Diffusion-Driven Shakespeare: Diffusion for Conditional Text Generation

Joseph Samuel josamuel@ucsd.edu
Xian Ying Kong xykong@ucsd.edu
Alexander Jensen adjensen@ucsd.edu
Arya Mazumdar arya@ucsd.edu

Abstract

The application of diffusion models in natural language has been an emerging area of research recently, especially for conditional text generation. While diffusion models have always been known for their powerful image generation capabilities, its limited application to text generation is largely due to the inherent discreteness of text. Despite that, recent research has successfully applied continuous diffusion on text and shown breakthrough in conditional text generation. The success of diffusion models in conditional text generation can be attributed to the end-to-end learning of feature and embedding space. In this paper, we implement DiffuSeq on conversations in Shakespeare's Plays to demonstrate its ability to learn complex text representations and sentence structures. We further supplement DiffuSeq with advanced fine-tuning techniques including layer-wise learning rate decay and warm up steps to achieve better training quality. We find that loss drops further by 40% when advanced fine-tuning techniques are adopted. With longer Shakespeare lines, we are able to create cohesive lines that can further possibly follow context and sentiment.

Code: https://github.com/xianyingkong/diffusion-text-generation/

1 Introduction

1.1 Overview

Diffusion models are a type of generative model that simulates the natural data generation process. They work by iteratively denoising noisy samples to generate high-quality samples that closely approximate its training data distribution. Ho et al. (2020) is one of the pioneering works in the landscape of diffusion models for image generation that has served as a foundational framework for subsequent research of diffusion models in the domain of computer vision and audio. More recently, Li et al. (2022) has pioneered the application of continuous diffusion models in text generation and shown comparable results with autoregressive-based language models. Li et al. (2022)'s Diffusion-LM is trained by introducing embedding space and rounding steps to the standard diffusion method. To facilitate conditional text generation, Diffusion-LM trains a separate classifier to guide the generation of text given a specific control task. Then, Gong et al. (2023) extends the application of diffusion models in conditional text generation to performing sequence-to-sequence (Seq2Seq) text generation in a classifier-free manner by introducing partial noising in the forward diffusion process and conditional denoising in the reverse diffusion process.

In this paper, we extend the application of diffusion models in modern English text generation to classical English text generation to demonstrate the ability of diffusion models to learn complex text representations and structures. We follow the architecture of DiffuSeq proposed by Gong et al. (2023) which learns the embedding space by partially corrupting the embeddings of a target sequence and using an source sequence for guidance. By doing so, the model is able to use the source sequence to predict the target sequence by denoising the noisy word embeddings, thus creating random and coherent generated text that follows a given narrative or subject.

1.2 Related Work

Denoising Diffusion Probabilistic Models (Ho et al. 2020), DDPM in short, has served as the basis for diffusion models in the domain of computer vision and audio. Ho et al. defines a noising schedule and probabilistic objective function for the diffusion process. Our underlying architecture follows a similar format of loss and noising process.

Diffusion-LM (Li et al. 2022) notes a significant change to continuous text embedding diffusion models, as Li et al. implements an embedding learning process to both train classifier-based text diffusion as well as a diffusion model denoising process to learn word embeddings, as well as a denoising process that follows guidance from a classifier. This paper proposes significant improvements on the embedding learning process and improved loss function on the premise that a classifier needs to be trained in order to create a controlled setting. This paper proposes a very complex training process, but is notable for it's contribution to subsequent researches.

DiffuSeq (Gong et al. 2023) is a pioneering work for diffused-based sequence to sequence

learning. Similar to Diffusion-LM using a classifier to guide denoising, DiffuSeq uses an unnoisy source sequence as a conditional for the denoising process, thus guiding what the model will generate text based on. Due to it's independence from other models, other than transfer learning for saving computation costs, it offers a complete end-to-end training based on unlabeled data.

The three works of literature above form the basis of the project. The first one is key to the understanding and development of DDPM, it goes in depth on how DDPM's function and the internal process of DDPM's. The second paper and third paper extend the basis of Ho et al. (2020) into the discrete text domain. Our model is primarily inspired by DiffuSeq by Gong et al. (2023).

1.3 Dataset

We use Shakespeare's plays in our task to demonstrate that diffusion models are able to capture complex semantics and structures of text. In addition, it may be necessary to learn both characters and words from scratch in order to create a suitable representation that fits Early Modern English instead of Modern English. To implement DiffuSeq's model architecture effectively, we arrange conversations in Shakespeare's plays following the format {src: <text>, trg:<text>} for each data point and each sequence is capped at a length of 200 characters. Our train data consists of 32531 data samples, each is a pair of source-target sequence while our validation data contains 8123 data samples.

