Generative Models for NLP

Denoising Diffusion and Language Models

Joseph Le Roux

January 9, 2026

Notes

Outline

Notes

DATE: 30/12/25

Notes

Recap

So far we have seeen:

1. How word (token) vectors are the basis of text representation;
2. How Language Models can generate texts fluently
3. How Transformers have become the ubiquitous neural architectures for LMs
4. How to *align* the generated texts with instructions

Today

All these models assume that we generate sentences one token at a time uni-directionally

What if we could generate all tokens simultaneously?

Several models have appeared recently, all based on diffusion processes (1) (3)

Notes

Diffusion Models

A mecha robot playing the guitar in a forest, low quality, 3d, photorealistic

Diffusion Models are known to be good at generating *realistic* images.
Can we use them to output *realistic* responses?

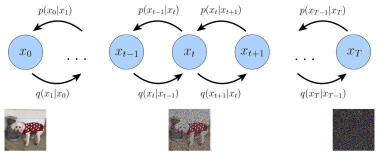
- Several propositions recently. We focus on the first one in this talk (2).

Notes

Diffusion Models (2)

Two reciprocal processes: forward and backward

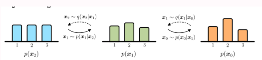
- diffusion distribution q (*forward*) generates noise from data
- generation outputs data by denoising via distribution p (*backward*) from noise to real data.



- q is fixed, we want to learn p

For discrete distributions

This paper discusses how to model diffusion for *multinomial* distributions (MD)



(b) Multinomial Diffusion: Each step $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ denoises the signal starting from a uniform categorical base distribution which gives the model $p(\boldsymbol{x}_0)$.

Notes

Definition

We want to generate data from a target multinomial distribution of K classes (*ie* we have a vocabulary of size K)

Data

We denote:

- $\mathbf{x}_0$, a piece of data generated by the target MD;
- $\mathbf{x}_t$, a piece of data generated by a noisy version of target MD after t forward steps.

$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T$ are all one-hot vectors of length K.

- we will note $\boldsymbol{\delta}_k$ the one-hot vector with 1 at position k.

Define a diffusion model

We need 2 conditional probabilities:

- forward $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ adding more noise
- backward $p(\mathbf{x}_t|\mathbf{x}_{t+1})$ substracting noise

p, q must be synchronized for every timestep t.

Notes

Forward diffusion process

$q(\mathbf{x}_t|\mathbf{x}_{t-1})$

- many possibilities, must be easy to sample from;
- For instance, in (2):
  1. flip a (biased) coin;
  2. if `head` then do not change $\mathbf{x}_{t-1}$, else (`tail`), choose a category at random (uniformly)

This amounts to:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1} = \boldsymbol{\delta}_k) = \begin{cases} (1-\beta_t) + \frac{\beta_t}{K} & \text{if } \mathbf{x}_t = \boldsymbol{\delta}_k \\ \frac{\beta_t}{K} & \text{otherwise.} \end{cases}$$

where β_t is a hyper-parameter corresponding to the `head` / `tail` ratio

But we will need more

1. $q(\mathbf{x}_t|\mathbf{x}_0)$ but be easily computable (apply t forward steps in a row)
2. the *posterior* $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ must also be easy to compute

We will see why in a moment

Notes

Combining steps of forward process (1)

Combine 2 steps

$t-1 \rightarrow t \rightarrow t+1$

$$
\begin{aligned}
q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_{t-1}=\boldsymbol{\delta}_k) &= \sum_{\boldsymbol{x}_t} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}=\boldsymbol{\delta}_k)q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t,\boldsymbol{x}_{t-1}=\boldsymbol{\delta}_k) \\
&= \sum_{\boldsymbol{x}_t} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}=\boldsymbol{\delta}_k)q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \\
&= q(\boldsymbol{x}_t=\boldsymbol{\delta}_k|\boldsymbol{x}_{t-1}=\boldsymbol{\delta}_k)q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t=\boldsymbol{\delta}_k) + \sum_{\boldsymbol{x}_t\neq\boldsymbol{\delta}_k} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}=\boldsymbol{\delta}_k)q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \\
&= \begin{cases} ((1-\beta_t)+\frac{\beta_t}{K})((1-\beta_{t+1})+\frac{\beta_{t+1}}{K})+(K-1)\frac{\beta_t}{K}\frac{\beta_{t+1}}{K} & \text{if } \boldsymbol{x}_{t+1}=\boldsymbol{\delta}_k \\ \frac{1-\text{above}}{K-1} & \text{otherwise} \end{cases} \\
&= \begin{cases} (1-\beta_t)(1-\beta_{t+1}) + \frac{1-(1-\beta_t)(1-\beta_{t+1})}{K} & \text{if } \boldsymbol{x}_{t+1}=\boldsymbol{\delta}_k \\ \frac{1-(1-\beta_t)(1-\beta_{t+1})}{K} & \text{otherwise.} \end{cases}
\end{aligned}
$$

