Replication Study: Controlled Generation using Diffusion-LM and FUDGE

Anubhav Goel
Department of Computer Science
The University of Texas at Austin
anubhav.goel@utexas.edu

Devyani Maladkar
Department of Computer Science
The University of Texas at Austin
devyani.maladkar@utexas.edu

Abstract

With the recent explosion in the popularity of large language models, controlling their behavior has become an important problem. The task of controlled generation, which entails performing control on the output of a language model without heavy retraining, has received increased attention from the research community in recent years. For this replication study, we look at Diffusion-LM and FUDGE, two popular methods for handling controlled generation. This paper is organized as follows: we begin with an overview of the task of controlled generation, followed by an exposition of the principles and methodology underpinning Diffusion-LM and FUDGE. Subsequently, we discuss various fine-tuning methods that serve as benchmarks for our investigation. Centering on semantic control and parts-of-speech (POS) tagging tasks, we delineate our experimental design to assess the aforementioned models. We conclude this study with an extensive analysis of the results that we obtain and their direct comparison with existing implementations.

1 Controlled Generation

In recent years, the field of natural language processing has witnessed a surge in the adoption and popularity of language models, which can be attributed to their remarkable ability to generate human-like text. Models such as GPT-4 have spearheaded this revolution, paving the way for numerous applications spanning content generation, virtual assistants, and sentiment analysis.

Despite these advancements, deploying language models in real-world settings remains a challenging problem, with a focus on controlled generation, which remains an open problem in this field. These methods aim to address potential pitfalls such as bias, offensiveness, and inaccuracy in generated content by fine-tuning and monitoring models. We underscore the importance of controlled generation in harnessing the immense potential of language models while mitigating the risks inherent in their deployment across various applications.

More concretely, given a language model whose outputs can be modeled as a probability distribution $P(X)$ over natural language sequences X (or $P(X|I)$ given input I, such as in the case of machine translation task), and an attribute a, we would like to sample outputs from the distribution $P(X|a)$ (or $P(X|I, a)$).

An initial approach to controlled generation is fine-tuning the language model according to a given control in a supervised learning setup, that is, given (control, text) pairs, the model weights can be retrained. This poses several challenges, starting with the fact that it requires heavy retraining of language models for every control. It also requires access to the entire model architecture which may not always be available. Additionally, the above methodology does not allow for the composition of controls either (without another round of retraining). This motivates the need for modular and lightweight models for controlled generation.

While there has been some success in achieving simple controls such as controlling the sentiment of the output, more complex controls such as generating an output corresponding to a sequence of POS tags or controlling the semantic content of output remain challenging and is an open problem in the field of natural language processing.

For this replication study, we look at *Diffusion-LM* (using continuous diffusion to create a hierarchical latent space representation which can be manipulated using gradient-based methods to incorporate the control) and *FUDGE (Future Discriminators for Generation)* (using a classifier for attribute a in conjunction with Bayes' Rule to condition the output of the model such that the control is satisfied).

2 Diffusion-LM

Diffusion-LM (Li et al., 2022) extends the idea of continuous diffusions, which was previously successfully applied to vision and audio domains, to text. However, adapting this idea requires the addition of embedding steps and decoding steps to the existing diffusion framework, learning the embedding function as well as proposing techniques for improved decoding. We begin with an overview of diffusion models.

2.1 Diffusion Models for Continuous Domains

Diffusion models are latent variable models which work as follows: given data $x_0 \in \mathbb{R}^d$, it is modeled as a sequence of latent variables $x_T, ..., x_0$, which each latent variable in $\mathbb{R}^d$ and x_T being a Gaussian random variable. The key idea is to incrementally denoise the sequence $x_{T:1}$ to approximate the samples from the target data distribution. Additionally, we model $x_T \sim \mathcal{N}(0, I)$ and the denoising process is modeled as

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

where the μ_θ and Σ_θ are modeled by a deep learning framework such as a transformer.
Additionally, a forward process is defined which computes the intermediate latent space representation $x_{1:T}$. This is achieved by incrementally adding Gaussian noise to the data, such that at step T, the samples x_T are approximately Gaussian. The forward process can then be modeled as

