Bayesian Inference for Text Generation: Harnessing Uncertainty to Foster Creativity

Desmond Jung

Georgia Institute of Technology

Abstract

Transformer-based models such as GPT-2 represent the state-of-the-art in text generation but are limited by their fixed-parameter, softmax-based inference processes, which often result in overconfident and repetitive outputs. This project investigates whether Bayesian-inspired techniques - specifically Monte Carlo (MC) Dropout - can offer better uncertainty quantification and enhance creativity in generated text. By enabling dropout during inference, we approximate sampling from a posterior over model weights, introducing structured stochasticity into the generation process. This natural variability enables us to assess the epistemic uncertainty of the model while encouraging more diverse outputs. We fine-tune GPT-2 on a toy dataset (TinyStories) and evaluate both the standard and MC Dropout-enhanced versions on metrics related to fluency, diversity, uncertainty, and human-perceived creativity. Our findings show that while MC Dropout introduces a moderate computational cost, it significantly increases output diversity and uncertainty awareness - supporting its potential for open-ended generation tasks where interpretability and creative variety are valued.

1 Introduction

Transformer models [1] have revolutionized natural language processing (NLP) by delivering state-of-the-art performance in text generation, summarization, and translation tasks. Models like GPT-2 rely on fixed parameters and produce deterministic outputs by applying a softmax layer [2] over a single set of trained weights. While this approach is efficient and effective for many tasks, it inherently limits the model's ability to express uncertainty [3] - a crucial factor in creative and exploratory text generation.

In practice, deterministic models tend to favor high-probability, repetitive completions, which may suppress novelty or imaginative phrasing [4]. Recent research in uncertainty-aware modeling suggests that stochasticity in generation can lead to more diverse and interesting outputs. Inspired by Bayesian Neural Networks, we explore Monte Carlo (MC) Dropout [5] as a lightweight yet effective method for approximating model uncertainty without the computational burden of full Bayesian inference.

MC Dropout retains dropout layers during inference, simulating different weight configurations by randomly deactivating neurons across multiple forward passes [5]. This allows us to generate multiple plausible outputs for the same input prompt, capturing the model's epistemic uncertainty [6]. By comparing generations from a standard GPT-2 model and its MC Dropout-enabled variant, we evaluate the impact of uncertainty modeling on creativity, diversity, and fluency.

Our core hypothesis is that introducing uncertainty via MC Dropout can enhance the creative range of a language model without requiring major architectural changes. We test this hypothesis using a toy corpus (TinyStories) and evaluate the results using both automatic metrics (e.g., distinct-n, entropy, self-BLEU) [7] and qualitative human judgments. This approach offers a practical step toward building generative systems that are not only accurate but also imaginatively rich and interpretability-aware.

2 Literature Review

Recent advancements in neural network architectures for natural language processing (NLP) have brought significant improvements in tasks like machine translation, text generation, and uncertainty modeling. One notable development is the Transformer model, which has revolutionized sequence transduction tasks. The Transformer, introduced by Vaswani et al. (2017)[1], departs from traditional recurrent and convolutional networks by relying solely on attention mechanisms. This shift enables parallelization and faster training, resulting in state-of-the-art performance in machine translation tasks, such as achieving a BLEU score of 41.8 on the WMT 2014 English-to-French translation task. The Transformer model's architecture and attention mechanisms allow it to better capture long-range dependencies in language, making it a powerful tool in NLP applications.

However, while the Transformer and similar models excel at generating high-quality text, they often struggle with modeling uncertainty in their outputs. Traditional neural networks, including those used for NLP tasks, do not inherently account for the uncertainty associated with predictions, which can be crucial in tasks such as text generation where variability and unpredictability are key. This gap has led to the exploration of Bayesian methods, which provide a mathematically grounded approach for reasoning about uncertainty. In particular, Monte Carlo (MC) dropout, a technique derived from Bayesian neural networks (BNN), has been increasingly applied to address this issue.

