Sequence-to-Sequence Continuous Diffusion Language Models for Control Style Transfering

Quang Minh Dinh
Simon Fraser University
Burnaby, Canada
qmd@sfu.ca

Hung Phong Tran
Hanoi University of Science and Technology
Hanoi, Vietnam
phong.tnh200465@sis.hust.edu.vn

***Abstract*—The capability to generate high-quality texts in a controlled context is necessary for several industrial tasks, yet there are still few studies on diffusion models for texts, which are highly capable of performing controls on the generation process. As diffusion models often require a large amount of computing resources and training time to produce high-quality results, we propose the Sequence-to-Sequence Diffusion Language Model with Pre-trained Embeddings (SDPE), a continuous diffusion language model for sequence-to-sequence tasks that uses a pretrained BERT model for word embedding. We train SDPE along with other continuous diffusion language models in a low-resource environment and successfully demonstrate their potentials in performing sequence-to-sequence style transferring tasks, at the same time show that using a pre-trained BERT model allows a faster training process.**

I. Introduction

Along with the appearance of large language models such as BERT [19], T5 [20], GPT-3 [15], and PaLM [21], the performances of several downstream natural language processing tasks have been increased tremendously. As such models with hundreds of billions of parameters can capture the meaningful syntactic and semantic structures of the natural language and even retrieve knowledge from their parameters themselves [20], fine-tuning large pre-trained models for downstream tasks has become extremely common. However, there are still many limitations to these models, considering deploying them in real-world applications [1]. Most industrial tasks require a specific form of representation that is not likely to exist on large datasets and can not directly be inferred from the natural language's structures learnt by large models, thus requiring a degree of control over the text generation process. This is also widely known as a notorious issue of large autoregressive models, as they generate texts in a left-to-right manner and can not impose controls on unprocessed contexts.

The task of mapping natural language to a target meaning representation is called semantic parsing [2]. Semantic parsing is different from other sequence-to-sequence style transfer tasks, such as translation or summarization, in the sense that it requires heavy control over the text outputs to be used in practice. Several attempts have been made to take control of the text generation process of large autoregressive models. However, the successes of those plug-and-play methods that keep the language models frozen and steer the generation process of the language models, including FUDGE [16], GeDi [17], and DExperts [18], have been limited to simple, attribute-level controls [1]. In 2014, Berant and Liang [6] proposed that the task of semantic parsing can be effectively equivalent to paraphrasing the original natural language sentence into a canonical form with a specific grammar structure before converting it to the target meaning representation. Shin et al. [2], Schucher et al. [4], and Shin and Van Durme [5] made use of this idea and attempted to use prompt-tuning on large language models such as GPT-3 [15], BART [22], T5 [20], and OpenAI Codex [23] on the task of converting natural language into a canonical representation. Shin et al. [2] found that controlling natural language using canonicalization was better than mapping it directly to the meaning representation when used with a pre-trained language model, and Schucher et al. [4] showed that prompt-tuning was better than fine-tuning large T5 models in the low-data regime.

Although canonicalization allows relaxation in the constraints of the target meaning representation's grammar structure, further control methods are necessary to make the large language models return acceptable results. Shin et al. [2] perform controls on the outputs of the canonicalization process by using a length-normalized variant of beam search. They assumed that a `nextTokens` function that returns a list of tokens that can immediately follow the token provided as the input is available for each task. This function would be used by the beam search algorithm on every generation step to choose the most probable word at each position. This method required a heavy feature engineering process to generate the `nextTokens` function based on the grammar of the target meaning representation provided in advance, thus demonstrating the limitations of autoregressive language models in constraining the outputs.

On the other hand, recent works on continuous diffusion language models [1] have shown promising results on different language-controlling tasks without depending on a feature engineering process beforehand. However, these works only attempted to reconstruct the original text inputs with constraints about representation formats, which was fundamentally different from style transferring. Diffusion models also have a critical drawback in that they require large amounts of computing resources and training time to produce high-quality results, making autoregressive models a more common choice for any natural language processing task.

In this work, we evaluate the performances of different continuous diffusion language models on sequence-to-sequence style transferring tasks, at the same time assess the effects of pre-trained BERT models in enhancing the training process of the continuous diffusion language models. We present the Sequence-to-Sequence Diffusion Language Model with Pre-trained Embeddings (SDPE), a continuous diffusion language model for style transferring. It uses a similar architecture to Diffusion-LM [1], with modifications for sequence-to-sequence tasks, and makes use of BERT [19], a transformer-based bidirectional encoding model, to reduce the training time. We evaluate the results of SDPE and several variations of it on the Overnight dataset [9] after being trained using low computing resources in a short amount of time and report ROUGE scores as the major performance metric. We also experiment with the diffusion model with pre-trained BERT embeddings (Diff-BERT) proposed by Jia [3] and use it as the baseline model for SDPE.

Our results show that in 2000 training steps, Modified Diff-BERT models outperform SDPEs in every domain. However, as the training time increases, SDPE models have a high potential of achieving higher results as the results of Modified Diff-BERT models start decreasing after only a few training steps. Furthermore, we demonstrate that increasing the size of the base BERT model allows SDPEs to perform better in capturing unigrams and high n-gram structures, while decreases their abilities in capturing bigrams.

