Controlling Conditional Language Models without Catastrophic Forgetting

Tomasz Korbak [*1] **Hady Elsahar** [2] **Germán Kruszewski** [2] **Marc Dymetman** [2]

Abstract

Machine learning is shifting towards general-purpose pretrained generative models, trained in a self-supervised manner on large amounts of data, which can then be applied to solve a large number of tasks. However, due to their generic training methodology, these models often fail to meet some of the downstream requirements (e.g., hallucinations in abstractive summarization or style violations in code generation). This raises the important question of how to adapt pre-trained generative models to meet all requirements without destroying their general capabilities ("catastrophic forgetting"). Recent work has proposed to solve this problem by representing task-specific requirements through energy-based models (EBMs) and approximating these EBMs using distributional policy gradients (DPG). Despite its effectiveness, this approach is however limited to unconditional distributions. In this paper, we extend DPG to conditional tasks by proposing Conditional DPG (CDPG). We evaluate CDPG on four different control objectives across three tasks (translation, summarization and code generation) and two pretrained models (T5 and GPT-Neo). Our results show that fine-tuning using CDPG robustly moves these pretrained models closer towards meeting control objectives and — in contrast with baseline approaches — does not result in catastrophic forgetting.

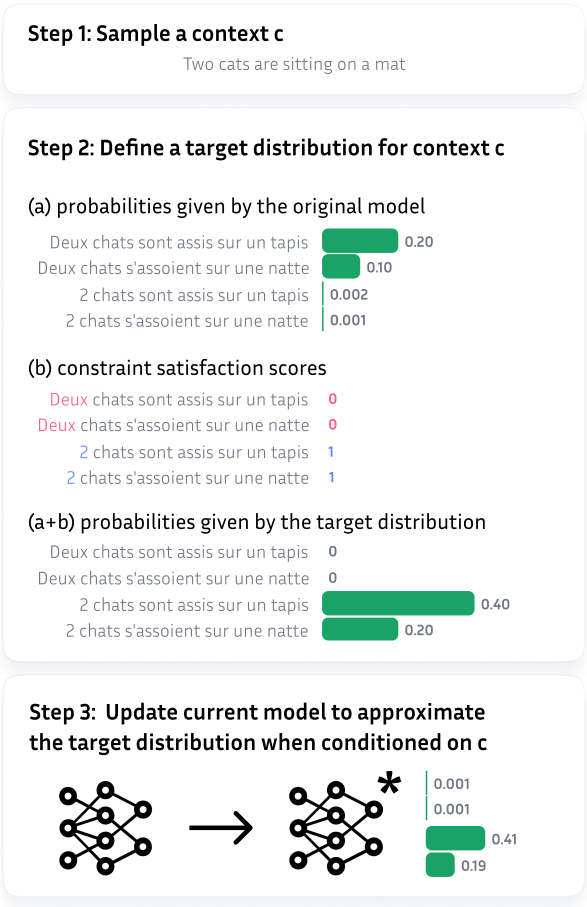


Figure 1: An overview of our algorithm for fine-tuning conditional language models, illustrated on the terminology-constrained translation task (see Section 3.2 for details).

1. Introduction

Pretrained generative models are shifting the landscape of machine learning research and practice. General purpose models such as the GPT family (Radford et al., 2019; Brown et al., 2020; Black et al., 2021), T5 (Raffel et al., 2020), CLIP (Radford et al., 2021) and Codex (Chen et al., 2021) are trained in a self-supervised manner on large amounts of uncurated data and can then be adapted to specific downstream tasks (e.g. generating Python code) or control objectives (e.g. controlling the style of generated code). Frequently, control objectives are motivated by the desire to address the shortcomings of certain pretrained models. These can be due to the uncurated nature of the original training data (e.g. a large portion of Python source code on the Internet violates PEP8 (van Rossum et al., 2001), the Python Style Guide) or the difficulty of learning a desired behaviour by purely self-supervised training (e.g.

*Work done during an internship at Naver Labs Europe. [1]University of Sussex [2]Naver Labs Europe. Correspondence to: Tomasz Korbak <tomasz.korbak@gmail.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

there is not enough training signal to ensure that a model trained on source code always generates compilable code or that a summarization model always produces factually correct summaries).

The practice of adapting and controlling pretrained generative models poses two open problems. The first is that control objectives frequently lack ground truth data that could be used for supervised fine-tuning; in general, we are only given an indicator $b(x)$ of whether a given sample x from the model satisfies a given control objective. Thus, this restriction calls for approaches that can employ this signal such as reinforcement learning (RL) (Pasunuru & Bansal, 2018; Ziegler et al., 2019), weighted decoding (Ghazvininejad et al., 2017; Holtzman et al., 2018; See et al., 2019) or decoding with perturbed activations (Dathathri et al., 2020).

The second problem is *catastrophic forgetting*: most approaches to enforcing a control objective result in a dramatic loss of capabilities of the original model beyond the scope of the control objective. Notably, there exists one approach largely avoiding both of these problems (Parshakova et al., 2019a; Khalifa et al., 2021): representing the control objective as an energy-based models (EBM) and approximating that EBM using distributional policy gradients (DPG). Although this approach shows great improvements in controlling pretrained language models while avoiding catastrophic forgetting (Khalifa et al., 2021), it is limited to unconditional generation tasks and cannot finetune conditional models that are behind the most impactful NLP tasks such as machine translation, summarization or dialogue systems.

In this paper, we present Conditional DPG (CDPG), an extension of DPG that can approximate *conditional* EBMs. A conditional EBM $\mathcal{P}$ defines an unnormalized distribution for each context c (e.g. a source document). Extending the approach of Khalifa et al. (2021) such a conditional EBM represents the ideal behaviour of a generative model, given context c, as the distribution that incorporates the control objectives while remaining as close as possible to the original distribution to prevent catastrophic forgetting. This corresponds to defining multiple distributions p_c indexed by c, where each is the normalization of an unconditional EBM P_c following Khalifa et al. (2021). We then define the training objective for the conditional model based on minimizing the *average* divergence for each p_c.

We demonstrate the effectiveness of CDPG in addressing shortcomings of pretrained generative models by considering three tasks: translation, summarization and code generation; and two corresponding pretrained generative models: T5 (Raffel et al., 2020) and GPT-Neo (Black et al., 2021).

We start by demonstrating the effectiveness of CDPG on a toy control objective for translation: ensuring that numeral nouns (e.g. "two") are translated as digits (e.g. "1") while other aspects of translation are unchanged. This problem is an simple instance of broader challenge of incorporating prior information in neural translation models. CDPG is able to make samples satisfying the constraint 116 times more likely.

For summarization, a similar big, unsolved problem is ensuring that summaries are factually faithful to source documents given that summarization models are prone to hallucinating named entities never mentioned in the source (Maynez et al., 2020). We show that a preference for factually faithful summaries (operationalized as entity-level factual consistency (Nan et al., 2021)) can be represented by a conditional EBM. Then, we show that using CDPG to finetune T5 to approximate this EBM increases the number of correct and relevant named entities in summaries and improves T5's Rouge score. In contrast with RL approaches, CDPG does not degrade the diversity and quality of summaries.

