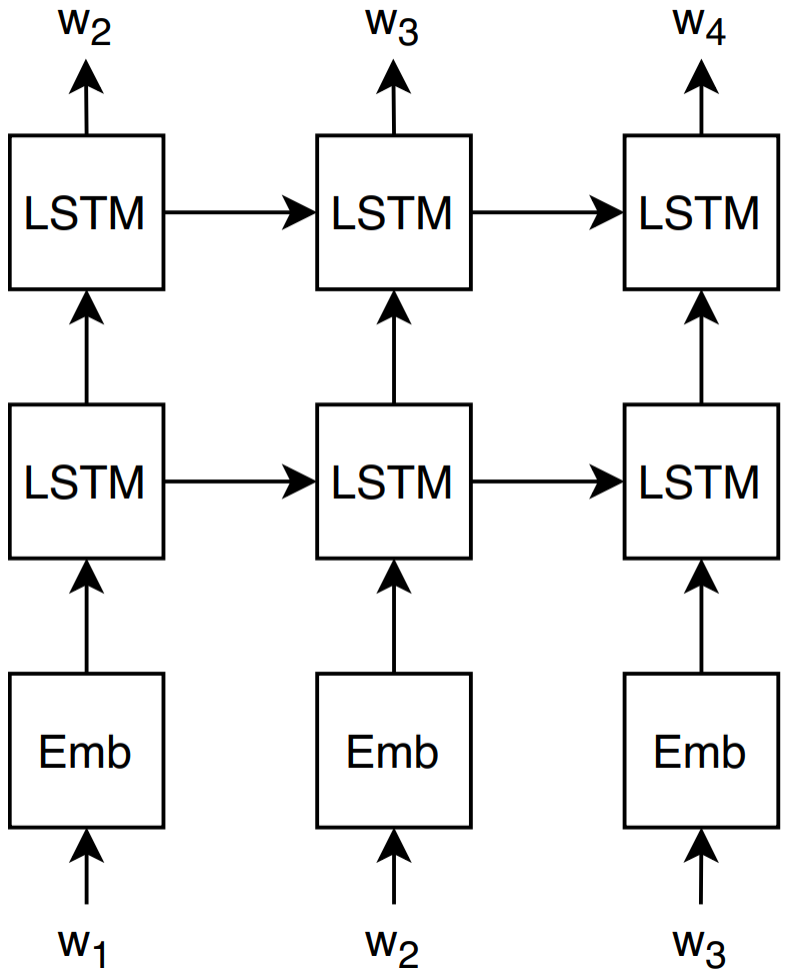
Эффективный пересчёт логитов: RNN (без внимания)

Сложность обработки кодировщиком в RNN равна $O(nd^2)$

При итеративной генерации m токенов, сложность равна
$O\left(d^2 + |W|d + \cdots + md^2 + |W|d\right) =$
$= O\left(m^2d^2 + m|W|d\right)$

Если кэшировать скрытые состояния с предыдущего шага, сложность равна
$O\left(d^2 + |W|d + \cdots + d^2 + |W|d\right) =$
$= O(md^2 + m|W|d)$

Затраты по памяти $O(d)$



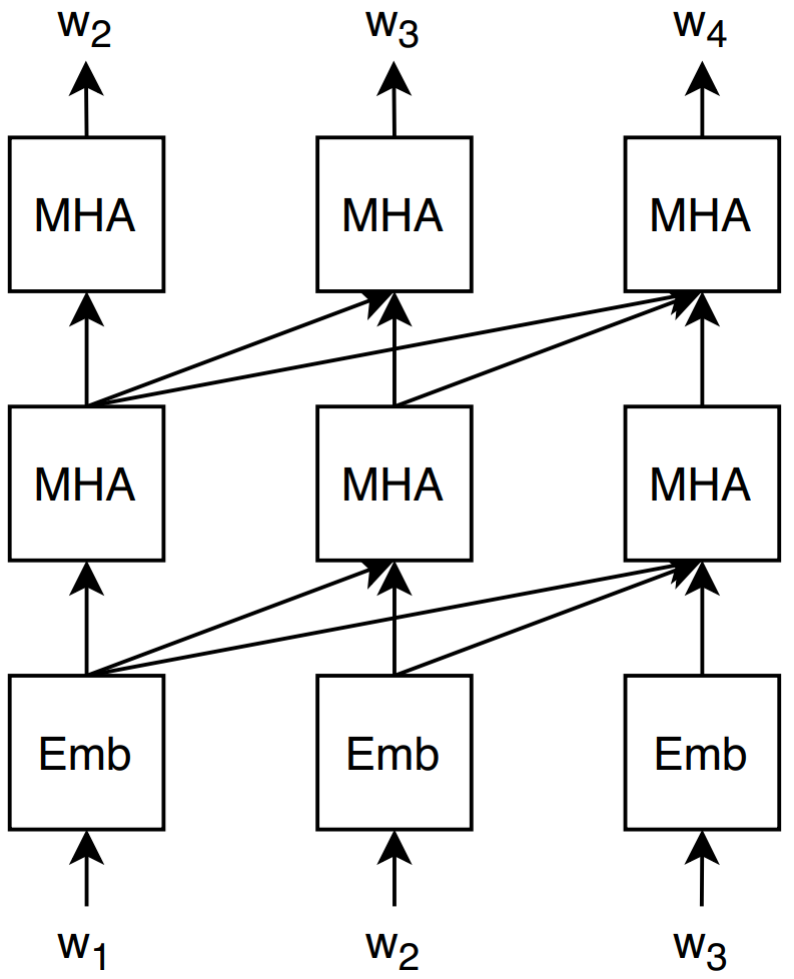
Эффективный пересчёт логитов: трансформер

