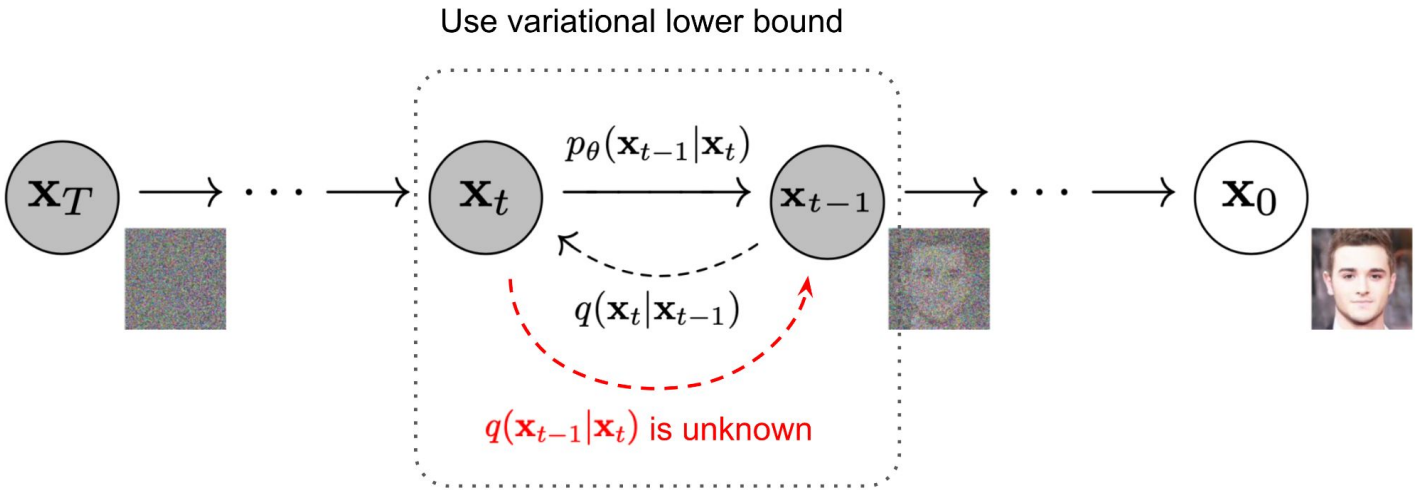
# Diffusion Models for Non-autoregressive Text Generation: A Survey

Докладчик: Мурат Апишев

# Диффузионная модель (DM)

**Идея:**

- *forward*: добавление случайного шума в объект в течение T шагов
- *reverse*: обучение сети итеративному восстановлению объекта из шума



# Forward-процесс

Объект x_0 ~ q(x) переводится в последовательность x_1, …, x_T последовательным сэмплированием из распределения

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$$

$\beta_t \in (0,1)$ - масштаб шума, растущий с номером итерации по заданному расписанию

За счёт репараметризации можно вычисление x_t можно производить напрямую из x_0, минуя промежуточные шаги:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, \sqrt{1-\bar{\alpha}_t}\mathbf{I})$$

$$\alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

# Reverse-процесс

Искомое апостериорное распределение $q(x_{t-1}|x_t)$ можно приблизить (при малых \beta) нормальным и параметризовать нейросетью:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Старт с $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$ с итеративным очищением до получения x_0

Лосс на основе VLB:

$$\mathcal{L}_{\text{vlb}} = \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(x_t|x_0)||p_\theta(x_T))}_{\mathcal{L}_T}] - \underbrace{\log p_\theta(x_0|x_1)}_{\mathcal{L}_0}$$

$$+ \mathbb{E}_q[\sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{\mathcal{L}_{t-1}}]$$

# Функционал качества

- Лосс может определяться не только напрямую через вариационную нижнюю оценку
- Можно упростить и свести всё к MSE с усреднением по всем объектам x_0 и итерациям t
- Предсказывать для объекта (и штрафовать в MSE) можно разные величины:
  - среднее апостериорного распределения $q(x_{t-1}|x_t, x_0)$
  - сам объект x_0
  - внесённый в объект x_0 до шага t шум