For code generation, we consider the task of generating a Python function given its signature (name and arguments). While general-purpose language models can generate idiomatic Python functions (Chen et al., 2021; Austin et al., 2021), they may still struggle to learn some desirable properties of generated code. For instance, a Python function generated by GPT-Neo will compile only 40% of the time and will contain on average 4 violations of PEP8. We show that using CDPG to approximate a conditional EBM expressing corresponding constraints improves both compilability and PEP8-compliance without hurting the diversity of generated Python code or leading to degeneration (Holtzman et al., 2020).

The contributions of this paper are as follows:

1. We introduce a formal framework for representing control objectives for *conditional* generative models as *conditional* EBMs while alleviating *catastrophic forgetting*,

2. We design CDPG, an extension of DPG suitable for approximating conditional EBMs,

3. We evaluate CDPG on three control objectives across three tasks: machine translation with number format constraints, summarization constrained to be factually correct, code generation constrained to generate compilable Python functions and code generation constrained to respect PEP8.

Code accompanying the paper will be available at `https://github.com/naver/gdc`.

2. Method

Unconditional EBMs A standard, "unconditional" EBM is a function P from a (discrete, i.e. finite or countable) space X to the non-negative reals, such that the partition function $Z \doteq \sum_{x \in X} P(x)$ is strictly positive and finite. We will denote by lowercase p the normalized distribution over X associated with P, namely $p(x) \doteq P(x)/Z$. Khalifa et al. (2021) show that the problem of fine-tuning a pretrained model $a(x)$ to satisfy a control condition $b(x) = 1 \; \forall x \in X$, where $b(x)$ is a binary scorer for a desired feature, while minimizing the divergence to the original a(x) has a unique solution given by the probability distribution p associated with the EBM

$$P(x) = a(x)b(x). \tag{1}$$

In information-geometric terms, p is the I-projection of a onto the manifold of all distributions satisfying the constraint given by b (Csiszár & Shields, 2004).

Conditional EBMs Let us now consider a discrete, potentially infinite set C of conditions c. Formally, $\mathcal{P}$, a *conditional* EBM over C, is defined as a function from C to the set of unconditional EBMs over X, in other words, a function that maps each $c \in C$ to an unconditional EBM P_c over X:

$$\mathcal{P} : c \mapsto P_c, \tag{2}$$

$$P_c : x \mapsto \mathbb{R}_+. \tag{3}$$

We denote by Z_c the partition function of $P_c(x)$, namely: $Z_c \doteq \sum_{x \in X} P_c(x)$, and by $p_c(x)$ the normalized version of $P_c(x)$, namely: $p_c(x) \doteq P_c(x)/Z_c$.

Representing constraints as conditional EBMs The problem of fine-tuning a pretrained model $a(x|c)$ to satisfy a control objective (e.g. generating factually correct summaries) can be seen as a constraint satisfaction problem: finding a model $p_c(x)$ that meets the demands of the control objective but at the same time stays as close as possible to the original pretrained model $a(x|c)$. We represent such an optimal model as an unconditional EBM $P_c(x)$. A control objective can be defined in terms of a binary scorer $b(x, c)$ such that $b(x, c) = 1$ if a sample (c, x) satisfies a constraint given by a control objective (e.g. x is factually correct with respect to c) and $b(x, c) = 0$ otherwise. Let us consider a set of contexts C. For each $c \in C$, we can frame the problem of finding the unique model $p_c(x)$ such that (i) $b(x, c) = 1$ for all samples $x \sim p_c(x)$, and (ii) $p_c(\cdot)$ has minimal KL divergence from $a(\cdot|c)$ as an instance of the unconditional case already considered by Khalifa et al. (2021). Following our example, p_c could be a distribution over factually correct summaries of c as similar as possible to a distribution over summaries which the original model a would produce for a document c. Therefore, p_c can be represented as an unconditional EBM $P_c(x)$ of the following form:[1]

$$P_c(x) \doteq a(x|c)b(x, c). \tag{4}$$

Approximating conditional EBMs While $\mathcal{P}$ represents the target conditional model optimally reconciling distance from $a(x|c)$ and the control objective, sampling and MAP decoding for $\mathcal{P}$ is intractable for two reasons. First, $\mathcal{P}$ actually represents a potentially infinite collection of unconditional models of the form $p_c(\cdot)$. Second, each of these unconditional models still cannot be easily sampled from because they do not admit an autoregressive factorisation: $b(x, c)$ is only defined for the whole sequence x.

The second problem was addressed by (Parshakova et al., 2019b) and (Khalifa et al., 2021) who used the distributional policy gradients (DPG) algorithm to approximate unconditional EBMs p using a *new* unconditional model π_θ trained to minimize the cross-entropy $\mathrm{CE}(p, \pi_\theta)$. Unfortunately, DPG is not directly usable for a conditional model covering many contexts c.

To address these problems, we instead try to find a *single* seq2seq model π_θ approximating p *on average* across contexts. Concretely, we minimize the expected cross-entropy between π_θ and multiple p_c's:

$$\mathcal{L}(\theta) = \mathbb{E}_{c \sim \tau(c)} \mathrm{CE}(p_c(\cdot), \pi_\theta(\cdot|c)), \tag{5}$$

where the expectation is over $\tau(c)$, a distribution over $c \in C$. The gradient of this objective takes the following form:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{c \sim \tau(c)} \nabla_\theta \mathrm{CE}(p_c(\cdot), \pi_\theta(\cdot|c)) \tag{6}$$

$$= -\mathbb{E}_{c \sim \tau(c)} \mathbb{E}_{x \sim p_c(x)} \nabla_\theta \log \pi_\theta(x|c) \tag{7}$$

$$= -\mathbb{E}_{c \sim \tau(c)} \mathbb{E}_{x \sim \pi_\theta(x|c)} \frac{p_c(x)}{\pi_\theta(x|c)} \nabla_\theta \log \pi_\theta(x|c), \tag{8}$$

$$= -\mathbb{E}_{c \sim \tau(c)} \mathbb{E}_{x \sim \pi_\theta(x|c)} \frac{P_c(x)}{Z_c \pi_\theta(x|c)} \nabla_\theta \log \pi_\theta(x|c), \tag{9}$$

where in (8) we applied importance sampling from π_θ and (9) expresses p_c in terms of unconditional EBM P_c and its partition function Z_c.[2]

[1] Khalifa et al. (2021) provide a more general, exponential family form of this EBM. The product-of-experts (Hinton, 2002) form in (4) is a special case of the exponential form, see (Khalifa et al., 2021, Appendix A.2).