Our contributions can be summarized as follows:

- We propose SDPE, a new continuous diffusion language model for sequence-to-sequence style transferring that uses a pre-trained BERT model for word embedding.
- Our work is one of the first works that study continuous diffusion language models for sequence-to-sequence tasks.
- We successfully demonstrate the potential of continuous diffusion language models in performing sequence-to-sequence style transferring tasks, at the same time show that taking advantage of pre-trained BERT models allows a faster diffusion training process.

II. Related Work

A. Transformer-based bidirectional encoding models and autoregressive language models

The original Transformer [26] is a classic sequence-to-sequence model that contains an encoder to capture the word representations from a given sequence and a decoder to generate another sequence from the word embeddings provided by the encoder. The Transformer outperformed every LSTM-based and RNN-based model at the time it was introduced, and Transformer-based models are still state-of-the-art in most natural language processing tasks at the time of writing.

The bidirectional encoding models such as BERT [19], RoBERTa [27], and ELECTRA [28] use several Transformer's encoder layers to learn word representations from corpora of texts by corrupting word sequences and trying to reconstruct the original inputs. This architecture is commonly used for classification or mask-filling problems that require knowledge of word relationships and the global context of the corpus. We use BERT [19] to denoise word data in SDPE as most of the diffusion process is performed in a continuous latent space, and pre-trained embeddings can speed up the training time.

The autoregressive language models like GPT-3 [15] use many Transformer's decoder layers for different text generation tasks. They process input and generate texts from left to right, using a Masked Multi-head Attention layer to mask all the words that they have not read. Although this architecture enables autoregressive language models to generate high-quality texts, it imposes a restriction on those models' text generation ability in different control contexts.

B. Diffusion Models for Text

Diffusion models have been a topic of interest among the general population recently after the introduction of DALL-E [24] and Stable Diffusion [25], and have observed several successes in the domain of visual and audio. When applied in natural language processing, Austin et al. [11] and Hoogeboom et al. [12], [13] proposed text diffusion models on discrete spaces using a corruption process. In contrast, [1] introduced Diffusion-LM, which was the first continuous diffusion model for text. Diffusion-LM enables the use of gradient-based methods on different control tasks by incrementally denoising Gaussian noise vectors and then converting them into discrete word vectors using a rounding step. Li et al. [1] reported a double in the success rates of the previous plug-and-play methods on four different classifier-guided control tasks, namely semantic content, parts-of-speech, syntax tree, and syntax spans, and significant outperformance of those plug-and-play methods on span-anchored controls tasks such as length control and infilling. Jia [3] argued that using the embedding layers of a pre-trained BERT model and finetuning its encoders allows Diffusion-LM to train faster and proposed a method to train Diffusion-LM using a pre-trained BERT model.

III. Background

A. Continuous Diffusion Language Model for Text

Diffusion models work by modelling the data x_0 as a Markov chain from x_T to x_0 with x_T being a standard Gaussian distribution, and x_{T-1} to x_0 being Gaussian.

For Diffusion-LM, Li et al. [1] proposed that continuous latent data x_0 can be obtained by embedding the original input sequence using a function $E_{MB}(w_i)$. The embedding of a sentence $\mathbf{w}$ with length n is defined as $E_{MB}(\boldsymbol{w}) = [E_{MB}(w_1), ..., E_{MB}(w_n)] \in \mathbb{R}^{nd}$. The transition from sentence $\mathbf{w}$ to x_0 is a Markov transition parametrized by $q_\phi(x_0|\boldsymbol{w}) = \mathcal{N}(E_{MB}(\boldsymbol{w}), \sigma_0 I)$, which allows mapping discrete text to continuous vectors. To generate the noise vectors x_T, in the forward process, Gaussian noise is incrementally added to x_0 through T steps until the results are approximately Gaussian. Each transition is parametrized by

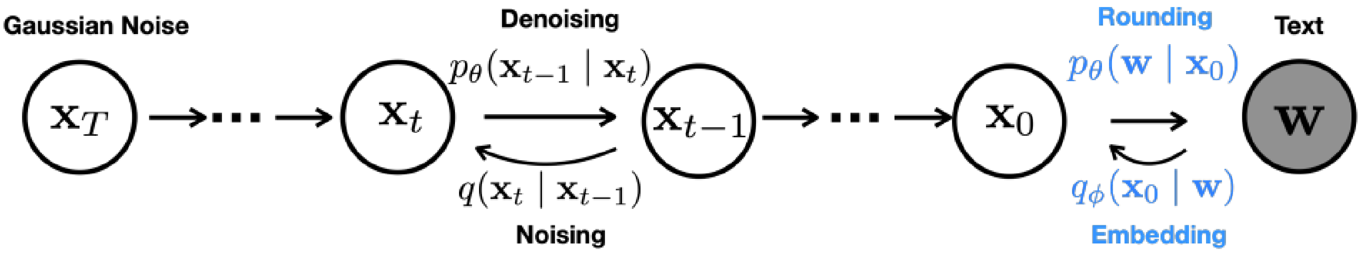
Gaussian Noise
Denoising
$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$
$q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$
Noising
Rounding
$p_\theta(\mathbf{w} \mid \mathbf{x}_0)$
$q_\phi(\mathbf{x}_0 \mid \mathbf{w})$
Embedding
Text
$\mathbf{x}_T$
$\mathbf{x}_t$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$
$\mathbf{w}$

Fig. 1: A graphical model representing the forward and reverse diffusion processes.
Adapted from Diffusion-LM Improves Controllable Text Generation, by Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., & Hashimoto, T. B. (2022). Use permitted under the Creative Commons Attribution License CC BY 4.0.)