Сложность обработки кодировщиком в трансформере равна $O(nd^2 + n^2 d)$

При итеративной генерации m токенов, сложность декодировщика равна
$O\left(d^2 + nd + |W|d + \cdots + md^2 + nmd + |W|d\right) =$
$= O\left(m^2 d^2 + m^2 nd + m|W|d\right)$

Если кэшировать все “ключи” и “значения” со всех предыдущих шагов, сложность равна
$O\left(d^2 + nd + |W|d + \ ... + d^2 + nd + |W|d\right) =$
$= O(md^2 + mnd \ + m|W|d)$

Затраты по памяти $O(nd + md)$



декодер

Стратегии выбора следующего токена

Наиболее вероятный токен:

$$y_i = \arg\max_{y \in Y} p(y|y_{<i})$$

- Стоит использовать при генерации коротких текстов
- Хорошо подходит для задачи автодополнения текста / кода
- Быстро зацикливается (генерация повторяющихся одинаковых фрагментов)

Сэмплирование токенов:

$$y_i \sim p(y|y_{<i})$$

- Стоит использовать при генерации длинного текста
- Хорошо подходит для режима свободной генерации (художественный текст, генеративная диалоговая система)

Получение распределения: softmax с температурой

Стандартный способ получения распределения – softmax:

$$p(y|y_{<i}) = softmax_{y\in Y}\big(q(y|y_{<i})\big) = \frac{\exp\big(q(y|y_{<i})\big)}{\sum_{u\in \mathrm{Y}} \exp(q(u|y_{<i})}$$

Повышение/понижение температуры в softmax:

$$p(y|y_{<i}) = softmax_{y\in Y}\left(\frac{q(y|y_{<i})}{T}\right)$$

При малых T распределение стремится к вырожденному.

При больших T распределение стремится к равномерному.

Получение распределения: topK и topP

Для уменьшения шума будем занулять все элементы распределения кроме нескольких максимальных.

TopK стратегия – оставляем k токенов с наибольшими вероятностями

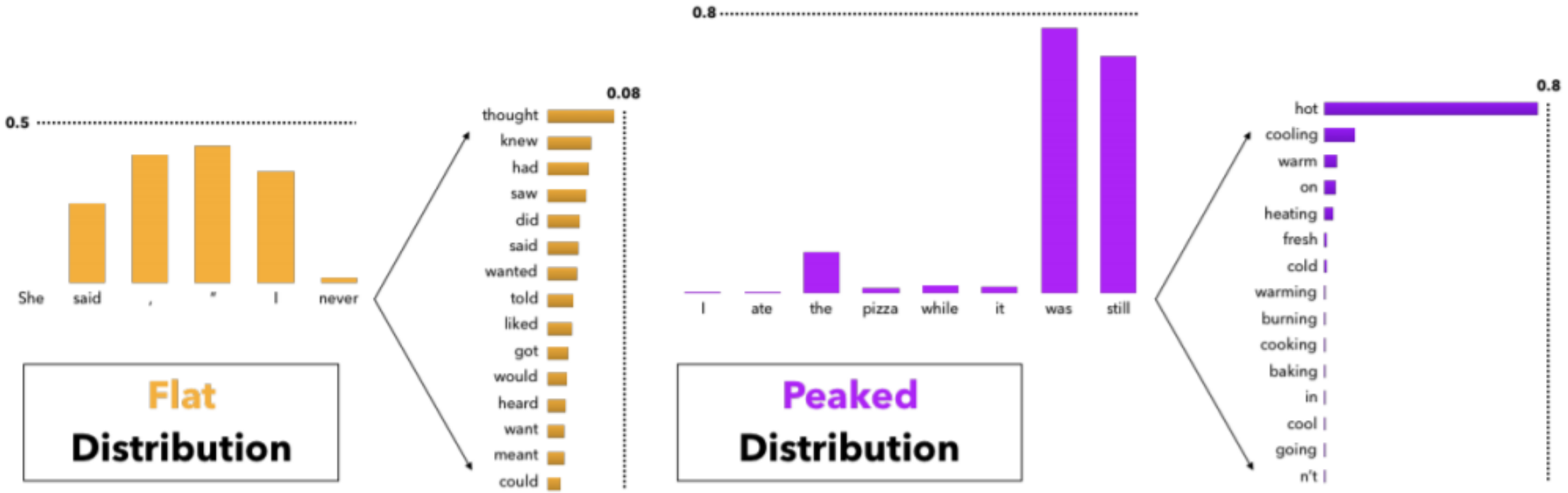
TopP стратегия – оставляем минимальное по размеру множество из k токенов, сумма вероятностей которых превышает p

