Data And Beyond · Follow publication

Member-only story

Lets go beyond GPT - A Deep Dive into Text Diffusion Language Models

A full breakdown of the Next Generation of Language Generators. Understand Denoising, Sampling, Conditioning, and How DLMs Challenge GPT's Reign.

8 min read · Oct 13, 2025

TONI RAMCHANDANI Following

1. The Genesis of Text Diffusion

> *"Imagine teaching a language model to write… not by feeding it words, but by teaching it to unblur a sentence from noise pixel by pixel, token by token."*

In 2020, while the world was distracted by the roaring rise of GPT-3 and transformer-based LLMs, a quieter revolution was brewing in a very different modality: vision. The paper that lit the match was DDPM: *Denoising Diffusion Probabilistic Models* by Ho, Jain, and Abbeel. It introduced a mind-bending concept: start with pure noise, and train a model to **reverse the noise** into a coherent image not in one go, but in hundreds of gradual steps.

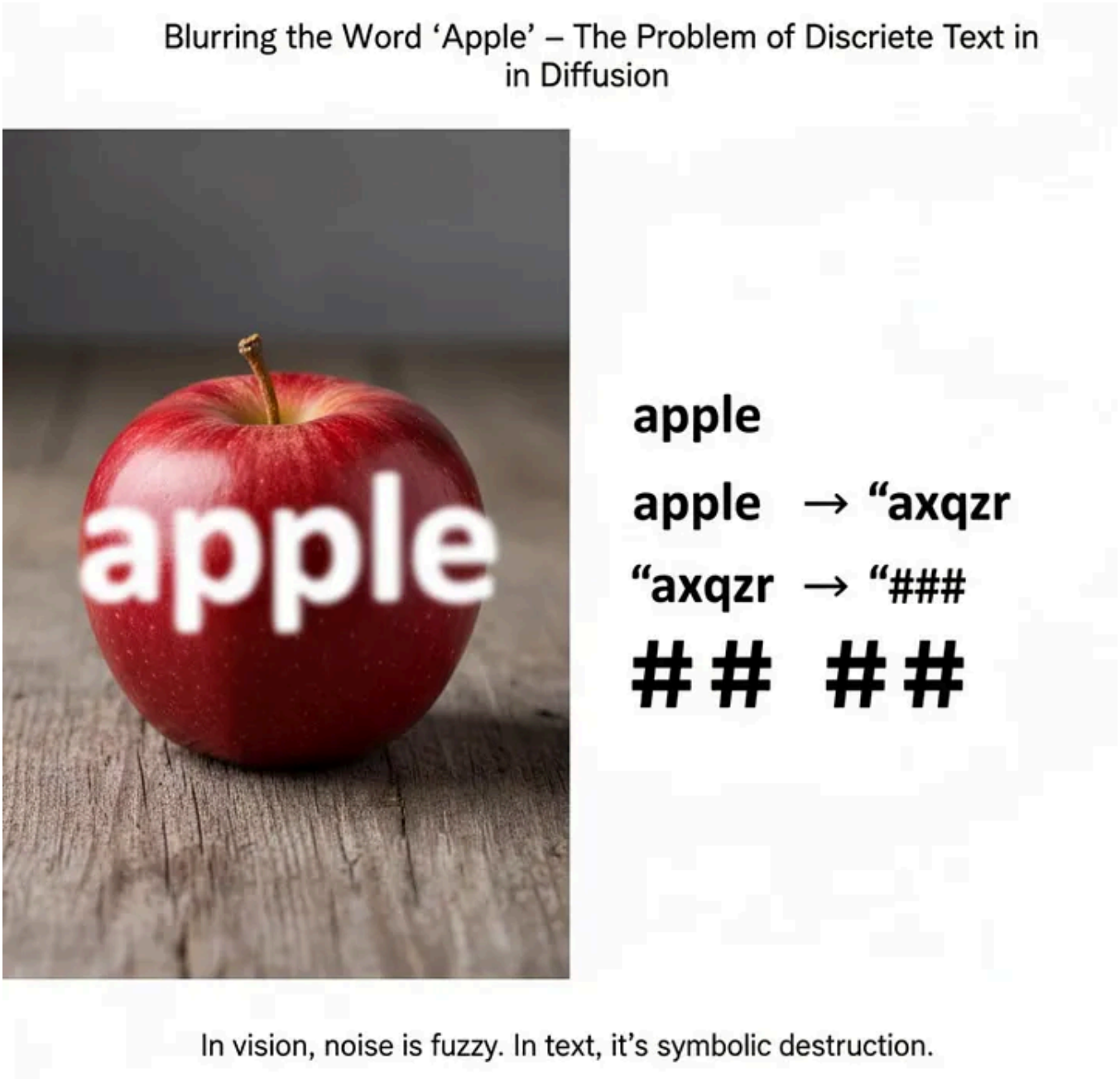
This idea exploded in computer vision. Stable Diffusion, Imagen, DALL·E 2 all rode the diffusion wave. But here's the twist: **text isn't an image.** You can't just add a bit of blur to the word "apple" and get a fuzzier version. Language is discrete. A token is either a token… or it isn't. There's no halfway between "apple" and "banana."