# Диффузионная модель для текстов

- **Зачем:** можно сильно ускорить инференс, если генерировать всё сразу
- DM хорошо работает для непрерывных объектов (изображения, аудио)
- В качестве базовой архитектуры обычно используется U-Net
- Тексты имеют дискретную природу, работать в чистом виде с токенами нельзя, нужны обходные пути:
  - использование дискретного вариант диффузии
  - переход от токенов к их эмбеддингам
  - переход от текстов к суррогатным изображениям
  - …
- Базовая архитектура для текстов - Transformer

# Дискретная диффузионная модель

Для $x \in 1, ..., K$ вводятся матрицы переходов $[\mathbf{Q}_t]_{ij} = q(x_t = j | x_{t-1} = i)$

Формула перехода между шагами:

$$q(x_t|x_{t-1}) = \mathrm{Cat}(x_t; \boldsymbol{p} = x_{t-1}\mathbf{Q}_t)$$

x - one-hot векторы **слов**, Cat - распределение над x с вероятностями p

Формула перехода к нужному шагу:

$$q(x_t|x_0) = \mathrm{Cat}(x_t; \boldsymbol{p} = x_0\bar{\mathbf{Q}}_t)$$

$$\bar{\mathbf{Q}}_t = \prod_{i=1}^{t} \mathbf{Q}_i$$

$$[\boldsymbol{Q}_t]_{ij} = \begin{cases} 1 & \text{if} \quad i = j = m \\ 1 - \beta_t & \text{if} \quad i = j \neq m \\ \beta_t & \text{if} \quad j = m, i \neq m \end{cases}$$

Q_t может быть разной, например, позволять переходы в токен MASK с заданной вероятностью или исходить из семантической близости слов

# Дискретная диффузионная модель

Формула для апостериорного распределения для слова:

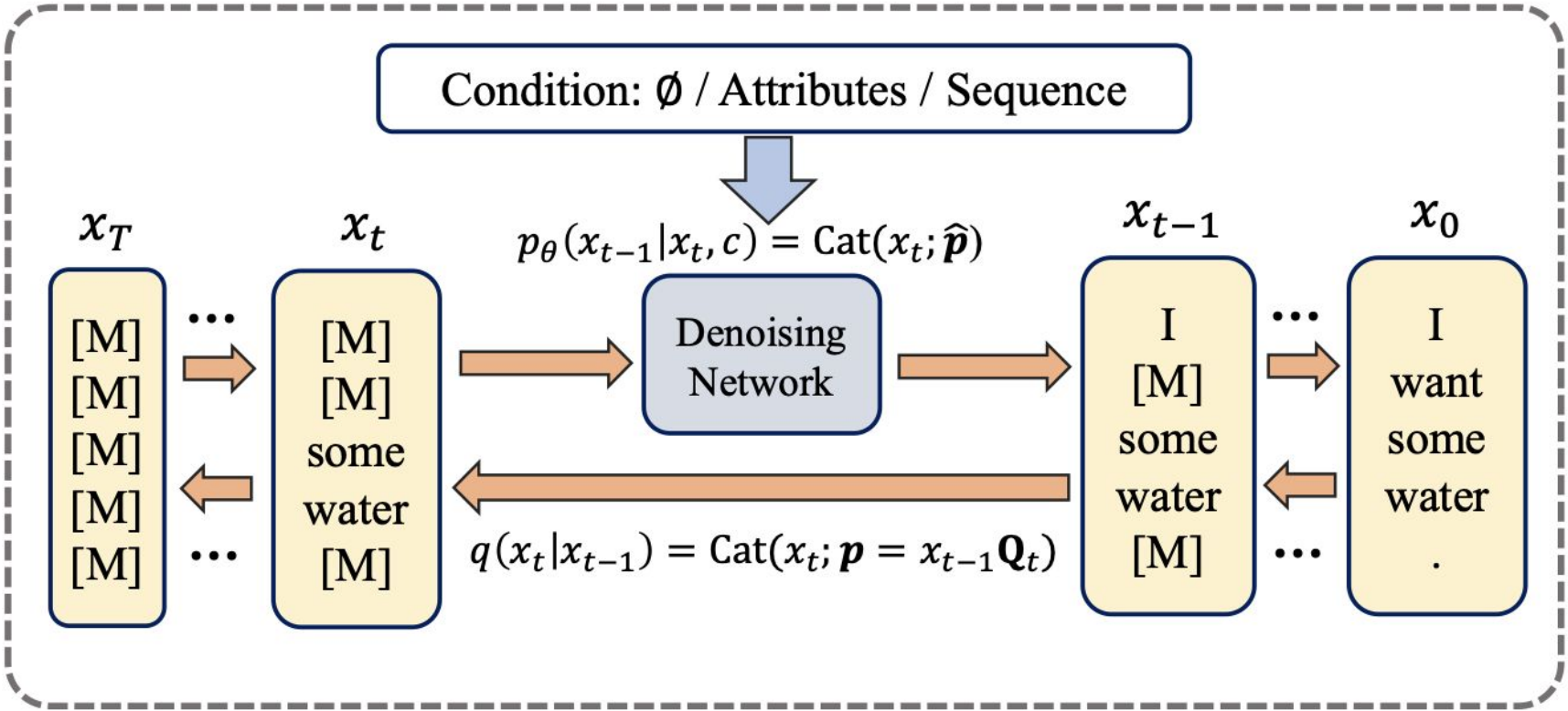
$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} = \text{Cat}\left(\boldsymbol{x}_{t-1}; \boldsymbol{p} = \frac{\boldsymbol{x}_t \boldsymbol{Q}_t^\top \odot \boldsymbol{x}_0 \overline{\boldsymbol{Q}}_{t-1}}{\boldsymbol{x}_0 \overline{\boldsymbol{Q}}_t \boldsymbol{x}_t^\top}\right)$$