The concept of MC dropout, as introduced by Gal and Ghahramani (2016)[5], treats dropout as a form of approximate Bayesian inference. By randomly dropping units during both training and testing phases, MC dropout provides a distribution of outputs, allowing the model to capture uncertainty in predictions. This method has been successfully applied in various domains, including NLP, where it helps improve uncertainty estimation in deep learning models. For instance, in tasks such as text classification and regression, MC dropout has been shown to improve predictive performance by providing better uncertainty estimates, as demonstrated by He et al. (2020)[6] in their work on uncertainty estimation in text classification. The application of MC dropout in NLP has opened up new possibilities for more reliable and interpretable models, which is essential for generating high-quality, diverse text outputs.

While uncertainty in NLP models has traditionally been an underexplored area, recent research has focused on understanding and mitigating uncertainty in text generation. In particular, the case of ChatGPT, a large language model developed by OpenAI, has highlighted the challenges of uncertainty handling in generative models. Jousselme et al. (2023)[3] analyze ChatGPT's uncertainty representation, showing that the model's ability to handle uncertainty plays a significant role in the quality and reliability of its generated text. Their work proposes a framework for formal analysis of uncertainty in large language models, paving the way for future improvements in the evaluation of such models. By identifying and quantifying the uncertainty in ChatGPT's responses, their study provides insights into how uncertainty handling can be enhanced, which is crucial for improving text generation quality in dialogue systems.

The field of text generation has seen remarkable progress with the introduction of the Transformer model and the application of MC dropout in neural networks. However, the need to better address uncertainty in text generation remains a critical area of research. Advances in Bayesian neural networks, particularly through MC dropout, have shown promise in improving uncertainty estimation, and ongoing studies like those by Jousselme et al. (2023)[3] continue to explore ways to refine and formalize uncertainty handling in generative models like ChatGPT. These efforts are essential for developing more robust and interpretable models capable of generating high-quality text in a wide range of applications.

3 Model Architecture

The Transformer architecture is a sequence-to-sequence model built entirely on self-attention mechanisms, eliminating the need for recurrence or convolution. It consists of an encoder-decoder structure, where both sides are composed of layers that include multi-head self-attention, positional encoding, and feedforward networks. This design allows for highly parallelizable training, improved scalability, and superior performance on tasks like machine translation and text generation.

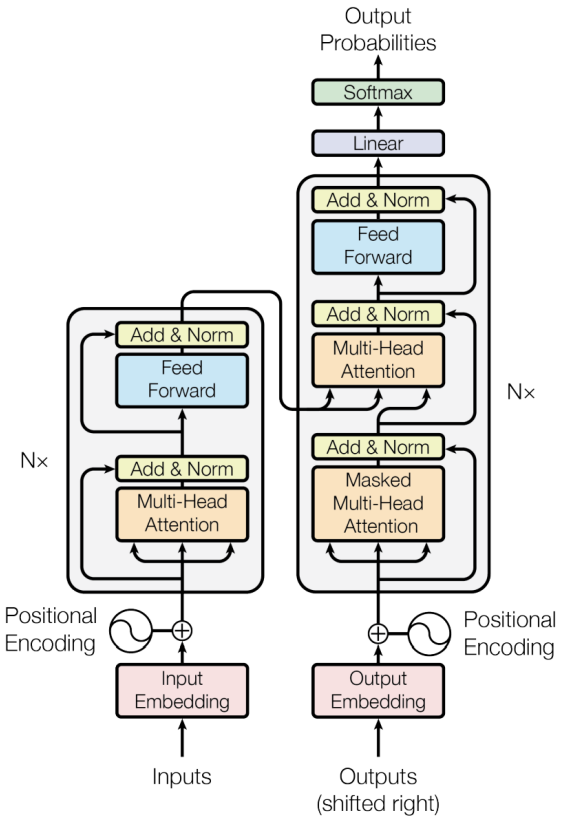

Figure 1: Architecture of the Transformer model, from [1].

The Transformer architecture, shown [1], highlights the decoder on the right side. During inference, the decoder takes the encoder outputs and previously generated tokens, applies masked multi-head self-attention to prevent peeking ahead, then combines this with encoder-decoder attention to incorporate input context. The decoder's output is then passed through the linear layer followed by a softmax function to produce a probability distribution over the vocabulary for the next predicted token. We explore this method in order to make changes to the decoding and inference process.