$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \boldsymbol{I})$, with hyperparameter β_t being the amount of noise added at step t. In the reverse process, each denoising transition is parametrized by $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$. To convert the predicted x_0 back to a discrete sentence, Li et al. [1] introduced a rounding step that chooses the most probable word at each position, based on $\operatorname{argmax} p_\theta(\mathbf{w}|\mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i|x_i)$, with $p_\theta(w_i|x_i)$ being a softmax distribution. The training objective is defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{simple}}^{\text{e2e}}(\mathbf{w}) = \underset{q_\phi(\mathbf{x}_{0:T}|\mathbf{w})}{\mathbb{E}} &\left[\mathcal{L}_{\text{simple}}(\mathbf{x}_0)\right. \\ &+ \|\mathrm{E}_{\mathrm{MB}}(\mathbf{w}) - \mu_\theta(\mathbf{x}_1, 1)\|^2 \\ &\left.- \log p_\theta(\mathbf{w} \mid \mathbf{x}_0)\right],\end{aligned} \tag{1}$$

with

$$\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \underset{q(\mathbf{x}_t|\mathbf{x}_0)}{\mathbb{E}} \|\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 \tag{2}$$

Control is performed on the continuous latent variables from x_0 to x_T, specifically, by performing gradient update on x_{t-1} at step t:

$$\begin{aligned}\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_{t-1}} &\log p(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \\ + \nabla_{\mathbf{x}_{t-1}} &\log p(\mathbf{c} \mid \mathbf{x}_{t-1}),\end{aligned} \tag{3}$$

with $\mathbf{c}$ being the control variable.

B. Re-parametrization and Clamping for faster computing

Li et al. [1] attempted to further simplify $\mathcal{L}_{\text{simple}}^{\text{e2e}}$ by re-parametrizing $\mathcal{L}_{\text{simple}}(\mathbf{x}_0)$. Specifically, the authors defined a new $\mathcal{L}_{\text{simple}}$ function that was parametrized via x_0:

$$\mathcal{L}_{\mathbf{x}_0\text{-simple}}^{e2e}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t} \|f_\theta(\mathbf{x}_t, t) - \mathbf{x}_0\|^2 \tag{4}$$

which set the model's training objective to predicting x_0 in every diffusion step. This approach produced a better result as the forward process was no longer needed to compute the loss, and the model learnt to center x_0 near a word embedding faster.

Li et al. [1] also introduced the clamping method. In every denoising step, the predicted word was mapped to the nearest word embeddings before being denoised. This method forced every latent word data to commit to exact words, reducing errors in the final step.

Another re-parametrization trick adopted from the implementation of the Variational Encoders [7] can also be applied to the forward diffusion process to make the transition from x_0 to x_T deterministic: $x_t = \sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon$.

C. Denoising Diffusion Implicit Models

Song et al. [29] introduced the Denoising Diffusion Implicit Models (DDIM) that replaced the Markovian backward diffusion process with another deterministic non-Markovian process with the same training objectives, thus removing the necessity to simulate a Markov chain for many steps to produce a sample. Specifically, in each step of the backward diffusion process, x_{t-1} can be obtained from x_t deterministically:

$$\begin{aligned}\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} &\underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{"predicted } \boldsymbol{x}_0 \text{ "}} \\ + &\underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{"direction pointing to } \boldsymbol{x}_t \text{ "}} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}}\end{aligned} \tag{5}$$

This method performed better than the traditional way of simulating a Markov chain, as Song et al. [29] reported a 10-50 times faster in terms of wall-clock time to produce high-quality data samples.

D. Diffusion model with pre-trained BERT embeddings (Diff-BERT)

Jia [3] attempted to train Diffusion-LM's forward process by finetuning a BERT encoder with the re-parametrization trick introduced by Li et al. [1] being the training objective.

In the forward process, the BERT embedding layers after receiving tokenized input sequences return an embedding matrix x_0 with each word's embedding dimension being the hidden size of BERT's last encoder layer, along with its encoded positional matrix. *step*, which is a random number from 1 to a max diffusion steps value given in advance, is fed to an embedding layer to create a time embedding matrix. Noise is generated from a standard Gaussian distribution afterward and is added to x_0 by using the re-parametrization trick with time embedding and positional embedding added to it to form x_1. x_1 is then fed to a normalization layer, then the BERT encoder layers to predict x_0. The first word embedding in x_0 then goes through a fully-connected layer to reconstruct the original tokenized inputs, and the loss value is the cross entropy loss between the predicted tokenized inputs and the original ones.

