Difficulty-Controllable Text Generation Aligned to Human Preferences

Stanford CS224N Custom Project

Allison Guman
Department of Electrical Engineering
Stanford University
aguman@stanford.edu

Udayan Mandal
Department of Computer Science
Stanford University
udayanm@stanford.edu

Nikhil Pandit
Department of Mathematics
Stanford University
npandit0@stanford.edu

Abstract

Controllable text generation (CTG) is a core problem in the field of natural language processing (NLP), and in recent years there have been a multitude of approaches to controlling the text generation of large language models (LLMs). Constraints researched for CTG include detoxifying generated text or aligning outputs to a particular sentiment. Despite the vast amount of existing research in CTG, we find there is a general lack of research in aligning controlled text generation towards human preferences or a classifier which has learned these human preferences. We bridge this gap by (1) implementing a DeBERTa-based readability predictor on teacher difficulty assessments and (2) training an LLM which can appropriately provide outputs aligned to teacher assessments given an explicit prepended prompt or a prepended code to the input. We showcase that methods used for pretraining Salesforce's LLM CTRL can be applied to successfully finetune a much smaller Distilled GPT-2 LLM ($\sim$ 20x fewer parameters). We also showcase that a novel method leveraging the classifier as a discriminator for further training of a finetuned Distilled GPT-2 LLM leads to vastly improved performance, reaching accuracies above 98% on a test dataset of teacher preferences, noticeably beating out a prompted Distilled GPT-2 baseline which achieves an accuracy of roughly 55%. Additionally, our approaches can be easily applied to guide LLMs to provide constrained outputs aligned to other human preferences outside of text difficulty.

1 Introduction

LLMs can generate fluent, readable text relevant to several domains and use cases, but the output is remarkably difficult to fit to specific constraints. In CTG, a model aims to generate text satisfying constraints or characteristics such as topic or sentiment. While LLMs like ChatGPT generate natural and high quality text, controllability of these models often requires complex mechanisms during training or inference in order to guide these models towards meeting prescribed constraints [11].

In the wide breadth of CTG approaches, while there have been successes in controlling text towards objective measurements such as ensuring outputs contain specific vocabulary [12], research in controlling LLMs towards finer-grained and subjective measurements aligned with user intent (such as generating a review for a 3/5 rating) has been previously done with unwieldy and computationally intensive pretraining approaches for massively large LLMs like Salesforce's CTRL model [4], and controlling LLM outputs to match human preferences has often been limited towards reinforcement learning (RL) based approaches which require a learned policy [6; 7].

In this paper, we produce a difficulty-controllable text generation model trained on teachers' assessments of the difficulties of reading passages. We demonstrate that pretraining approaches leveraged for the CTRL LLM with 1.6 billion parameters can be successfully applied to finetune a Distilled GPT2 model of only 82 million parameters, and successfully produce (1) a regression model predicting difficulty scores, and (2) a generative language model producing text at a user-specified difficulty level. We additionally demonstrate that the alignment of the generative model towards the regression model can be significantly improved by a novel discriminator-based approach, adapted from the existing DisCup approach for guiding LLMs towards meeting an objective metric given by a binary classifier [10], leading to a model vastly outperforming a prompted Distilled GPT-2 model.

2 Related Work

2.1 Overview of Controlled Language Generation

Existing approaches to CTG range from finetuning existing LLMs, pretraining LLMs with specific architectures or labeled data, leveraging postprocessing or decode-time methods to modify outputs, and reinforcement learning based approaches to learn from human preferences and feedback [11]. Our approach is a finetuning approach which builds upon limitations from pretraining LLMs, decode-time methods, and prompt-tuning based approaches.

Decode-time methods like DExperts serve as a postprocessing layer to the LLM's outputs, and often leverage a discriminator or classifier as an "expert" to ensure generated tokens by a causal langauge model meet the intended constraints [5]. These approaches sometimes come at the cost of fluency and additionally dramatically increase decoding and inference time [8], motivating finetuning approaches like DisCup which use discriminator-based or adversarial learning for training instead of leveraging the discriminator during post-processing [10]. Pretraining methods like CTRL and CoCon learn to control textual outputs from prompts by leveraging control codes in inputs and control blocks in the transformer architecture, respectively [4; 1]. However, these existing methods often function over massive LLMs, require large amounts of data to function, and additionally have worse performance in handling a versatile range of tasks [11]. Prompt-tuning approaches like Tailor learn to embed tokens into an LLM's prompt to better align it to constraints [9]. However, such approaches often freeze the model's parameters during fine-tuning and necessitate additional training procedures to learn prompt embeddings—as well as decode-time methods to ensure the output is aligned to the prompt—leading to increased inference time. We circumvent many of these issues by avoiding pretraining, replacing prompt tuning with fixed prepended prompts or control codes, and leveraging a discriminator during the training process instead of during decode time, similar to approaches in CTRL and DisCup.