[2] This last form of the gradients estimate bears some resemblance to policy gradients methods in reinforcement learning (Sutton et al., 1999) with term $P_c(x)/Z_c\pi_\theta(x|c)$ playing the role of a pseudoreward. This similarity, however, is fairly superficial: our minimization objective (5) is expected cross-entropy (over contexts), not expected (negative) reward. See (Korbak et al., 2022) for extended discussion.

We approximate both expectations in (9) by sampling. Intuitively, this corresponds to building unconditional EBMs $P_c(\cdot)$ on the fly for each $c \sim \tau(c)$, computing the EBM "score" $P_c(x)$ for each sample from the seq2seq model $x \sim \pi_\theta(\cdot|c)$ and then using this score as a "pseudoreward" term $P_c(x)/(Z_c\pi_\theta(x|c))$ in the policy gradient estimate.

Algorithm 1 Conditional DPG (CDPG)

Input: conditional EBM $P_c(x)$, initial model $a(x|c)$
  $\pi_\theta \leftarrow a$
  for each iteration **do**
    $\mathcal{B} \leftarrow \{\}$
    sample batch $\{c_1, ..., c_i, ..., c_N\}$ from $\tau(c)$
    for each c_i **do**
      sample batch $\{x_1, ..., x_j, ..., x_M\}$ from $\pi_\theta(x|c_i)$
      $\hat{Z}_{c_i} = \frac{1}{M}\sum_{j=1}^{M} \frac{P_{c_i}(x_j)}{\pi_\theta(x_j|c_i)}$
      for each x_j **do**
        $\mathcal{B} \overset{+}{\leftarrow} (x_j, c_i, \hat{Z}_{c_i})$
    for $(x, c, \hat{Z}_c)$ in shuffle($\mathcal{B}$) **do**
      $\theta \leftarrow \theta + \alpha^{(\theta)} \frac{1}{\hat{Z}_c + \epsilon} \frac{P_c(x)}{\pi_\theta(x|c)} \nabla_\theta \log \pi_\theta(x|c)$

Output: π_θ

Estimating Z_c The one term in (9) that remains difficult to evaluate is the partition function Z_c. For a single unconditional EBM, the (Khalifa et al., 2021) approach had no need to evaluate the partition function Z as it just scaled gradient estimates and therefore, Z could be absorbed into the learning rate. In the conditional case, Z_c varies with c. Therefore for each c, we compute Z_c using a batch of M samples $\{x_1, \ldots, x_j, \ldots x_M\}$ from $\pi_\theta(x|c_i)$. Then, Z_c is estimated using importance sampling by reweighting samples x_j by their likelihood according to $\pi_\theta(\cdot|c_i)$.

Training loop We train π_θ by stochastic gradient descent using the gradient estimate (9). At each epoch, we first sample N contexts c and then, for each c, M samples x from $\pi_\theta(x|c)$. We maintain a buffer $\mathcal{B}$ storing each (c_i, x_j) pair along with its corresponding partition function Z_{c_i}. Finally, we shuffle $\mathcal{B}$ and iterate over it to perform gradient steps using (9) with learning rate $\alpha^{(\theta)}$. The procedure is summarized in Algorithm 1. $\alpha^{(\theta)}$, N and M are hyperparameters. Values used in experiments are reported in Tables 1-2 in the Appendix.

3. Experiments

We evaluate CDPG as well as three baselines on four control objectives across three tasks: translation, summarization and code generation. Each task is associated with C_{train}, a set of contexts c used for prompting the model: these are English source sentences for translation, Python function signatures in case of code generation and source documents in case of summarization. When computing evaluation metrics, we sample contexts from a held out set C_{test} not used for training. In addition to that, for each experiment, we measure $\mathbb{E}_{c\sim\tau(c)} D_{\text{KL}}(p_c, \pi_\theta)$, the expected forward KL divergence from the optimal distribution p_c, as well as $\mathbb{E}_{c\sim\tau(c)} D_{\text{KL}}(\pi_\theta, a)$, the expected reverse KL divergence from the original pretrained model.[3].

3.1. Baselines

DPG-like ablation We compare our algorithm with an ablation (labeled as "DPG" on all figures) that sets Z_c in the denominator of (9) to a constant Z which is the running mean of $P_c(x)$ across x's and c's. This ablation resembles the original DPG algorithm for unconditional EBMs developed by (Parshakova et al., 2019a) and extended by (Khalifa et al., 2021). While the partition function is constant for unconditional EBMs, in conditional EBMs Z_c varies with c. Therefore, the DPG-like ablation performs gradient updates using biased gradient estimates.

Reinforcement learning The problem of fine-tuning a pretrained model to satisfy a pointwise constraint $b(x, c)$ can be posed as maximising tbe expected reward $\mathbb{E}_{c\sim\tau(c)}\mathbb{E}_{x\sim\pi_\theta(x|c)} R(x, c)$. We consider two instances of this approach: Reinforce (Williams, 1992) and Ziegler (Ziegler et al., 2019). For Reinforce, we simply define $R(x, c) = b(x, c)$. Ziegler prevents too large departures from a by adding a KL penalty term and defining $R(x, c) = b(x, c) - \beta D_{\text{KL}}(\pi_\theta, a)$, where β is a hyperparameter updated using an adaptive schedule.

3.2. Translation

Dataset For the translation task, $\tau(c)$ from Algorithm 1 is a uniform distribution over a fixed set of English sentences. We sampled 5k English sentences containing numeral nouns from the English-French subcorpus of the Europarl dataset, version 7 (Koehn, 2005). Metrics are computed for generated translations of another set of 5k English sentences from the test split of Europarl. Note that neither CDPG nor the baselines utilise ground truth translations (references); we compute $b(x, c)$ based on source documents and *generated* translations. Ground-truth translations are only used for evaluating the BLEU score of generated translations.

Model We conduct our experiments on the T5 architecture (Raffel et al., 2020), using the pre-trained model `t5-small` as π_θ. During fine-tuning, we generate translations x conditioned on a source sentence c by pure ancestral sampling from π_θ. For evaluation, we follow the setup described by (Raffel et al., 2020) and use beam search decoding with beam size 4.

[3] See Appendix A for details of metrics calculations.

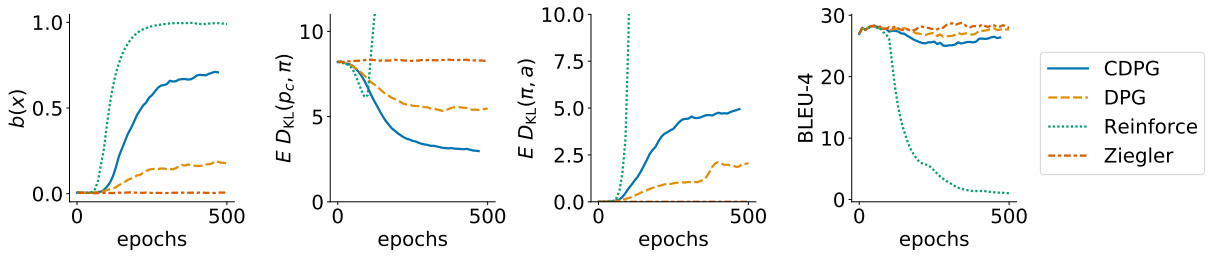


Figure 2: Translation with terminology constraint. Evaluation metrics: fraction of samples satisfying constraint $b(x)$ (↑ better), expected $D_{\mathrm{KL}}(p_c, \pi_\theta)$ (↓ better) and $D_{\mathrm{KL}}(\pi_\theta, a)$ (↓ better), and BLEU-4 score (↑ better) for models obtained from fine-tuning with conditional DPG, DPG, Ziegler and Reinforce.