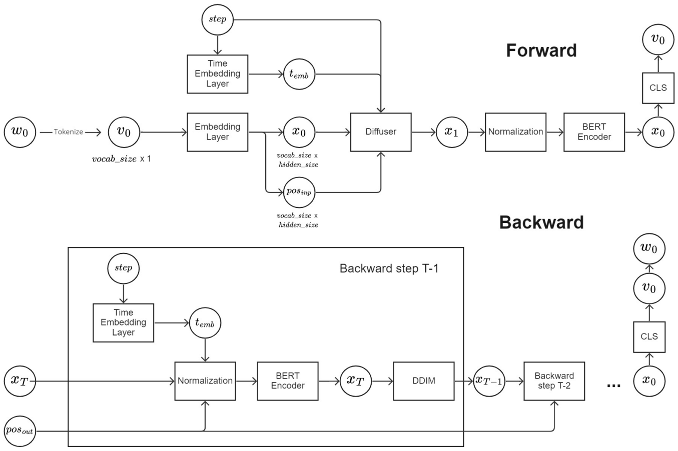

Fig. 2: The forward and backward diffusion processes with pre-trained BERT Embedding and Encoder layers.

The backward diffusion process is not trainable, and is used to generate word data using the finetuned BERT model. x_T is sampled from a standard Gaussian distribution, and is fed to the normalization layer along with its corresponding positional encoding matrix and the time embedding matrix. In the backward diffusion process, $step$ starts from the max diffusion step value and keeps decreasing until it reaches one. The normalized data is fed to the BERT encoder to predict an x_T that is similar to the x_T in the forward diffusion process. x_{T-1} is then obtained from x_T using the formula used by DDIM, and the process repeats until x_0 is obtained. The tokenized word data is acquired from x_0 in the same manner as the forward diffusion process and is fed to a tokenizer to decode and obtain the target sequences.

IV. Methodology

To evaluate the performance of the continuous diffusion language models on style transfer tasks in a low-resource training environment, we introduce the Sequence-to-Sequence Diffusion Language Model with Pre-trained Embeddings (SDPE), at the same time modify the diffusion model proposed by Jia [3] to work with sequence-to-sequence tasks. We look at the ROUGE scores of all models and compare them to have a comprehensive view of their performance in a resource-limited environment. We also test SDPT using two different BERT base models to evaluate the effects of pre-trained BERT models on SDPT's performances.

A. Modified diffusion model with pre-trained BERT embeddings

The diffusion model with pre-trained BERT embeddings [3] was designed to work with only input sequences, as it tries to reconstruct the original sequences from the noisy data. In order to make it work with sequence-to-sequence tasks, we modified its architecture by changing w_0 to the target sequences and adding the word embeddings and positional embeddings of the input sequences in the same stages that the time embedding is added to the noisy data, both in the forward and the backward process.

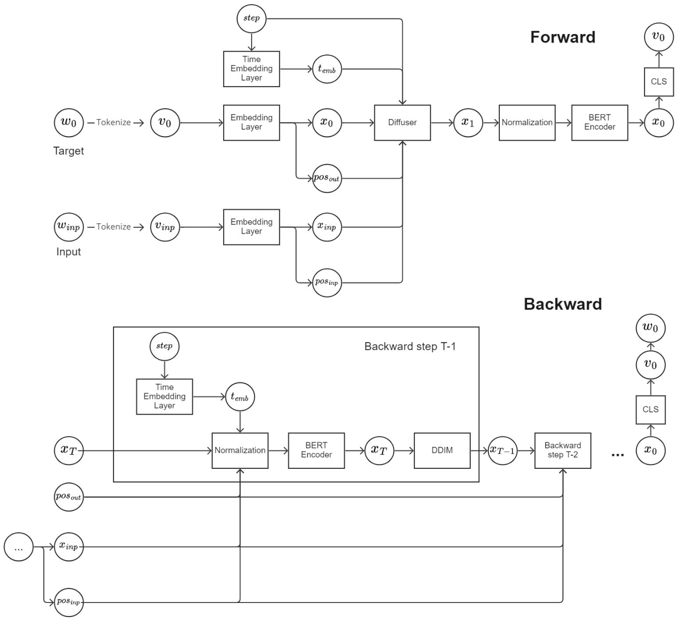

Fig. 3: The modified Diff-BERT for sequence-to-sequence tasks.

We also attempted to modify the method to generate the diffusion step in the forward diffusion process of the model so that the diffusion step increases monotonically and compare its performance to the other models.

B. Sequence-to-Sequence Diffusion Language Model with Pre-trained Embeddings (SDPE)

The architecture of SDPE is similar to the backward diffusion process of the modified Modified Diff-BERT shown in Figure 3. However, we make few major differences. We follow the re-parametrization trick introduced by Li et al. [1], removing the forward diffusion process and making the backward diffusion process trainable. This is fundamentally different from the Modified Diff-BERT as it only trains the forward diffusion process and uses the backward diffusion process for inference. We also add the word embeddings for the target sequences and use them to calculate the loss value. We define our loss function as similar to the loss function proposed by Li et al. [1](1), which is the summation of three other losses. The first one is the mean of the mean squared error (MSE) loss of the noise data in every diffusion step