2.2 Controlled Language Generation with CTRL

One effective strategy for CTG, developed in CTRL [4], involves prepending "control codes" to training data before training time, in addition to a large amount of data appropriately aligned with these "control codes". For training, text is drawn from several different sources (numerous Reddit forums, Wikipedia, etc.), and is prepended with a code specifying its source (e.g. `Wikipedia`, `Books`, or `Politics` for text from r/Politics). Some multiply-specified pieces of text receive multiple control codes: e.g., ratings receive the control code `Reviews` together with a control code `Rating:`n specifying the numerical rating corresponding to the text. After prepending control codes, the large Transformer model is trained on the modified training data, and the resulting generative language model is successful in imitating domain-specific text (see Figure 1).

`Reviews Rating: 1.0\n\n`I bought this for my daughter and it was a complete waste of

Figure 1: CTRL embedding for reviews

2.3 Controlled Language Generation with DisCup

Instead of placing constraints on outputs at decode-time, DisCup controls text generation in a two-part process by leveraging a discriminator during the training process [10]. Specifically, the discriminator leveraged during DisCup is trained to provide a probability of an input i meeting some constraint

or attribute a. During the training process, from a training sequence of tokens (Step 1 of Figure 2), the top-k most probable next tokens following this sequence are provided, and each of these tokens are attached to form k sequences. These k sequences are then fed into the discriminator to assign a probability to each sequence of the constraint being met. From this, the k tokens are re-ranked according to their associated sequence probability (Step 2 of Figure 2). Finally, token logit probabilities of the model with control-prompt embeddings (Step 3 of Figure 2) along with the re-ranked distributions are used to guide the language model to produce logit rankings which are aligned with the re-ranked distribution, through a loss function from unlikelihood training.

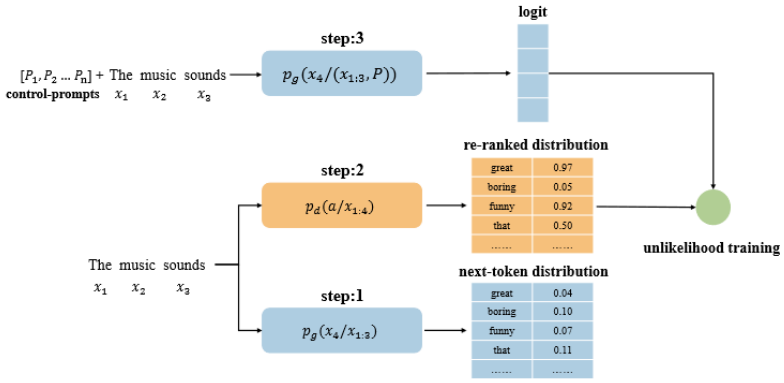

Figure 2: DisCup's training approach

3 Approach

3.1 Training DeBERTa-Based Discriminator Aligned To Teacher Preferences

We start by training a discriminator to predict BT-Easiness scores using regression over the training set. Given our training passages $P_1, \ldots, P_n$ with associated BT-Easiness scores $\gamma_1, \ldots, \gamma_n$, our objective is to train a model to find a real-valued function f minimizing the L^2 loss

$$\mathcal{L}(f; P_1, \ldots, P_n) = \sum_{i=1}^{n} (f(P_i) - \gamma_i)^2.$$

We finetune a pre-trained DeBERTa V3 Small model with a classification head for a single class with logits representing BT Easiness scores over reading passages with teacher-labelled BT Easiness scores to obtain our regression predictor f. This is a small Transformer model with 6 hidden layers with a size of 768, and 44M backbone parameters. We compare the performance of our discriminator against the average BT Easiness standard error from teachers.

3.2 Control Code and Prompt Embeddings with Distilled GPT-2

Inspired by pretraining methods for CTRL, we finetune a generative language model by prepending control codes for controlling the generation of excerpts of "easy" or "hard" text. Given a training dataset with excerpts labelled as "easy" or "hard", we prepend the control code <1> to the text of each easy passage and <2> to the text of each difficult passage. We then finetune a pretrained 82M parameter Distilled GPT-2 model to generate language similar to these training inputs.

For controllable language generation of this type, given a control code C and a sequence of tokens $x_1, \ldots, x_n$, the goal is to learn the conditional probability:

$$p(x|C) = \prod_{i=1}^{n} p(x_i|x_{<i}, C).$$

At generation time, the model should produce the next token x_{n+1} maximizing $p(x_1, \ldots, x_{n+1}|C)$. Performance is compared against a Distilled GPT-2 model with no fine-tuning.