Metrics In addition to measuring expected $D_{\mathrm{KL}}(p_c, \pi_\theta)$ and $D_{\mathrm{KL}}(\pi_\theta, a)$, we evaluate the forgetting of T5's capabilities in terms of BLEU-4 score (Papineni et al., 2002), a measure of translation quality understood as overlap between generated and ground-truth translation.

Constraint We implement the constraint scorer as table lookup: $b(x, c) = 1$ if for every occurrence of a given numeral noun (e.g. "two") in a source sentence c, a corresponding digit ("2") occurs in its translation x. Otherwise, $b(x, c) = 0$.

Results We present the results of the translation task on Figure 2. Initial constraint satisfaction is very low: 0.006. Intuitively, it's very unlikely for T5 to translate "two" as "2", not as "deux". However, CDPG is able to boost that number to 0.7 and reduce the expected divergence from its target distributions p_c almost twofold, outperforming DPG by a wide margin. It also outperforms Reinforce by staying closer to the original distribution a and not suffering from almost any drop in BLEU-4 score. (Note that some drop is necessary for satisfying the constraint, because ground truth translations with respect to which BLEU-4 is computed almost don't satisfy the constraint themselves.) In contrast, Reinforce improves constraints satisfaction only at the cost of heavy divergence from a: it learns to append all the digits at the end of each translation, thus ensuring constraint satisfaction (see Appendix C.1 for sample translations). This is reflected in a catastrophic drop in BLEU-4 score. Ziegler, on the other hand, fails to improve constraint satisfaction and stays *too close* to the original distribution a.

3.3. Summarization

Dataset To conduct our summarization experiments, we use the CNN/DailyMail dataset (Nallapati et al., 2016) and sample 5k source documents from the train and test subsets to use for fine-tuning and evaluation, respectively. We use ground truth summaries only for computing reference-based evaluation metrics such as ROUGE score or recall-target. Ground truth summaries are not used in training.

Model We use the same model as in the translation task (`t5-small`). For fine-tuning, we generate summaries x conditioned on a source document c by pure ancestral sampling from π_θ; for evaluation, we use beam search with beam size 4.

Constraints Following (Nan et al., 2021), we define an entity-level factual consistency constraint as a product of two constraints: there must be at least four named entities in the summary x and all the named entities x must have occurred in the source c. More formally, let $\mathrm{NER}(\cdot)$ denote the set of named entities found in a text and $|\cdot|$ the number of elements of a set. Then, $b(x, c) = 1$ iff $[|\mathrm{NER}(x)| \geq 4] \wedge [\mathrm{NER}(x) \subseteq \mathrm{NER}(c)]$ and $b(x, c) = 0$ otherwise.

Metrics In addition to measuring expected $D_{\mathrm{KL}}(p_c, \pi_\theta)$ and $D_{\mathrm{KL}}(\pi_\theta, a)$, we evaluate the quality and factual consistency of generated summaries using the following metrics:

1. Precision-source (Nan et al., 2021), defined as $[|\mathrm{NER}(x) \cap \mathrm{NER}(x)|]/|\mathrm{NER}(x)|$ is the percentage of named entities in the summary that can be found in the source. Low precision-source indicates severe hallucination,
2. Recall-target (Nan et al., 2021), defined as $[|\mathrm{NER}(x) \cap \mathrm{NER}(x)|]/|\mathrm{NER}(t)|$, is the percentage of named entities in the target summary t that can be found in the generated summary x.
3. Distinct-2 (Li et al., 2016), a measure of text diversity in terms of the frequency of bigram repetitions within a single continuation x,
4. ROUGE-L (Lin, 2004), a measure of summarization quality in terms of unigram overlap between the source document and ground truth summary.

See Appendix A.4 for on how scorers b and metrics are computed for summarization experiments.

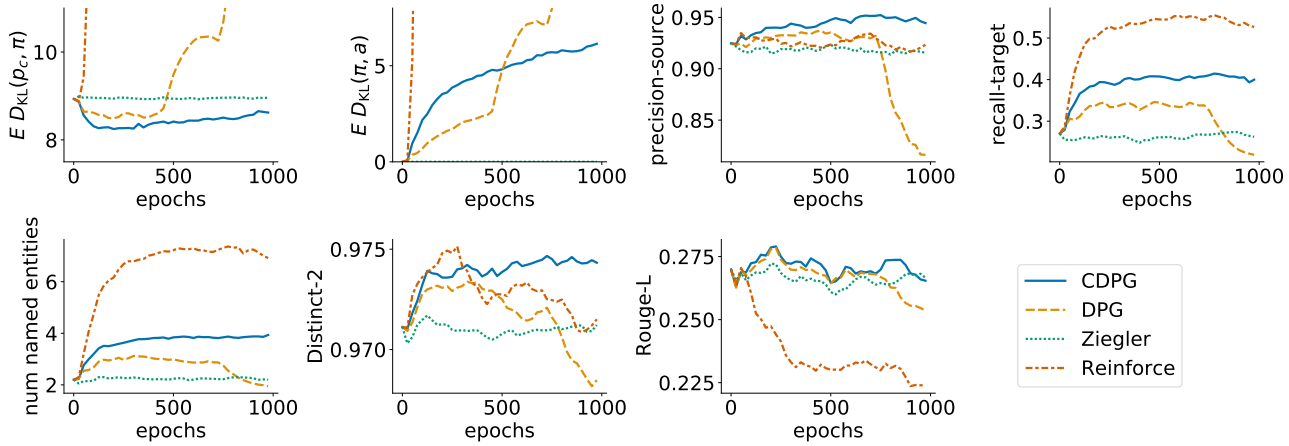


Figure 3: Summarization with factual consistency constraint. Evaluation metrics: expected $D_{\mathrm{KL}}(p_c, \pi_\theta)$ (↓ better) and $D_{\mathrm{KL}}(\pi_\theta, a)$ (↓ better), precision-source (↑ better), recall-target (↑ better), number of named entities (↑ better), Distinct-2 (↑ better), ROUGE-L (↑ better) for models obtained from fine-tuning with conditional DPG, DPG, Ziegler and Reinforce.