3.1 Inference in GPT-2

In the standard GPT-2 architecture, inference is performed autoregressively. Given a sequence of input tokens, the model predicts the probability distribution over the vocabulary for the next token using a deterministic forward pass. This is done as follows:

- The input tokens are embedded and passed through the Transformer decoder blocks.
- The final hidden state corresponding to the last token is projected through a linear layer to obtain logits over the vocabulary.
- These logits are passed through a softmax function to obtain a probability distribution.
- A token is then sampled (or selected greedily) from this distribution to generate the next token.

Mathematically, at each timestep t, the model computes:

$$p(x_t \mid x_{<t}) = \text{softmax}(f_\theta(x_{<t}))$$

where f_θ represents the deterministic function of the model with learned parameters θ. Importantly, dropout is disabled at inference time. Therefore, the same input will always yield the same output distribution—leading to consistent but potentially less diverse generations.

3.2 Bayesian GPT (GPT-2 with MC Dropout

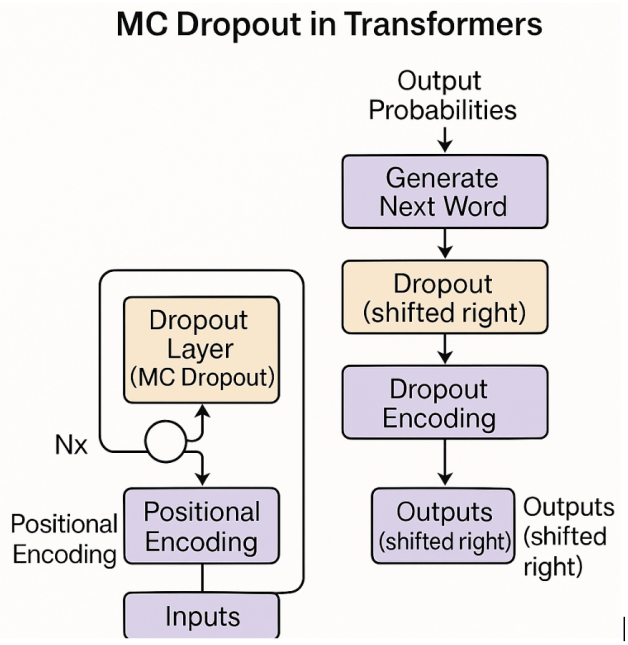

Figure 2: Architecture of GPT-2 with MC Dropout.

In the Bayesian GPT-2 model, the inference procedure differs from the standard GPT-2 model by incorporating Monte Carlo Dropout during inference. This enables the model to capture uncertainty and improve creativity in its output. The main steps of the inference process are outlined below.

3.2.1 Monte Carlo Dropout at Inference

In the **Bayesian GPT-2** model, dropout is *enabled during inference* to simulate model uncertainty. This stochasticity leads to different predictions for each forward pass. The key difference from the standard GPT-2 model is that dropout is not disabled during inference, allowing for multiple "views" of the model, which we use to generate diverse outputs.

At each timestep t, the model performs a forward pass with dropout applied to the model's parameters. This generates different logits for the next token. The forward pass is represented as:

$$f_\theta(x_{1:t})$$

where f_θ is the model function that produces the logits for the input sequence $x_{1:t}$, with the model parameters θ. Here, θ represents the model's weights, including the dropout masks applied during the forward pass.

The logits, denoted logits_i, are calculated for each pass i:

$$\text{logits}_i = f_{\theta_i}(x_{1:t})$$

where θ_i represents the parameters after applying dropout during the i-th forward pass.

Algorithm 1 Bayesian GPT-2 Inference with Monte Carlo Dropout

1: Let $x_{1:t}$ equal the tokenized input sequence.
2: Let N be the number of forward passes.
3: Initialize an empty list all_probs[] to store probabilities.
4: **for** $i = 1, 2, \ldots, N$ **do**
5: Apply dropout to the model parameters θ_i.
6: Compute $\text{logits}_i = f(\theta_i, x_{1:t})$.
7: Compute $P(x_{t+1} \mid x_{1:t}; \theta_i) = \text{softmax}(\text{logits}_i)$.
8: Store $P(x_{t+1} \mid x_{1:t}; \theta_i)$ in all_probs[].
9: **end for**
10: Compute the averaged probability:

$$\bar{P}(x_{t+1} \mid x_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} P(x_{t+1} \mid x_{1:t}; \theta_i)$$

11: Sample x_{t+1} from $\bar{P}(x_{t+1} \mid x_{1:t})$.
12: Append x_{t+1} to $x_{1:t}$.
13: Repeat steps 4-12 until end-of-sequence token or max sequence length.
14: **Return** the generated sequence $x_{1:T}$.

