Latent Diffusion Language Models

Samuel Dolman

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2023

Abstract

Diffusion language models bring together the concept of diffusion - a form of generative AI based on sequential denoising, commonly used for image generation - into the domain of language models, unlocking the ability for global control over the generated text. The challenge of combining the two lies in the inherent discrete nature of natural language (when expressed as a sequence of tokens), whereas diffusion requires gradient computations on continuous noise perturbations. In this project I investigate one approach for implementing diffusion-LMs for controllable text generation: adding perturbation noise (and running the diffusion process) in a continuous latent space obtained via a pre-trained encoder/decoder network, rather than in the discrete sentence space.

Specifically, I extend existing work by showing that the choice of encoder LM is vital: increasing the model size/capacity of the pre-trained encoder LM has a greater impact on generated text fluency, than varying the size of the learnt denoising transformer network. Furthermore, replacing with an alternative pre-trained LM of similar capacity (e.g. BART vs T5) has a significant impact. This suggests that fine-tuning of the encoder LM alongside the diffusion training process should improve results further. 'Low Rank Adaptive' (LoRA) fine-tuning was implemented for this purpose, but failed to improve experimental results for either fluency or control.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Samuel Dolman*)

Table of Contents

Chapter 1

Introduction

During recent years, the accepted paradigm for training language models for text generation has been to train large, transformer based auto-regressive neural networks with a masking/next word prediction objective. Such models have essentially solved the problem of generating fluent, high quality text (Brown et al. (2020), Chowdhery et al. (2022), Vaswani et al. (2017)). *Controllable* text generation, where the output must satisfy some user-defined restrictions (e.g. on semantic content, sentiment, or syntactic structure), is an essential next step for deploying any real world generative NLP system.

This problem has received significant recent attention in the literature: existing approaches usually involve either further fine-tuning of the model (Keskar et al. (2019)), or appending smaller 'plug and play' classification models to steer generation during test time (Dathathri et al. (2020)). However fine-tuning approaches are often prohibitively compute-intensive, whilst plug and play steering is successful for relatively simple control commands only (e.g. sentiment).

An alternative approach to controllable text generation may lie with diffusion (Ho et al. (2020)) based models. Here, output is generated in a non-autoregressive way; beginning with random noise in some feature space and performing sequential denoising steps, culminating with a representation of the final generated output (learning the denoising network is the main objective during model training). Each denoising step can also be conditioned on whatever classifier we desire, allowing flexible and powerful control during sentence generation. Note that this conditioning is global rather than left-to-right, allowing complex controls such as target syntactic structure.

After initial success in the image domain (Nichol and Dhariwal (2021), Rombach et al. (2022)), where output representations are fully continuous, Li et al. (2022) demonstrated that successfully applying denoising diffusion methods to the discrete domain of text generation was also possible. Recently, a variety of differing approaches (outlined in Section 2.3) have been proposed to train improved diffusion language models.

In this dissertation, we base our experiments on the approach taken by Lovelace et al. (2022). Here, the authors trained a diffusion model within the latent space of an existing (pre-trained) encoder/decoder language model, achieving improved text fluency (MAUVE score / perplexity) compared to the original approach in Li et al. (2022). We run experiments to confirm that the choice of pre-trained LM is at least as important as the architecture of the diffusion transformer itself, via varying the size of the pre-trained LM and diffusion transformer independently. An alternative pre-trained model (T5, Raffel et al. (2020)) is also tested in place of the BART (Lewis et al. (2020)) model used in Lovelace et al. (2022). Note that training data size ($\sim$100k samples) and model architecture ($\sim$100M parameters) will be kept modest in comparison to SOTA, in order to ensure models can easily be trained using the available computational resources.

Our experiments are evaluated via a range of metrics measuring the fluency and relevance of the generated text on both an unconditional task (generating restaurant reviews based on the E2E dataset), and conditional generation task (generating either positive or negative film reviews using the SST dataset). For the conditional task, we additionally measure how well our model output adheres to the class condition (see Section 3.4 for further details).

Furthermore, in an attempt to improve upon the results in Lovelace et al. (2022), we fine-tune the encoder/decoder network as part of the training process. As a full fine-tuning process (over all layers) would be prohibitively expensive in terms of computation, we employ the LoRA technique (Hu et al. (2021)). This has the benefit of adding relatively little computational overhead during training, whilst adding *no* overhead during inference (in Section 4.4 we present average inference times across all our experiments). Unfortunately, our results here were inconclusive - Section 5.1 suggests some possible reasons for this.

The remainder of this report is set out as follows:

Throughout Section 2, we review the existing literature on controllable language models, the denoising diffusion method, and alternate approaches for training diffusion language models. In Section 3 we explain in detail our model architecture (as based upon Lovelace et al. (2022)), and describe our experimental setup. Results are discussed in Section 4, before conclusions and possible further work are outlined in Section 5. Samples of generated text output from our models can be found in Appendix A.3.

Chapter 2

Background

In this section we outline the main concepts required to train our latent diffusion language models, and provide motivation for our experiments based on existing literature.

2.1 Controllable language models

Existing approaches for controlling text generation for auto-regressive language models include additional supervised fine-tuning on (control, text) sentence pairs, such as in Keskar et al., 2019. Whilst successful, this approach has the limitation that significant effort must be expended to re-tune the model every time a new control is required. Furthermore, simultaneous composition of multiple controls is impossible.

The current state of the art is obtained via reinforcement learning through human feedback (RLHF), coupled with few-shot prompting (provided by the user), as seen in InstructGPT (Ouyang et al. (2022)) and ChatGPT. The downside here is an extremely extensive and expensive training process, which is unavailable to most researchers.

An alternative approach is to freeze the underlying LM itself, and apply a separate 'plug and play' guidance classifier during generation (Dathathri et al. (2020)). Such classifiers are much easier to train (orders of magnitude smaller than the underlying LM)[1], and several can be applied in parallel to achieve fully compositional control. For example, it is possible to simultaneously condition for semantic content, sentiment, and syntactic structure.

This approach represents an excellent trade-off between performance and training expense, and the conditional generation method used in this project is based upon it.

[1]in fact, depending on architecture, an existing pre-trained text classifier can often be substituted - removing the requirement to train a custom one each time

The main limitation is that when applied in an auto-regressive setting, the left-to-right conditioning precludes complex constraints from being imposed (such as sentence parse tree, which requires both left- and right-context). This constraint is removed when we instead work within a non-auto-regressive setting, such as diffusion based approaches. This is a primary motivation for developing diffusion based language models.

2.2 Denoising diffusion models

In order to describe diffusion based language models, we first provide a brief overview of the underling diffusion process itself:

Denoising diffusion probabilistic models were introduced in Ho et al. (2020) as a parameterized Markov chain, trained using variational inference. During training, the *forward diffusion process* is defined by sequentially adding Gaussian noise β_t to the original training samples x_0 over a series of T timesteps. For example if the original training sample x_0 is an image, then $x_1, ..., x_T$ represent a series of progressively noisy images culminating in an image indistinguishable from random static - see Figure 2.1.

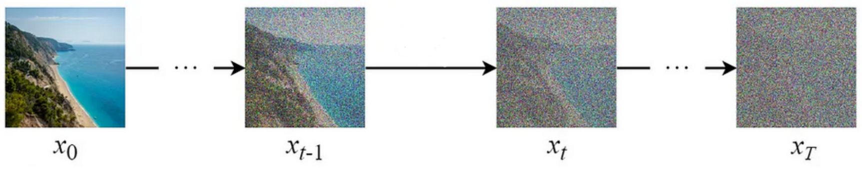


Figure 2.1: example forward diffusion process, taken from https://medium.com/@steinsfu/diffusion-model-clearly-explained-cd331bd41166

Whilst the training examples are altered in this way, a separate 'denoising' autoencoder network $p_\theta(x_{t-1}|x_t)$ is trained with the objective of reversing this process and recovering the original data distribution. This network usually has a transformer architecture. The learned parameters θ define the mean of a Gaussian transformation, which represents the *reverse diffusion process*. At test time, pure Gaussian noise is fed through the denoising network in order to reconstruct a generated image which approximates the training distribution.

Various improvements upon this basic approach have since been proposed: cosine scheduling of the noise parameter variance β_t (rather than constant or linear scheduling), and learning both the mean and variance for the backwards denoising process (rather than just the mean), were both introduced in Nichol and Dhariwal (2021).

A further breakthrough was to perform the diffusion process in a lower-dimensional latent space, rather than the original pixel space. This was used successfully in the famous stable diffusion paper (Rombach et al. (2022)), to generate higher quality images than previous (non-diffusion) approaches such as CLIP (Radford et al. (2021)). This approach also allows for classifier guidance, enabling fine-grained control over the generated image (for example by following a text prompt).