Results We present the evolution of our 7 metrics through time on Figure 3. CDPG is the only method stably decreasing expected $D_{\mathrm{KL}}(p_c, \pi_\theta)$ and thus approaching (as opposed to drifting away from) optimal distributions p_c. This is reflected in moderate divergence from a and translates into downstream metrics. Summaries generated by the fine-tuned model contain, on average, more named entities. Moreover, name entities in summaries are both more factually consistent with source (an increase in precision-source) and more relevant (an increase in recall-target). The tendency towards mentioning more factually consistent named entities increases the bigram diversity within summaries (Distinct-2) and the overall quality of generated summaries compared to ground truth (ROUGE-L). This last results might seem surprising given that CDPG did *not* have access to ground truth summaries. A plausible explanation is that the original pretrained model was biased towards mentioning too few factually correct entities, at least compared to ground truth summaries. Satisfying the factual consistency constraint reduced this bias.

Baseline approaches fall short of achieving similar results. The DPG-like ablation, the closest contender, still leaves a significant gap in terms of *all* metrics and is far less stable than CDPG (e.g. its $D_{\mathrm{KL}}(p_c, \pi_\theta)$ starts to diverge again after around 500 epochs). Ziegler stays extremely close to the original model a, failing to improve its shortcomings. In contrast, Reinforce heavily departs from a pushing it to mention a large number of named entities. This results in artificially inflated recall-target but no increase in precision-source and a *decrease* in ROUGE-L. The additional named entities frequently are frequently irrelevant (i.e. not mentioned in ground truth summaries) or simply hallucinated. See Tables 8-12 in the Appendix for randomly chosen summaries of documents in the test set.

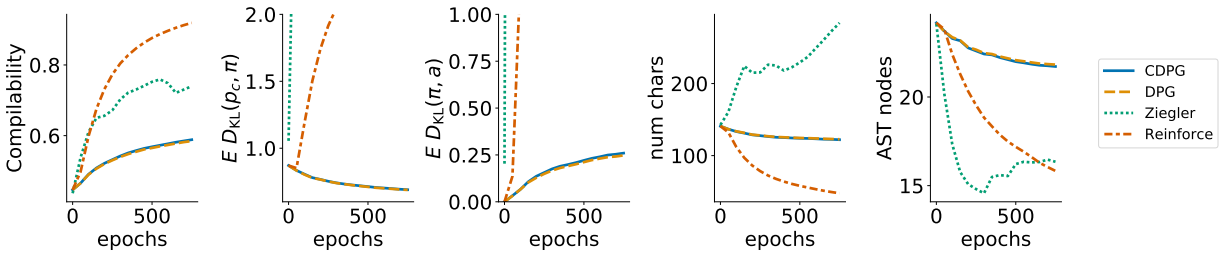
3.4. Code generation

Dataset For code generation experiments, we condition a language model on Python function signatures (both of methods and standalone functions) extracted from the Python150 dataset which consists of Python source code obtained from GitHub (Raychev et al., 2016). We use the code provided by (Roziere et al., 2020) for function extraction and randomly choose 5k functions for C_{train} and 5k for C_{test}. $\tau(c)$ is a uniform distribution over these signatures. Note that neither in fine-tuning nor in evaluation do we use ground truth function bodies.

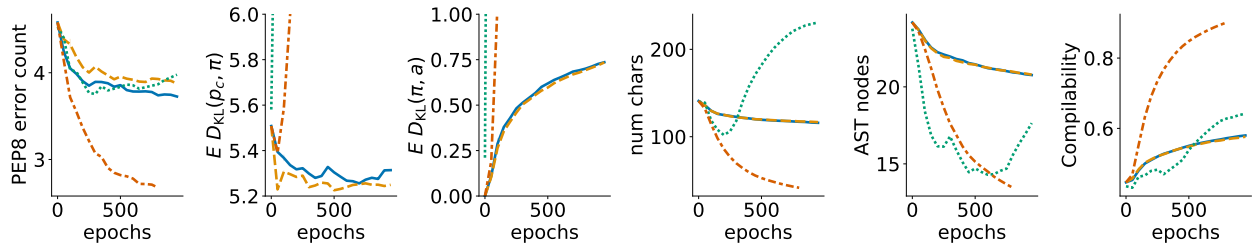
Model We conduct experiments using GPT-Neo (Black et al., 2021): an off-the-shelf, freely available autoregressive language model mirroring the GPT-3 architecture (Brown et al., 2020). GPT-Neo's training set included 85 GiB of source code from GitHub which endowed it with some code completion abilities (Gao et al., 2020). We use the `gpt-neo-125` variant available on Huggingface Transformers (Wolf et al., 2019). During both fine-tuning and evaluation we generate function bodies by conditioning on signatures using pure ancestral sampling.

Constraints For experiments with compilability control condition, we check compilability of a Python function declaration obtained by concatenating $[c, x]$ and trying to execute it. $b(x, c) = 1$ if the Python interpreter raises an exception and 0 otherwise. See Appendix A.4 for more details.

For experiments with PEP8-compliance control condition, we check whether a function declaration given by $[c, x]$ violates PEP8 (van Rossum et al., 2001), the style guide for Python, by running `pycodestyle`, an off-the-shelf linter



(a) Compilability constraint



(b) PEP8 constraint

Figure 4: Code generation with compilability (a) and PEP8 (b) constraint. Evaluation metrics: compilability (↑ better), number of PEP8 errors (↓ better), expected $D_{\mathrm{KL}}(p_c, \pi_\theta)$ (↓ better) and $D_{\mathrm{KL}}(\pi_\theta, a)$ (↓ better), number of characters, AST node count (↑ better) for models obtained from fine-tuning with CDPG, DPG, Ziegler and Reinforce.

(static code analysis tool).[4] $b(x, c) = 1$ if the number of PEP8 violations found by `pycodestyle` is 0, otherwise $b(x, c) = 0$.

Metrics We evaluate the quality of generated Python functions using the following metrics:

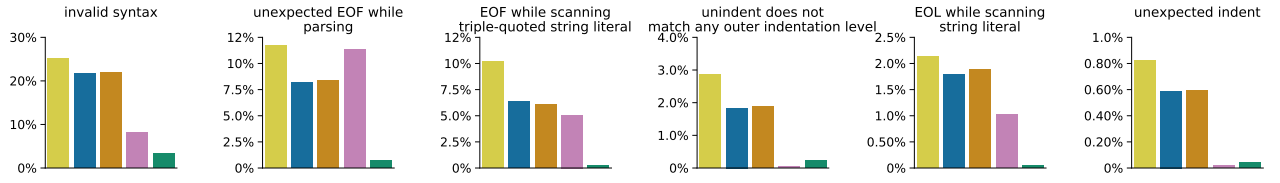
1. PEP8 error count, the average number of violations of PEP8,
2. Compilability, the fraction of samples $[c, x]$ that compile,
3. The average number of characters in $[c, x]$ (after detokenization),
4. The average number of nodes in an abstract syntax tree (AST) of sequences that compile. Intuitively, this metric indicates the logical (as opposed to surface) complexity of generated programs.