As an alternative to the CTRL method, we also investigate prepending the data with a natural language prompt rather than a control code to guide outputs towards a difficulty. In this setup, we replace the control codes <1> and <2> by the phrases "This is written by a 1st grader." and "This is written by a PhD student". Finetuning proceeds in exactly the same way as with the control codes.

3.3 Discriminator-Based Training with Likelihood and Unlikelihood Training for Regression

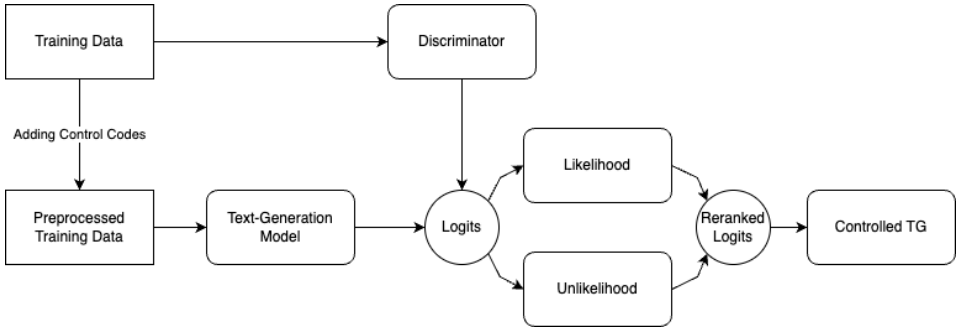

Figure 3: Discriminator-Based Training Pipeline

Taking inspiration from DisCup, we now fine-tune a text generation model using a modified loss function leveraging a discriminator f over training inputs with control codes or prepended prompts.

Let $S_1, \ldots, S_N$ be all the length-n sequences in the training data (we take $n = 12$). Each sequence S_j has a control code/prepended prompt C_j derived from the BT easiness score of the passage to which it belongs. Consider a single sequence $S_j = \{x_1, \ldots, x_n\}$, and consider the problem of generating x_n from $x_{<n}$. We wish for our model to predict an x_n which produces a sentence which is (1) fluent, and (2) conforms to constraint C_j. To ensure fluency, we restrict our attention to the top k (we let k=10) most likely tokens $y_1, \ldots, y_k$ for the n'th token. We then compute the BT Easiness scores for k concatenated sequences from the original sequence and the top-k tokens:

$$d_i = f(\{x_1, \ldots, x_{n-1}, y_i\}) \quad 1 \leq i \leq k.$$

d_i measures the difficulty of the sentence completed with y_i. Additionally, we use the logits (l^k) of the top k tokens for x_n and compute the softmax distribution of these k logits:

$$\mathbf{s}^k = \text{softmax}(l^k)$$

We define a target BT Easiness score for the given constraint C_j, which we denote t_{C_j}, and define

$$\mathcal{L}_{\text{likelihood}}(S_j) = \sum_{i=1}^{k} \mathbf{s}_i^k(|d_i - t_{C_j}|)$$

$$\mathcal{L}_{\text{unlikelihood}}(S_j) = \sum_{i=1}^{k} (1 - s_i^k)(-|d_i - t_{C_j}|)$$

to be the likelihood loss and the unlikelihood loss of a training sequence S_j. The likelihood loss guides top-k tokens with higher logit probabilities to produce sequences closer to the target BT Easiness score, while the unlikelihood loss guides top-k tokens with lower logit probabilities to produce sequences further away from the target BT Easiness score in order to prevent the model from only recommending tokens with improved discriminator scores and improving fluency.

3.4 Extending Discriminator-Based Training With Larger Windows of Generated Text

We further extend our previous approach by considering larger windows of generated text tokens in our likelihood and unlikelihood loss functions. For each input length-n sequence $S_j = \{x_1, ..., x_n\}$,

we consider a window of length w. We then define the loss functions for this sequence as follows:

$$\mathcal{L}_{\text{window likelihood}}(S_j) = \sum_{i=0}^{w-1} \mathcal{L}_{\text{likelihood}}(x_1, ..., x_{n-i})$$

$$\mathcal{L}_{\text{window unlikelihood}}(S_j) = \sum_{i=0}^{w-1} \mathcal{L}_{\text{unlikelihood}}(x_1, ..., x_{n-i})$$

This allows our discriminator-based training to more appropriately consider the performance of our model in generating multiple tokens, instead of a singular token, at the cost of more intensive training. We set w to be 3 and additionally lower k to 4 in order to offset the increased training computation.