Notes

Combining steps of forward process (2)

Combinining t steps from the beginning

- Apply the same process as before n times
- we have a recurrence:

$$
q(\boldsymbol{x}_t|\boldsymbol{x}_0=\boldsymbol{\delta}_k) = \begin{cases} \bar{\alpha}_t + \frac{1-\bar{\alpha}_t}{K} & \text{if} \boldsymbol{x}_t=\boldsymbol{\delta}_k \\ \frac{1-\bar{\alpha}_t}{K} & \text{otherwise.} \end{cases}
$$

with $\alpha_t = \prod_{i=0}^{t}(1-\beta_i)$ and $\bar{\alpha}_t = 1-\alpha_t$

Notes

Computing the posterior (1)

The posterior will be needed in the loss function

Derivation of posterior

$$
\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) &= \frac{q(\mathbf{x}_{t-1},\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}\\
&= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}\\
&= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}
\end{aligned}
$$

Notes

Computing the posterior (2)

The posterior will be needed in the loss function

Derivation of posterior

Let us rewrite the previous derivation with concrete data values:

$$
\begin{aligned}
q(\mathbf{x}_{t-1}=\boldsymbol{\delta}_p|\mathbf{x}_t=\boldsymbol{\delta}_c,\mathbf{x}_0=\boldsymbol{\delta}_k) &= \frac{q(\mathbf{x}_t=\boldsymbol{\delta}_c|\mathbf{x}_{t-1}=\boldsymbol{\delta}_p)q(\mathbf{x}_{t-1}=\boldsymbol{\delta}_p|\mathbf{x}_0=\boldsymbol{\delta}_k)}{q(\mathbf{x}_t=\boldsymbol{\delta}_c|\mathbf{x}_0=\boldsymbol{\delta}_k)}\\
&= \frac{q(\mathbf{x}_t=\boldsymbol{\delta}_c|\mathbf{x}_{t-1}=\boldsymbol{\delta}_p)q(\mathbf{x}_{t-1}=\boldsymbol{\delta}_p|\mathbf{x}_0=\boldsymbol{\delta}_k)}{\sum_{\boldsymbol{\delta}_{p'}} q(\mathbf{x}_t=\boldsymbol{\delta}_c,\mathbf{x}_{t-1}=\boldsymbol{\delta}_{p'}|\mathbf{x}_0=\boldsymbol{\delta}_k)}\\
&= \frac{q(\mathbf{x}_t=\boldsymbol{\delta}_c|\mathbf{x}_{t-1}=\boldsymbol{\delta}_p)q(\mathbf{x}_{t-1}=\boldsymbol{\delta}_p|\mathbf{x}_0=\boldsymbol{\delta}_k)}{\sum_{\boldsymbol{\delta}_{p'}} q(\mathbf{x}_t=\boldsymbol{\delta}_c|\mathbf{x}_{t-1}=\boldsymbol{\delta}_{p'})q(\mathbf{x}_{t-1}=\boldsymbol{\delta}_{p'}|\mathbf{x}_0=\boldsymbol{\delta}_k)}\\
&= \frac{\boldsymbol{\theta}(t,k,c,p)}{\sum_{p'=1}^{K}\boldsymbol{\theta}(t,k,c,p')}
\end{aligned}
$$

Take home message: we can precompute all posteriors and store them in tables.

Notes

Backward Process $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ as Denoising

The distribution that we want to learn and implement via a neural network

- Actually, T different distributions, too difficult: so rewrite backward process and model only part of it

Denoising with posterior from step t to step $t-1$:

1. Complete denoising: predict clean from noisy (ie perform t backward steps)
2. From predicted data use the posterior to perform $(t-1)$ forward steps.