$$\hat{p}_y = softmax_{y \in Y}\left(q(y|y_{<i})\right)$$

$$p(y|y_{<i}) = \frac{\hat{p}_y \, \mathbb{I}[y \in V_k]}{\sum_{u \in Y} \hat{p}_u \, \mathbb{I}[u \in V_k]}$$

Holtzman et al (ICLR 2020); The Curious Case Of Neural Text Degeneration

Почему topP лучше topK?



Penalized sampling

Генеративные модели могут зацикливаться: повторять один и тот же токен много раз.

$$Z_y = \begin{cases} \theta, & if\ y \in \{y_1, \dots, y_{i-1}\} \\ 1, & \text{иначе} \end{cases}$$

$$p(y|y_{<i}) = softmax_{y \in Y}\left(\frac{q(y|y_{<i})}{Z_y}\right)$$

Чтобы этот трюк работал, все $q(y|y_{<i})$ должны иметь один знак!

Shirish et al (2019); CTRL: A Conditional Transformer Language Model for Controllable Generation

Beam search (лучевой поиск)

Вход модели — множество входных последовательностей $V = [< START >]$

Повторяем пока $|A| \neq m$ или $i \neq M$:

1. Для всех последовательностей $y_{<i} \in V$ получаем $q(y|y_{<i}, x)$
2. Для всех $y_{<i} \in V$ получаем $p(y|y_{<i}, x)$
3. Для каждой $y_{<i} \in V$ выбираем m вариантов следующего токена

Beam search (лучевой поиск)

Вход модели — множество входных последовательностей $V = [< START >]$

Повторяем пока $|A| \neq m$ или $i \neq M$:

1. Для всех последовательностей $y_{<i} \in V$ получаем $q(y|y_{<i}, x)$
2. Для всех $y_{<i} \in V$ получаем $p(y|y_{<i}, x)$
3. Для каждой $y_{<i} \in V$ выбираем m вариантов следующего токена
4. Оцениваем вероятности всех получившихся последовательностей
$$p(y_1, \dots, y_i|y_0, x) = p(y_i|y_{<i}, x)\, p(y_{i-1}|y_{i-2}, x) \dots p(y_1|y_0, x)$$
5. Выбираем m последовательностей с наибольшими вероятностями