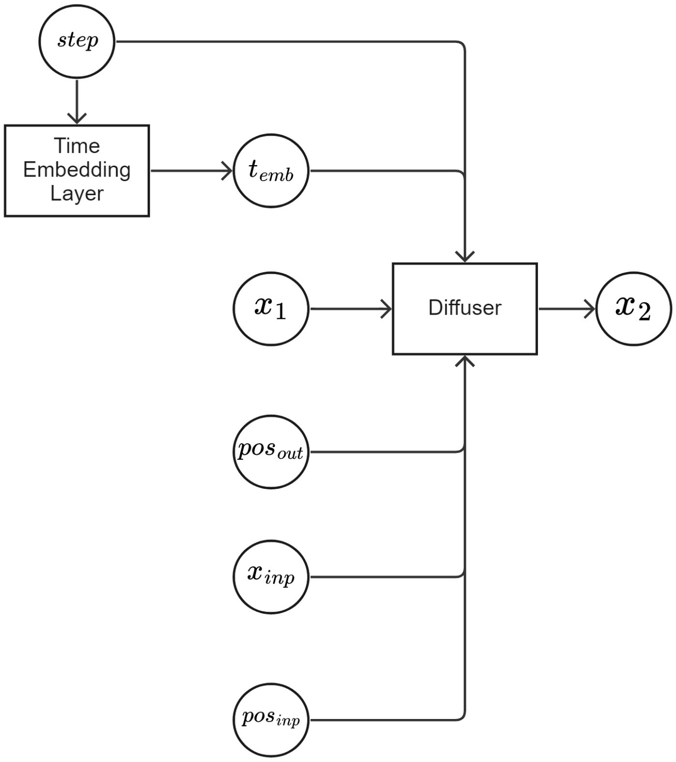

Fig. 4: The modified Diff-BERT for sequence-to-sequence tasks with a monotonically increasing diffusion step.

and the target word embeddings. The second loss is the mean squared error of the last denoised data and the target word embeddings. The last loss is the cross entropy loss between the final predicted sequences and the target sequences. We also test the clamping method by feeding the predicted vector x_t to the CLS layer to get the predicted tokenized vector and re-feeding it to the BERT embedding layer to map it to exact word embeddings before denoising it to get x_{t-1}.

C. Experimental setup

All models are trained on the Overnight dataset [9], using an NVIDIA A6000 GPU with 45 GiB RAM, in 20 epochs, with 250-500 diffusion steps, and using bert-tiny with hidden size 128 and 2 hidden layers as the base model. For SDPE, we also experiment using bert-mini hidden size 256 and 4 hidden layers to evaluate the effects of pre-trained BERT models on its performances.

For evaluation, we use the ROUGE scores as the major performance metric. Intuitively, ROUGE scores range from 0 to 1 depending on how many words in the generated sequences appear in the target sequence, with ROUGE-1 and ROUGE-2 measuring unigram and bigram overlap respectively, and ROUGE-L measuring the longest matching sequences. For a style transfer task in a resource-limited environment, ROUGE is a good metric since it measures how close the generated sequence is to the target sequence, and it is not likely that diffusion models would generate a high enough result for the other metrics to be effective. We do not expect our model to reach state-of-the-art performance, as it is trained using limited computing resources, and it is well-known that diffusion models take long times and lots of computing power to generate quality results.

D. Datasets

Overnight was collected by Wang et al. [9] and contains 13,682 triples of natural language sentence, canonical form, and meaning representation across eight different domains. They initially used a human-defined grammar to generate canonical utterance and meaning representation pairs, then utilized crowdsourcing to paraphrase the canonical utterances into their natural language form. The dataset was used in their study of building a semantic parser in a short amount of time. We follow Shin et al. [2] and use hold-out validation to sample 200 out of 640-3,535 examples for each domain for the training set. 20% of the remaining data is used for validation. This dataset is commonly used for different semantic parsing tasks and contains data from eight different domains. Hence, it is suitable for our work.

V. Main Results

We train SDPE and Modified Diff-BERT models on the Overnight dataset. In 2000 steps, both Modified Diff-BERT models outperform SDPEs in every domain. However, SDPE models have a high potential of achieving the highest results as the results of Modified Diff-BERT models start decreasing after only a few training steps.

A. SDPEs vs Modified Diff-BERTs

As shown in Table I, both Modified Diff-BERT models perform better than SDPE in all domains, with Modified Diff-BERT with an increasing step having higher ROUGE-L scores than Modified Diff-BERT on 6 out of 8 domains. SDPE produces similar ROUGE-1 and ROUGE-L scores as Modified Diff-BERT on two domains, namely Publication and Restaurants.

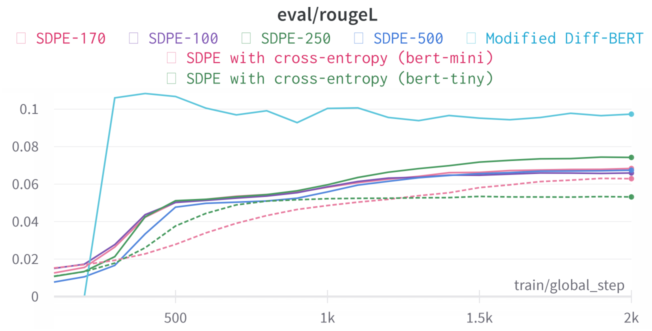

Fig. 5: ROUGE-L scores of Modified Diff-BERT and SDPE variations: base SDPE trained using bert-tiny with 100, 170, 250 and 500 diffusion steps, SDPE trained using bert-tiny and bert-mini with the MSE losses changed to cross-entropy losses.