4 Experiments

4.1 Data

Our datasets are derived from the CLEAR Corpus, which is an open-source dataset of over 5,000 grade-level reading passages annotated with various measures of reading difficulty. In our analysis, we focus on the BT Easiness score, the Bradley-Terry coefficient of the difficulty ratings of 1116 teachers [2]. If passages i and j have Bradley-Terry coefficients γ_i and γ_j, then

$$\mathbb{P}(\text{passage } i \text{ more difficult than passage } j) = \frac{\gamma_i}{\gamma_i + \gamma_j}.$$

Thus, higher Bradley-Terry coefficients correspond to more difficult passages. For our experiments, we consider three data categories: the original training dataset, the original test dataset, as well as an augmented training dataset consisting of all 50 word subsections of excerpts used in the original training dataset, with the same assigned BT Easiness score and standard error. We additionally split our training data into tertiles and let the BT Easiness upper bound of the lower tertile determine the threshold for a passage being considered hard from teacher preferences, and the lower bound of the upper tertile determinine the threshold for a passage being considered easy from teacher preferences. We show the total number of data entries, the number of easy passages, as well as the average BT Easiness Standard Error for the number of hard passages in Table 1 for each of our datasets.

Dataset	Total	Total Easy	Total Hard	Avg BT Easiness S.E.
Training Data Augmented	10997	3624	3695	0.4915
Test Data	1890	629	621	0.4909
Training Data	2834	944	944	0.4914

Table 1: Dataset statistics

4.2 Evaluation method

For our discriminators, we assess root mean squared error and mean accuracy error metrics.

For our CTG models, we generate text for evaluation by passing as input the first word in a data entry's excerpt prepended with the appropriate control code or phrase, and generate the next 30 tokens greedily to build an output. The resulting output (excluding the prepended control code or phrase) is then passed to our discriminator to provide a BT Easiness score, and this score is translated into the generated text being categorized as easy or difficult depending on whether the score is above the training data's median score. We assess these categorizations using accuracy, recall, and F1 metrics. We compute the root mean squared error and mean accuracy error metrics between the discriminator's outputted score and a target score. This target score is the minimum score of the training dataset if the intention is to produce a hard output, and otherwise the maximum score if the intention is to produce an easy output.

Finally, we compute a human evaluation for our final set of CTG models using control codes in Table 4. Two human evaluators are given 5 sets of 2 generated outputs, where each set contains two

generated outputs from one starting word in our test dataset prepended with either <1> or <2>. These evaluators give a fluency metric (Fluency column) out of 5 for each set of output, and try to assign each output to a control code, and the correctness of this is measured (H. Acc column).

4.3 Experimental details

All our models are trained using the AdamW optimizer correcting bias without any warmup steps, and are trained on a NVIDIA 3050 Laptop GPU. We train both our DeBERTa-Based discriminators and non-baseline CTG models on either the training data (denoted NDA) or the augmented training data (denoted DA) in Table 1, and all data entries are truncated to the first 60 tokens.

Our discriminators (denoted Regression) are trained with a batch-size of 64 over 10 epochs with a learning rate denoted in Table 2. We use the best performing discriminator in 2 (DA + Regression) to both evaluate and train (with discriminator-based training) our CTG models.

Our CTG models leveraging control codes (denoted CTRLF) or leveraging prepended prompts (denoted PromptedF) are trained with a batch-size of 64 over 10 epochs with a learning rate of 5×10^{-5}. The corresponding baselines without any finetuning are CTRLU and PromptedU.

Our CTG models in Table 3 trained with both likelihood and unlikelihood training (denoted FullDisc) are trained over 1 epoch with a batch-size of 8, with data entries truncated to the first 60 tokens, with a learning rate of 5×10^{-5}. Our CTG models in Table 4 leveraging either only likelihood training (denoted Likelihood), both likelihood and unlikelihood training (denoted FullDisc), only likelihood training with larger windows (denoted MultLikelihood), and both likelihood and unlikelihood training with larger windows (denoted MultFullDisc) are trained over 10 epochs from the NDA + CTRLF model, with a batch size of 8 with a a learning rate of 5×10^{-6}.

4.4 Results

Configuration	Learning Rate	MSE	RMSE	MAE
NDA + Regression	5×10^{-4}	0.4857	0.6951	0.5545
DA + Regression	5×10^{-4}	0.6149	0.7805	0.6272
NDA + Regression	5×10^{-5}	**0.3697**	**0.6052**	0.4819
DA + Regression	5×10^{-5}	0.3723	0.6077	**0.4818**

Table 2: Experimental results for aligned difficulty predictor on test set

We are able to achieve a mean accuracy error of 0.4818 for the test set with our best regression model in Table 2, which does even better than the average human BT Easiness standard error of 0.4909 reported by the CLEAR dataset 1, indicating great discriminator performance.