2.3 Diffusion language models

As discussed above, diffusion models have achieved great success in continuous domains such as image generation. The main hurdle to adapting the same approach to text generation is the discrete nature of language tokens: the requirement to backpropagate a loss gradient is a limitation when working in discrete space. Two broad classes of solution have emerged:

2.3.1 Discrete approaches

Some attempts have been made to run the language diffusion process directly as a sequence of discrete sentence edits. In Hoogeboom et al. (2021), the concepts of *Argmax Flow* (composition of continuous noise prior to an argmax function to transform into categorical space) and *Multinomial Diffusion* (adding categorical noise) are introduced. Both can be used to perform diffusion on categorical data.

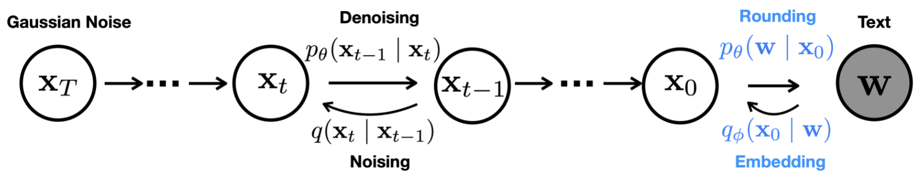
Discrete Denoising Diffusion Probabilistic Models (D3PMs) were introduced in Austin et al. (2021). Here, discrete sentence corruptions are based on transition matrices designed to mimic Gaussian kernels in continuous space.

More recently, DiffusER (Reid et al. (2023)) is a method to train diffusion LMs via representing noise perturbations as discrete levenshtein edits on the original sentence. One advantage is that the resulting model has the ability to revise existing text, as well as generating new sentences (by conditioning on a prototype example).

In addition to these initial forays into diffusion-LMs, the literature contains general techniques developed for estimating gradients of discrete functions. Examples include discrete stein operators (Shi et al. (2022)), and Implicit MLE (Niepert et al. (2021), Minervini et al. (2023)). The application of such techniques to diffusion language models is one potentially fruitful area for further research.

2.3.2 Continuous approaches

One way to (partially) side-step the problem of back-propagating a discrete loss function, is to repeat the trick from Rombach et al. (2022) and perform the diffusion in a continuous latent space (word embeddings). The first such implementation was described in Li et al. (2022).



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

Figure 2.2: Illustration of the forward and reverse diffusion process used in *Diffusion-LM*, taken from Li et al. (2022). All variables $\mathbf{x}_i$ lie in the continuous latent/embedding space.

As shown in Figure 1, the diffusion/denoising process is performed on sentence representations $\mathbf{x}_0 \in \mathbb{R}^{nd}$, where n is the number of words in the sentence and d is the embedding dimension. These sentence representations are formed as concatenations of continuous word embedding vectors: $\mathbf{x}_0 = \text{EMB}(\mathbf{w}) = [\text{EMB}(w_1), \ldots, \text{EMB}(w_n)]$, where $\text{EMB}(w_i) \in \mathbb{R}^d$ is an embedding mapping allowing translation between the discrete sentence space and latent space.

During the forward diffusion process, a 'rounding'/argmax operation is carried out to obtain the closest sentence for a given series of latent word embeddings. As the majority of the diffusion process occurs in continuous space, we need only to employ discrete gradient estimators to this final step during backpropagation of the loss.

The promise of this approach was later confirmed in Gulrajani and Hashimoto (2023), which trained a likelihood-based diffusion language model with a substantial compute budget called *Plaid 1B*. This model was shown to outperform a version of GPT2 (Radford et al. (2019)) in terms of zero-shot likelihood on common benchmarks, the first time a diffusion language model had beaten a large, widely-adopted autoregressive model in this way.

With this approach, the choice of embedding mapping is vital: both papers showed that using a fixed pre-trained embedding (such as word2vec, Mikolov et al. (2013)) performs relatively poorly. Instead, learning a custom embedding during the training process was significantly better: achieving NLL of 3.89 versus 4.54 for using a pre-trained embedding, according to one ablation study in Gulrajani and Hashimoto (2023). This observation suggests that focusing further on the embedding part of the architecture, and not just the denoising transformer itself, is worthwhile.

Note that the embeddings described in this section are relatively simple *static* word embeddings, i.e. that $\text{EMB}(w_i)$ is independent of all w_j, for $j \neq i$. In Lovelace et al., 2022, this assumption is relaxed: diffusion is carried out in the latent space of a pre-trained *encoder/decoder* model (BART, Lewis et al. (2020)). This brings two benefits: a more richly defined set of contextual embeddings based on the current state-of-the-art in auto-regressive language transformers, and an easy mechanism (the pre-trained decoder) for decoding latent samples into natural language.

One may then ask: if learning our own embeddings is better than using pre-trained static embeddings, and if using a modern encoder/decoder model is better than using word2vec, would even better performance be possible if we trained our own encoder/decoder network alongside the denoising transformer? These observations motivate the main experiments of this dissertation: we will use the architecture from Lovelace et al., 2022 as a baseline, and finetune the encoder network during training. The methodology is more fully described in the following Section 3, but first we review one more technique that we will need:

2.4 LoRA

Learning a set of static word embeddings (as in Li et al. (2022)) is relatively simple compared to the main task of training the denoising transformer for the diffusion process, due to the relatively low parameter count. However in our case, we wish to finetune BART-base which contains $\sim$140M parameters. This is at least as large as the denoising transformer itself (across all of our experiments), making a full fine-tune prohibitively computationally expensive. Furthermore, as the BART encoder appears earlier in the overall architecture (see Figure 3.1), vanishing gradients may prove problematic.

Our chosen solution to avoid finetuning the entire encoder network is to employ *Low Rank Adaptive finetuning* (LoRA, Hu et al. (2021)). As shown in Figure 2.3, LoRA fixes each block of original transformer weights W whilst introducing two new weight matrices A and B alongside it. A downscales the input onto dimension r (a hyperparameter), before B projects back into the original output dimension and combines additively with the original output of W. Only A and B are trained: the claim is that these small residual layers have sufficient capacity to encompass the weight updates required for effective fine-tuning.

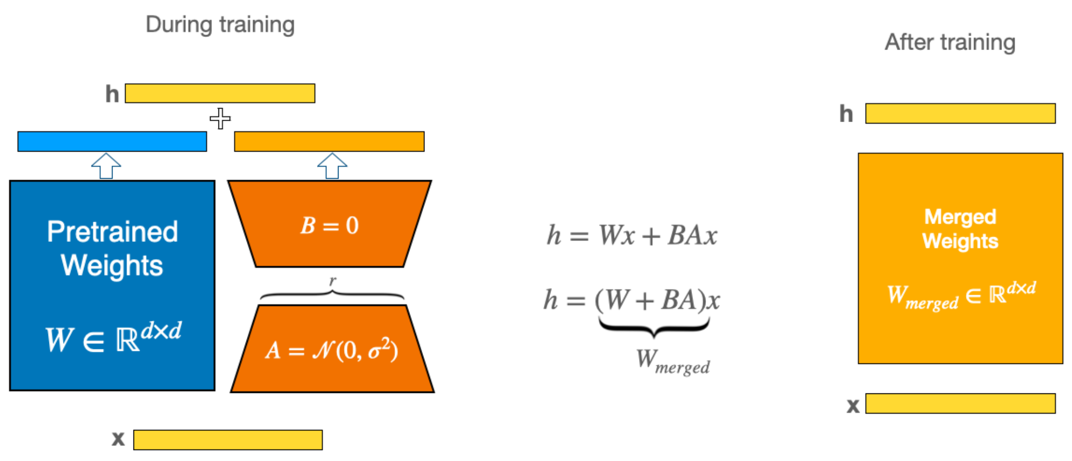

During training
After training
h
x
Pretrained Weights
$W \in \mathbb{R}^{d \times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
$h = Wx + BAx$
$h = \underbrace{(W + BA)}_{W_{merged}}x$
Merged Weights
$W_{merged} \in \mathbb{R}^{d \times d}$

Figure 2.3: Illustration of LoRA finetuning during training versus inference. Taken from https://huggingface.co/docs/peft/conceptual_guides/lora

As we can pick dimension $r \ll d$, only a small fraction of the original parameter count needs to be trained[2]. This results in minimal computational overhead during training. Furthermore, as BA can be merged into the original weights W when storing the final model, there is a complete lack of additional inference latency. This stands in contrast to some alternative fine-tuning techniques, which append additional tuning layers on top of the frozen model architecture.

In Hu et al. (2021), it is shown that LoRA performs at least on-par with a full fine-tune of RoBERTa, GPT-2 and GPT-3. These properties make the technique perfect for our purposes.