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

where β_t is a hyperparameter (amount of noise added at step t). The forward process does not contain any trainable parameters. The diffusion model aims to maximize the marginal likelihood of the data which leads to the following variational lower bound on $log(p_\theta(x_0))$

$$\begin{aligned}\mathcal{L}_{vlb}(x_0) = \mathbb{E}_{q(x_{1:T}|x_0)}[log\frac{q(x_T|x_0)}{p_\theta(x_T)} \\ + \Sigma_{t=2}^T log\frac{q(x_{t-1}|x_0, x_t)}{p_\theta(x_{t-1}|x_t)} \\ - logp_\theta(x_0|x_1)].\end{aligned}$$

However, this objective is numerically unstable and is generally substituted by

$$\mathcal{L}(x_0) = \Sigma_{t=1}^T \mathbb{E}_{q(x_t|x_0)} \|\mu_\theta(x_t, t) - \hat{\mu}(x_t, x_0)\|^2$$

where $\hat{\mu}(x_t, x_0)$ is the mean of the distribution $q(x_{t-1}|x_0, x_t)$, which can be shown to be a closed form Gaussian.

2.2 Diffusion Models for Language

Extending the above idea to discrete text, we need an embedding function that maps each word to a vector in $\mathbb{R}^d$. An end-to-end framework is proposed which learns the diffusion model parameters as well as the word embedding function. Similarly, we need to add a trainable rounding function as well which can be used to decode text from the continuous vector representation.
The decoder function is slightly harder to model in this case, since the x_0 obtained using the model does not map exactly to a (unique) word. This problem arises due to the fact that the loss does not enforce a constraint on the structure of x_0 to force it to commit to one word, except at time steps near $t = 0$. The workaround suggested for this problem is to model x_0 at every step and re-parametrize the loss function as follows

$$\mathcal{L}(x_0) = \Sigma_{t=1}^T \mathbb{E}_{x_t} \|f_\theta(x_t, t) - x_0\|^2$$

where $f_\theta()$ denotes the model output. Forcing the neural network to predict x_0 at every step forces it to learn the word embedding structure. The denoising step can be written as follows in this case

$$x_{t-1} = \sqrt{\bar{\alpha}_t} f_\theta(x_t, t) + \sqrt{1-\bar{\alpha}_t}\epsilon$$

where $\bar{\alpha}_t = \Pi_{u=0}^t(1-\beta_u)$ and $\epsilon \sim \mathcal{N}(0, I)$. The authors introduce a *clamping trick*, which maps $f_\theta(x_t, t)$ to its closest word embedding at each step such that the denoising equation now becomes

$$x_{t-1} = \sqrt{\bar{\alpha}_t} clamp(f_\theta(x_t, t)) + \sqrt{1-\bar{\alpha}_t}\epsilon$$

which further forces commitment to a single word and reduces decoding errors in the model.

2.3 Controllable Text Generation

The approach to controlling the output of the LM is motivated by the Bayesian formulation and instead of being performed on the discrete text, the control is performed on the sequence of latent representations $x_{0:T}$. This control can be modeled as decoding from the distribution $p(x_{0:T}|c) = \Pi_{t=1}^T p(x_{t-1}|x_t, c)$ which at each time step can be simplified by Bayes' rule as follows

$$\begin{aligned}p(x_{t-1}|x_t, c) &\propto p(x_{t-1}|x_t)p(c|x_{t-1}, x_t) \\ &\propto p(x_{t-1}|x_t)p(c|x_{t-1})\end{aligned}$$

where the second term is simplified using conditional independence assumption for diffusion models. The gradient update on x_{t-1} can now be templatized as follows:

$$\nabla_{x_{t-1}} logp(x_{t-1}|x_t, c) = \nabla_{x_{t-1}} logp(x_{t-1}|x_t) + \nabla_{x_{t-1}} logp(c|x_{t-1})$$

Fluency Regularization can be performed by multiplying one of the terms with a hyperparameter λ to control the tradeoff between *fluency* and *control*.