По сути общее распределение q на всём объекте-тексте факторизуется на множество независимых распределений для отдельных элементов-слов

Это позволяет рассчитать KL-дивергенции для VLB-лосса суммированием по всем компонентам объекта

Нейросеть может предсказывать логиты для $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ напрямую или с использованием дополнительной информации, например, для учёта шаблона разреженности Q_t

# Дискретная диффузионная модель



# Непрерывные модели

- Дискретные компоненты w_i объекта **w** можно перевести в векторы **x** во время forward-процесса

$$q_\phi(\mathbf{x}_0|\mathbf{w}) = \mathcal{N}(\text{Emb}(\mathbf{w}), \sigma_0 I)$$

*обучаемые эмбеддинги лучше фиксированных предобученных*

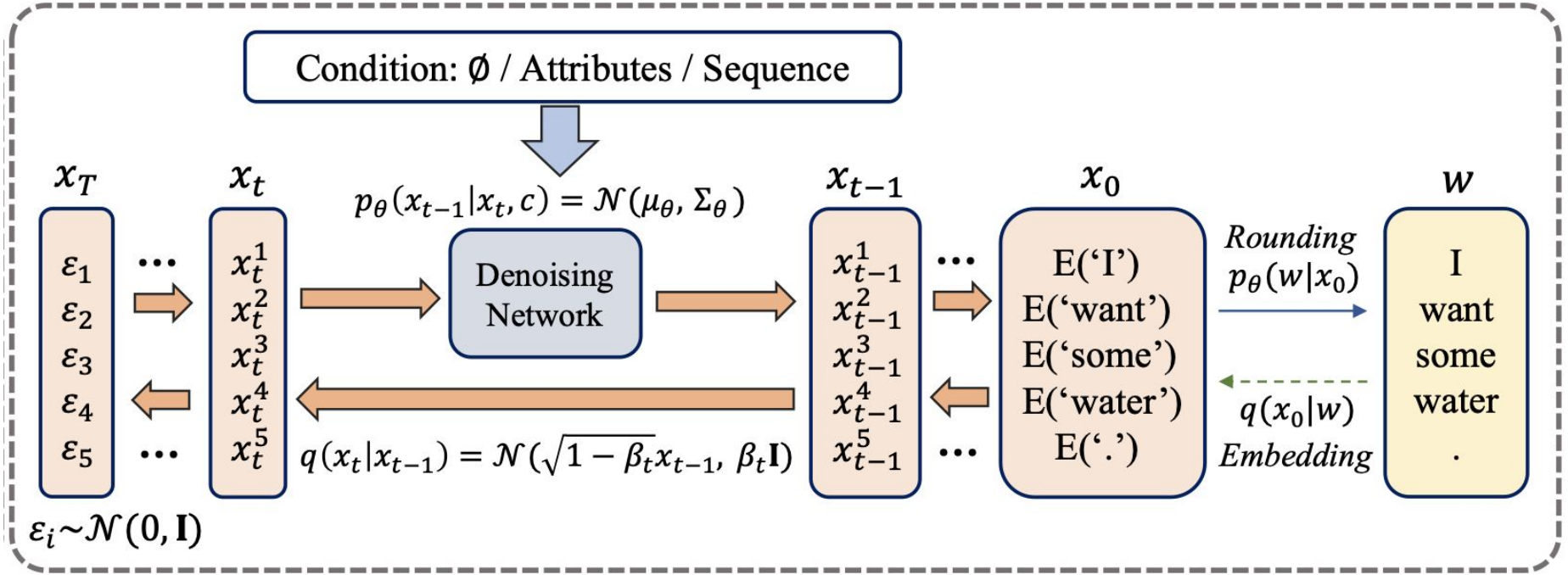
- В конце reverse-процесса добавляется обучаемый шаг, делающий обратное преобразование:

$$p_\theta(\mathbf{w} \mid \mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i \mid x_i)$$

$p_\theta(w_i \mid x_i)$ - распределение, получаемое после softmax

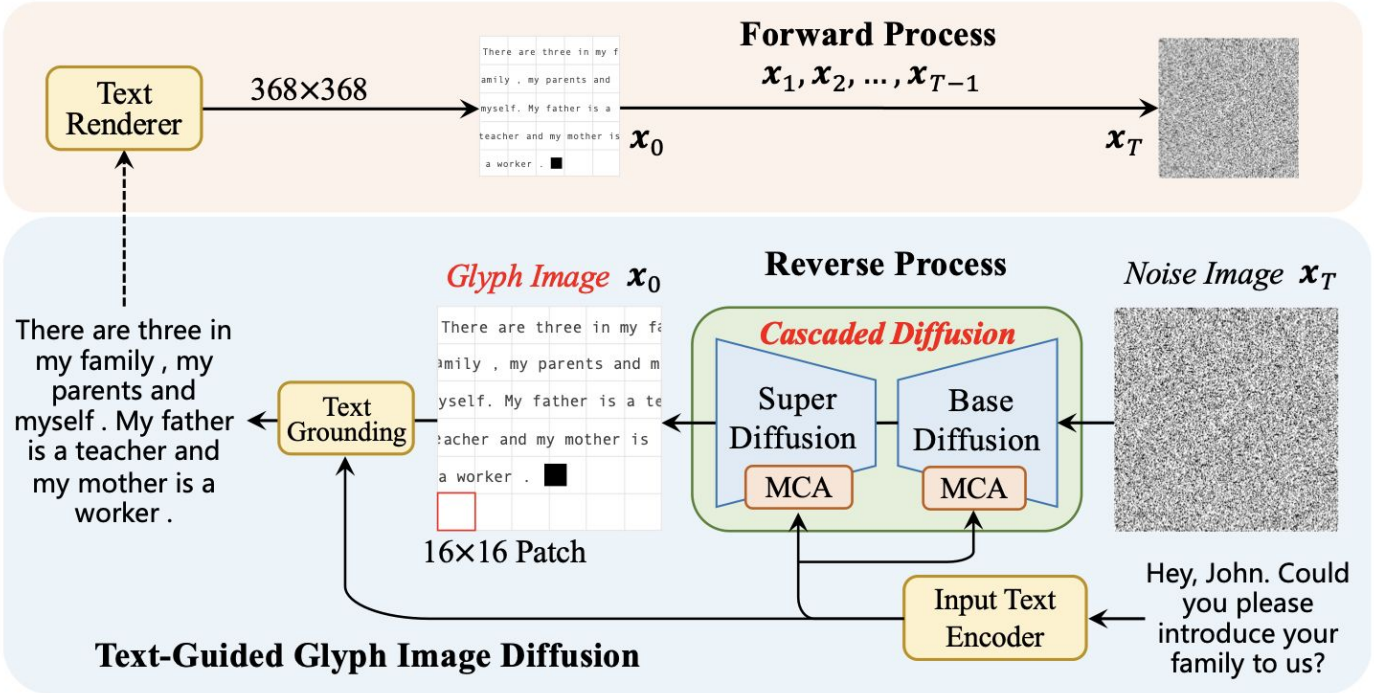
- Обучение диффузии, эмбеддингов и обратного шага обычно совместное