So how do you apply a *gradual noise-and-denoise* process to something inherently symbolic?

> 👉 *Liked the article? Smash those claps* ***(50 if you're feeling generous!)***
> ☕ *Appreciate the effort? Support my work on* ***Buy Me A Coffee link***
> 🔗 *Let's connect on* ***LinkedIn*** *— I love meeting curious minds.*
>
> *Thank you for reading — your support helps fuel the research, writing, and experiments that make articles like this possible.*

This was the paradox at the heart of *text diffusion*. And solving it would eventually lead to **a new paradigm in generative language modeling** one that doesn't predict the next word left-to-right like GPT, but rather refines entire sequences **in parallel**, offering better control, speed, and bidirectional understanding.

But the road here wasn't linear. It was iterative. Just like the diffusion process itself.



. . .

2. What is diffusion?

> *“What if instead of predicting one word at a time, your model could sculpt sentences from noise like carving meaning out of static?”*

Before we dive into text, let’s first understand how **diffusion models work at their core.**

They are not new. They’re rooted in the idea of **thermodynamic reversibility** a concept borrowed from physics. But what makes them magical in AI is this:

Diffusion models don’t generate data directly.

They **start with noise** and **learn to remove it, step by step** until something structured, meaningful, and real appears.

This is the **opposite of GANs** (which try to fool a discriminator) and also quite different from **autoregressive models** (which predict one token at a time, left to right).

Diffusion instead says:
"Let's destroy the data gradually. Then train a model to reverse that destruction."

Step 1: The Forward Process

Imagine you have a clean sentence:

```
"The owl hunts silently in the night."
```

The **forward process** will corrupt this sentence over time. In images, this means adding Gaussian noise to the pixels. In text, it's more complicated we'll get to that but let's stick with the visual intuition first.

At each timestep t, a little more noise is added. Formally, the forward process is a **Markov chain** defined as:

Each timestep slightly blurs the image or the latent text representation. Do this for, say, 1000 steps, and you end up with **pure noise**.

Think of it like this:

```
Step 0: 🦉 The owl hunts silently in the night.
Step 200: 🧠 Some structure remains, but words are fuzzy.
```

```
Step 800:  Nearly gibberish.
Step 1000:  Pure static.
```

This process is **hand-crafted and fixed**. No learning here. We're just building a clean-to-noise mapping.

Step 2: The Reverse Process

Here's the learning magic.

We now train a neural network $\epsilon\theta$ to **reverse** the noise.

This network gets:

- A noisy input *Xt*,
- The timestep *t*,
- (Sometimes) conditioning information like a prompt or label,

And it learns to **predict the noise that was added.** Once it knows the noise, we can subtract it and get closer to the original image or text.

This reverse process is:

This **step-by-step denoising** gradually reconstructs coherent data whether that's an image of a cat or a sentence about an owl.

Why this works

Because the forward process is known, the model doesn't need to guess how the data was corrupted. It just needs to **learn how to denoise it** at each step.

This means:

- **Stable training** (no adversarial game like GANs).
- **Full likelihood estimation** (you can compute how probable your data is).

- **Flexibility** (you can guide generation at any step).

And it turns out, with enough steps and training, **you can generate incredibly realistic outputs** by just starting from random noise and reversing it.

Sidebar

There's a deeper theory here and it's beautiful.

Diffusion models are a form of **score-based generative modeling**. The "score" is the gradient of the log probability of the data:

The model learns this gradient at different noise levels. By following this "score field," we can **sample from complex distributions** — even ones as weird as human language.

This connection grounds diffusion in **probabilistic foundations**, unlike GANs which are more heuristic.

Text? But text is discrete!

Here's the catch: all of this works wonderfully for **continuous data** like images.