2.4 Implementation

We trained the language model as well as all the classifiers that we used in our experiments ground up using the implementation provided by the authors (Li et al., 2022) (Transformer architecture with 80M parameters and diffusion steps (T) set to 2000).

3 FUDGE

Future Discriminators for Generation (Yang and Klein, 2021) provides a flexible method for the task of controlled text generation. Given attribute a (for example sentiment) and a language model, FUDGE uses Bayes' rule to decompose the desired conditional distribution $P(x|a)$, to introduce control $P(a|x)$ onto the output of the language model $P(x)$. This method requires access to the output probabilities (logits) of the language model only. In order to model $P(a|x)$, a binary predictor is learned which indicates whether the attribute a will be satisfied in the final completed sentence, based on the incomplete sentence prefix.

Let $x = x_{1...n}$ be a complete sentence where each x_i represents a word in the sentence (please note the difference in notation from the previous section where each x_t denotes an intermediate latent space representation of a sequence of words). The language generation is done by an autoregressive model which models $P(x_i|x_{1:i-1})$ (note that this is in contrast to Diffusion-LM which does not use the autoregressive model). We can obtain the entire sequence $P(x) = P(x_{1:n})$ by sampling at each point in the sequence using the below decomposition,

$$P(x_{1:n}) = \prod_{i=1}^{n} P(x_i|x_{1:i-1})$$

Consequently, it can be extended to model controlled generation as follows

$$P(X = x_{1:n}|a) = \prod_{i=1}^{n} P(x_i|x_{1:i-1}, a)$$

The Bayes' rule gives us

$$P(x_i|x_{1:i-1}, a) \propto P(a|x_{1:i-1})P(x_i|x_{1:i-1})$$

$P(a|x_{1:i-1})$ is modeled using the binary predictor as mentioned earlier. The performance of the binary predictor is crucial to condition the output of the language model. We train our classifier using a dataset of the form $(x_{1:n}, a)$. However, the important point to note here is that we need to train a future predictor, which is a classifier that can output the probability of an attribute given not just the complete sentence but incomplete prefixes of the sentence as well. This makes designing such a classifier challenging and necessitates the need to train it on incomplete prefixes of the training data as well $((x_{1:i}, a), i < n)$.

Additionally, as in the previous section, we use a hyperparameter λ to control the tradeoff between *fluency* and *control*. This is referred to as the temperature term in the Bayesian decomposition, which controls the sensitivity of the generation to the binary predictor's output. With this term, the equation can be rewritten as follows

$$P(x_i|x_{1:i-1}, a) \propto P(a|x_{1:i-1}) \left(P(x_i|x_{1:i-1})\right)^{\lambda}$$

3.1 Implementation

The FUDGE implementation as presented in (Yang and Klein, 2021) does not extend naturally to our control tasks (Section 6), so we implement the framework from scratch, following the outline in (Li et al., 2022).

We choose the GPT-2 small architecture model for the language model in the FUDGE framework. We fine-tune the pre-trained model on the E2E dataset (Section 5). For both the tasks during sentence generation, we choose the top 20 values of $P(x_i|x_{1:i-1})$, that is we take only the words with the 20 highest probabilities by the language model, and then apply the controls, as opposed to applying the control on all 50257 words of the vocab. Since the language model is task-independent, we train it once and use the same model for all control tasks. For the semantic control task, we train a future discriminator for each of the 8 attributes. For each

Model	Test	Train	Valid
bert-name	99.1	99.8	99.7
bert-near	82.6	99.9	82.6
bert-price	92.69	96.57	86.97
bert-customer-rating	90.5	96.21	85.8
gpt2-area	97.30	97.05	97.50
gpt2-food	64.79	82.15	55.82
gpt2-type	61.87	86.44	92.55
gpt2-family-friendly	92.77	94.03	93.59

Table 1: Accuracy score for best future discriminator models.

binary predictor, we use a transformer-based architecture. Since the predictor performance is crucial to obtaining high control accuracy, we experiment with two transformer architectures - distilled-BERT and GPT-2 small. For some attributes such as name and price, distilled-BERT gives a higher performance while for other attributes, GPT-2 small outperforms BERT. The predictor performances for different attributes have been presented in Table 1. For each control, we pick the classifier that performs the best.