# Непрерывные модели



# Непрерывные модели

Интересный подход - GlyphDiffusion: **шум** -> **изображение текста** -> **текст**



# Стратегии управления шумом

- **Linear**:
  - линейный рост с итерациями \beta_t
  - упрощение на старте, усложнение ближе к T
- **Cosine**:
  - меняется сразу $\bar{\alpha}_t = \frac{f(t)}{f(0)}$, где $f(t) = \cos(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2})^2$
  - замедление роста шума к последним итерациям
- **Mutual Information**:
  - для дискретных DM, поскольку cosine напрямую не применяется
  - \beta_t вычисляется как линейная интерполяция взаимной информации между x_0 и x_t
  - для основного случая с MASK это сводится к $\beta_t = (T - t + 1)^{-1}$

# Стратегии управления шумом

- **Sqrt**:
  - в моделях с эмбеддингами слабый шум в начале будет приводить слишком простой задаче восстановления, т.к. в небольшой окрестности вектора слова других векторов нет
  - нужно усилить стартовый шум, не увеличивая сильно шум в конце
  - $\bar{\alpha}_t = 1 - \sqrt{t/T + s}$ где s - небольшая константа стартового шума
- **Spindle**:
  - первыми должны генерироваться более частые слова для контекста
  - редкие слова будут заменены на MASK в начале forward и восстановлены в самом конце reverse
- **Adaptive**:
  - Sqrt шедулер + обновление уровня шума в зависимости от текущего значения лосса

# Функционалы качества

- Для текстовых моделей предсказание среднего апостериорного распределения $q(x_{t-1}|x_0, x_t)$ может приводит к несходимости
- Часто в MSE предсказывают напрямую эмбеддинги текста (**x_0-param**)
- Лосс для моделей с эмбеддингами:
  - для обучения получения слов из векторов вводится лосс $\mathcal{L}_{\text{round}} = -\log p_\theta(w|x_0)$
  - без этого совместное обучение параметров и эмбеддингов приводит к тому, что все эмбеддинги получаются очень близкими
  - но и этого лосса недостаточно, поскольку x_0 получен из оригинального эмбеддинга с небольшим шумом
  - предлагается дополнительный лосс $\mathcal{L}_{\text{anchor}} = -\log p_\theta(w|\hat{x}_0)$ оперирующий не правильным эмбеддингом, а тем, который был сгенерирован моделью

# Подходы к обуславливанию генерации

- **Unconditional**: по сути бейзлайн для оценки способности к генерации
- **Attribute-to-text**:
  - **classifier-guidance**: обученный классификатор сдвигает градиент в свою сторону

$$\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) + \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{c} \mid \mathbf{x}_{t-1})$$

  - **classifier-free**:
    - часть сэмплов обучается с вектором-условием, часть - без (с нулевым вектором)
    - в качестве условий могут использоваться всё, что можно векторизовать
    - на инференсе к предсказанию без условия добавляется с весом s разность между предсказанием с условием и без

$$\tilde{\boldsymbol{x}}_{0,s}^{t} = \hat{\boldsymbol{x}}_0\big(\boldsymbol{x}_t, 0, 0, t, \theta\big) + s \cdot \Big(\hat{\boldsymbol{x}}_0\big(\boldsymbol{x}_t, \boldsymbol{c}, \tilde{\boldsymbol{x}}_0^{t+1}, t, \theta\big) - \hat{\boldsymbol{x}}_0\big(\boldsymbol{x}_t, 0, 0, t, \theta\big)\Big)$$