Configuration	MSE	RMSE	MAE	Accuracy	Recall	F1
CTRLU	7.6770	2.7578	2.6754	0.4916	0.4868	0.4065
PromptedU	6.9567	2.6264	2.5609	0.5565	0.5584	0.4710
NDA + CTRLF	5.2368	2.2649	2.1308	0.7322	0.7329	0.6789
DA + CTRLF	6.0774	2.4386	2.3049	0.6408	0.6222	0.5355
NDA + PromptedF	4.5918	2.1396	2.0525	**0.8691**	**0.8706**	0.8252
DA + PromptedF	**4.1663**	**2.0359**	**1.9523**	0.8663	0.8639	**0.8330**
NDA + CTRL + FullDisc	7.2265	2.6881	2.6864	0.5267	0.5302	0.3972
DA + CTRL + FullDisc	7.1767	2.6786	2.6769	0.5417	0.5565	0.4923
NDA + Prompted + FullDisc	7.2164	2.6849	2.6771	0.4972	0.5058	0.3217
DA + Prompted + FullDisc	7.3084	2.7019	2.6965	0.4948	0.5000	0.3107

Table 3: Experimental results for baselines and various aligned language models on test set

We also see that our baseline GPT-2 models prompted with control codes (CTRLU) and with special prompt phrases (PromptedU) have an accuracy close to random guessing with values of 0.4916 and 0.5565 respectively, which is far poorer than the accuracy from finetuned models like DA + CTRLF and NDA + PromptedF. While the CTRLU baseline is not particularly surprising (a baseline

GPT-2 model should not know how to respond to an arbitrary control code), the PromptedF baseline mimics how a user would typically prompt GPT-2 in order to generate language of either hard or easy difficulty. Additionally, we surprisingly find that applying discriminator-based methods on the pretrained model over 1 epoch leads to even poorer performance than baselines.

Configuration	RMSE	MAE	Accuracy	Recall	F1	H. Acc.	Fluency
NDA + CTRLF	2.2649	2.1308	0.7322	0.7330	0.6789	**1**	4.15
NDA + CTRLF + Likelihood	2.0276	**1.9282**	0.9595	0.9608	0.9341	0.4	2.45
NDA + CTRLF + FullDisc	2.9013	2.8946	0.0288	0.0321	0.0278	0.4	**4.55**
NDA + CTRLF + MultLikelihood	**2.0158**	1.9939	**0.9867**	**0.9885**	**0.9739**	0.95	4.1
NDA + CTRLF + MultFullDisc	2.9790	2.9374	0.2804	0.2685	0.2104	0.6	1.55

Table 4: Results for CTRL and discriminators, including human assessments

Using our finetuned model NDA + CTRLF as a baseline, we show that further likelihood training over 10 epochs with our discriminator, both with or without extended token windows, leads to vastly improved performance with good human accuracy, boasting accuracy more than 0.98. Most of our models fine-tuned with discriminators also maintain good fluency measurements. Surprisingly, we find that additionally incorporating unlikelihood training can even lead text to gravitate towards the opposite of its intended constraints, likely due to how it can encourage lower discriminator scores.

5 Analysis

Of the five models in Table 4, two score highly both on human-evaluated measures of accuracy and fluency, namely the NDA + CTRLF model and the NDA + CTRLF + MultLikelihood model. Table 5 below shows some example outputs of these two models:

Input	NDA + CTRLF + MultLikelihood Output	NDA + CTRLF Output
Input:<1>	The next day, when the sun was setting, the little girl went out to play. She was very hungry. She went to bed. She went	The first thing I remember about the first day was that I was in a coma. I was very sleepy. I was very sleepy. I was very
Input:<2>	The first step in the construction of a battery is to obtain a suitable temperature of 1.5°C. The first step is to obtain a suitable	The first step in the process of determining the composition of the gas is to determine the composition of the gas by the ratio of the two elements in the
Input:<1> A beginning	A beginning-day boy called "Bobby" and said, "Bobby is going to school today. He is going to school today. He is going'	A beginning-day visit to the village of Sarnia was a very pleasant one. The children were very fond of the village. They were very fond of
Input:<1> A beginning is the time for taking the most delicate care	A beginning is the time for taking the most delicate care of your baby. You will take care of your baby's food and water	A beginning is the time for taking the most delicate care of your baby. You will not be surprised if your baby is not very happy. You will not be surprised if your baby is not very happy.
Input:<2> A beginning	A beginning of the first phase of the study of the nervous system is the development of a nervous system. The nervous system consists of a series of nerves, called	A beginning of the first century B.C. was a city of about 300,000 people. It was the center of the British Empire, and was the
Input:<2> A beginning is the time for taking the most delicate care	A beginning is the time for taking the most delicate care of the body. The body is made up of a single piece of tissue, called a tube, which is filled with blood. The tube is filled	A beginning is the time for taking the most delicate care of the body. The body is not only made up of a single piece of tissue, but is also made up of a whole body of cells