Due to computational constraints, for each training pair of the form $(x_{1:n}, a)$ in the train split of the E2E dataset we randomly generate $(x_{1:i}, a)$ for a few values of i (as described earlier), rather than training on all prefixes as suggested in (Yang and Klein, 2021). We perform hyperparameter tuning to establish a good tradeoff between fluency and control.

We also extend the semantic control task to multi-control settings. Given a set of attributes (type, food, area), we combine the future discriminator values of $P(a_i|x)$ for each attribute to obtain $P(x|a_1, a_2, a_3)$.

For the POS tags control task, we use an off-the-shelf POS tagger for the future discriminator model, namely the flair POS tagger (Akbik et al., 2019).

4 Fine-Tuning

As we mentioned earlier, Fine-tuning is a straightforward and intuitive way to guide a language model for controlled generation. Conditional training aims to embed the control into the text generation by carefully fine-tuning. The common approaches involve using RL and using controlled datasets. For this project, we make use of the controlled dataset approach. We follow a similar approach as introduced in CTRL (Keskar et al., 2019).

For the controlled generation task, we use a pre-trained language model and fine-tune the model on the data using control code prefixes.

4.1 Implementation

We use a fine-tuned GPT-2 model for comparison of controlled text generation tasks. We use this for baseline comparison as done in (Li et al., 2022). The fine-tuned GPT-2 model consists of a pre-trained GPT-2 small model which we fine-tune directly on the controlled generation task. We frame the controlled generation task as a language-modeling task and fine-tune the model. The training data and the prompt used as control at generation time are shown in Table 3.

For the semantic control task, we set the controls as the prefix of the training text, followed by the review. During generation given only the prefix is given to the model to generate the text. When training the model, we shuffle the order and number of controls in the prefix to prevent the model from learning any spurious relation to the position and count of controls. For the POS control task, we train using the POS tag sequence for the sentence as the prefix.

We train a GPT-2 model separately for each control task. At the time of generation, we use different methods of decoding. We present the controlled generation results for beam search, nucleus sampling, and greedy decoding.

attribute	values	example
name	34	The Eagle, Cotto
near	19	Café Sicilia, Burger King
area	2	riverside, city centre
price	6	cheap, more than £ 30
Type	3	coffee shop, pub, restaurant
food	7	English, Japanese, Indian
family-friendly	2	yes, no
customer rating	6	low, 5 out of 5

Table 2: E2E dataset attributes.

5 Dataset

We use the E2E dataset (Novikova et al., 2017) for training all the language models and classifiers

Semantic Control Task	
Train Instance	name : The Vaults | Type : pub | price : more than £ 30 | customer rating : 5 out of 5 | near : Café Adriatic || The Vaults pub near Café Adriatic has a 5 star rating . Prices start at £ 30.
Generation Prompt	name : The Vaults | Type : pub | price : more than £ 30 | customer rating : 5 out of 5 | near : Café Adriatic ||
POS Control Task	
Train Instance	START ADV NOUN ADV ADP PROPN PROPN PUNCT DET PROPN NOUN NOUN VERB ADJ NOUN NOUN CCONJ AUX PART VERB NOUN NOUN PUNCT ||Only feet away from Cafe Sicilia, The Punter Coffee Shop offers low price coffee and does not have family restrooms.
Generation Prompt	START ADV NOUN ADV ADP PROPN PROPN PUNCT DET PROPN NOUN NOUN VERB ADJ NOUN NOUN CCONJ AUX PART VERB NOUN NOUN PUNCT PROPN ||

Table 3: Sample instances of training and generation for Fine Tune models for controlled Text Generation

(predictors) which we have described above. This dataset contains 50K reviews for restaurants. Each review has 8 attributes. The details of the attributes and the values are in Table 2.