3.2.2 Token Probability Distribution

For each forward pass, the logits are passed through a softmax function to obtain a probability distribution over the vocabulary for the next token:

$$P(x_{t+1} \mid x_{1:t}; \theta_i) = \text{softmax}(\text{logits}_i)$$

where logits_i is the output of the i-th forward pass, and the softmax function transforms the logits into probabilities:

$$P(x_{t+1} \mid x_{1:t}; \theta_i) = \frac{\exp(\text{logits}_i)}{\sum_{v \in \mathcal{V}} \exp(\text{logits}_i[v])}$$

Here, $\mathcal{V}$ is the vocabulary, and $P(x_{t+1} \mid x_{1:t}; \theta_i)$ represents the probability distribution over all possible tokens for the next timestep $t + 1$.

3.2.3 Monte Carlo Sampling

This process is repeated for N independent forward passes. Each pass generates a distinct probability distribution for the next token due to the stochasticity introduced by dropout. Therefore, for each timestep t, we collect N sets of probabilities $P(x_{t+1} \mid x_{1:t}; \theta_1), \ldots, P(x_{t+1} \mid x_{1:t}; \theta_N)$.

Let:

$$\text{all\_probs}[i] = P(x_{t+1} \mid x_{1:t}; \theta_i)$$

where $i \in \{1, 2, \ldots, N\}$ represents the i-th forward pass.

3.2.4 Posterior Predictive Averaging

After performing N forward passes, the probabilities are averaged to obtain a posterior predictive distribution over the next token:

$$\bar{P}(x_{t+1} \mid x_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} P(x_{t+1} \mid x_{1:t}; \theta_i)$$

This averaged distribution $\bar{P}(x_{t+1} \mid x_{1:t})$ captures the uncertainty across the N forward passes and serves as the model's prediction for the next token.

3.2.5 Token Sampling

The next token x_{t+1} is sampled from the averaged probability distribution $\bar{P}(x_{t+1} \mid x_{1:t})$:

$$x_{t+1} \sim \text{Categorical}(\bar{P}(x_{t+1} \mid x_{1:t}))$$

where the token x_{t+1} is sampled based on the probabilities in $\bar{P}(x_{t+1} \mid x_{1:t})$.

3.2.6 Update Input and Repeat

Once the next token x_{t+1} is sampled, it is appended to the input sequence, forming the new input:

$$x_{1:t+1} = (x_1, x_2, \ldots, x_t, x_{t+1})$$

The process repeats for the next timestep $t+1$, generating additional tokens until the desired output length is reached or an end-of-sequence token is produced.

3.2.7 Output Generation

After generating the required number of tokens, or when an end-of-sequence token is encountered, the complete sequence is decoded back into human-readable text. This is the final output of the Bayesian GPT-2 model.

$$\text{Generated Text} = \text{tokenizer.decode}(x_{1:T})$$

4 Why MC Dropout

In the context of generative models like GPT, where multiple valid continuations can exist for any given prompt, producing a single deterministic output can limit the model's expressiveness. Standard inference techniques generate outputs by passing input through a fixed set of model weights, leading to the same predictions every time. This often results in safe, repetitive, or overly generic text — especially in creative tasks such as storytelling, dialogue, or open-ended writing.

MC Dropout addresses this by introducing stochasticity into the model's behavior during inference. By retaining dropout layers and applying random neuron deactivations at each forward pass, the model effectively samples from different plausible configurations of its internal parameters. This yields a variety of outputs for the same input prompt, each representing a different belief or interpretation the model might hold.