Model	Metric	Calendar	Basketball	Blocks	Housing	Publication	Recipes	Restaurants	Social Network
SDPE	ROUGE-1	0,0733	0,0919	0,0983	0,0781	0,0825	0,0675	0,0843	0,0804
	ROUGE-2	0,0042	0,0004	0,0119	0,0063	0,0054	0,0033	0,0094	0,0067
	ROUGE-L	0,0658	0,0783	0,0873	0,0684	0,0746	0,0599	0,0750	0,0733
Modified Diff-BERT	ROUGE-1	0,0867	0,2089	0,1270	0,1029	0,0881	0,0762	0,0869	0,0926
	ROUGE-2	0,0170	0,0934	0,0517	0,0310	0,0206	0,0168	0,0277	0,0172
	ROUGE-L	0,0781	**0,1638**	**0,1117**	0,0895	0,0785	0,0711	0,0785	0,0825
Modified Diff-BERT with an increasing step	ROUGE-1	0,0901	0,1769	0,1090	0,0981	0,0980	0,0818	0,0936	0,0925
	ROUGE-2	0,0272	0,0537	0,0391	0,0281	0,0313	0,0254	0,0302	0,0197
	ROUGE-L	**0,0861**	0,1481	0,1017	**0,0916**	**0,0913**	**0,0772**	**0,0874**	**0,0861**

TABLE I: Both Modified Diff-BERT models outperform SDPEs in every domain, with Modified Diff-BERT with an increasing step having higher ROUGE-L scores than Modified Diff-BERT in 6 out of 8 domains.

Base Model	Steps	Loss	Clamp	Metric	Calendar	Basketball	Blocks	Housing	Publication	Recipes	Restaurants	Social Network
bert-tiny	100	MSE	No	ROUGE-1	0,0628	0,0813	0,0831	0,0676	0,0699	0,0577	0,0715	0,0687
				ROUGE-2	**0,0047**	0,0005	0,0123	0,0062	**0,0078**	**0,0037**	0,0108	**0,0074**
				ROUGE-L	0,0587	0,0715	0,0775	0,0632	0,0650	0,0539	0,0673	0,0649
bert-tiny	170	MSE	No	ROUGE-1	0,0678	0,0842	0,0885	0,0704	0,0737	0,0590	0,0765	0,0720
				ROUGE-2	0,0045	0,0007	0,0118	0,0057	0,0065	0,0033	0,0101	0,0073
				ROUGE-L	0,0603	0,0724	0,0801	0,0642	0,0672	0,0540	0,0691	0,0670
bert-tiny	250	MSE	No	ROUGE-1	**0,0733**	0,0919	**0,0983**	**0,0781**	**0,0825**	**0,0675**	**0,0843**	**0,0804**
				ROUGE-2	0,0042	0,0004	0,0119	**0,0063**	0,0054	0,0033	0,0094	0,0067
				ROUGE-L	**0,0658**	**0,0783**	**0,0873**	**0,0684**	**0,0746**	**0,0599**	**0,0750**	**0,0733**
bert-tiny	500	MSE	No	ROUGE-1	0,0641	0,0818	0,0850	0,0713	0,0729	0,0581	0,0766	0,0707
				ROUGE-2	0,0041	0,0004	**0,0131**	0,0061	0,0077	0,0034	**0,0109**	0,0073
				ROUGE-L	0,0597	0,0713	0,0788	0,0649	0,0673	0,0540	0,0699	0,0661
bert-tiny	500	MSE	Yes	ROUGE-1	0,0471	0,0690	0,0684	0,0515	0,0530	0,0391	0,0580	0,0533
				ROUGE-2	0,0045	**0,0011**	0,0091	0,0056	0,0064	0,0031	0,0076	0,0059
				ROUGE-L	0,0444	0,0620	0,0629	0,0483	0,0505	0,0369	0,0540	0,0512
bert-mini	500	MSE	Yes	ROUGE-1	0,0683	**0,0978**	0,0826	0,0686	0,0693	0,0590	0,0721	0,0039
				ROUGE-2	0,0037	0,0006	0,0066	0,0032	0,0035	0,0022	0,0045	**0,0074**
				ROUGE-L	0,0571	0,0765	0,0700	0,0581	0,0575	0,0495	0,0603	0,0625
bert-tiny	500	cross-entropy	Yes	ROUGE-1	0,0115	0,0043	0,0165	0,0150	0,0137	0,0078	0,0203	0,0073
				ROUGE-2	0,0000	0,0000	0,0007	0,0003	0,0000	0,0003	0,0010	0,0000
				ROUGE-L	0,0115	0,0041	0,0159	0,0147	0,0133	0,0075	0,0198	0,0071

TABLE II: Variations of SDPE using different base BERT models, with different number of diffusion steps, different loss components and with or without clamping.

Modified Diff-BERT models can take advantage of the pre-trained BERT base model better than SDPE as the inputs of two Modified Diff-BERT models contain richer information about the target output, which allows a shorter time in fine-tuning two BERT models. As diffusion models require a long training time to produce high-quality samples, SDPE models that lean toward the diffusion nature more while only making use of the pre-trained BERT's embeddings do not perform well in only 2000 training steps. The Modified Diff-BERT model with an increasing step tends to perform better than the Modified Diff-BERT as its process is more similar to the forward diffusion process.