Table 5: Comparison of text generation with different control codes and prompt truncation lengths

As evidenced by the table, inputs with control code <2> produce saliently more difficult outputs than inputs with control code <1>, as desired, noticeably containing more specific terms like "nervous system". However, because of the greedy text-generation strategy, many of the outputs have repeated text; for longer output generation, a repetition penalty should be implemented. Additionally we see that our NDA+CTRLF model sometimes fails to make text easy: its outputs can contain relatively difficult words like "coma", contributing to its lower accuracy in Table 4.

Interestingly, we find that outputs of the model are recognizably in the form either of fiction or of a piece of scientific information from a book, likely an artifact of the reading comprehension passages on which the model was finetuned. However we find these book-like outputs do not appear on a pretrained model, since feeding inputs like "A beginning" and "He was the" to a pretrained Distilled GPT-2 model produces general phrases like "A beginning of the year" or "He was the first person to be arrested for a hate crime in the United States". This leads us to believe that finetuning should be done on a multitude of corpora covering a range of textual data. Such corpora could be generated with language models like GPT-4 in order to automatically label while maintaining language variety, as done in TinyStories [3].

Finally, we observe that despite good fluency of outputs, some outputs by our models do not make factual sense, like the phrase "The body is made up of a single piece of tissue, called a tube, which is filled with blood". This is likely due to the poor reasoning qualities of an LLM of this size, compounded with the emphasisis on stitching together language for it to be deliberately difficult. We wonder if leveraging larger LLMs can prevent these factual inaccuracies, and additionally believe an assessment of factual accuracy would be useful in understanding tradeoffs from imposing greater controllability of text.

5.1 Investigating unlikelihood training

Due to the surprisingly poor accuracy given from models incorporating unlikelihood training like NDA + CTRLF + MultFullDisc and NDA + CTRLF + FullDisc over our evaluations, we peer into some example inputs in our evaluation set, along with their generated outputs.

Most significantly, we find that NDA + CTRLF + MultFullDisc fails to generate any coherent sentences and suffers heavily from repetition, producing phrases such as “What was it then? Rome was Rome; Rome was Rome; Rome was Rome; Rome was Rome; Rome was Rome; Rome was Rome; Rome was” and “It was a country people dwelt in America was Rome; Rome dwelt Rome dwelt Rome; Rome dwelt Rome dwelt Rome; Rome dwelt”. Additionally, we find a repeated “Rome" and “America" in almost all generated textual inputs, showcasing that unlikelihood training is somehow forcing the model to latch onto particular words in order to meet the discriminator’s score. We see very similar responses irrespective of control codes.

While NDA + CTRLF + FullDisc does not showcase the same level of repetition with specific words “Rome” and “America”, we still observe generally greater repetition with many phrases being repeated, with generated phrases like “What is the name of the animal? It is a very common name for the animal. It is a very common name for the animal. It is a”. Additionally, the model seems to present easy outputs regardless of provided control code, as it presents a very simple sentence “Crows, and the other birds, were all very happy. They were all very happy. They were all very happy. They were all very happy. They were” as one given output for a hard control code. These findings emphasize to us the necessity to explore unlikelihood training being done with an explicit repetition penalty, and some level of regularization to prevent overfitting to particular words.

6 Conclusion

In this paper we showcase how language models can be finetuned to control their outputs in alignment to user intent. We specifically demonstrate successes with a Distilled GPT-2 LLM being able to control the human-specified difficulty of generated text given prepended prompts or control codes from the user. To do so, we leverage approaches used to pretrain Salesforce’s CTRL model, and develop novel discriminator-based methods taking inspiration from existing literature.

Due to the limited memory of the 3050 GPU used to finetune our models, the size of models used for experimentation as well as the amount of training iterations was limited. Additionally, we note that our results are demonstrated over limited control codes and prompts, and also only demonstrate control with respect to the difficulty of generated English text.

As a result, we believe future work should explore our finetuning methods with larger language models and applying control over different human-specified metrics. Finally, due to repetitions in observed outputs and poor performance of our unlikelihood training approach, we believe repetition penalties and a modified unlikelihood training scheme should be investigated in future work.

7 Key Information to include

Mentor: Tony Wang. This project has no external collaborators and is not shared. All group members made equal contributions with work evenly split among coding, writing, and researching literature.

References

[1] Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2022. Cocon: A self-supervised approach for controlled text generation.