But **text is a different beast**:

- Words are **discrete tokens** you can't "add Gaussian noise" to "hello."
- There's no smooth interpolation between "owl" and "tiger."

So how do we handle this?

There are two schools of thought:

1. Continuous Diffusion for Text

✅ Map tokens to continuous space (embeddings), diffuse there.

2. Discrete Diffusion for Text

✅ Define noise as replacing, masking, or corrupting tokens with probabilities.

> **Diffusion** isn't just a model.
> It's a *framework for generation* based on the principle of **gradual transformation** from

> randomness to structure.

. . .

3. Continuous vs discrete text diffusion: The fork in the road

> *"When you can't add noise to words, you have to choose: either embed them into a space where noise makes sense… or redefine what noise means entirely."*

In image diffusion, the process is intuitive: add Gaussian noise to pixel values smooth, continuous values. It's mathematically elegant and visually grounded. But for text?

We hit a wall.

The problem: text is discrete

Let's say you have this sentence:

> `"The hawk dives swiftly toward its prey."`

Now imagine trying to add Gaussian noise to the word `"hawk"`.

You can't.

There is no in-between version of "hawk" no numeric perturbation that results in a blurry hawk. You either have `"hawk"`, or you have `"tree"`, `"lamp"`, or a mask token like `[MASK]`. There's no continuous axis between them.

Text is not a signal. It's a sequence of **discrete symbols** from a **finite vocabulary**.

So, how do you apply diffusion?

This leads to two powerful and radically different schools of thought in the world of text diffusion:

A. Continuous diffusion: operating in embedding space

> *"If you can't add noise to tokens, map them to vectors first."*

In this approach, pioneered by models like **Diffusion-LM (Li et al., 2022)** and **DiffuSeq**, the idea is simple yet clever:

1. **Map words into continuous space** (using word embeddings or learned vector representations).
2. **Apply Gaussian noise** to those embeddings.
3. **Denoise** those vectors back to original embeddings.
4. **Decode** the final vectors back to tokens using nearest neighbors or a softmax decoder.

It looks like this:

```
"hawk" → [1.3, -2.1, 0.4, ...] → [1.25, -2.0, 0.38, ...] ← noisy vector
```

Over time:

- You corrupt these embeddings with Gaussian noise.
- The model learns to **predict the original clean embedding** or the **noise itself**.
- At inference, you **start from pure Gaussian noise vectors** and iteratively denoise them into valid text embeddings.

Eventually, you **decode the embeddings back into tokens**, using either:

- Nearest neighbor lookup in the embedding space.
- A projection head + softmax over vocabulary.

This brings the entire **image diffusion pipeline** into text without touching the symbolic level at all.

Pros of continuous diffusion:

- Leverages **mature diffusion math** (from image/audio domains).
- Easily integrates **score-based techniques**, ODE/SDE solvers, and classifier-free guidance.
- Works well with **pretrained word embeddings** or transformer layers.

- Enables **gradient-based control** (you can steer embeddings with gradients to enforce sentiment, style, etc.)

Challenges of continuous diffusion:

- Embedding spaces are **fuzzy**: multiple tokens can map to nearby points, leading to decoding ambiguities.
- Final decoding step is **non-trivial** nearest neighbors can be semantically close but syntactically invalid.
- High-quality embedding prediction doesn't always translate to **valid or fluent sentences**.
- Struggles with **rare words** or **subword tokenization** (especially in open-vocab models like T5 or GPT-2).

Example Model: Diffusion-LM

- Operates on continuous word embeddings.
- Predicts the added noise at each step (score-based training).
- Can condition generation on sentiment, length, syntax using gradient guidance.
- Produces diverse and coherent sentences after denoising vectors for ~200 steps.

> "The embedding of 'hawk' is diffused into noise and then recovered like pulling back meaning from a latent soup."

B. Discrete diffusion: noising in token space

> *"What if instead of blurring words… we just replace them?"*

This approach, pioneered by **D3PM (Austin et al., 2021)** and later refined by **DiffusionBERT**, **Reparam DLM**, and **LLaDA**, redefines the noising process **entirely within the discrete token space**.

Instead of adding real-valued noise, we apply **probabilistic corruption** to each token in a sequence.

Forward process:

- At each timestep ttt, each token has a probability $\beta t$of being:
- Replaced with a random token,

- Replaced with a special token (like `[MASK]`),
- Replaced based on a noise matrix QQQ, which defines corruption rules.