Figure 5 shows that the Modified Diff-BERT's ROUGE-L score surges quickly and reaches its peak in the first 200-300 training steps. That number then goes down and tends to keep decreasing for the rest of the training process. On the contrary, all the SDPE variations' ROUGE-L scores increase much slower, yet keep rising throughout the entire training process. As the training time increases, it is likely that SDPE models would outperform Modified Diff-BERT models.

B. Accuracy tradeoffs for different numbers of diffusion steps

Table II summarizes our results on different variations of SDPE. Among four models using the same bert-tiny base model, using the original SDPE loss and without the clamping method, SDPE model with 250 diffusion steps (SDPE-250) achieves the highest ROUGE-1 and ROUGE-L scores on all 8 domains. Figure 5 shows that after the first 1000 training steps, SDPE-250 performs significantly better than all three other models with 100, 170 and 500 diffusion steps. SDPE-100, SDPE-170 and SDPE-500 all produce similar ROUGE-L results throughout the training process.

However, SDPE has the lowest ROUGE-2 scores on 5 out of 8 domains. Despite using the lowest number of diffusion steps, SDPE-100 produces the highest scores on 4 domains, with its ROUGE-2 scores on two out of the remaining four, namely Housing and Restaurants, being comparable to the highest scores.

In summary, these results show that SDPE-250 is most capable of capturing unigrams and high n-gram structures, while SDPE-100 performs best in capturing bigram structures.

C. Effects of the Clamping method on SDPEs' performances

The SDPE model with the bert-tiny base model, using the original SDPE loss and 500 diffusion steps that does not use the clamping method has higher ROUGE-1 and ROUGE-L scores than its correspondent with the clamping method on every domain. The SDPE variation without the clamping method also produces higher ROUGE-2 scores on 6 out of 8 domains.

The first component of SDPE's loss function minimizes the distances between each latent word embedding and the target word embedding. When applying the clamping method to SDPE models, this component is calculated after the latent word data is clamped using the BERT embedding layers. However, every BERT embedding layer is frozen for the sake of faster computing, which makes gradients can not be tracked by the optimizer of the first loss component. Thus, applying the clamping method to SDPE does not produce the same effect as Diffusion-LM [1], and results in a worse performance.

D. Effects of different base BERT models' sizes on SDPEs' performances

As we experiment with the SDPE-500 variations with the original loss and clamping method applied, we observe significant increases in ROUGE-1 and ROUGE-L scores on every domain when changing the base BERT model from bert-tiny to bert-mini. However, we also notice decreases in ROUGE-2 scores on 7 out of 8 domains.

As the size of the base BERT model increases, the frozen embedding layers allow for performing higher dimensional word embeddings. The higher dimensional word embeddings contain richer information about the target outputs, for which reason SDPEs can capture the unigrams and higher n-gram structures more precisely. As the sizes of the BERT encoding layers also increase, they need a longer training time to learn the denoising process, which might be the reason for the worse performance in capturing bigram structures.

E. MSE vs cross-entropy loss for SDPEs

We test the effectiveness of our loss function by replacing the MSE losses used in the first two components of our loss function with the cross-entropy losses. As shown in Table II, this replacement results in a 4-15 times reduction in ROUGE-1 and ROUGE-L scores, while the new SDPE variation is unable to capture bigram structures on 4 out of 8 domains.

VI. Discussion

We successfully demonstrated the potential of continuous diffusion language models in performing sequence-to-sequence style transferring tasks, at the same time showed that taking advantage of pre-trained BERT models allows a faster training process. The effectiveness of the pre-trained BERT's word embeddings could be seen in the swift increase of Modified Diff-BERT's ROUGE-L scores at the beginning of the training process and the significant outperformance of SDPE variation with bert-mini on several domains in terms of ROUGE-1 and ROUGE-L scores compared to its correspondent that used bert-tiny. The potential of continuous diffusion language models was shown in section V-A when analyzing the growth of SDPEs' ROUGE-L scores throughout the training process and when comparing the Modified Diff-BERT with an increasing diffusion step with the Modified Diff-BERT.

Our results affirmed the effectiveness of the method used by DDIM [29], the re-parametrization trick, and the loss function proposed by Li et al. [1]. We also reached the same conclusion as Jia [3] about the effectiveness of pre-trained BERT models in enhancing the training process of continuous diffusion language models. However, in contrast to the remark of Li et al. [1] about continuous diffusion language models performing better as the numbers of diffusion steps increase, we showed that when being trained in a low-resource environment with the enhancement of pre-trained BERT embeddings, continuous diffusion language models with a high number of diffusion steps might not perform as well as their correspondents using a lower number of diffusion steps.

Due to the limitations of available computing resources, we could not conduct a comprehensive study on the capability of our diffusion language models on sequence-to-sequence style transferring tasks. Hence, we do not guarantee that the conclusions and insights gained in our works still apply to continuous diffusion language models trained in a high-resource setting. Further work is necessary to reach a conclusion about the capability of continuous diffusion language models compared to autoregressive language models.