Extensive effort is taken to clean the dataset as we remove non-conversational lines and non-English lines (mostly in Henry V) in the plays, segment conversations by the last punctuation within the specified length of sequence, and group lines of conversations of each character based on their turns in each act scene. Furthermore, we train a new tokenizer using the original Shakespeare's plays consisting of 30268 vocabularies.

2 Methods

Diffusion models are characterized by a forward process and a reverse process. Firstly, the forward diffusion process is where the model iteratively adds noise sampled from a specific distribution, typically Gaussian, to the input until it reaches a state of pure Gaussian noise. Secondly, the reverse diffusion process is where the model learns to reverse the noising process by mapping a pure noise input back to one that approximates the original data distribution, thus attempting to get from a noisy representation back into a unnoisy example. In diffusion, we define a hyperparameter T for number of timesteps. The higher the timestep, the noisier the inputs are, that is $T = 0$ being a completely noise-free input, and higher timesteps ($T \rightarrow 1000$) being a very noisy input.

Our input sequence is a concatenation of the source and target sequences. More specifically, an input sequence consists of a source sequence (w^x) of size $\mathbb{R}^{m*d}$ followed by an end token, a separator token, then by a target sequence (w^y) of size $\mathbb{R}^{n*d}$, where m and n are the

respective number of tokens in the sequence, and d is the embedding dimension of each token. After this sequence is another end token to represent the ending of the sequence. Representing this sequence as z^{m+n+3}, we apply partial noising to this sequence whereby:

- The source sequence w^x has slight amount noise added to improve generalization.
- The target sequence w^y has noise applied to it according to a noise schedule similar to DDPM's noising schedule.

Our diffusion model has 2000 diffusion timesteps and a square-root noise schedule which follows that of Diffusion-LM (Li et al. 2022) and DiffuSeq (Gong et al. 2023). We treat the number of diffusion steps as a hyperparameter and we experiment with 500, 1000, 2000 and 2500 diffusion steps.

In the reverse process, we use a BERT-based transformer to predict and reconstruct the original target sequence guided by the source sequence as part of the input sequence. The same transformer is used in all diffusion steps to recover the original sequence. Our transformer consists of 91M parameters, with a sequence length of 128 and hidden dimension of 128 units.

To generate text samples, the model starts by sampling from a standard Gaussian distribution, which serves as the initial corrupted representation. This corrupted representation is then passed through the reverse process, where it undergoes iterative denoising to gradually recover the original data. At each denoising step, the model produces a sequence of word embeddings, which are then rounded to obtain discrete word tokens. This process continues until the original data is fully reconstructed.

Algorithm 1: DiffuSeq Learning Objective

Input: $w^z = w_1^x, \ldots, w_m^x, [END], [SEP], w_1^y, \ldots, w_n^y, [END]$
Output: err, the error of the current batch/epoch
$\sigma \leftarrow$ noise according to noise schedule $(0 < \sigma \leq .01)$
$\epsilon \leftarrow$ Model prediction of w_0^z
$e \sim N(0, 1)$

1. $w_0^z \leftarrow w^z + \sigma * e$ (Applied to entire embedding matrix)
2. $w_t^z \leftarrow \sqrt{\tilde{\alpha}_t} w_0^z + \epsilon\sqrt{1-\tilde{\alpha}}$ (Only applied to (w^y embeddings))
3. $L_{MSE} = \|w_0^z - \epsilon_\theta(w_t^z, t)\|^2$ If t = 0
4. $L_{MSE} = q(w_{t-1}^z | w_t^z, x_0^z)$ If t != 0
5. $L_t = (\sqrt{\tilde{\alpha}} w_0^z)^2$
6. $L_{enc} = -log(p(w|w_0^z))^2$ (Embedding regularization)
7. Loss $= L_{MSE} + L_t + L_{enc}$

3 Experiments

3.1 Baseline

Our baseline model is a scaled-down version of DiffuSeq trained on 1 GPU without using PyTorch's Distributed package. Our final baseline model is trained with 30,000 epochs, 2000 diffusion steps, 0.1 dropout rate, sequence length of 128, and hidden dimension of 128 units. We mainly focused on changing the number of diffusion steps, ranging from 500 to 2500, which suggests a trade-off between evaluation time and complexity of the model's prediction task. We select the best-trained model by evaluating against loss on the validation set at the end of every 10 training epochs.