$$
\begin{aligned}
p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) &= \sum_{\boldsymbol{x}_0} p_\theta(\boldsymbol{x}_{t-1}, \boldsymbol{x}_0|\boldsymbol{x}_t) \\
&= \sum_{\boldsymbol{x}_0} p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t) q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0, \boldsymbol{x}_t) \\
&= \mathbb{E}_{\boldsymbol{x}_0 \sim p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}\left[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0, \boldsymbol{x}_t)\right] \approx q(\boldsymbol{x}_{t-1}|\mathbb{E}_{\boldsymbol{x}_0 \sim p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)}[\boldsymbol{x}_0], \boldsymbol{x}_t) \\
&= q(\boldsymbol{x}_{t-1}|\widehat{\boldsymbol{x}_0}, \boldsymbol{x}_t) \text{ with } \widehat{\boldsymbol{x}_0} = \mu_\theta(\boldsymbol{x}_t, t) \text{ the neural network.}
\end{aligned}
$$

Remarks

1. $\widehat{\boldsymbol{x}_0}$ is ≥ 0, sums to 1, but not one-hot.
2. $p(\boldsymbol{x}_0|\boldsymbol{x}_1) = q(\boldsymbol{x}_0|\widehat{\boldsymbol{x}_0}, \boldsymbol{x}_1)$ is simply $\widehat{\boldsymbol{x}_0} = \mu(\boldsymbol{x}_1, 1)$ seen as a distribution

We now have all the tools to learn our diffusion model

Notes

Learning Problem (1)

Maximize the log-likelihood with latent diffusion

$$
\begin{aligned}
\log p(\boldsymbol{x}_0) &= \log \sum_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T} p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \log \sum_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T} \frac{q(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\boldsymbol{x}_0)}{q(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\boldsymbol{x}_0)} p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \\
&= \log \mathbb{E}_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T \sim q}\left[\frac{p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)}{q(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\boldsymbol{x}_0)}\right] \\
&\geq \mathbb{E}_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T \sim q}\left[\log \frac{p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)}{q(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T \sim q}\left[\log \frac{p(\boldsymbol{x}_T) p(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots |\boldsymbol{x}_T)}{q(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\boldsymbol{x}_0)}\right] \\
&= \mathbb{E}_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T \sim q}\left[\log \frac{p(\boldsymbol{x}_T) \prod_{t=1}^T p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{\prod_{t=1}^T q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right] \\
&= \mathbb{E}_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T \sim q}\left[\log p(\boldsymbol{x}_T) + \sum_{t=1}^T \log \frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}\right]
\end{aligned}
$$

- Maximize last line, a lower bound of the log-likelihood, as a *surrogate* loss.
- ... but because of sampling, this has high variance, we need more maths!

Notes

Learning Problem (2)

Forget constant terms

$$\mathbb{E}_q[\log p(\mathbf{x}_T) + \sum_{t=1}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] = \mathbb{E}_q[\log p(\mathbf{x}_T)] + \mathbb{E}_q[\sum_{t=1}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] = C + \mathbb{E}_q[\sum_{t=1}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}]$$

Use the special definition of $p(\mathbf{x}_0|\mathbf{x}_1)$

$$\begin{aligned}\mathbb{E}_q[\sum_{t=1}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] &= \mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] + \mathbb{E}_q[\log \frac{p(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}] \\ &= \mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] + \mathbb{E}_q[\log \frac{\mu(\mathbf{x}_1, 1)}{q(\mathbf{x}_1|\mathbf{x}_0)}] \\ &= \mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] + \mathbb{E}_q[\log \mu(\mathbf{x}_1, 1)] - C\end{aligned}$$

Notes

Learning Problem (3)

Use the posterior

$$\begin{aligned}\mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}] &= \mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}] \\ &= \mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}] + \mathbb{E}_q[\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}] = \mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}] + C\end{aligned}$$

Use Kullback-Liebler divergence

$$\begin{aligned}\mathbb{E}_q[\sum_{t=2}^{T} \log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}] &= \sum_{t=2}^{T} \mathbb{E}_q[\log \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}] \\ &= \sum_{t=2}^{T} \mathbb{E}_q[-KL\big(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p(\mathbf{x}_{t-1}|\mathbf{x}_t)\big)] \\ &= \sum_{t=2}^{T} \mathbb{E}_q[-KL\big(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||q(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{x_0})\big)]\end{aligned}$$

Notes

Learning Problem (4)

Congratulations! You (and I) survived

We have our loss function defined as:

$$\mathcal{L}(\boldsymbol{x}_0) = \mathbb{E}_q \log p(\boldsymbol{x}_0|\boldsymbol{x}_1) + \sum_{t=2}^{T} \mathbb{E}_q[-KL(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \hat{\boldsymbol{x}}_0))]$$