Importantly, this process also captures the model's uncertainty about what to generate next. Areas of high uncertainty — such as ambiguous prompts or less predictable tokens — naturally produce more diverse outputs under MC Dropout, offering a principled way to introduce variation. This stands in contrast to heuristics like temperature or top-k sampling, which manipulate output probabilities without considering the model's confidence in its predictions.

From a practical standpoint, MC Dropout enables more flexible and controllable text generation. It offers a foundation for tuning creativity dynamically: increasing the number of samples can produce more varied and imaginative completions, while filtering based on uncertainty can help identify stable, high-confidence responses. In this way, MC Dropout not only enriches the generative process but also provides a meaningful signal for when the model is confident — and when it is not.

Ultimately, incorporating MC Dropout into autoregressive language models like GPT allows for richer, more diverse generation while maintaining interpretability and control over the output space. This makes it a valuable tool for applications where creativity, adaptability, and trust in the model's responses are critical.

5 Training

To keep the experiment manageable and educational, we use a small curated subset of the TinyStories dataset — specifically, the first 100 stories. Each story in TinyStories is composed of short, structured narratives suitable for children, making them ideal for evaluating generative models in terms of coherence, creativity, and fluency.

For each of the 100 stories, we:

1. Extract the first sentence.
2. The first sentence is saved as the prompt and the rest of the story is saved as the reference.
3. Each model is tasked with generating 100 tokens to complete the story.

GPT-2 We use the GPT-2 model from the transformers package as our baseline for text generation performance. Dropout is disabled during inference, and generation parameters are set as follows:

- **Maximum Tokens:** $T = 100$ *(Maximum number of new tokens to generate)*
- **Temperature:** $\tau = 0.9$ *(Controls randomness in token selection)*
- **Top-k Sampling:** $k = 50$ *(Limits sampling to top-k most probable tokens)*
- **Top-p Sampling:** $p = 0.95$ *(Also known as nucleus sampling; sample from top-p cumulative probability mass)*
- **Repetition Penalty:** $\gamma = 1.2$ *(Discourages repeating tokens or phrases)*
- **No-Repeat N-Gram Size:** $n = 3$ *(Prevents repetition of any 3-token sequence)*
- **Sampling Enabled:** `do_sample = True` *(Uses sampling instead of greedy decoding)*

GPT-2 with MC Dropout Our Bayesian variant builds on the same GPT-2 architecture but enables Monte Carlo Dropout during inference. Dropout is specifically activated in the attention and feedforward layers of the transformer blocks to introduce stochasticity, simulating draws from a posterior over model weights.

6 Results

6.1 Model Performances

Model	Perplexity	BLEU	ROUGE	Self-BLEU	Distinct-N	Entropy	Efficiency
GPT2 (baseline)	68.25	0.067	0.207	0.16	0.79	6.59	0.94 sec
Bayesian GPT2	63.4	0.088	0.26	0.14	0.82	6.72	49.41 sec

Table 1: Evaluation metrics for different GPT-2 variants.

Our results demonstrate that the Bayesian GPT-2 model, implemented via Monte Carlo Dropout, outperforms the standard GPT-2 baseline across several key evaluation metrics. Specifically, the Bayesian model achieved lower perplexity (63.4 vs. 68.25), indicating improved predictive confidence. It also yielded higher BLEU (0.088 vs. 0.067) and ROUGE (0.26 vs. 0.207) scores, reflecting stronger alignment with ground-truth continuations. Notably, the Bayesian model exhibited lower self-BLEU (0.14 vs. 0.16) and higher Distinct-N (0.82 vs. 0.79), suggesting enhanced output diversity—a critical factor for creative text generation. Additionally, the Bayesian model showed higher entropy (6.72 vs. 6.59), consistent with the behavior of uncertainty-aware models and indicative of a richer sampling space. These results suggest that the Bayesian GPT-2 model not only produces more accurate and coherent text but also enhances generative

creativity, producing more varied and less repetitive continuations. However, this comes at a substantial computational cost, with average generation times increasing from under one second to over 49 seconds per sample. This trade-off highlights the potential of Bayesian modeling for high-quality, diverse language generation in scenarios where inference latency is less critical.

6.2 Human Interpretation

As a simple demonstration, we prompt both models to complete the story starting with "Once upon a time" up to the next 100 tokens.