# Подходы к обуславливанию генерации

- **Text-to-text**:
  - Если условием является последовательность, то использовать напрямую подходы с классификатором нельзя
  - Можно конкатенировать текст-условие и текст-результат в одну последовательность:
    - текст-условие не подвергается изменениям на этапе forward-процесса
    - на этапе reverse текст-условие соединяется с шумом и тоже не изменятся
  - Другой вариант - закодировать текст с помощью модели-кодировщика и связать эти эмбеддинги с эмбеддингами текста-результата в декодировщике через cross-attention

# Дополнительные улучшения

- **Clamping Trick**:
  - выходы модели на инференсе приближают к ближайшему реальному эмбеддингу
  - это заставляет модель приближает модель к настоящим словам
  - считать близости может быть затратно, иногда применяют только на последнем шаге
- **Self-conditioning**:
  - в латентных моделях генерация на шаге t обусловлена на текущую латентную переменную и t
  - можно добавлять к этому ещё и выход модели (получаемый из латентной переменной) с предыдущего шага
  - обычно это делается конкатенацией и позволяет улучшить качество
  - на инференсе такой выход есть всегда, на обучении он аппроксимируется на основе шага и модели шума и добавляется в половине случаев

# Дополнительные улучшения

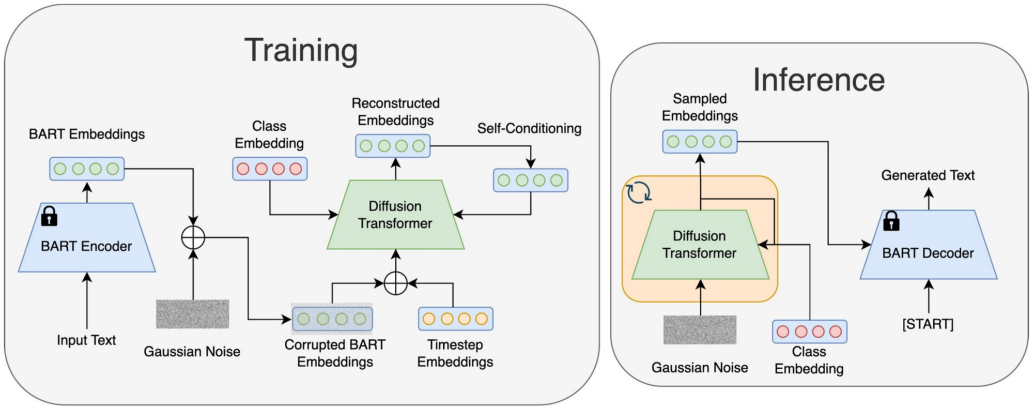
- **Semi-NAR Generation**:
  - средний вариант между AR и NAR - генерация блоками токенов
  - блок генерируется в контексте уже сгенерированного, после чего сам идёт в контекст
- **Additional Normalization**:
  - замечено, что норма векторов редких слов оказывается ощутимо больше, чем у частых, из-за этого один и тот же уровень шума действует по-разному
  - для борьбы с этим добавляется LayerNorm поверх слоя эмбеддингов
- **Timestep Sampling**:
  - обычно при обучении шаг t сэмплируется равномерно
  - вместо этого можно сэмплировать так, чтобы чаще использовались шаги, дающие наибольший вклад в лосс

# Пример архитектуры: SED

- Модель с непрерывными эмбеддингами и управлением classifier-free
- В основе кодировщик Transformer
- Относительное позиционное кодирование из Transformer-XL
- Зашумлённые эмбеддинги сперва проектируются в размерность входа
- Кодирование шага диффузии:
  - на шаге t диффузии вычисляется синусоидальный эмбеддинг шага размерности входа
  - он дополнительно пропускается через полносвязный слой той же размерности
  - далее этот вектор добавляется ко всем входным
- На выходе сети тоже ставится слой линейный слой для проектирования результатов в размерность эмбеддингов слов
- Дополнительные условия могут добавляться конкатенацией с входными векторами сети

# Использование предобученных LM (PLM)

- Задача демаскирования токенов в DM очень схожа с задачами PLM
- Но использовать напрямую PLM проблематично, т.к. они не обусловлены на шаг t
- Пробуют разные варианты:
  - Инициализация эмбеддингов выходами предобученных кодировщиков
  - Использование PLM как генератора латентного пространства для DM