[2] Scott Crossley, Aron Heintz, Joon Suh Choi, Jordan Batchelor, Mehrnoush Karimi, and Agnes Malatinszky. 2023. A large-scaled corpus for assessing text readability.

[3] Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english?

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

[5] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts.

[6] Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041, Online. Association for Computational Linguistics.

[7] Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. Quark: Controllable text generation with reinforced unlearning.

[8] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Discriminative adversarial search for abstractive summarization.

[9] Kexin Yang, Dayiheng Liu, Wenqiang Lei, Baosong Yang, Mingfeng Xue, Boxing Chen, and Jun Xie. 2022. Tailor: A prompt-based approach to attribute-based controlled text generation.

[10] Hanqing Zhang and Dawei Song. 2022. Discup: Discriminator cooperative unlikelihood prompt-tuning for controllable text generation.

[11] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models.

[12] Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions.

A Appendix

We explore different hyperparameters for CTRLF tuning, and find relatively comparable performance regardless of learning rate or data augmentation. We believe more training epochs are necessary to get stable performance:

Configuration	MSE	RMSE	MAE	Accuracy	Recall	F1
DA + CTRLF, lr=5×10^4	5.8219	2.3860	2.2574	0.7152	0.7219	0.6654
NDA + CTRLF, lr=5×10^4	6.6521	2.5528	2.4245	0.6666	0.6760	0.6111
DA + CTRLF, lr=5×10^5	6.3065	2.4807	2.3407	0.6675	0.6792	0.6071
NDA + CTRLF, lr=5×10^5	6.4047	2.5020	2.3724	0.6855	0.7012	0.6357

Table 6: Test results over hyperparameter search for CTRLF after 10 epochs of training

We explore training of our discriminator-tuned models in Table 4. We find that unlikelihood training seems to lead to overfitting on the training set, as while training metrics hover close to 1, test metrics dramatically fall to near 0.