[2]in our experiments, we pick $r = 8$. Correspondingly, less than 1% of the original total parameter count are available for finetuning

Chapter 3

Methodology

In this section we describe in greater detail our model architecture (adapted from Lovelace et al. (2022)) and training setup, including the datasets and hyperparameter settings used for our experiments.

3.1 Baseline model

As described in the previous section, we will base our experiments on the architecture described in Lovelace et al., 2022. The authors trained their models on four datasets in total - we will restrict ourselves to the smallest two of these, one for unconditional generation (E2E) and one for conditional generation (SST). The first task is to re-implement their baseline model on these two datasets.

Whilst their model is relatively lightweight by modern standards (a transformer based architecture with $\sim$214M parameters), available GPU hardware was limited to NVidia GTX-1080Ti cards (with 11GB VRAM) rather than the RTX-3090 and A6000 (24GB / 48GB VRAM) used in their paper. Consequently, we slightly truncate the architecture to use as our primary baseline - see Section 3.3 for details. The metrics used to evaluate our models for fluency and control are detailed in Section 3.4.

The original PyTorch code implementation is freely available on github [1], which I have forked and adapted [2]. The overall approach is shown in Figure 3.1.

[1] https://github.com/justinlovelace/latent-diffusion-for-language

[2] https://github.com/dolmas1/dissertation_public

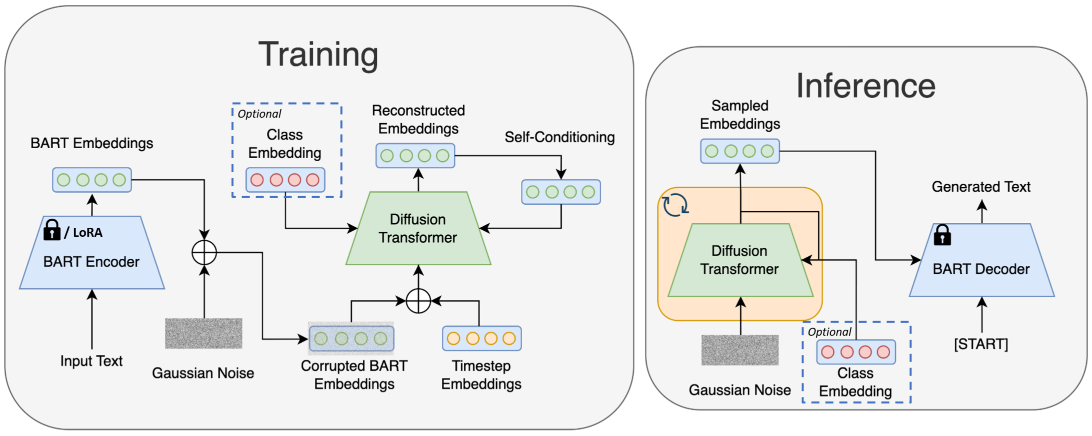


Figure 3.1: Outline of approach for latent diffusion for language generation (adapted from Lovelace et al., 2022)

During model **training**, we sample natural language $\mathbf{w}$ of length l from our training distribution $\mathcal{D}$, and represent it as a sequence of one-hot vectors over our vocabulary, $\mathbf{w} \in \mathbb{R}^{l \times \mathcal{V}}$. The BART encoder, $E()$, transforms this sequence into continuous latent space: $\mathbf{x} = E(\mathbf{w}) \in \mathbb{R}^{l \times d}$.

Next, sample a diffusion timestep $t \in \{1, ..., T\}$ and some Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. We obtain the corrupted latent embedding $\mathbf{z}_t$ by applying the noise ϵ to our original latent embedding $\mathbf{x}$, according to $\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{x} + \sqrt{1 - \alpha_t}\epsilon$.
Here, $\alpha_t \in (0, 1)$ controls the amount of noise for diffusion timestep t according to our chosen noise schedule.

The corrupted embedding $\mathbf{z}_t$, along with the timestep t and (optional) class embedding y, are fed into our denoising network $\hat{\mathbf{x}}_\theta$ (labelled *diffusion transformer* in Figure 3.1). We train this denoising network against the loss function:

$$L = ||\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t, y) - \mathbf{x}||_2^2 \tag{3.1}$$

During **inference**, the denoising network recovers embeddings from random noise, which are then decoded into text by the BART decoder. First, we must determine the length of the sample to generate - during training, l was specified by the training sample but during inference it must be predetermined. The chosen approach is to sample l_i from the empirical distribution of sentence lengths from the training set.

This allows us to sample a fully corrupted embedding, $\mathbf{z}_T \in \mathbb{R}^{l_i \times d} \sim \mathcal{N}(0, \mathbf{I})$.

An iterative process is then performed across $t \in \{T, ..., 1\}$: the trained denoising network $\hat{\mathbf{x}}_\theta$ is used to first denoise the current latent $\mathbf{z}_t$ into $\tilde{\mathbf{x}}_t$, before the latent corresponding to timestep $t-1$, $\mathbf{z}_{t-1}$, is reconstructed via use of $\tilde{\mathbf{x}}_t$, $\mathbf{z}_t$, and α_t.

Once $\tilde{\mathbf{x}}_1$ has been reached, it is passed through the BART decoder to obtain the final decoded text output. Beam search (with a beam size of 4) is used at this point.

Note that this iterative process acts as a bottleneck, and is the main reason why inference for diffusion based LMs is substantially slower than for standard autoregressive LMs.

3.1.1 Diffusion transformer architecture

As outlined in Lovelace et al. (2022), the diffusion transformer $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t, y)$ is a bidirectional Pre-LN transformer (Xiong et al. (2020)) with GeGLU activation functions (Shazeer (2020)). The latent $\mathbf{z}_t$ is projected to the input dimension of the transformer, passed through the transformer, processed with a LayerNorm, and finally passed through a linear layer to obtain the denoised latent.

Default sizes for the transformer itself are 12 layers with hidden dimension $d_{tx} = 768$; however in this dissertation we will instead be testing a range of smaller sizes (partly due to memory limitations, but also to gauge the importance of diffusion transformer size in comparison to encoder/decoder LM size).

The timestep is fed into the transformer in an identical manner as for image diffusion: via a sinusoidal positional encoding $\psi(t)$ passed through a learnable MLP, to obtain a time embedding $MLP(\psi(t)) \in \mathbb{R}^{d_{tx}}$.

3.1.2 Conditional generation

If conditional generation is required for a specific task (e.g. topic or sentiment conditioning), the ground truth class labels from our training examples are directly encoded and fed into the diffusion transformer during model training (see ‘class embedding’ component in figure 3.1). For class labels $c \in \{1, 2, ..., C, \emptyset\}$ - including the null class - this is done via learnable class embeddings $y(c) \in \mathbb{R}^{(C+1) \times d_{tx}}$. These class embeddings are combined with the latent language embeddings $\mathbf{z}_t$ via a cross-attention layer in the denoising network.

In this way, the model learns how to generate examples from each class present in the training set: during inference, simply pass the appropriate class embedding $y(c)$ into the denoising network. To retain the ability for unconditional generation, for ten percent of training updates we randomly mask the class label with $\emptyset$. This empty class label can be passed during inference when no class conditioning is required.

Whilst this framework differs slightly from the classification-model guided approach in Li et al. (2022), it can still be adapted for scenarios where we have access to a classifier but not ground truth labels. For example, one could use an existing sentiment classification model to classify an unlabelled set of training sentences, and pass the encoded prediction to the diffusion transformer in place of discrete class label. During inference, we could also specify an arbitrary value for the classifier output rather than a discrete label (assuming the label encoder was modified accordingly).

3.1.3 Noise schedule

$T = 1000$ diffusion timesteps are used for every model. The noise schedule α_t is parameterized according to linear schedule introduced by Ho et al. (2020):

α_1 = 1e-4 to α_T = 0.02.

Such a schedule 'ensures that the noise scale is relatively small at the beginning, making it easier for the denoising network to recover the data, while eventually corrupting the original data to random noise by increasing the noise scale'.

3.1.4 Objective

Note that in the image domain, diffusion transformers are often parameterized to predict the noise: $\hat{\varepsilon}_\theta(\mathbf{z}_t, t)$. But our diffusion transformer $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ has instead been reparameterized with the objective of recreating an estimate of the original (noiseless) latent embedding - see our loss function, Equation 3.1. This is commonly denoted as '$\mathbf{x}_0$-parameterized loss'.

According to the survey paper Li et al. (2023), this prevents instability where the network would otherwise fail to converge onto precise word embeddings, and has been widely adopted in recent work.

3.1.5 Self-conditioning

The technique of *self-conditioning* is the final adjustment to the method described above. Introduced by Chen et al. (2023), it was found that improved sample quality could be obtained by additionally conditioning the denoising network on its output from the previous timestep: instead of $\tilde{\mathbf{x}}_t = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t, y)$, we have $\tilde{\mathbf{x}}_t = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t, y, \tilde{\mathbf{x}}_{t+1})$.