6 Tasks

We consider 3 control tasks to test the performance (in terms of perplexity (*fluency*) as well as accuracy (*control*) of the aforementioned controlled text generation models. The control tasks consist of semantic control, POS sequence control, and length control. The setup as well as the control tasks have been borrowed from the Diffusion-LM paper (Li et al., 2022). The tasks vary in ease of control with semantic control being considerably easier as compared to POS tags sequence. The accuracy score is calculated based on exact match for the controls. We describe the calculation of our accuracy as well as perplexity in the following subsections.

6.1 Accuracy

Semantic Control: Given the (attribute, value) pair as the control, we generate text such that it satisfies attribute = value. For accuracy, we check if a given generation contains the exact match and we set the accuracy to be the number of generations containing the exact match divided by the total number of generations.

POS Tags Sequence Control: Given a Parts-Of-Speech tags sequence, we want to generate a sentence that matches the given sequence. We use Universal POS tags. For the accuracy score, we tried matching the exact POS sequence as mentioned in the Diffusion-LM paper (Li et al., 2022) but found that it gave us very low accuracy numbers for all the methods which were hard to compare. In this case, we switched our approach such that we measured the fraction of POS tags that match with the target sequence and labeled that as the accuracy.

Length Control: Given a target integer length we want to generate a review of that length. For the accuracy score, we allow a margin of 4 (*length*$\pm$ 4).

6.2 Perplexity

We train a teacher LM (GPT-2) following the implementation provided by (Li et al., 2022) to obtain perplexity scores for the generated sentences. This teacher LM is trained with the UNK token included.

The Diffusion-LM implementation uses a word-level tokenizer including the UNK token, whereas the FUDGE and FT implementation use the default tokenizer for GPT-2 which is a sub-word tokenizer. The Diffusion-LM implementation creates a vocabulary from the training dataset replacing the low-frequency words with UNK. As a result of this difference, we evaluate two values for perplexity as follows

1. **Evaluation without re-tokenization:** We obtain the perplexity scores for the FUDGE outputs as is, without re-tokenizing it in the space of the tokens used by Diffusion-LM (which contains UNKs).

2. **Handling UNK:** We use the word level tokenizer of the Diffusion-LM to re-encode the outputs of FUDGE. This introduces UNK to-

kens in the outputs. We then generate perplexity scores for these outputs.

We report both the perplexity values because the outputs containing UNK are qualitatively poor as they can be difficult to interpret for humans. Note: the perplexity value for Diffusion-LM will be the same for both approaches.
Additionally, this occurrence of UNK is solely due to the design choice employed in the implementation of Diffusion-LM (Li et al., 2022) for which we were unable to find any reasoning. While in FUDGE and FT, to maintain the integrity of the GPT-2 implementation we choose the default tokenizer and generate outputs without UNKs.

7 Experiments

This being a replication study, we use Table 2 from (Li et al., 2022) as a reference for our results. For each of the following subsections, we first present the results from (Li et al., 2022) and then document the results that we obtain followed by a direct comparison and analysis of the two.

7.1 Accuracy

We present the accuracy results on the semantic control and the POS tags control task as reported in (Li et al., 2022).

Model	Semantic Control	POS Tags
Diffusion-LM	81.2%	90.0%
FUDGE	69.9%	27.0%
FT-Sample	72.5%	89.5%
FT-Search	89.9%	93.0%

Table 4: Control Accuracy results as presented in (Li et al., 2022). Note that the results for FT-Greedy were not reported in the paper.

We present the accuracy results on the semantic control and the POS tags control task as obtained by us.

Model	Semantic Control	POS Tags
Diffusion-LM	77.44%	84.95%
FUDGE	41.77%	12.87%
FT-Sample	91.13%	60.02%
FT-Search	91.77%	63.42%
FT-Greedy	87.34%	62.40%

Table 5: Control Accuracy results as obtained during the replication study.