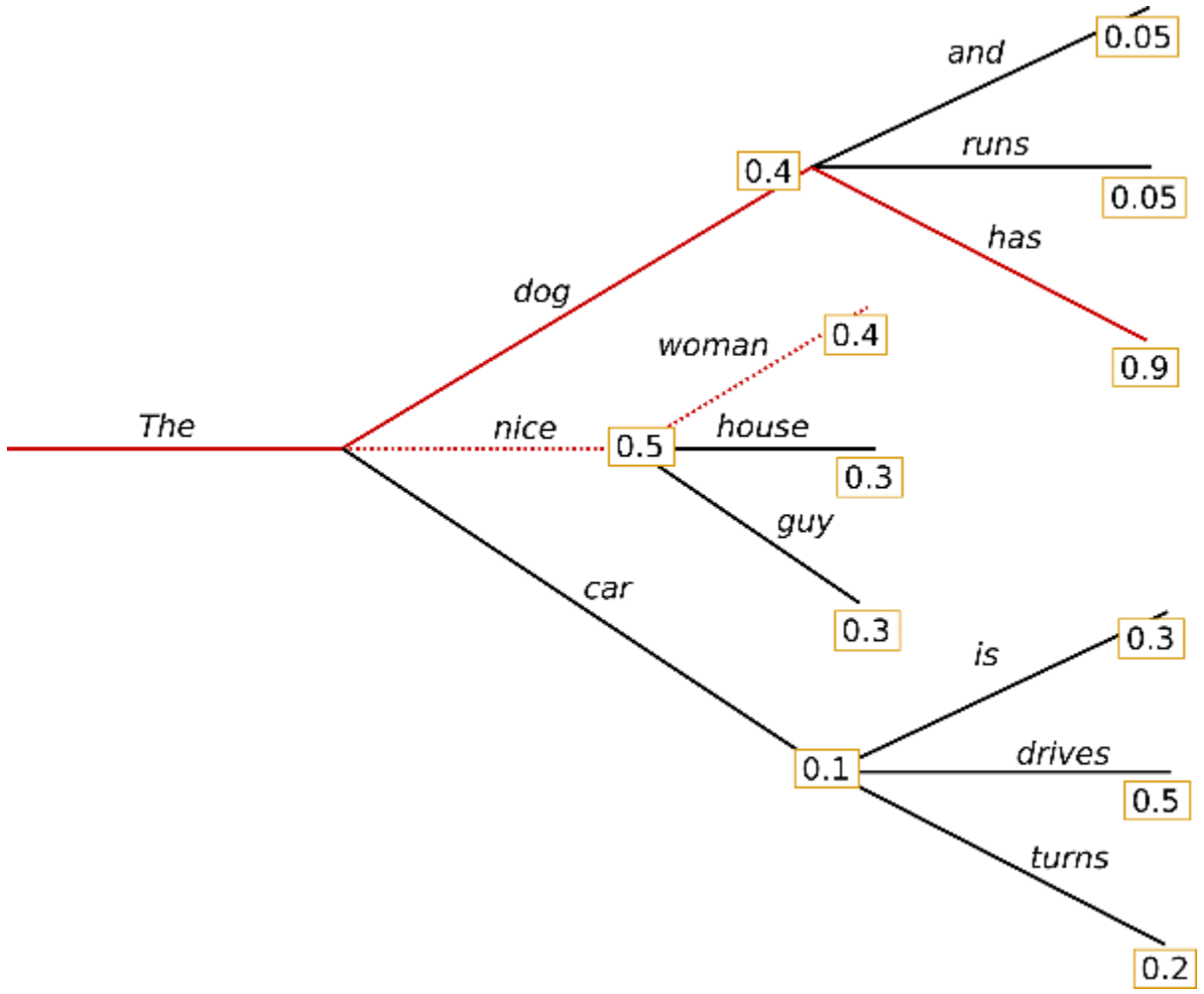
Beam search (лучевой поиск)

Вход модели — множество входных последовательностей $V = [< START >]$

Повторяем пока $|A| \neq m$ или $i \neq M$:

1. Для всех последовательностей $y_{<i} \in V$ получаем $q(y|y_{<i}, x)$
2. Для всех $y_{<i} \in V$ получаем $p(y|y_{<i}, x)$
3. Для каждой $y_{<i} \in V$ выбираем m вариантов следующего токена
4. Оцениваем вероятности всех получившихся последовательностей
$$p(y_1, \dots, y_i|y_0, x) = p(y_i|y_{<i}, x)\, p(y_{i-1}|y_{i-2}, x) \dots p(y_1|y_0, x)$$
5. Выбираем m последовательностей с наибольшими вероятностями
6. Если среди m последовательностей есть законченные (кончаются токеном $< END >$ или максимальной длины), добавляем их в A
7. Оставшиеся последовательности добавляем в V

Иллюстрация работы beam search



Свойства beam search

- Beam search выбирает наиболее вероятную последовательность, а не последовательность наиболее вероятных токенов
- Beam search необходим, если мы хотим выдавать пользователю несколько вариантов ответа
- Beam search с разумными m практически всегда позволяет получить более адекватные результаты, чем при $m = 1$
- Большие m в beam search приводит к практически одинаковым «безопасным» последовательностям
- При больших m большинство последовательностей будут короткими
- Beam search часто используют с детерминированными стратегиями выбора токена и температурой, реже с семплированием

Сравнение разных стратегий 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**

Что делать на практике?

- Одна и та же обученная модель может вести себя по разному в зависимости от подобранных параметров генерации
- Оптимальный вариант – подбор параметров генерации на валидации, замер качества по тесту