3.2 Custom Fine-tuning Strategies

We explore various fine-tuning strategies by making different yet significant changes to the dataset, i.e. the extent of data cleaning, preprocessing methods, and a combination of different datasets, as well as training methods, i.e. transfer learning and advanced deep learning methods.

Our final best-trained model has 1000 diffusion steps, 128 hidden dimension, 0.1 dropout rate, 0.0 weight decay, batch size of 30 and layer-wise learning rate decay of 0.9 with a starting learning rate of 0.0001 at the bottom-most layer (closest to input). The model is trained on 1 2080Ti GPU with 30,000 epochs and 500 warm up steps, taking roughly 7-8 hours. 2-3 GPUs cuts down the training to about 3-4 hours.

3.2.1 Transfer Learning

Our first few experiments are designed to improve the ability of the baseline model to generate coherent sentences. Inspired by DiffuSeq's open-domain dialogue sequence generation, we first train our model on a combination of CommonSense dialogues (Gong et al. 2023) and Shakespeare's plays with a ratio of 90% to 10% to allow the model learn most of modern text structures. Next, we retrain the model on a more targeted dataset, i.e. only Shakespeare's plays. We train with and without reinitializing the final layer of the transformer.

Our experiments end up with the model not being able to generate relevant words at all, with some experiments having words in other languages being generated, which is worse than our baseline model. However, these experiment have served as an important foundation to our experiments with the dataset as we observe the importance of the quality of conversations and tokenizer in training the model.

3.2.2 Feature Engineering and More Shakespeare's

We further experiment the baseline model with different data pre-processing methods including text cleaning, definition of conversation, inspection on languages in each play. Our final experimental settings include minimal text cleaning where we decide to keep all words in its original forms, for example 'tis instead this and 'd which is equivalent to -ed in past tense in most cases, and setting a maximum conversation length of 128 characters and using its immediate next conversation as a response instead of solely using the next line (as written in the original Shakespeare's plays) as response. Furthermore, we experiment with training the model play by play and without (i.e. shuffling the dataset). We find that shuffling the dataset plays a crucial role in model training, possibly due to the large correlation between data samples of the same play when model is trained play by play.

We have also attempted to increase our data sample by introducing Shakespeare's Sonnets, which are poems written by Shakespeares that follow a strict rule of structure. While the baseline model trained on both Shakespeare's poems and plays is able to generate Shakespearean style text, there is a strong lack of sentence coherence. We hypothesize that the vast difference in the structure of dialogue and poetry has posed a challenge to the baseline model to learn efficiently with the same training setup.

Nonetheless, experiments with more granular feature engineering have given us better results that the model is capture semantic relationship between words as seen in the generated response. The model is able to identify gender related words such as the pronouns he or she and return the same in its generated response.

We have also greatly benefited from increasing the length of the input sequence during training as the model is able to further understand the topic and sentiment of the source input, and thus generating more coherent sequence.

3.2.3 Warm Up Steps

The first custom fine-tuning technique for transformers we implement is warm up steps with cosine learning rate scheduler in the training loop. With a total of 30000 training epochs, we use 500 warm up steps.

3.2.4 Layer-Wise Learning Rate Decay

The second custom fine-tuning technique for transformers we implement is layer-wise learning rate decay (LLRD) using AdamW optimizer. We use a decay rate of 0.9, whereby the learning rate decreases from the top-most layer to the bottom-most layer at a rate of 0.9. Table A 1 shows the learning rate of each layer of our own categorization, from the bottom-most to the top-most layer:

3.2.5 Reinitialization of Last Layer of Transformer

The third custom fine-tuning technique for transformers we implement is reinitializing the last layer, i.e. the 12th layer of the transformer. We find that the results do not differ much from a model with warm up steps and LLRD.

We experiment with decay rates 0.75, 0.9 and 0.99, with different combinations of starting learning rate. We generally find that training with 0.75 decay results in worse training process as loss starts rising back to approximately the early losses towards later timesteps.

3.2.6 Hyperparameters Fine-tuning

A few more notable hyperparameters that we closely experiment and observe are the number of diffusion steps (500/ 1000/ 2000/ 2500), dropout rate (0.1/ 0.15/ 0.2), batch size (20/ 30/ 50/ 100), and weight decay (0.0/ 1e-5). Our final experiment settings are 2000 diffusion steps, 0.1 dropout rate, batch size of 30 for training and 60 for validation, and weight decay of 0.0.