We expect the results for the Diffusion-LM model to be close to what has been reported in the paper since we made use of the code provided by the authors. However, (Li et al., 2022) does not provide any details of how FUDGE was implemented for the semantic control task, nor is there any detailed description of training language modeling or classifiers in their setup. Hence, we expect to see a larger difference in the results for FUDGE. Moreover, (Li et al., 2022) does not talk about its fine-tuning approach in detail either so we observe some differences in the accuracy scores there as well.
We observe that semantic control is an easier task (models achieve reasonably high accuracy on it), whereas generating a sentence given a sequence of POS tags is considerably harder and we observe that the performance from all the models is low on this task except Diffusion-LM which still manages to do well.
Fine-tuning models performs better overall in both metrics (with the exception of control accuracy on the POS tags task). This is not surprising, it is known that fine tuning large language models performs well on a wide variety of unseen downstream tasks. However, this can require significant compute (in terms of memory and runtime) and will not be robust to frequently changing controls as the fine-tuning will require heavy retraining. Additionally, depending on the nature of the control task, a significant amount of data is required for obtaining good performance and avoid memorization by the language model. On the contrary, plug-and-play models such as FUDGE can be advantageous as they require very less compute and can perform well if the task is well formulated for the future discriminator. Unlike FT models, they do not require access to the entire language model which may not be available in closed implementations. Even Diffusion-LM is quite lightweight and requires training only a classifier which can then be used to alter the gradients of the diffusion language model.
We perform the length task as well (which is a classifier-free task in the case of Diffusion-LM). The method of implementing the length control in Diffusion-LM is that the authors use infilling setting the word at the location of the given length to be the END token. Diffusion-LM gives an accuracy of **92%** (our implementation in contrast to 99.9% as presented in the paper). There is no descrip-

tion of how to extend the length task to FUDGE and FT models. For FUDGE, we implemented a classifier that predicts when the END token would occur, thus setting the length in terms of tokens, however, FUDGE makes use of subword tokens so this methodology did not work. Additionally, FUDGE requires a good future discriminator for the control to work. In case of a task such as length it is difficult train such a model.

7.2 Perplexity

The perplexity results for semantic control and POS tags control tasks as reported in (Li et al., 2022) have been shown below.

Model	Semantic Control	POS Tags
Diffusion-LM	2.55	5.16
FUDGE	2.83	7.96
FT-Sample	2.87	4.72
FT-Search	1.78	3.31

Table 6: Perplexity results as presented in (Li et al., 2022). Note that the results for FT-Greedy were not reported in the paper.

As we described above, we use two approaches to calculate the perplexity during our experiments. We present results from the approaches in the tables below.

Model	Semantic Control	POS Tags
Diffusion-LM	8.54	6.95
FUDGE	7.04	17.68
FT-Sample	2.26	5.12
FT-Search	2.98	10.89
FT-Greedy	4.84	4.52

Table 7: Perplexity results as obtained during the replication study using the perplexity approach suggested in (Li et al., 2022) (using UNK in the tokenizer space).

Model	Semantic Control	POS Tags
Diffusion-LM	8.54	6.95
FUDGE	4.48	15.04
FT-Sample	2.80	4.18
FT-Search	2.58	3.98
FT-Greedy	2.68	4.08

Table 8: Perplexity results as obtained during the replication study using the perplexity approach as the modification without re-encoding the output of FUDGE and FT.

As noted in the section above, the observations about the task of semantic control being easier compared to POS tags still follows as we overall lower perplexity scores for semantic control compared to POS tags (with the exception of Diffusion-LM which performs better on the POS tags task).
Additionally, we observe that when we re-encode using the Diffusion-LM tokenizer the perplexity increases due to the introduction of UNKs. Despite the re-encoding scheme, the FUDGE model still does well compared to the Diffusion-LM for the semantic control task. Without handling the UNK tokens, the outputs from FT and FUDGE are very well placed in terms of perplexity. Though the control for FUDGE may not be as powerful as Diffusion-LM, the usability of the outputs from Diffusion-LM significantly decreases due to the introduction of UNKs. Therefore, we see a trade-off wherein although Diffusion-LM exerts stronger control accuracy, the UNKs as well as higher perplexity values make FUDGE a more usable scheme in real-world applications.