In practice, do not optimize for every timestep:

- sample $1 \leq t \leq T$ at random
- diffuse $\boldsymbol{x}_0$ for t timesteps (or better, sample from $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ directly)
- optimize KL for timestep t only
- move to the next example

Notes

Experiments on Language

Datasets

- `text8`: *data has First billion characters from wikipedia (clean data), can be used in word2vec, glove etc*
  - 27 categories (26 letters + space)
  - chunked in sequences of length 256
  - train/dev/test sizes: 90000000/5000000/5000000
- `enwik8`: *first 100,000,000 (100M) bytes of the English Wikipedia XML dump on Mar. 3, 2006 and is typically used to measure a model's ability to compress data*
  - 256 categories (bytes)
  - chunked in sequences of length 320
  - train/dev/test sizes: 90000000/5000000/5000000

Architecture

- 12 layer transformer (encoders only), 8 heads, layer size is 512
- 1000 diffusion steps for `text8`
- 4000 diffusion steps for `enwik8`

Notes

Results (1)

Compression metrics

Table 3: Comparison of different methods on `text8` and `enwik8`. Results are reported in negative log-likelihood with units bits per character (bpc) for `text8` and bits per raw byte (bpb) for `enwik8`.

Model type	Model	text8 (bpc)	enwik8 (bpb)
ARM	64 Layer Transformer (Al-Rfou et al., 2019)	1.13	1.06
	TransformerXL (Dai et al., 2019)	1.08	0.99
VAE	AF/AF* (AR) (Ziegler and Rush, 2019)	1.62	1.72
	IAF / SCF* (Ziegler and Rush, 2019)	1.88	2.03
	CategoricalNF (AR) (Lippe and Gavves, 2020)	1.45	-
Generative Flow	Argmax Flow, AR (ours)	1.39	1.42
	Argmax Coupling Flow (ours)	1.82	1.93
Diffusion	Multinomial Text Diffusion (ours)	1.72	1.75

⋆ Results obtained by running code from the official repository for the `text8` and `enwik8` datasets.

- worse than autoregressive models
- better than non-AR with continuous embeddings

Notes

Results (2)

Sampling

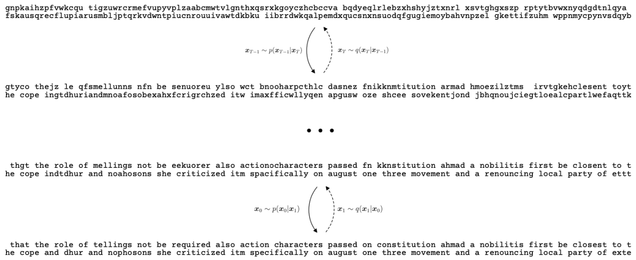


Figure 7: Intermediate steps of the generation chain of the Multinomial Text Diffusion model trained on `text8`.

Notes

Results (3)

Spell Checking

as a by-product, assume that input text is $\boldsymbol{x}_1$ and predict $\boldsymbol{x}_0$

```
mexico city the aztec stadium estadio azteca home of club america is on
e of the world s largest stadiums with capacity to seat approximately o
ne one zero zero zero zero fans mexico hosted the football world cup in
 one nine seven zero and one nine eight six
```

(a) Ground truth sequence from `text8`.

```
mexico citi the aztec stadium estadio azteca home of clup amerika is on
e of the world s largest stadioms with capakity to seat approsimately o
ne one zeto zero zero zero fans mexico hosted the footpall wolld cup in
 one nine zeven zero and one nyne eiggt six
```

(b) Corrupted sentence.

```
mexico city the aztec stadium estadio aztecs home of club america is on
e of the world s largest stadiums with capacity to seat approximately o
ne one zero zero zero zero fans mexico hosted the football world cup in
 one nine seven zero and one nine eight six
```

(c) Suggested, prediction by the model.

Figure 5: Spell checking with Multinomial Text Diffusion.

Notes

Bibliography

Austin, Jacob and Johnson, Daniel D. and Ho, Jonathan and Tarlow, Daniel and van den Berg, Rianne (2021). *Structured Denoising Diffusion Models in Discrete State-Spaces*, Curran Associates, Inc..

Hoogeboom, Emiel and Nielsen, Didrik and Jaini, Priyank and Forré, Patrick and Welling, Max (2021). *Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions*, Curran Associates, Inc..

Lou, Aaron and Meng, Chenlin and Ermon, Stefano (2024). *Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution*, PMLR.

Notes