# Диффузионные модели для текстов

| Model | NAR | Diffusion space | Noise schedule | Tasks | $x_0$-param | PLMs | Clamping |
|---|---|---|---|---|---|---|---|
| D3PM [2021] | ✓ | Discrete | Mutual information | UCG | ✓ | ✗ | ✗ |
| Diffusion-LM [2022] | ✓ | Continuous | Sqrt | A2T | ✓ | ✗ | ✓ |
| Diffuseq [2022] | ✓ | Continuous | Sqrt | T2T | ✓ | ✗ | ✓ |
| SED [2022] | ✓ | Continuous | Cosine | UCG, A2T | ✓ | ✗ | ✗ |
| SSD-LM [2022] | ✓ | Continuous | Cosine | UCG, A2T | ✓ | ✓ | ✗ |
| DiffusionBERT [2022] | ✓ | Discrete | Spindle | UCG | ✓ | ✓ | ✗ |
| CDCD [2022] | ✓ | Continuous | - | T2T | ✓ | ✗ | ✗ |
| Difformer [2022] | ✓ | Continuous | Linear | T2T | ✓ | ✓ | ✗ |
| LD4LG [2022] | ✗ | Continuous | Linear | UCG, A2T | ✓ | ✓ | ✗ |
| SeqDiffuSeq [2022] | ✓ | Continuous | Adaptive | T2T | ✓ | ✗ | ✗ |
| Diff-Glat [2022] | ✓ | Discrete | - | T2T | ✓ | ✗ | ✗ |
| GENIE [2022] | ✓ | Continuous | - | T2T | ✗ | ✗ | ✓ |
| DINOISER [2023] | ✓ | Continuous | Linear | T2T | ✓ | ✗ | ✗ |
| GlyphDiffusion [2023] | ✓ | Continuous | - | T2T | ✗ | ✗ | ✗ |
| Diffusion-NAT [2023] | ✓ | Discrete | Linear | T2T | ✓ | ✓ | ✗ |

**UCG** - безусловная генерация, **A2T** - генерация по атрибутам, **T2T** - текст в текст

Еще больше моделей тут: https://github.com/heejkoo/Awesome-Diffusion-Models#natural-language

# Примеры генерации

| | |
|---|---|
| input (Semantic Content)<br>output text | food : Japanese<br>Browns Cambridge is good for Japanese food and also children friendly near The Sorrento . |
| input (Parts-of-speech)<br>output text | PROPN AUX DET ADJ NOUN NOUN VERB ADP DET NOUN ADP DET NOUN PUNCT<br>Zizzi is a local coffee shop located on the outskirts of the city . |
| input (Syntax Tree)<br>output text | (TOP (S (NP (*) (*) (*)) (VP (*) (NP (NP (*) (*)))))<br>The Twenty Two has great food |
| input (Syntax Spans)<br>output text | (7, 10, VP)<br>Wildwood pub serves multicultural dishes and is ranked 3 stars |
| input (Length)<br>output text | 14<br>Browns Cambridge offers Japanese food located near The Sorrento in the city centre . |
| input (left context)<br>input (right context)<br>output text | My dog loved tennis balls.<br>My dog had stolen every one and put it under there.<br>One day, I found all of my lost tennis balls underneath the bed. |

Diffusion-LM Improves Controllable Text Generation

# Примеры генерации


AR-DIFFUSION: Auto-Regressive Diffusion Model for Text Generation

**Ссылки**:

- Diffusion Models for Non-autoregressive Text Generation: A Survey
- Structured Denoising Diffusion Models in Discrete State-Spaces
- Diffusion-LM Improves Controllable Text Generation
- Self-Conditioned Embedding Diffusion For Text Generation
- GlyphDiffusion: Text Generation Is Also Image Generation
- Difformer: Empowering Diffusion Models on the Embedding Space for Text Generation
- Diffusion-NAT: Self-Prompting Discrete Diffusion for Non-Autoregressive Text Generation
- Latent Diffusion for Language Generation

Спасибо за внимание!