4 Results

Our models are able to sample new continuations of Shakespeare lines that match a similar sentiment to the Shakeaspearean line input into the model. Our models perform better with longer input sequences and the samples are also of high diversity. These responses are not entirely coherent all the time however, as some generations may not follow the given line well, and thus some human verification may be required. Table 1 shows some example responses that the diffusion model generated at each of the following diffusion steps and the prompt that was used.

Table 1: Comparison of generated response by different diffusion steps

Diffusion Steps	Prompt	Generated Response
500	Call her forth to me	Ay.
	No, sir, I live by the church.	Come, well, well.
	Wherefore art thou?	Farewell be, let thou home.
1000	Call her forth to me	O, none! Haste, in a measure enough. But we have of in to and her plain.
	No, sir, I live by the church.	Is a law it prove a christian, the.
	Wherefore art thou?	Ay, ay, find you could.
2000	Call her forth to me	Be her, there.
	No, sir, I live by the church.	Good, save grace!
	Wherefore art thou?	O not Demetrius. I see.
2500	Call her forth to me	If prithee, away, here, if day how have he.
	No, sir, I live by the church.	Take'll know i'my.
	Wherefore art thou?	Ay,, as come.

Figure 1 shows the training loss and validation loss of our model. Loss shrinks quickly during the early epochs but converges slowly beyond that as we typically get our best-trained model at the 23,000th epoch. Despite that, we recognize that loss for diffusion is not an accurate measure of sample quality, and cannot be used solely to determine a model's performance compared to other models.

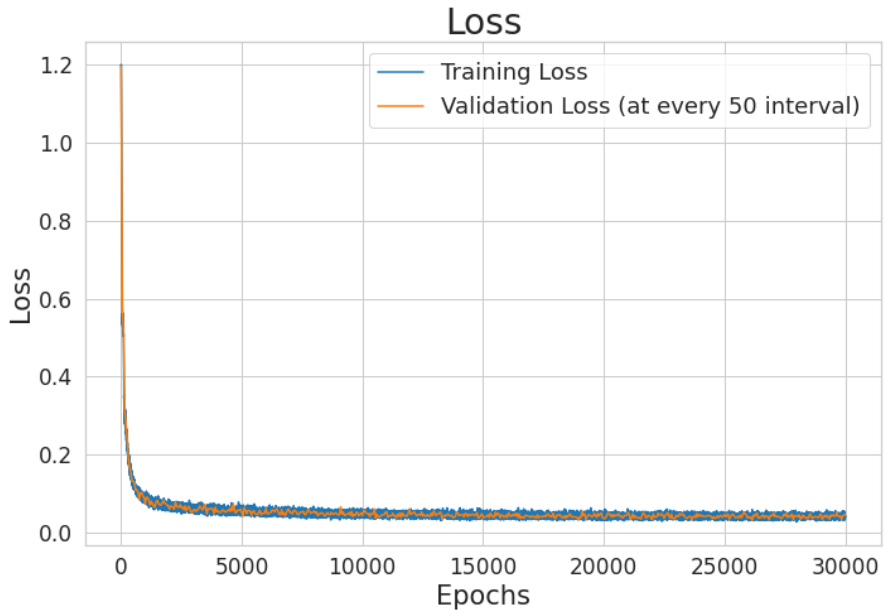

Figure 1: Training vs. validation loss for 2000 diffusion steps

Figure 2 visually depicts the embedding space of positive and negative words learned by our model and shows how the model is able to cluster similar words belonging to comedies or tragedies.

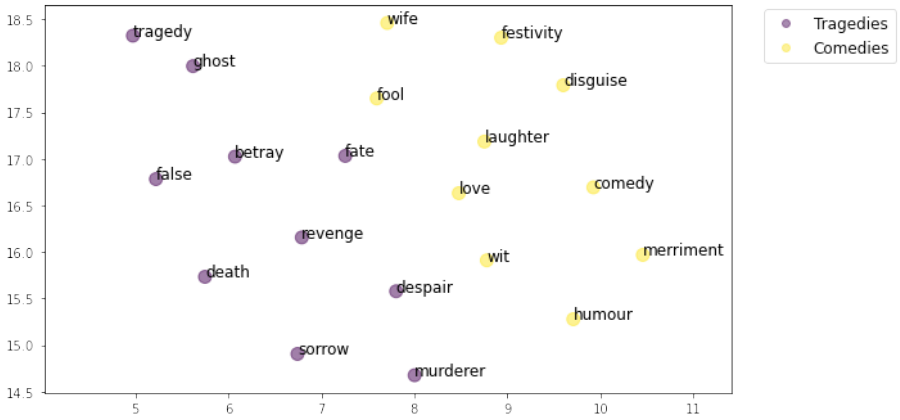

Figure 2: UMAP projection of word embeddings by positive and negative sentiments

When the model is trained with character names at the start of each conversation in the source-target pairs, we find the our model is able to capture some level of social network

amongst the characters. Figure 3 visually depicts the embedding space of character names in selected plays and shows how our model is able to associate characters of the same play.