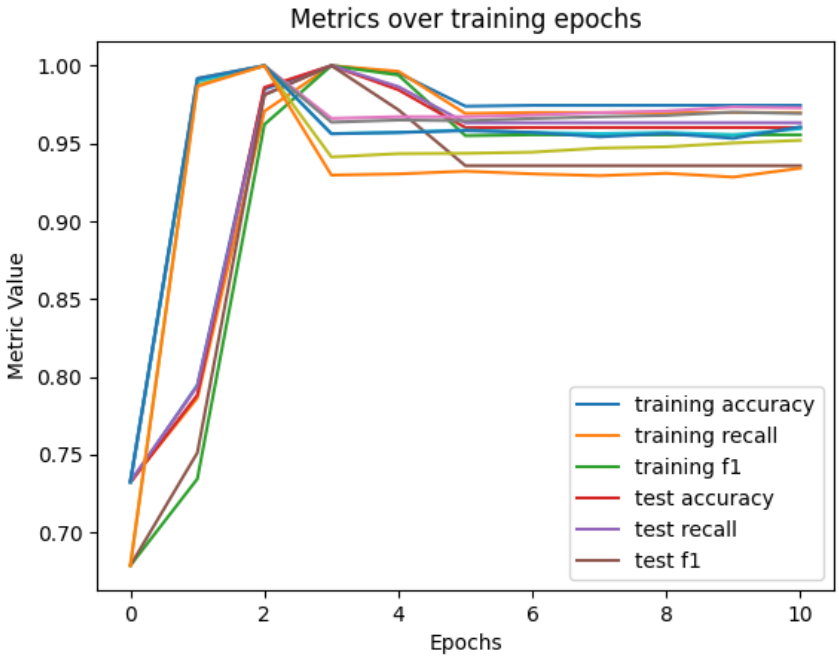

Figure 4: NDA + CTRLF + Likelihood Training

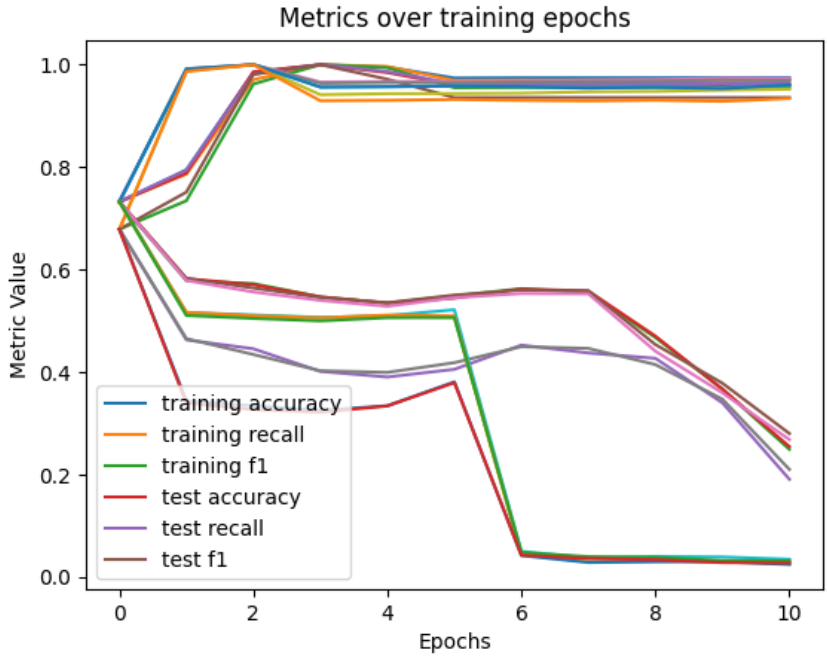

Figure 5: NDA + CTRLF + FullDisc Training

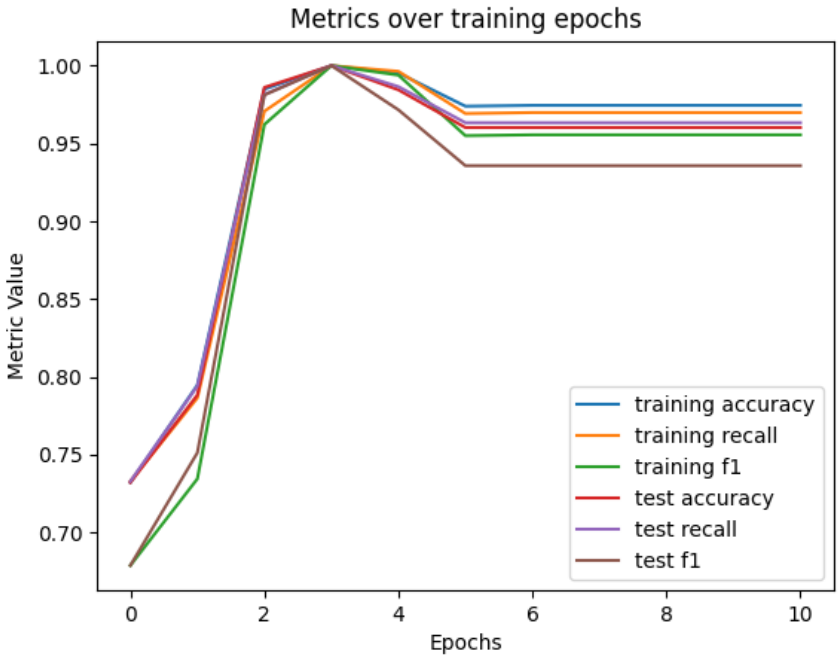

Figure 6: NDA + CTRLF + MultLikelihood Training

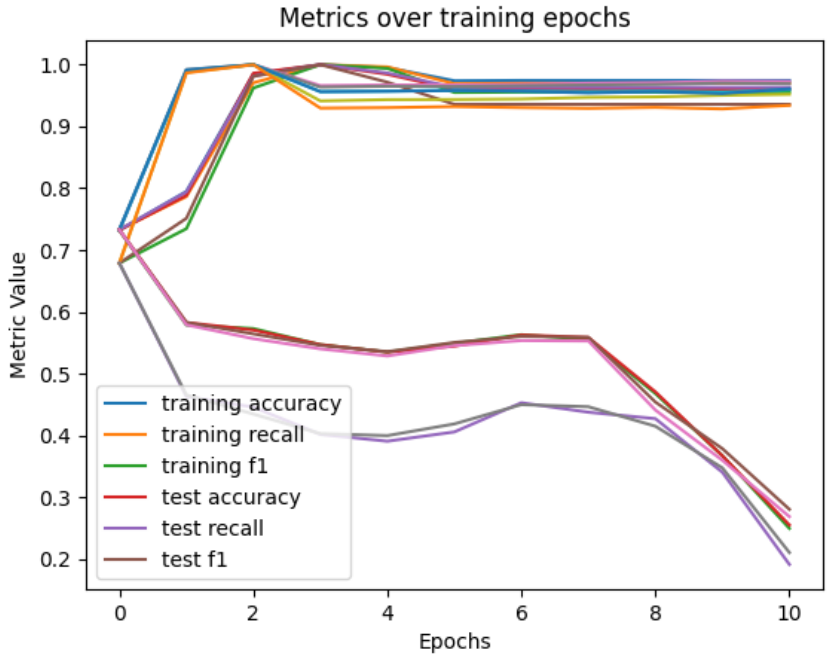

Figure 7: NDA + CTRLF + MultFullDisc Training