In Lovelace et al. (2022), experiments showed that the perplexity of the generated text was substantially improved when self-conditioning was used. Furthermore, no additional calls to the denoising network are required during inference due to the iterative nature of the sampling procedure. The only tricky detail is that for the very first denoising step during inference, no previous latent exists - a slight modification to the training procedure is therefore required, to retain the ability for the denoising network to condition only upon $(\mathbf{z}_t, t, y)$.

To allow this, during training we randomly replace the conditioning on $\tilde{\mathbf{x}}_{t+1}$ with a null latent embedding $\emptyset$ half of the time.

3.2 Datasets

We will train Diffusion-LMs for two different tasks: our first task will be to (unconditionally) generate sentences of restaurant reviews. The second task is conditional generation, where we will learn to generate either positive or negative movie reviews. Each task has an associated training dataset of sample sentences, which our model will learn to emulate.

E2E Novikova et al. (2017), dataset containing 51,426 restaurant reviews. Each review describes between 3 and 8 attributes (e.g. price range, location, type of cuisine etc). The data is split into 42,061 training instances, 4,672 validation, and 4,693 test instances. The dataset is often used for attribute detection tasks, but we use it for unconditional generation only - by design, the wide range of attributes contained in the data ensures sufficient diversity (within the domain of restaurant reviews). An example training sentence is as follows:

The Phoenix is located in the city centre. It offers English food, has a high price range, and a customer rating of one out of five.

Stanford Sentiment Treebank Socher et al. (2013) contains 11,855 sentences extracted from movie reviews, taken from rottentomatoes.com. Each review was originally labelled (via Amazon Mechanical Turk) on a five point scale. However, we use a version of the data where the labels have been simplified into binary ‘positive’ or ‘negative’ sentiment only. These binary labels are fed into the diffusion transformer as the class embedding, along with the sentence embeddings. The data is split into 8,544 training instances, 1,101 validation, and 2,210 test instances. A pair of example training sentences are as follows:

Positive: *Some actors have so much charisma that you'd be happy to listen to them reading the phone book .*

Negative: *I was surprised at how quickly it faded from my memory.*

3.3 Experiments

The experiments included in this dissertation are designed to extend the results in Lovelace et al. (2022) by answering three research questions:

- For latent diffusion language models, would additional model capacity be deployed more optimally (in terms of output quality and inference speed) by expanding the encoder/decoder LM, or the diffusion transformer/denoising network?
- Instead of fixing the encoder/decoder LM, can better performance be obtained by applying LoRA finetuning to it (alongside the usual process of learning the denoising network)?
- Do different encoder/decoder LMs, of equivalent size, give similar results?

To investigate we run a fully-crossed experiment design. For each task (E2E & SST) we train a latent diffusion LM for each combination of:

- Variable dimension / layer count of denoising transformer
- Original (BART-base) vs reduced parameter (BART-small) encoder LM
- LoRA finetuned encoder LM vs fixed

Additionally, we run a handful of experiments swapping out BART-small with T5-small. T5 is an autoregressive encoder/decoder LM similar to BART, chosen because it boasts a range of modern architectural and training procedure improvements over BART (see Raffel et al. (2020)). Yet in terms of model size, T5-small has comparable parameter count to BART-small making a fair comparison possible.

The full list of experiments is shown in Table 3.1.

3.3.1 Implementation details

BART-base (Lewis et al. (2020)) has 6 encoder and decoder layers, and a hidden dimension of 768, for a total of 140M parameters. Model weight checkpoints are taken from huggingface[3].

[3]huggingface.co/facebook/bart-base

Dataset	Pre-trained LM	Denoising transformer size	LM finetuning
SST	BART-small	(384, 10) (dimension d_{tx}, layers)	None LoRA
		(512, 10)	None LoRA
		(512, 11)	None LoRA
		(640, 10)	None LoRA
	BART-base	(384, 10)	None LoRA
		(512, 10)	None LoRA
		(512, 11)	None LoRA
		(640, 10)	None LoRA
		(768, 12)	None
	T5-small	(512, 10)	None LoRA
E2E	BART-small	(384, 10)	None LoRA
		(512, 10)	None LoRA
		(640, 11)	None LoRA
	BART-base	(384, 10)	None LoRA
		(512, 10)	None LoRA
		(768, 12)	None
	T5-small	(512, 10)	None LoRA

Table 3.1: Full list of our completed experiments: 18 models were trained on the SST dataset, and 12 on the E2E dataset. The highlighted rows represent the final model settings used in Lovelace et al., 2022, however a lack of VRAM prevented us from successfully training at these model sizes.

BART-small is a variant of BART trained by Luca Di Liello (University of Trento, Italy). It uses a smaller hidden dimension of 512, fewer attention heads, and reduced feed forward size. The total parameter count is 70M. All other training details are identical to BART-base. Model weights are taken from huggingface[4].
T5-small model weights are taken from huggingface[5]. The total parameter count is 61M.
LoRA finetuning implementation was greatly eased through use of the PEFT python package[6]. We use a default value for the LoRA dimension parameter, $r = 8$.

Training settings common to all models are shown in Appendix Table A.1. Experiments were run on the University of Edinburgh ILCC computing cluster. Each experiment took between 16 and 36 hours (dependent on settings) to train on a single NVidia GTX 1080Ti GPU with 11GB VRAM.

3.4 Evaluation metrics

For each experiment, we sample 1000 sentences from the latent diffusion LM. To measure the fluency and quality of generated text, we calculate four different metrics:

- **MAUVE Score**. This metric, introduced in Pillutla et al. (2021), compares the distribution of generated text to a reference, using divergence frontiers. It is intended to measure how ‘human like’ the generated text is. The calculation of divergences requires both text samples to be embedded in some space - we report scores utilizing a modern autoregressive language model, GPT-2 Large (Radford et al. (2019)), as the embedding model for this purpose. For the reference sample of human-generated text, we sample 1000 instances from the test set.

- **Perplexity** (Ppl) measures how likely to occur the generated text is, according to some oracle model. We report scores using GPT-2 Large.

- **Diversity** (Div). We want our generated text to not only be fluent, but non-repetitive. Following Su et al. (2022), we report a diversity metric over our generated text as follows: $\textbf{diversity} = \prod_{n=2}^{4} \frac{|\text{unique n-grams}|}{|\text{total n-grams}|}$

[4]huggingface.co/lucadiliello/bart-small
[5]huggingface.co/t5-small
[6]https://github.com/huggingface/peft

- **Memorisation** (Mem). We note that the previous metrics could be near-maximised by simply regurgitating examples from the train set (overfitting). To measure the tradeoff between perplexity and memorisation, we report the proportion of generated 4-grams which were present in the training set.

We repeat this sampling of 1000 sentences and metric calculation five times, and report the mean and standard deviation of each metric as $\text{mean}_{\text{stdev}}$.

As an additional comparison for our models, we generate *reference metrics* by taking a natural sample of 1000 instances from the training set. This acts a proxy as a sample from a hypothetical 'reference model'. Metrics are generated using this sample in the same way as described above.

3.4.1 Metrics for controlled generation

- **MAUVE score class alignment**. For our generic MAUVE metric, we compare 1000 sentences sampled unconditionally from our model with 1000 random test set samples (containing sentences from every ground-truth class label). However; we can instead choose to compare test set samples from a single class against sentences sampled from our model using a specific label conditioning. We iterate this method to implement a fully crossed design, generating MAUVE scores to compare each ground-truth label against each conditioning label. We include the full matrix in our report[7]. If the model is generating sentences highly similar to data from the same class it has been conditioned on, this implies that the conditional generation is working as intended. Conversely, we should expect lower MAUVE scores whenever the conditioning and ground-truth labels are misaligned. We can summarise this matrix into a single metric:

$$\textbf{Class alignment} = \frac{\sum_{i=j} \text{MAUVE}_{ref=j}^{cond=i}}{\sum_{i\neq j} \text{MAUVE}_{ref=j}^{cond=i}}, \text{ where } i, j \text{ range over the class labels.}$$

[7]For the conditional dataset (SST) used in this dissertation, this results in a 2x2 matrix only - but the same approach can be extended to any number of class labels

Chapter 4

Results and analysis

The full table of evaluation results (including MAUVE scores from both validation and test sets) is available in Appendix A.2. Here, we highlight certain relevant subsets of these results in order to answer our specific research questions.

4.1 Varying the size of each network

We want to know whether varying the size of the pre-trained encoder/decoder language model has a greater effect on the quality of generated text, compared to varying the size of the denoising transformer by a similar degree. To determine this, we ignore experiments involving either LoRA finetuning or the T5 language model (which has a different architecture to BART, and so results are not directly comparable). MAUVE and Perplexity scores for the remaining models are shown in Table 4.1.