7.3 Qualitative Results

We present some qualitative outputs (control, sentence) pairs to demonstrate the performance of various models in Table 9. For the semantic control task, we have presented examples where all the models perform well in terms of both control and fluency.
For the POS control task which is arguably harder, we present an output from Diffusion-LM which prioritizes fluency over control accuracy (the sentence is supposed to end with punctuation and proper noun, but in the output this order is reversed to make the sentence sensible). The other models do not exhibit the same behavior with FT adding a proper noun (restaurant name) after the end of the sentence. FUDGE shows the worst performance here which is exhibited given its poor control accuracy in this task.

7.4 Fluency Regularization in FUDGE

Since we implemented FUDGE from the ground up, we had to perform extensive analysis to tune the fluency regularization control (hyperparameter λ in our analysis) to obtain strong control accuracy. We present an analysis of the hyperparameter control for both control tasks. We plotted both control accuracy and perplexity against λ for both tasks and we present our results below.

Semantic Control Task	(name, The Vaults)
Diffusion LM	The Vaults is a coffee shop near Café Brazil in the city centre . It has a customer rating of 3 out of 5 .
FUDGE	A coffee shop called The Vaults is located near Café Rouge.
FT-Search	The Vaults is a coffee shop providing Italian food in the moderate price range. It is located in the city centre.
FT-Sample	The Vaults is a coffee shop that is family friendly and serves French food. It has a low customer rating and the price range is less than £ 20. It is located in the riverside area near Café Brazil.
FT-Greedy	The Vaults is a fast food restaurant with a price range of £ 20 - 25.
Semantic Control Task	DET NOUN NOUN ADP PROPN VERB DET NOUN NOUN ADP NUM ADP ADP NUM CCONJ DET ADJ NOUN NOUN PUNCT PROPN
Diffusion LM	A coffee shop in riverside with a customer rating of 1 out of 5 and a high price range is Zizzi .
FUDGE	Alimentum in Riverside provides a family atmosphere with a mid to upper £££price range. Aliment
FT-Search	The coffee shop near Avalon has a customer rating of 5 out of 5 and a moderate price range. Fitzbillies
FT-Sample	The coffee shop near Avalon has a customer rating of 5 out of 5 and a moderate price range, Wildwood
FT-Greedy	The coffee shop near Avalon has a customer rating of 5 out of 5 and a moderate price range. Wildwood

Table 9: Sample instances of training and generation for different models for controlled Text Generation

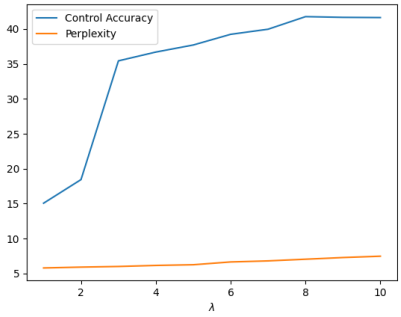

Figure 1: Control Accuracy and Perplexity while varying λ for the Semantic Control task.

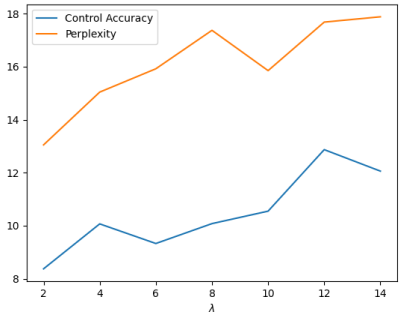

Figure 2: Control Accuracy and Perplexity while varying λ for the POS Tags Sequence task.