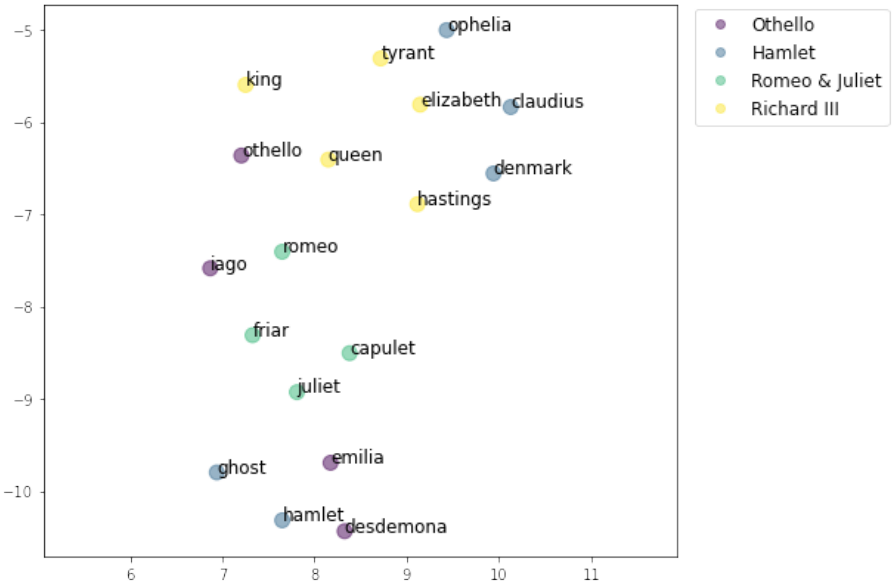

Figure 3: UMAP projection of character names in selected famous plays

5 Discussion

In conclusion, our model is able to learn and generate sequences that follow the style of Shakespeare's. While we are able to prove the capability of diffusion models in conditional text generation, there are a few areas that can be expanded.

Firstly, language is not only about stringing words together but also about conveying meaning, context, and coherence. We find that our modified DiffuSeq lacks at upholding sentence coherence. We hypothesize that this could be attributed to the end-to-end training of embedding space and quality of training data used to train these models, as conversations in Shakespeare's plays contain high inconsistencies in terms of sequence length and word meaning. Moreover, language is inherently context-dependent, and generating coherent sentences requires understanding and maintaining context throughout the text generation process.

Secondly, echoing with Li et al. (2022), sampling in diffusion models for text generation is slow. Generating a sentence with 2000 diffusion steps for a sequence length of 5 words takes approximately 30 seconds.

Lastly, it is important to take into account the many different hyperparameters that can be adjusted, and to also consider the tradeoffs that these hyperparameters pose. Diffusion steps is an important hyperparameter we looked into, as it gives the model an easier prediction task with a larger amount of timesteps at the cost of sampling new examples and the possibility of overfitting due to the smaller noise required to be predicted between each timestep.

References

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation, 2022.

Appendices

A.1 Training Details

We used GPU 1080i/2080i from UC San Diego's Data Science and Machine Learning Platform (DSMLP) for all experiments.

A.2 Learning Rates

Table A 1: Layer-Wise Learning Rate

Layer Name	Learning Rate	Layer Group Index
Word embeddings	0.0001	1
Linear head layer	0.0001	1
Time embeddings	0.0001	1
Input projection	0.0001	1
Encoder layer 0	0.000111	2
Encoder layer 1	0.000123	3
Encoder layer 2	0.000137	4
Encoder layer 3	0.000152	5
Encoder layer 4	0.000169	6
Encoder layer 5	0.000188	7
Encoder layer 6	0.000209	8
Encoder layer 7	0.000232	9
Encoder layer 8	0.000258	10
Encoder layer 9	0.000287	11
Encoder layer 10	0.000319	12
Encoder layer 11	0.000354	13
Layer norm	0.000393	14
Output projection	0.000393	14