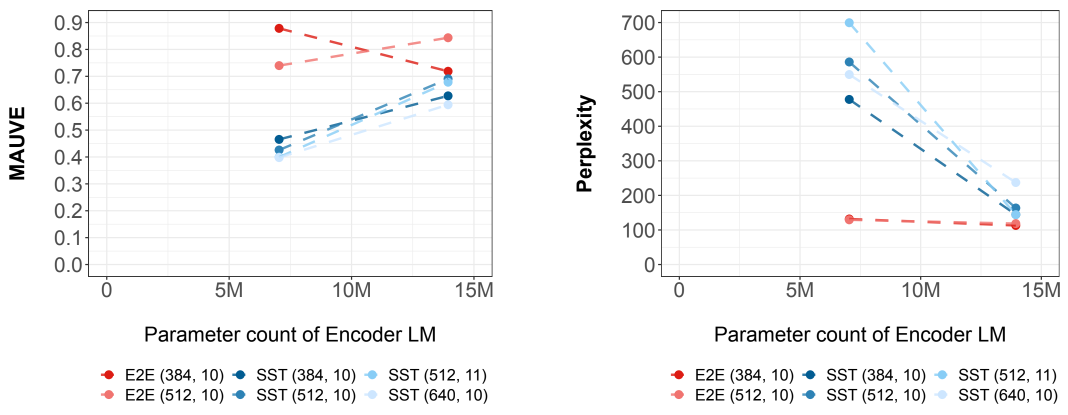
We observe that increasing the parameter count of the encoder LM brings a significant improvement in terms of both MAUVE and Perplexity scores - see Figure 4.1 for a clear illustration of this.

Conversely, many of our architecture experiments show minimal (or even negative) gains when parameter count of the diffusion transformer is increased. See Figure 4.2 for an illustration.

This confirms our hypothesis that additional model capacity should primarily be spent improving the encoder network, rather than the denoising transformer. This agrees with findings from previous work (Li et al. (2022)) that learning a better set of word embeddings can significantly improve diffusion language models.

Task	Encoder LM Size	Diffusion Transformer Size	MAUVE (test) ↑	Ppl ↓
SST	BART-small (70M)	(384, 10) - 46M	$0.465_{0.07}$	$477.6_{66.3}$
		(512, 10) - 81M	$0.426_{0.06}$	$585.9_{93.4}$
		(512, 11) - 88M	$0.399_{0.11}$	$699.5_{293.4}$
		(640, 10) - 126M	$0.398_{0.11}$	$549.7_{86.9}$
	BART-base (140M)	(384, 10) - 46M	$0.627_{0.13}$	$144.6_{84.8}$
		(512, 10) - 81M	$0.691_{0.09}$	$163.5_{29.2}$
		(512, 11) - 88M	$0.678_{0.1}$	$144.3_{17.9}$
		(640, 10) - 126M	$0.594_{0.07}$	$237.2_{100.4}$
E2E	BART-small (70M)	(384, 10) - 46M	$0.878_{0.1}$	$131.7_{23.5}$
		(512, 10) - 81M	$0.74_{0.05}$	$129.4_{14.8}$
		(640, 11) - 137M	$0.797_{0.05}$	$140.9_{11.6}$
	BART-base (140M)	(384, 10) - 46M	$0.719_{0.05}$	$113_{13.4}$
		(512, 10) - 81M	$0.843_{0.07}$	$118.5_{15.5}$

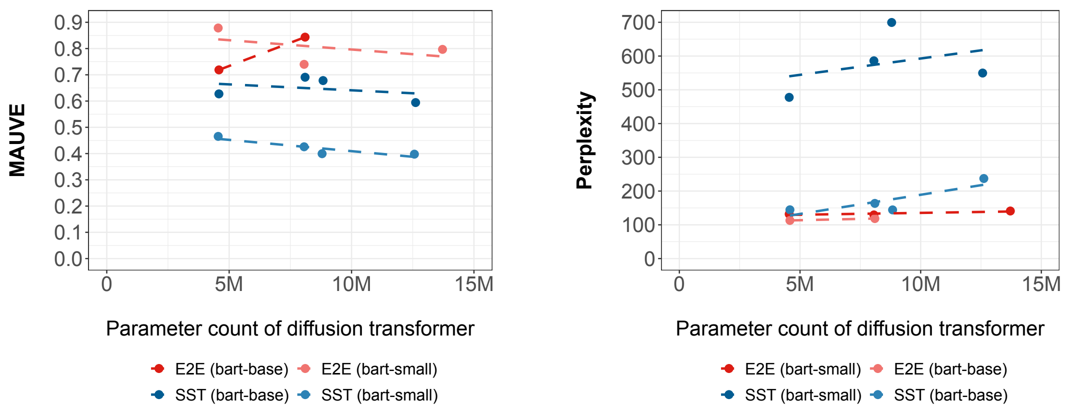
Table 4.1: Experimental results for varying parameter counts of each model



(a) MAUVE (test set)

(b) Perplexity

Figure 4.1: Effect of Encoder LM parameter count on text fluency: larger models are strongly associated with improved MAUVE and Perplexity scores, across the range of experiments



(a) MAUVE (test set) (b) Perplexity

Figure 4.2: Effect of Diffusion Transformer parameter count on text fluency: no clear improvement in MAUVE or Perplexity from increased model size

4.2 LoRA tuning of encoder LM

We now know that improving the pre-trained encoder LM is a promising direction for improving our diffusion language model as a whole. With LoRA finetuning, we test whether we can train a custom embedding to bring further improvements. In Table 4.2, we show full experimental results (using unconditional generation) for our main baseline model[1], with and without LoRA finetuning. Within each task, the highlighted cells show whether the LoRA or non-LoRA model was superior.

Task	Model	MAUVE (test) ↑	Div ↑	Mem ↓	Ppl ↓
SST	Reference metrics	$0.968_{0.02}$	$0.914_{0.014}$	$0.048_{0.008}$	$108.8_{23.2}$
	Baseline latent diffusion LM	$0.691_{0.09}$	$0.787_{0.027}$	$0.06_{0.007}$	$163.5_{29.2}$
	Latent diffusion LM + LoRA	$0.673_{0.04}$	$0.796_{0.021}$	$0.062_{0.006}$	$153.8_{25.1}$
E2E	Reference metrics	$0.819_{0.06}$	$0.164_{0.012}$	$0.872_{0.011}$	$128.5_{10.8}$
	Baseline latent diffusion LM	$0.843_{0.07}$	$0.179_{0.025}$	$0.839_{0.02}$	$118.5_{15.5}$
	Latent diffusion LM + LoRA	$0.74_{0.08}$	$0.173_{0.014}$	$0.853_{0.025}$	$122.3_{24.7}$

Table 4.2: Experimental results for LoRA finetuning (unconditional generation)

Unfortunately, LoRA has completely failed to improve the quality of generated text for the **E2E** task according to our chosen metrics - although note that both models attain similar performance to the *reference* metrics, suggesting that performance on this easier task is already near saturated.

[1]BART-base, with denoising transformer dimension (512, 10)

On the more difficult **SST** task (where performance is substantially below the reference metrics), we see mixed results. Implementing LoRA has improved the diversity of our generated text from 78.7% to 79.9%, and improved the perplexity from 163.5 to 153.8. No improvement in MAUVE score or memorisation was observed.

Whilst these results are not tremendously encouraging, the relatively large standard deviations around each metric make drawing conclusions difficult. We attempt to use the remainder of our experiments (i.e. not only the BART-base / (512, 10) baseline shown in Table 4.2) to narrow the uncertainty:

For each of our 15 model architectures, we have performance metrics across five random seeds for both {LoRA and non-LoRA} versions of the architecture. This gives us 15 * 5 = **75** independent comparisons of LoRA vs non-LoRA for each metric. We can therefore calculate the proportion of occurrences, across these 75 comparisons, that the LoRA model was superior to the non-LoRA model:

For diversity, LoRA was superior **53%** of the time
For MAUVE, LoRA was superior **44%** of the time
For memorisation, LoRA was superior **57%** of the time
For perplexity, LoRA was superior **40%** of the time

Therefore, we find no significant performance impact from implementing LoRA finetuning.

4.2.1 Conditional generation

In Table 4.5, we show MAUVE class alignment scores for the conditional generation task (SST dataset). The same baseline model as above is shown, with and without LoRA finetuning.

As expected, we see higher MAUVE scores where the ground truth reference label aligns with the label used for conditioning. This demonstrates that our models are correctly distinguishing between the two classes during conditional generation, and holds true for both the baseline model and LoRA model. Interestingly, we see greater differentiation when generating sentences conditioned on the ‘Negative’ class - perhaps negative movie reviews have more uniquely distinguishing characteristics than positive reviews.