For more details on how scorers b and metrics are implemented, see Appendix A.4.

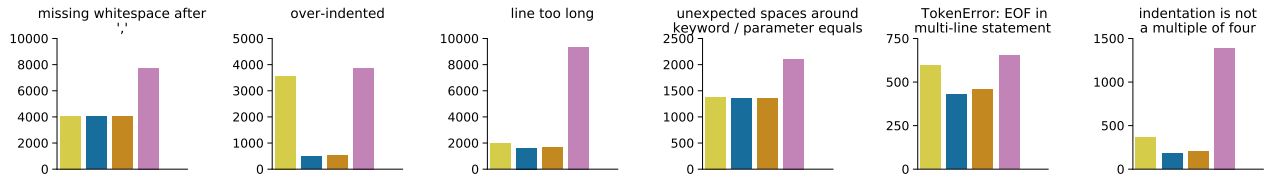
Results We present the evolution of metrics through time on Figure 4. CDPG was able to increase the fraction of compilable functions from around 40% to around 65% and decrease the average number of PEP8 violations. Incidentally, the PEP8 control objective also leads to an increase in compilability because many PEP8 violations are also compilation errors. Here we note similarly to the results of previous experiments that, CDPG and its DPG-like ablation are the only methods actually approaching optimal distributions p_c and diverging moderately from a. This allows them to maintain the original statistics of a: length and the number of nodes in AST trees of generated functions. In contrast, Reinforce learns to generate shorter functions (having less opportunity for mistakes) and Ziegler produces heavily degenerated samples (Holtzman et al., 2020): syntactically simple functions with severe repetitions. This is reflected in an increase in length and a decrease in AST nodes count. See Tables 13-15 and Tables 16-18 for randomly chosen samples from the compilability and PEP8 experiments, respectively.

Note that the performance gap between CDPG and its DPG-like ablation is much closer for code generation (especially with compilability control objective) than for summarization. This can be accounted for by the normalized standard deviation of partition functions Z_c for EBMs P_c in the range of conditional EBMs $\mathcal{P}$ for each control objective. For code generation, this standard deviation is lower meaning that Z_c in (9) is better approximated by a constant which can be absorbed into the learning rate $\alpha^{(\theta)}$. For summarization, this variance is higher, therefore ignoring the Z_c term incurs higher bias which translates into worse performance. See Appendix A.5 for a comparison.

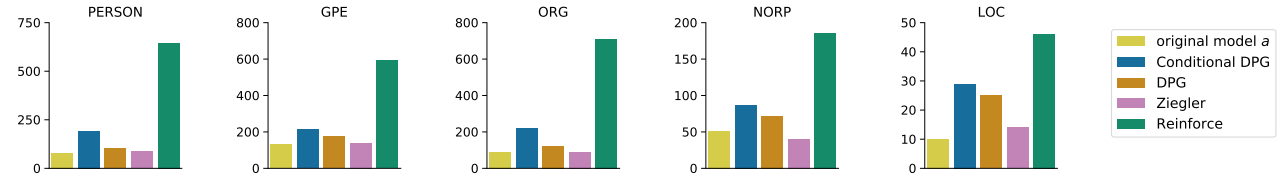
[4] https://github.com/PyCQA/pycodestyle



(a) Relative frequencies of compilation errors



(b) Absolute frequencies of PEP8 violations



(c) Absolute frequencies of named entities

Figure 5: Relative frequencies of most common compilation errors (a), absolute frequencies of most common PEP8 violations (b) and absolute frequencies of named entities (c) in a batch of 10280 samples from the original model a as well as models obtained from fine-tuning with CDPG, DPG, Ziegler and Reinforce.

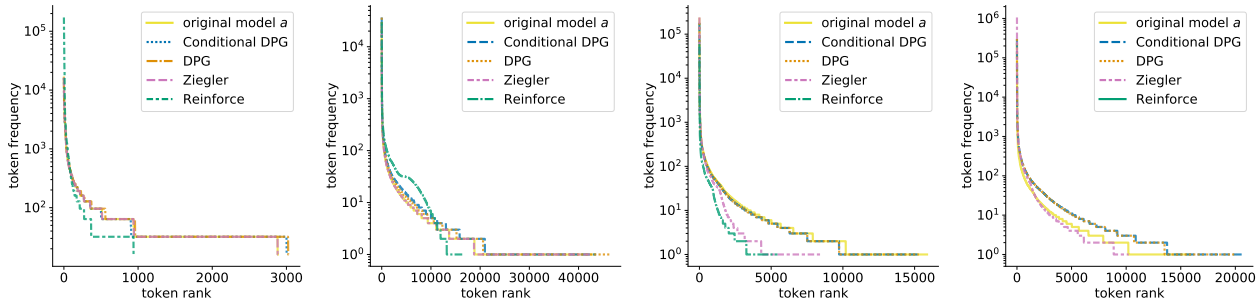
3.5. Qualitative analysis

In the previous sections, we showed how CDPG is able to fine-tune a pretrained model a to satisfy certain constraints without destroying a's capabilities. Here we attempt to gain a better understanding of how different fine-tuning approaches affect the distributions of final models. On Figure 5 we present frequencies of errors (for the code generation task) and named entities (for the summarisation task) obtained from fine-tuned models. While errors and named entities differ significantly in their frequency, CDPG consistently decreases frequencies of these errors and consistently increases the frequencies of all kinds of named entities, including the long tail of rare ones.

To compare lexical diversity of samples obtained from fine-tuned models (for all four tasks), we plot the frequency of each token (the number of times it occurs) and its rank (its index in a sorted list of tokens) in Figure 6. CDPG and its DPG-like ablation are able to closely match original token frequencies while Ziegler and Reinforce tend to have shorter tails of rare tokens.

4. Related work

Injecting prior information in machine translation The "Statistical Machine Translation" paradigm (Koehn, 2010), which was dominant before the deep learning revolution in NLP, heavily exploited log-linear models over predefined features of translation pairs, but without the ability to learn representations typical of neural approaches. Building over such prior work, Zhang et al. (2017) propose to inject a regularization term over a neural translation model which asks for the posterior distribution to be close to a log-linear model using predefined features. This approach is similar to ours in that it allows to incorporate arbitrary features into the posterior distribution. In contrast to our work, the conditional model must be trained jointly with the log-linear one, instead of allowing to control an existing pre-trained model towards directly satisfying constraints. The task here explored is in the spirit of machine translation with terminological constraints (Chatterjee et al., 2017; Hasler et al., 2018; Dinu et al., 2019; ibn Alam et al., 2021). Some approaches to tackle it include constrained decoding (Chatterjee et al., 2017; Hasler et al., 2018), adding the desired terminology as part of the source context (Dinu et al., 2019), among others. Unlike the here-presented approach, these approaches are specific to this



(a) Translation with terminology constraint **(b)** Summarization with factual consistency constraint **(c)** Code generation with compilability constraint **(d)** Code generation with PEP8 constraint

Figure 6: Token frequency against token rank computed for tokens in 10280 samples from a, Conditional DPG and baselines. Longer tails imply more diverse samples.

task only and will not generalize to arbitrary constraints.