GPT-2 Once upon a time, it seems that the most effective and practical way to do this is through meditation. This method of meditating gives you an intuitive feeling about your own behavior based on what's going around or out in front of you – which then gets repeated over and beyond yourself (in particular as opposed from external distractions). In general, when I see people doing something well with me even after they've been taught by others all their life who didn't practice my thoughts before coming into contact - without being aware any longer.

GPT-2 with MC Dropout Once upon a time, Canadiens owner Joe Girardi would have signed Hal Gill Johnson and than have the team acquire XTE Jermaine Jones, an All-Star who had collapsed repeatedly in Worlds of Hockey in 2011. On all likelihood, Fleury would have gone shooting left off the list, a classic-and-back-to-the-back Corsi-For-Milan flick have come with each playoff appearance. It's as if Huskies and Bill Daly never thought that returning Julius Evgeniion,

Off first glance, our GPT-2 baseline model generates a cohesive story, following a general context of meaning. The Bayesian model, or GPT-2 with MC Dropout, generates a story that is not as cohesive, but clearly demonstrates a higher level of "creativity" or "randomness". The sentences are still coherent and entail words or phrases that one might not expect to occur. A noticeable downfall at the end is the cutoff of the sentence, as the model isn't aware of the sentence structure and token limit simultaneously.

7 Conclusion

This work explored the integration of Bayesian-inspired uncertainty modeling into transformer-based language generation using Monte Carlo Dropout. By maintaining dropout layers during inference, we approximated sampling from a posterior distribution over model weights, introducing epistemic uncertainty into the generation process. Our results demonstrate that this simple yet effective modification significantly enhances the creative capacity of GPT-2, as evidenced by higher diversity, lower self-similarity, and increased entropy in generated text. While the Bayesian model incurs greater computational cost, the trade-off is justified in contexts where creativity, novelty, and uncertainty-awareness are paramount. These findings validate our core hypothesis: structured stochasticity via MC Dropout can enrich text generation without requiring full Bayesian inference or retraining. Future work may explore combining MC Dropout with large-scale models and reinforcement learning to further refine the balance between coherence, novelty, and efficiency in open-ended language generation.

References

[1] A. Vaswani et al. "Attention is all you need". In: *arXiv preprint* arXiv:1706.03762 (2017).

[2] A. Radford et al. "Improving language understanding by generative pre-training". In: *arXiv preprint* arXiv:1801.06146 (2018).

[3] A.-L. Jousselme et al. "Uncertain about ChatGPT: enabling the uncertainty evaluation of large language models". In: *IEEE* (2023).

[4] Ari Holtzman et al. "The Curious Case of Neural Text Degeneration". In: *arXiv preprint* arXiv:1904.09751 (2019). Published in ICLR 2020; DOI: `https://doi.org/10.48550/arXiv.1904.09751`.

[5] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *arXiv preprint* arXiv:1506.02142 (2015). Submitted on 6 Jun 2015 (v1), last revised 4 Oct 2016 (this version, v6); Published in ICML 2016.

[6] Jianfeng He et al. “Towards More Accurate Uncertainty Estimation In Text Classification”. In: *arXiv preprint* (2020). Discovery Analytics Center, Virginia Tech, Falls Church, VA, USA; Computer Science and Engineering, Mississippi State University, Starkville, Mississippi, USA; Department of Computer Science, American University, Washington, DC, USA; Emails: {jianfenghe, xuczhang, slei, fanglanc, hamdani, ctlu}@vt.edu, zchen@cse.msstate.edu, bei.xiao@american.edu.

[7] Jianing Li et al. “On the Relation between Quality-Diversity Evaluation and Distribution-Fitting Goal in Text Generation”. In: *arXiv preprint* (2020). Supported by Beijing Academy of Artificial Intelligence (BAAI) under Grants No. BAAI2019ZD0306, and BAAI2020ZJ0303, the National Natural Science Foundation of China (NSFC) under Grants No. 61722211, 61773362, 61872338, 61902381, and 61906180, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, the National Key RD Program of China under Grants No. 2016QY02D0405, the LenovoCAS Joint Lab Youth Scientist Project.