		Baseline latent diffusion LM		Latent diffusion LM + LoRA	
		Reference Label		Reference Label	
		Negative	Positive	Negative	Positive
Conditioning	Negative	$0.713_{0.17}$	$0.346_{0.06}$	$0.682_{0.05}$	$0.286_{0.07}$
Label	Positive	$0.486_{0.16}$	$0.554_{0.17}$	$0.564_{0.18}$	$0.591_{0.07}$

Table 4.3: Experimental results for LoRA finetuning (conditional generation: MAUVE class alignment scores on SST dataset)

The **class alignment ratio** (see Section 3.4.1) for the baseline model is **1.52**, compared to **1.50** for the LoRA fine-tuned model. This again shows that both models are performing at roughly the same level, but that customising our embedding via fine-tuning has not improved the extent of conditional control we have over the diffusion model. Examples of conditionally generated text can be found in Table A.5.

4.3 Replacement of enconder LM

Instead of improving the suitability of the encoder LM to our task via by direct finetuning, in these experiments we replace it with an alternative pre-trained LM. We hope to improve text generation by picking a model that has roughly the same size (T5-small has 60M parameters, compared to 70M for BART-small), but better performance when evaluated as a standalone auto-regressive LM. In Table 4.4, we show full experimental results for our main BART-small model[2], compared to T5-small. Within each task, the highlighted cells show which of the T5 or BART model was superior.

Task	Model	MAUVE (test) ↑	Div ↑	Mem ↓	Ppl ↓
SST	Reference metrics	$0.968_{0.02}$	$0.914_{0.014}$	$0.048_{0.008}$	$108.8_{23.2}$
	Latent diffusion LM: BART-small	$0.426_{0.06}$	$0.854_{0.011}$	$0.036_{0.008}$	$585.9_{93.4}$
	Latent diffusion LM: T5-small	$0.202_{0.06}$	$0.866_{0.016}$	$0.037_{0.006}$	$406.7_{313.2}$
E2E	Reference metrics	$0.819_{0.06}$	$0.164_{0.012}$	$0.872_{0.011}$	$128.5_{10.8}$
	Latent diffusion LM: BART-small	$0.74_{0.05}$	$0.158_{0.013}$	$0.861_{0.015}$	$129.4_{14.8}$
	Latent diffusion LM: T5-small	$0.507_{0.13}$	$0.231_{0.019}$	$0.784_{0.025}$	$147.4_{24.4}$

Table 4.4: T5 experimental results (diffusion transformer dimension (512, 10), non-LoRA)

[2]with denoising transformer dimension (512, 10) and non-LoRA

As in the previous LoRA section, results are mixed. We observe some significant improvements in terms of diversity, memorisation, and perplexity, suggesting that the T5 embeddings are more successful for our diffusion process.

However, we also observe a very large deterioration in MAUVE score: from 0.426 to 0.202 in the SST task, and from 0.74 to 0.507 in the E2E task. This seems contradictory at first - we will take a closer look at samples of the generated text to understand why.

4.3.1 Generated text

Inspection of example sentences generated from each model immediately reveals the issue: many of the sentences generated by the T5 model talk about films in the expected way, but are in German[3]! This is due to the inclusion of German, French, and Romanian translation tasks[4] in T5's pre-training procedure, unlike BART's.

Whilst all of our training data is in English, note that during training the denoising transformer is only learning to recover the correct *latent embedding* of the training sentences. We hypothesize that roughly the correct embeddings (i.e. representing semantic content of film/restaurant reviews) are being recovered, but that embeddings for German and English translations of the same sentence are very close (in terms of L2-norm) inside the T5 latent space. Investigating further would be an interesting problem, but outside the scope of this dissertation. Our main conclusion is that switching to a different pre-trained encoder LM has the potential to deliver significantly different results, but that care must be taken in choosing the correct model.

Example sentences generated from our **BART-small** latent diffusion LM, SST task:

```
1: A tedious drama of a a trangling family man that the slap-buttones
   into the confrontation.
2: It's a solid family romantic drama that's proves to achieve it.
3: A dark, deeply emotional portrait of a story.
4: The film's depth may may may not gravitate, but the script is
   resurrecting and the morality of their conscience.
```

[3]multiple languages in our output easily explains the observed diversity and memorisation improvement. As MAUVE and perplexity both depend on large auto-regressive 'oracle' models for their calculation, how they respond to multilingual data will heavily depend on this model choice

[4]note that mT5, a multilingual version of T5 covering 101 languages, was trained by (Xue et al. (2021)) and is widely available - but is not the version used here

Example sentences generated from our **T5-small** latent diffusion LM, SST task:

```
1: No cinematic moments in any real action, but there’s no film
   clever moments in a real action.
2: Vielleicht hat es hier nicht diesen Charme, weil Sie eine
   schwierige Methode haben, ihn zu distanzieren.
3: Ein schmerzhaftes, tiefe und flötvolles Film.
4: Ein entertaining, wenn auch letzten Endes satisfying, documentary.
```

4.3.2 Conditional generation

In Table 4.5, we show MAUVE class alignment scores for the conditional generation task (SST dataset). The same BART-small and T5-small models as above are shown.

		Latent diffusion LM: BART-small		**Latent diffusion LM: T5**	
		Reference Label		Reference Label	
		Negative	Positive	Negative	Positive
Conditioning Label	Negative	$0.683_{0.18}$	$0.395_{0.08}$	$0.163_{0.05}$	$0.141_{0.07}$
	Positive	$0.548_{0.25}$	$0.619_{0.05}$	$0.154_{0.04}$	$0.184_{0.03}$

Table 4.5: Experimental results for T5 models (conditional generation: MAUVE class alignment scores on SST dataset)

Again, we see higher MAUVE scores where the ground truth reference label aligns with the label used for conditioning, and higher differentiation for generated sentences conditioned on the ‘Negative’ class. As discussed above, overall MAUVE scores for the T5-small model are significantly below the BART-small baseline. This contributes to a low class alignment ratio of 1.18 for the T5-small model, in comparison to 1.38 for BART-small: we conclude that switching to the pre-trained T5 language model has harmed capacity for controllable generation, as well as fluency.

4.4 Inference speed

We investigate the time taken to run model inference, sampling 1000 sentences from the model trained in each experiment. We use the same NVidia GTX 1080Ti as used during training. Each inference task was repeated five times with differing random seeds. The full set of results are shown in Table 4.6.

To interrogate these results, we train a simple linear regression model to predict the number of seconds taken. Due to relatively low variability, this is a straightforward task (R^2 of 91% was achieved). We obtain the following regression coefficients[5]:

```
SST task is 3.2 seconds quicker than E2E  (statistically insignificant)
LoRA models are 8.8s slower than non-LoRA models (statistically insignificant)
Each additional 1M encoder LM parameters will add 0.47s to inference
Each additional 1M diffusion transformer parameters will add 4.8s to inference
```

Task	Encoder LM	Diffusion Dim	Inference (s), non-LoRA ↓	Inference (s), LoRA ↓
SST	BART-small	(384, 10)	467.7 $_{5.8}$	422.8 $_{0.8}$
		(512, 10)	545.5 $_{19.8}$	561.8 $_{12.7}$
		(512, 11)	546.6 $_{8.1}$	597 $_{7.7}$
		(640, 10)	816.1 $_{2.4}$	803.7 $_{4.7}$
	BART-base	(384, 10)	462 $_{1.5}$	476.8 $_{21.2}$
		(512, 10)	550.8 $_{2.8}$	701.8 $_{27.9}$
		(512, 11)	622.4 $_{1}$	584 $_{3.2}$
		(640, 10)	842.4 $_{8.8}$	859.6 $_{3.1}$
	T5-small	(512, 10)	580.3 $_{2.6}$	568.7 $_{7.4}$
E2E	BART-small	(384, 10)	442.5 $_{5}$	417.9 $_{2.6}$
		(512, 10)	525.4 $_{2.6}$	674.9 $_{36.2}$
		(640, 11)	872.2 $_{2.4}$	871 $_{3.7}$
	BART-small	(384, 10)	423.2 $_{1.2}$	432.3 $_{4}$
		(512, 10)	683.5 $_{4.4}$	545.5 $_{4}$
	T5-small	(512, 10)	545.5 $_{1.3}$	542.4 $_{1}$

Table 4.6: Average wall time (seconds) to generate 1000 samples, for each experiment

[5]Note that these coefficients must be considered in relation to an average inference time of 600 seconds

Accordingly, we conclude:

- The task (SST or E2E) has negligible influence on inference speed
- LoRA models **do not suffer from significantly slower inference** compared to non-LoRA models
- Increasing the size of the encoder LM and/or diffusion transformer slows down inference, although the effect of diffusion transformer size is **ten times greature**. This is expected, due to the iterative nature of the denoising process during inference

Chapter 5

Conclusions and further work

In this project we have expanded upon previous work by training a variety of diffusion language models within the latent space of pre-trained encoder/decoder LMs. These diffusion-based models currently lag behind current foundational auto-regressive models in most language modelling tasks, but have the potential to unlock very powerful forms of classifier control during text generation. In this newly emerging field of study (entirely within the last ~ 18 months); the outcome of this project can help guide the direction of future work.