Reducing hallucinations in summarization Neural abstractive summarization is highly prone to hallucinate content in the summary that is unfaithful to the source document. Maynez et al. (2020) found that hallucinations occur in more than 70% of single-sentence summaries and most of these are *extrinsic hallucinations*: adding information not directly inferable from the input document. Therefore, a substantial effort was devoted to to improving factual consistency of abstractive summarization. Some notable attempts include reranking summaries based on their correctness predicted by entailment classifiers (Falke et al., 2019) or fine-tuning using RL with a reward derived from an entailment classifier (Pasunuru & Bansal, 2018). The notion of *entity-level* factual consistency – a property such that all named entities in the summary are actually mentioned in the source document – was introduced by Nan et al. (2021) as one way of operationalizing the notion of extrinsic hallucinations.

Controllable code generation Generating source code is an established application of language models (Nguyen et al., 2013; Raychev et al., 2014; Karpathy et al., 2015; Bielik et al., 2016) that since recently has enjoyed renewed interest (Lu et al., 2021; Chen et al., 2021; Austin et al., 2021). The task is formulated both as unconditional generation (with applications in code completion, e.g. Codex (Lu et al., 2021) or GitHub Copilot[5]) and as conditional generation (e.g. program synthesis or generating a program satisfying a given input-output specification, e.g. (Austin et al., 2021)). Our task of function generation can be seen as a simplified program synthesis with the specification given by function signature (a name of a function and a list of arguments). Previous work found compilability errors to be a signification failure mode of neural code generation (Roziere et al., 2020). Previous attempts at improving compilability of generated code include (Maddison & Tarlow, 2014), who augment neural probabilistic context free grammars with semantic constraints and use them for unconditional generation or (Zhong et al., 2017), who used policy gradients to train a model translating natural language questions to corresponding SQL queries and – in addition for rewarding for query execution results – added a penalty for syntactically invalid queries. Most in line with our work, Korbak et al. (2021) used DPG to improve compilability of unconditional language models for code.

[5] https://copilot.github.com

5. Conclusion

We presented CDPG, a principled approach to fine-tuning conditional language models to satisfy arbitrary constraints. In contrast with other methods, CDPG does not require ground truth training data and is able to shift model distribution in a minimally invasive way. In consequence, models fine-tuned with CDPG share desired characteristics, such as improved factual consistency or compilability, with the fluency and diversity of the original model.

Future work could evaluate CDPG on other tasks — such as dialogue — as well as explore other control objectives such as constraining the semantics (as opposed to syntax) of generated Python functions. Another future direction consists in extending CDPG to approximate conditional analogues of the more general, *exponential-form* (Khalifa et al., 2021) EBMs which can represent *distributional* constraints, namely, desired expected values for certain features of generated samples.

References

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language mod-

els, 2021.

Bielik, P., Raychev, V., and Vechev, M. Phog: Probabilistic model for code. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2933–2942. JMLR.org, 2016.

Black, S., Leo, G., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL https://doi.org/10.5281/zenodo.5297715.

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., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Chatterjee, R., Negri, M., Turchi, M., Federico, M., Specia, L., and Blain, F. Guiding neural machine translation decoding with external knowledge. In *WMT 2017*, pp. 157–168, 2017.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

Csiszár, I. and Shields, P. C. Information theory and statistics: A tutorial. *Commun. Inf. Theory*, 1(4):417–528, December 2004. doi: 10.1561/0100000004. URL https://www.stat.berkeley.edu/~binyu/212A/papers/cs.pdf.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

Dinu, G., Mathur, P., Federico, M., and Al-Onaizan, Y. Training neural machine translation to apply terminology constraints. In *ACL*, pp. 3063–3068, 2019.

Falke, T., Ribeiro, L. F. R., Utama, P. A., Dagan, I., and Gurevych, I. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2214–2220, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1213. URL https://aclanthology.org/P19-1213.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Ghazvininejad, M., Shi, X., Priyadarshi, J., and Knight, K. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pp. 43–48, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://aclanthology.org/P17-4008.

Hasler, E., de Gispert, A., Iglesias, G., and Byrne, B. Neural machine translation decoding with terminology constraints. In *NAACL-HLT 2018*, pp. 506–512, 2018.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018. URL https://doi.org/10.1162/089976602760128018.

Holtzman, A., Buys, J., Forbes, M., Bosselut, A., Golub, D., and Choi, Y. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1638–1649, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1152. URL https://www.aclweb.org/anthology/P18-1152.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

ibn Alam, M. M., Kvapilíková, I., Anastasopoulos, A., Besacier, L., Dinu, G., Federico, M., Gallé, M., Koehn, P., Nikoulina, V., and Jung, K. W. Findings of the wmt shared task on machine translation using terminologies. *WMT 2021*, pp. 652, 2021.

Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. *ArXiv*, abs/1506.02078, 2015.

Khalifa, M., Elsahar, H., and Dymetman, M. A distributional approach to controlled text generation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jWkw45-9AbL.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koehn, P. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pp. 79–86, Phuket, Thailand, September 13-15 2005. URL https://aclanthology.org/2005.mtsummit-papers.11.

Koehn, P. *Statistical Machine Translation*. Cambridge University Press, 2010.

Korbak, T., Elsahar, H., Dymetman, M., and Kruszewski, G. Energy-based models for code generation under compilability constraints, 2021.

Korbak, T., Elsahar, H., Kruszewski, G., and Dymetman, M. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting, 2022. URL https://arxiv.org/abs/2206.00761.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL https://www.aclweb.org/anthology/N16-1014.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C. B., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., Sundaresan, N., Deng, S. K., Fu, S., and Liu, S. Codexglue: A machine learning benchmark dataset for code understanding and generation. *CoRR*, abs/2102.04664, 2021.

Maddison, C. J. and Tarlow, D. Structured generative models of natural source code. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–649–II–657. JMLR.org, 2014.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.173. URL https://aclanthology.org/2020.acl-main.173.

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., and Xiang, B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL https://aclanthology.org/K16-1028.

Nan, F., Nallapati, R., Wang, Z., Nogueira dos Santos, C., Zhu, H., Zhang, D., McKeown, K., and Xiang, B. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 2727–2733, Online, April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.eacl-main.235.

Nguyen, T. T., Nguyen, A. T., Nguyen, H. A., and Nguyen, T. N. A statistical semantic language model for source code. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pp. 532–542, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450322379. doi: 10.1145/2491411.2491458. URL https://doi.org/10.1145/2491411.2491458.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pp. 311–318, USA,

2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Parshakova, T., Andreoli, J.-M., and Dymetman, M. Global Autoregressive Models for Data-Efficient Sequence Learning. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pp. 900–909, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/K19-1084. URL https://www.aclweb.org/anthology/K19-1084.