We can observe in both the plots that both the accuracy and the perplexity increase with increasing λ. This is in line with our expectations since increasing λ promotes control accuracy over fluency, which results in higher accuracy scores at the cost of higher perplexity (which implies lower fluency). However, we note that the accuracy starts stagnating after a certain value of λ. We look at several qualitative results to reason this as follows. Since we sample only the top 20 outputs from the language model and calculate the classifier scores for those samples only, if the required value of the control is not present in the top 20 samples, the classifier score will be low for all the samples, and increasing λ, in this case, will not lead to increase in the accuracy. We use the value of λ where the accuracy starts to saturate ($\lambda = 8$ for the semantic control task and $\lambda = 12$ for the POS tags sequence task).

7.5 Multiple Control Composition

We observe that the nature of FUDGE (using different classifiers for each attribute) not only allows us to combine multiple controls but allows us to assign different weights (different λ_i for each a_i) to different controls. This task can be thought

of as controlling the strength of each control.
We perform this analysis (which is novel from both the papers that we study) and observe the effects of assigning different weights to different controls. We perform experiments over the semantic control task where we pick 3 controls (type, area, food) and assign different weights to each and record the accuracy with respect to each control for a different combination of weights. We record our results in Table 10.

Hyperparameter Values	**Type**	**Area**	**Food**
$\lambda_{Type} = 8$ $\lambda_{Area} = 1$ $\lambda_{Food} = 1$	78.57	26.19	35.71
$\lambda_{Type} = 8$ $\lambda_{Area} = 6$ $\lambda_{Food} = 1$	54.76	59.52	35.71
$\lambda_{Type} = 8$ $\lambda_{Area} = 6$ $\lambda_{Food} = 6$	59.52	52.38	52.38

Table 10: Results showing the weighted composition of different controls using different λ for each control and the effect on accuracy for each control.

We can observe that we set $\lambda_{Type} = 8$ and $\lambda_{Area} = \lambda_{Food} = 1$, a higher priority is assigned to incorporate the given value of the type control as against the given values of the area and the food controls. This is evident in the higher accuracy with respect to the type control. Increasing λ_{Area} to 6 increases its accuracy showing a clear correlation between the associated weight and the control accuracy. However, this comes at the cost of reduced accuracy for the λ_{Type}. Subsequently, increasing the value of λ_{Food} also results in an increase in its accuracy.
In the process of tuning the λ values for different classifiers, we observe that for the single attribute control, we can obtain better performance by using different λ values. The result for FUDGE presented in Section 7.1 can be further improved by considering this fine hyperparameter tuning. Using this scheme we get a significant increase, resulting in a **48%** accuracy for FUDGE (compared to 41.77% earlier).

8 Conclusion

We motivate the study of Controlled Generation by emphasizing its importance in the deployment of language models for real-world tasks and define it formally. We observe that even though fine-tuning is a straightforward approach to this task but it requires heavy retraining and does not allow for the easy composition of controls motivating the need for modular and lightweight methods for controlled generation.
We look at two important models for controlled generation, namely Diffusion-LM, and FUDGE, and study and explore the key ideas behind their methodologies. We evaluate their performance on semantic control and POS control tasks while using fine-tuned language models as a baseline. We use two primary metrics for our evaluation: control accuracy and perplexity. We conclude that Diffusion-LM outperforms FUDGE for both the control tasks in terms of accuracy but FUDGE performs better in terms of fluency in the semantic control task whereas, for POS control, which is a harder task, FUDGE does poorly in terms of perplexity as well. Even though FT models perform well, they are compute-intensive and not suitable for the composition of controls.
Having implemented FUDGE ground up, we explore tuning the tradeoff between fluency and control as well as the composition of multiple controls and how to potentially strengthen one control with respect to others in the multiple control setting.

9 Code

We have uploaded all the code written by us during this project to the following GitHub repository: github.com/YANI-ALT/Controlled-Text-Generation

Acknowledgements

We would like to thank Prof. Greg Durrett and Kaj Bostrom for their constant assistance in this work. The discussions with Kaj gave us a lot of insight into the issues we faced for reproducing the baselines and understanding the implementations in the literature.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019.

FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

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

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.