We have shown that varying the capacity of the pre-trained LM has a large effect on the quality of generated text, but a modest effect on inference speed. Conversely; varying the capacity of the denoising network brings a lesser improvement to the quality of generated text, yet a significant effect on inference speed. Swapping between different pre-trained LMs brings a large difference in results, although care must be taken to consider the specific properties of the chosen model.

All of this suggests that it should be possible to successfully fine-tune an existing LM in parallel with learning the denoising network, but our attempts in this direction (using LoRA fine-tuning) proved inconclusive. This is disappointing, but we have suggestions for how further work may overcome this.

Our investigations were able to confirm that conditional generation can successfully be used to control latent diffusion language models, but again we observed no improvement over the baseline level of control when varying our experimental setup.

5.1 Limitations and further work

Due to limited compute budget, the models trained in this work are small in comparison to modern autoregressive models such as Llama (Touvron et al. (2023)). The LoRA finetuning approach may be a lot more powerful when applied to larger ($>$1B parameter) models - it is not clear whether our results will extrapolate. Additionally, there are avenues for exploration not covered in this work:

- Varying the dimension parameter r. In our experiments, the number of LoRA-tuneable parameters is $\sim 0.4\%$ of the original parameter count: higher or lower values may be optimal
- Varying optimiser settings (learning rate etc) for the LoRA specific weight updates, rather than sharing an optimiser with the denoising network
- Other methods of fine-tuning the encoder LM: for example unfreezing the last few layers

Other extensions of this work could involve changing the class conditioning approach:

- Finer grained control could be achieved by passing a series of floats to the class label encoder during inference, rather than a binary/categorical label. This would allow generation of 'mildly positive' sentences, even though the training data may only have contained discrete 'positive' or 'negative' classes. The success of this approach could be tested by calculating the class alignment ratio of a series of generated sentences (conditioned on labels e.g. $\{0.0, 0.1, 0.2, ..., 0.9, 1.0\}$) against the discretely classified reference sentences.
- Instead of encoding ground truth class labels for each sample during training, we could experiment with using arbitrary pre-trained plugin classifier models. As well as enabling finer grained control, this would also reduce the need for human annotation of training datasets

Bibliography

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., & Berg, R. v. d. (2021). Structured Denoising Diffusion Models in Discrete State-Spaces.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners.

Chen, T., Zhang, R., & Hinton, G. (2023). Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning.

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.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., & Liu, R. (2020). Plug and Play Language Models: A Simple Approach to Controlled Text Generation.

Gulrajani, I., & Hashimoto, T. B. (2023). Likelihood-Based Diffusion Language Models.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., & Welling, M. (2021). Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). CTRL: A Conditional Transformer Language Model for Controllable Generation.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 7871–7880.

Li, X. L., Thickstun, J., Gulrajani, I., Liang, P., & Hashimoto, T. B. (2022). Diffusion-LM Improves Controllable Text Generation.

Li, Y., Zhou, K., Zhao, W. X., & Wen, J.-R. (2023). Diffusion Models for Non-autoregressive Text Generation: A Survey.

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization.

Lovelace, J., Kishore, V., Wan, C., Shekhtman, E., & Weinberger, K. (2022). Latent Diffusion for Language Generation.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.

Minervini, P., Franceschi, L., & Niepert, M. (2023). Adaptive Perturbation-Based Gradient Estimation for Discrete Latent Variable Models.

Nichol, A., & Dhariwal, P. (2021). Improved Denoising Diffusion Probabilistic Models.

Niepert, M., Minervini, P., & Franceschi, L. (2021). Implicit MLE: Backpropagating Through Discrete Exponential Family Distributions.

Novikova, J., Dušek, O., & Rieser, V. (2017). The E2E Dataset: New Challenges For End-to-End Generation. *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 201–206.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback.

Pillutla, K., Swayamdipta, S., Zellers, R., Thickstun, J., Welleck, S., Choi, Y., & Harchaoui, Z. (2021). MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.

Reid, M., Neubig, G., & Hellendoorn, V. J. (2023). DIFFUSER: DIFFUSION VIA EDIT-BASED RECON- STRUCTION.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models.

Shazeer, N. (2020). GLU Variants Improve Transformer.

Shi, J., Zhou, Y., Hwang, J., Titsias, M. K., & Mackey, L. (2022). Gradient Estimation with Discrete Stein Operators.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642.

Su, Y., Lan, T., Wang, Y., Yogatama, D., Kong, L., & Collier, N. (2022). A Contrastive Framework for Neural Text Generation. *Advances in Neural Information Processing Systems*, *35*, 21548–21561.

Touvron, H., Lavril, T., & Gautier, I. (2023). LLaMA: Open and Efficient Foundation Language Models - Meta Research.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., & Liu, T.-Y. (2020). On Layer Normalization in the Transformer Architecture.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer.

Appendix A

Appendix

A.1 Default training settings

Parameter	Setting
Max Seq Length	64
Diffusion steps (T)	1000
Noise Schedule (for α_t)	Linear
Regression Loss	L1
Transformer Layers	*variable*
Transformer Dimension	*variable*
Optimizer	AdamW (Loshchilov and Hutter (2019))
Learning Rate	1e-4
(β_1, β_2)	(0.9, 0.999)
Batch Size	64
Warmup Steps	1000
Learning Rate Schedule	Cosine Decay
Adam Weight Decay	0.01
Dropout	0.1
Gradient Clipping	1.0
EMA Decay	0.9999
Iterations	50k

Table A.1: Default training settings used across all our experiments

A.2 Exhaustive results

Task	Encoder LM	Diffusion Dim	LoRA	MAUVE (val) ↑	MAUVE (test) ↑	Div ↑	Mem ↓	Ppl ↓
SST	Reference metrics			0.976	$0.968_{0.02}$	$0.914_{0.014}$	$0.048_{0.008}$	$108.8_{23.2}$
	BART-small	(384, 10)	0	0.421	$0.465_{0.07}$	$0.842_{0.005}$	$0.042_{0.006}$	$477.6_{66.3}$
			1	0.517	$0.405_{0.13}$	$0.857_{0.009}$	$0.032_{0.004}$	$546.8_{23.1}$
		(512, 10)	0	0.589	$0.426_{0.06}$	$0.854_{0.011}$	$0.036_{0.008}$	$585.9_{93.4}$
			1	0.345	$0.372_{0.07}$	$0.841_{0.013}$	$0.043_{0.011}$	$560.1_{87.8}$
		(512, 11)	0	0.327	$0.399_{0.11}$	$0.859_{0.016}$	$0.033_{0.007}$	$699.5_{293.4}$
			1	0.451	$0.451_{0.08}$	$0.862_{0.013}$	$0.037_{0.009}$	535.2_{40}
		(640, 10)	0	0.495	$0.398_{0.11}$	$0.855_{0.012}$	$0.038_{0.008}$	$549.7_{86.9}$
			1	0.422	$0.448_{0.09}$	$0.844_{0.013}$	$0.039_{0.008}$	$596.8_{79.3}$
	BART-base	(384, 10)	0	0.548	$0.627_{0.13}$	$0.756_{0.022}$	$0.065_{0.01}$	$144.6_{84.8}$
			1	0.753	$0.616_{0.04}$	$0.731_{0.014}$	$0.066_{0.008}$	$990_{1858.4}$
		(512, 10)	0	0.609	$0.691_{0.09}$	$0.787_{0.027}$	$0.06_{0.007}$	$163.5_{29.2}$
			1	0.803	$0.673_{0.04}$	$0.796_{0.021}$	$0.062_{0.006}$	$153.8_{25.1}$
		(512, 11)	0	0.571	$0.678_{0.1}$	$0.78_{0.012}$	$0.061_{0.009}$	$144.3_{17.9}$
			1	0.672	$0.646_{0.17}$	$0.787_{0.015}$	$0.057_{0.009}$	$165.9_{42.4}$
		(640, 10)	0	0.479	$0.594_{0.07}$	$0.806_{0.022}$	$0.058_{0.009}$	$237.2_{100.4}$
			1	0.637	$0.665_{0.08}$	$0.819_{0.019}$	$0.057_{0.005}$	$227.3_{88.9}$
	T5-small	(512, 10)	0	0.136	$0.202_{0.06}$	$0.866_{0.016}$	$0.037_{0.006}$	$406.7_{313.2}$
			1	0.19	$0.184_{0.05}$	$0.864_{0.016}$	$0.038_{0.012}$	$278.8_{36.1}$
E2E	Reference metrics			0.907	$0.819_{0.06}$	$0.164_{0.012}$	$0.872_{0.011}$	$128.5_{10.8}$
	BART-small	(384, 10)	0	0.996	$0.878_{0.1}$	$0.15_{0.014}$	$0.873_{0.014}$	$131.7_{23.5}$
			1	0.906	$0.831_{0.09}$	$0.189_{0.015}$	$0.819_{0.005}$	$139.6_{6.8}$
		(512, 10)	0	0.949	$0.74_{0.05}$	$0.158_{0.013}$	$0.861_{0.015}$	$129.4_{14.8}$
			1	0.914	$0.83_{0.11}$	$0.17_{0.017}$	$0.841_{0.015}$	135.9_{11}
		(640, 11)	0	0.994	$0.797_{0.05}$	$0.19_{0.019}$	$0.831_{0.019}$	$140.9_{11.6}$
			1	0.913	$0.763_{0.12}$	$0.191_{0.014}$	$0.833_{0.007}$	$139.9_{18.5}$
	BART-base	(384, 10)	0	0.773	$0.719_{0.05}$	$0.146_{0.021}$	$0.871_{0.019}$	$113_{13.4}$
			1	0.907	$0.755_{0.07}$	$0.155_{0.016}$	$0.856_{0.011}$	$135.5_{40.8}$
		(512, 10)	0	0.925	$0.843_{0.07}$	$0.179_{0.025}$	$0.839_{0.02}$	$118.5_{15.5}$
			1	0.95	$0.74_{0.08}$	$0.173_{0.014}$	$0.853_{0.025}$	$122.3_{24.7}$
	T5-small	(512, 10)	0	0.556	$0.507_{0.13}$	$0.231_{0.019}$	$0.784_{0.025}$	$147.4_{24.4}$
			1	0.478	$0.488_{0.17}$	$0.219_{0.029}$	$0.791_{0.042}$	$144.5_{13.1}$