This turns the sentence:

```
"The hawk dives swiftly" → "The [MASK] dives [MASK]" → "lamp [MASK] pencil swif
```

As t→T, the sentence approaches **pure noise** typically all `[MASK]` or randomly shuffled tokens.

Reverse process:

- The model is trained to predict the **original token** given the corrupted version and the timestep.
- It outputs a **distribution over vocabulary** for each position (cross-entropy loss).
- At inference, we **start from masked/random sequence** and **iteratively denoise** it by sampling from the model's output at each position.

Pros of discrete diffusion:

- Operates directly on tokens no ambiguity in decoding.
- Naturally supports **mask-based infilling and editing** (great for applications like sentence rewriting or code patching).
- Easily bootstrapped from **pretrained masked LMs** (e.g. BERT, RoBERTa).
- Can generate sequences **in parallel**, no left-to-right constraint.

Challenges of discrete diffusion:

- Harder to align with **score-based diffusion math** discrete transitions are not differentiable.
- Sampling can be **less stable**: jumping from one token to another can break fluency.
- Transition matrices QQQ must be **carefully designed** to balance learning and diversity.

- Requires **more steps** to reach fluency than AR models unless guided.

Example Model: D3PM (Discrete Denoising Diffusion Probabilistic Models)

- Defines transition matrices for different corruption types:
- Uniform replacement,
- Nearest neighbor in embedding space,
- Absorbing state (mask token).
- Trains with a variational objective + auxiliary cross-entropy.
- Demonstrated that **discrete diffusion can match AR perplexity** on language modeling benchmarks.

> “Instead of softening tokens, it replaces them with structured chaos then learns to rebuild language from noise.”

Summary: two paradigms, one goal

Both approaches strive to **reverse corruption** but take entirely different roads:

- **Continuous** tries to **blur and sharpen**.
- **Discrete** tries to **mask and fill.**

> *"Diffusion isn’t just a new modeling trick it demands a rethinking of the language model’s skeleton: how it sees, learns, and generates."*

Llm | Diffusion Models | NLP | AI | Gpt

beyond-b83ff0f3813e

Following

Written by TONI RAMCHANDANI

991 followers · 371 following

https://www.linkedin.com/in/toni-ramchandani/

No responses yet

Bgerby

What are your thoughts?

More from TONI RAMCHANDANI and Data And Beyond

In Data And Beyond by TONI RAMCHANDANI

IBM’s Granite Docling 258M & Its DocTag Revolution: The Model That Doesn’t Flatten Your Data

A storytelling journey into how IBM turned vision, language, and structure into a layout-preserving AI built for the RAG era

Sep 24 168 2

In Data And Beyond by Shahzaib

You NEED to Use n8n RIGHT NOW!! (Free, Local, Private)

The Ultimate Guide to Unleashing Your Inner Automation Genius Without Spending a Dime

Sep 7 245 2

In Data And Beyond by Pavan Belagatti

Vector Databases: A Beginner's Guide!

In the age of burgeoning data complexity and high-dimensional information, traditional databases often fall short when it comes to...

Aug 25, 2023 1.6K 12

In Data And Beyond by TONI RAMCHANDANI

Part 1: Introduction to n8n — What It Is and How It Works

Hey everyone, welcome to this series! Today, we're kicking off our journey into the world of automation with n8n — the one-stop workflow...

Mar 26 353 6

See all from TONI RAMCHANDANI

See all from Data And Beyond

Recommended from Medium

In Towards AI by Teja Kusireddy

We Spent $47,000 Running AI Agents in Production. Here's What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic's Model Context Protocol (MCP) are revolutionary. But...

Oct 16 1.6K 37

In Data Science Collective by Ida Silfverskiöld

Agentic AI: Single vs Multi-Agent Systems

Building with a structured data source in LangGraph

3d ago 378 9

In Level Up Coding by Gao Dalie (*高達烈*)

PaddleOCR VL + RAG: Revolutionize Complex Data Extraction (Open-Source)

Not even a month ago, I made a video about MistralOCR that many of you liked.

5d ago 52

In Artificial Intelligence in Plain English by DrSwarnenduAI

A Geometric Interpretation of Attention: The Linear Algebra of Finding Your Match

Good day and Welcome back 😎 !

Oct 24 163 3

Alain Airom (Ayrom)

Using Chonkie

Testing Chonkie 🧪

Oct 24 56

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You'll Install This Year

I didn't expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 297 12

See more recommendations