VII. Conclusion

To evaluate the performances of continuous diffusion language models on sequence-to-sequence tasks and assess the effects of pre-trained BERT models in enhancing the training process of the diffusion models, we introduce SDPE, a continuous diffusion language model for style transferring that takes advantage of a pre-trained BERT model for word embedding, at the same time modify the diffusion model proposed by Jia [3] and train them on the Overnight dataset [9] in a low-resource environment. We are able to demonstrate the potential of continuous diffusion language models in performing sequence-to-sequence style transferring tasks and show that taking advantage of pre-trained BERT models allows a faster training process.

As diffusion is still a new direction for text generation, further research is still needed to be performed to prove its capabilities against autoregressive language models. The enhancement to the diffusion training process by using a pre-trained BERT model for word embedding shown in this work can be extended and applied to other diffusion architectures. Since the remarkable performance of the large autoregressive language models comes from their massive size and the fact that they are trained using a large amount of data in an extensive amount of time, a good research direction for future works might be designing a pre-training and fine-tuning paradigm for diffusion models, such that new controls can be applied in the fine-tuning process. Further optimizations and simplifications in the diffusion process can also be made in future works to address the inherent issue of diffusion models, which is the long training and inference time.

References

[1] Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., & Hashimoto, T. B. (2022). Diffusion-LM Improves Controllable Text Generation. arXiv. https://doi.org/10.48550/arXiv.2205.14217

[2] Shin, R., Lin, C. H., Thomson, S., Chen, C., Roy, S., Platanios, E. A., Pauls, A., Klein, D., Eisner, J., & Van Durme, B. (2021). Constrained Language Models Yield Few-Shot Semantic Parsers. arXiv. https://doi.org/10.48550/arXiv.2104.08768

[3] Yinjun Harold Jia, Diffusion-LM, (2022), GitHub repository, https://github.com/EBGU/Diffusion-LM

[4] Schucher, N., Reddy, S., & de Vries, H. (2021). The Power of Prompt Tuning for Low-Resource Semantic Parsing. arXiv. https://doi.org/10.48550/arXiv.2110.08525

[5] Shin, R., & Van Durme, B. (2021). Few-Shot Semantic Parsing with Language Models Trained On Code. arXiv. https://doi.org/10.48550/arXiv.2112.08696

[6] Berant, J., & Liang, P. (2014). Semantic Parsing via Paraphrasing. ACL.

[7] Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv. https://doi.org/10.48550/arXiv.1312.6114

[8] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. arXiv. https://doi.org/10.48550/arXiv.2002.08155

[9] Wang, Y., Berant, J., & Liang, P. (2015). Building a Semantic Parser Overnight. ACL.

[10] Machines, S., Andreas, J., Bufe, J., Burkett, D., Chen, C., Clausman, J., Crawford, J., Crim, K., DeLoach, J., Dorner, L., Eisner, J., Fang, H., Guo, A., Hall, D., Hayes, K., Hill, K., Ho, D., Iwaszuk, W., Jha, S., . . . Zotov, A. (2020). Task-Oriented Dialogue as Dataflow Synthesis. arXiv. https://doi.org/10.1162/tacl_a_00333

[11] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., & Berg, R. V. (2021). Structured Denoising Diffusion Models in Discrete State-Spaces. arXiv. https://doi.org/10.48550/arXiv.2107.03006

[12] Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., & Welling, M. (2021). Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions. arXiv. https://doi.org/10.48550/arXiv.2102.05379

[13] Hoogeboom, E., Gritsenko, A. A., Bastings, J., Poole, B., Berg, R. V., & Salimans, T. (2021). Autoregressive Diffusion Models. arXiv. https://doi.org/10.48550/arXiv.2110.02037

[14] Lester, B., & Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. arXiv. https://doi.org/10.48550/arXiv.2104.08691

[15] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv. https://doi.org/10.48550/arXiv.2005.14165

[16] Yang, K., & Klein, D. (2021). FUDGE: Controlled Text Generation With Future Discriminators. arXiv. https://doi.org/10.18653/v1/2021.naacl-main.276

[17] Krause, B., Gotmare, A. D., McCann, B., Keskar, N. S., Joty, S., Socher, R., & Rajani, N. F. (2020). GeDi: Generative Discriminator Guided Sequence Generation. arXiv. https://doi.org/10.48550/arXiv.2009.06367

[18] Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., & Choi, Y. (2021). DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. arXiv. https://doi.org/10.48550/arXiv.2105.03023

[19] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv. https://doi.org/10.48550/arXiv.1810.04805

[20] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv. https://doi.org/10.48550/arXiv.1910.10683

[21] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., . . . Fiedel, N. (2022). PaLM: Scaling Language Modeling with Pathways. arXiv. https://doi.org/10.48550/arXiv.2204.02311

[22] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv. https://doi.org/10.48550/arXiv.1910.13461

[23] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., . . . Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. arXiv. https://doi.org/10.48550/arXiv.2107.03374

[24] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. arXiv. https://doi.org/10.48550/arXiv.2102.12092

[25] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. arXiv. https://doi.org/10.48550/arXiv.2112.10752

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. arXiv. https://doi.org/10.48550/arXiv.1706.03762

[27] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv. https://doi.org/10.48550/arXiv.1907.11692

[28] Clark, K., Luong, M., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. arXiv. https://doi.org/10.48550/arXiv.2003.10555

[29] Song, J., Meng, C., & Ermon, S. (2020). Denoising Diffusion Implicit Models. arXiv. https://doi.org/10.48550/arXiv.2010.02502