Table A.2: Full list of experimental results (unconditional generation fluency metrics)

Task	Encoder LM	Diffusion Dim	LoRA	‘Neg’ conditioning / ‘Neg’ reference	Neg / Pos	Pos / Neg	Pos / Pos
SST	BART-small	(384, 10)	0	$0.49_{0.12}$	$0.208_{0.06}$	$0.25_{0.08}$	$0.438_{0.15}$
			1	$0.389_{0.09}$	$0.271_{0.07}$	$0.263_{0.11}$	$0.467_{0.06}$
		(512, 10)	0	$0.683_{0.18}$	$0.395_{0.08}$	$0.548_{0.25}$	$0.619_{0.05}$
			1	$0.716_{0.2}$	$0.371_{0.14}$	$0.472_{0.2}$	$0.731_{0.06}$
		(512, 11)	0	$0.476_{0.14}$	$0.31_{0.08}$	$0.296_{0.04}$	$0.459_{0.17}$
			1	$0.409_{0.14}$	$0.248_{0.07}$	$0.269_{0.13}$	$0.385_{0.09}$
		(640, 10)	0	$0.616_{0.12}$	$0.288_{0.05}$	$0.563_{0.16}$	$0.566_{0.1}$
			1	$0.538_{0.1}$	$0.359_{0.18}$	$0.636_{0.14}$	$0.418_{0.1}$
	BART-base	(384, 10)	0	$0.44_{0.09}$	$0.212_{0.06}$	$0.307_{0.13}$	$0.388_{0.08}$
			1	$0.428_{0.19}$	$0.215_{0.07}$	$0.294_{0.1}$	$0.43_{0.12}$
		(512, 10)	0	$0.713_{0.17}$	$0.346_{0.06}$	$0.486_{0.16}$	$0.554_{0.17}$
			1	$0.682_{0.05}$	$0.286_{0.07}$	$0.564_{0.18}$	$0.591_{0.07}$
		(512, 11)	0	$0.383_{0.14}$	$0.233_{0.07}$	$0.361_{0.18}$	$0.358_{0.1}$
			1	$0.387_{0.15}$	$0.196_{0.03}$	$0.298_{0.07}$	$0.371_{0.06}$
		(640, 10)	0	$0.524_{0.13}$	$0.258_{0.1}$	$0.453_{0.16}$	$0.565_{0.08}$
			1	$0.64_{0.11}$	$0.387_{0.11}$	$0.388_{0.11}$	$0.724_{0.04}$
	T5-small	(512, 10)	0	$0.163_{0.05}$	$0.141_{0.07}$	$0.154_{0.04}$	$0.184_{0.03}$
			1	$0.193_{0.04}$	$0.138_{0.04}$	$0.14_{0.02}$	$0.166_{0.02}$

Table A.3: Full list of experimental results (MAUVE label alignment scores for conditional generation). Labels are ‘Neg’: negative sentiment, and ‘Pos’: positive sentiment

A.3 Qualitative Examples

id	generated text
1	For a average price of less than £ 20 £ 20, try you can at Giraffe. You will find it near The Bakers.
2	If you are looking for a restaurant at The Golden Curry. It serves Chinese food. It is very expensive with a good rating.
3	The Eagle has an average rating and serves English food.
4	At The Portland Arms, The Waterman pub has a customer rating of 5 out of 5.
5	Green Man is a restaurant providing Indian food in the more than £ 30 price range. It is near All Bar One in the riverside area near All bar One.
6	The Rice Boat in the £ 20 - £ 25 - 25, customer Rating is in riverside and is near the Express by Holiday Inn.
7	Green Man, located near All Bar One, offers Indian food in the moderate price range.
8	The Phoenix is a good place with a customer rating of 5 out of 5. It is located in riverside and serves Japanese food for moderate price.
9	Midsummer House Near Café Rouge It is near Café Rouge. It has average customer ratings.
10	The Rice Boat serves Japanese food with a 3 out of 5 rating. It is located in city centre.
11	There is a Japanese restaurant called The Loch Fyne which is child friendly.
12	Loch Fyne is a family friendly restaurant serving Japanese food less than £ 20
13	In the riverside area there is a pub called The Waterman. It has been a customer rating of 1 out of 5. It serves Japanese food.
14	The Plough is a pub located in city centre near Café Rouge. It is family - friendly.
15	The Dumpling Tree a pub that serves Japanese food. It is near The Portland Arms.

Table A.4: Further examples of generated text: E2E task, BART-base, (512, 10), no-LoRA

id	generated text	label
1	I don't think the thought why anyone else else... I dont think that he seems to think to think even thought.	Negative
2	If a better picture of this twisted, shameless organization will be forthcoming, this new effort.	Negative
3	The film offers a few few moments moments moments.	Negative
4	Adam Sandler's script's flaws are easily mitigated by a small-scale sense of empathy is little more than mediocre.	Negative
5	A good-hearted movie that trumps to use old-hearted humor and pends for the audience. A movie that pays homage to old-fashioned humor and wides for their sympathy, regardless if you're willing the film or despise the audience or care for the cast or its characters to care about.	Negative
6	Amitabh Bachchan has a lot of problems in the film industry. It is a big problem in the industry. But, it has a big problems.	Negative
7	This new nihilism genre becomes forgettableable.	Negative
8	Why why would such such such an such such a thing?	Negative
9	When you see you, love each other, you never quite see the story together.	Negative
10	Eagle creates an almost impossible impossible piece of work – and, of, most of it, of course, is forgotten.	Negative
16	The most of it is fairly funny, despite mostly all the actors, the actors giggle through interesting dialogue in ways – creepy, awfuly and annoying.	Positive
17	An American American fable that skits just just slightly amusing enough to the audience that the audience a slightly fling.	Positive
18	Watching through a play of Prince Charming, the " fairytale of the fairies " and the story of Mr. and Mrs. Claus is an exhilar experience.	Positive
19	What more powerful the film is it is that Shawan manages to transcend an culture of black black and cultural of black who have become our oppressors.	Positive
20	Unfortunately Depp, Shostakovich, Tarantino and Hollywood's lauded film debut all up for this ambitious effort.	Positive
21	The cast performances are outstanding, and the performances are performances, and The performances are top-notch.	Positive
22	Director David Lean beautifully delivers such a paean to the faith to the God of the universe, but co-director/director/filmmaker/director-director-producer-turned-directorial-writer-director Nichelle Nichols still oozes raw and raw, while his messages and message	Positive
23	An intriguing French French Frenchthriller-turned-comedy-dramarama about the family and their lives – and how their families.	Positive
24	This is a small-budget film and deserves a high level of rating and audience value.	Positive
25	It's on the big screen.	Positive

Table A.5: Further examples of generated text: SST task, BART-base, (640, 10), no-LoRA