Parshakova, T., Andreoli, J.-M., and Dymetman, M. Distributional Reinforcement Learning For Energy-Based Sequential Models. *CoRR*, 2019b. URL https://arxiv.org/abs/1912.08517.

Pasunuru, R. and Bansal, M. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 646–653, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2102. URL https://aclanthology.org/N18-2102.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Post, M. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Raychev, V., Vechev, M., and Yahav, E. Code completion with statistical language models. *SIGPLAN Not.*, 49(6):419–428, June 2014. ISSN 0362-1340. doi: 10.1145/2666356.2594321. URL https://doi.org/10.1145/2666356.2594321.

Raychev, V., Bielik, P., and Vechev, M. Probabilistic model for code with decision trees. *SIGPLAN Not.*, 51(10):731–747, 2016. ISSN 0362-1340. doi: 10.1145/3022671.2984041. URL https://doi.org/10.1145/3022671.2984041.

Roziere, B., Lachaux, M.-A., Chanussot, L., and Lample, G. Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, 33, 2020.

See, A., Roller, S., Kiela, D., and Weston, J. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1702–1723, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1170. URL https://aclanthology.org/N19-1170.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

van Rossum, G., Warsaw, B., and Coghlan, N. Style guide for Python code. PEP 8, 2001. URL https://www.python.org/dev/peps/pep-0008/.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pp. 229–256, 1992.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL http://arxiv.org/abs/1910.03771.

Zhang, J., Liu, Y., Luan, H., Xu, J., and Sun, M. Prior knowledge integration for neural machine translation using posterior regularization. In *ACL*, 2017.

Zhong, V., Xiong, C., and Socher, R. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL http://arxiv.org/abs/1909.08593.

A. Details of metric and score calculation

A.1. KL divergences

Calculation of metrics relative to p_c's, such as $\mathbb{E}_{c\sim\tau(c)} D_{\mathrm{KL}}(p_c, \pi_\theta)$, requires estimating Z_c's. This is done using importance sampling from π_θ in a manner analogous to the training loop (Algorithm 1). Then, expected KL can be simplified to the following form:

$$\mathbb{E}_{c\sim\pi(c)} D_{\mathrm{KL}}\Big[p_c(x) \mid\mid \pi_\theta(x|c)\Big] = \mathbb{E}_{c\sim\pi(c)} \sum_x p_c(x) \log \frac{p_c(x)}{\pi_\theta(x|c)} \tag{10}$$

$$= \mathbb{E}_{c\sim\pi(c)} \sum_x p(x|c) \log \frac{P_c(x)}{Z_c \pi_\theta(x|c)} \tag{11}$$

$$= \mathbb{E}_{c\sim\pi(c)} \Big[-\log Z_c + \sum_x p(x|c) \log \frac{P_c(x)}{\pi_\theta(x|c)} \Big] \tag{12}$$

$$= \mathbb{E}_{c\sim\pi(c)} \Big[-\log Z_c + \sum_x \pi_\theta(x|c) \frac{P_c(x)}{\pi_\theta(x|c)} \log \frac{P_c(x)}{\pi_\theta(x|c)} \Big] \tag{13}$$

$$= \mathbb{E}_{c\sim\pi(c)} \Big[-\log Z_c + \frac{1}{Z_c} \mathbb{E}_{x\sim\pi_\theta(x|c)} \frac{P_c(x)}{\pi_\theta(x|c)} \log \frac{P_c(x)}{\pi_\theta(x|c)} \Big]. \tag{14}$$

A small ϵ is added to Z_c for stability. We approximate both expectations (over τ and π_θ) using importance sampling. For a complete procedure, see Algorithm 2.

$\mathbb{E}_{c\sim\tau(c)} D_{\mathrm{KL}}(\pi_\theta, a)$ is computed in a simpler manner as it doesn't require estimating Z_c's and we can directly sample from π_θ. It boils down to sampling a batch of N contexts c_i, a batch of M samples x_j from $\pi_\theta(x|c_i)$ for each c_i and evaluating:

$$\mathbb{E}_{c\sim\tau(c)} D_{\mathrm{KL}}(\pi_\theta, a) \approx \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{\pi_\theta(x_j|c_i)}{a(x_j|c_i)}. \tag{15}$$

To avoid bias, when computing KL divergences we always sample from π_θ using pure ancestral sampling (as opposed to top p sampling or beam search decoding).

Algorithm 2 Estimating $\mathbb{E}_{c\sim\tau(c)} D_{\mathrm{KL}}(p_c, \pi_\theta)$

Input: a distribution over queries $\tau(c)$
Input: conditional model π_θ
Input: N, number of contexts
Input: M, number of samples for each context
1: sample batch $\{c_1, ..., c_i, ..., c_N\}$ from $\tau(c)$
2: **for** $i \in \{1, \ldots, N\}$ **do**
3: sample batch $\{x_1, ..., x_j, ..., x_M\}$ from $\pi_\theta(x|c_i)$
4: $\hat{Z}_{c_i} = \frac{1}{M} \sum_{j=1}^{M} \frac{P_{c_i}(x_j)}{\pi_\theta(x_j|c_i)}$
5: $D_{\mathrm{KL}}(p, \pi_\theta) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left[\frac{1}{\hat{Z}_{c_i} + \epsilon} \frac{P_{c_i}(x_j)}{\pi_\theta(x_j|c_i)} \Big[-\log \hat{Z}_{c_i} + \log \frac{P_{c_i}(x_j)}{\pi_\theta(x_j|c_i)} \Big] \right]$
Output: An estimate of $\mathbb{E}_{c\sim\tau(c)} D_{\mathrm{KL}}(p_c, \pi_\theta)$

A.2. Translation

We implement the scorer for number normalization as a lookup table mapping a numeral noun (e.g. "one") to a digit ("1"). Digits range from 1 to 9. A constraint is satisfied if for every occurrence of a given numeral noun in source sentence x, a corresponding digit occurs in its translation x.

To compute BLEU-4 score, we use the SacreBLEU implementation (Post, 2018).

A.3. Summarization

Following (Nan et al., 2021), we implement NER$(\cdot)$ as using a pretrained SpaCy (Honnibal et al., 2020) named entity recognizer. We use the `en_core_web_sm` model and restrict the named entities we extract to the following categories: `PERSON`, `FAC` (buildings, airports, highways, bridges, etc.), `GPE` (geopolitical entities: countries, cities, etc.), `ORG` (companies, agencies, institutions, etc.), `NORP` (nationalities or religious or political groups), `LOC` (Non-GPE locations: mountain ranges, bodies of water, etc.), `EVENT` (named hurricanes, battles, wars, sports events, etc.). Also following (Nan et al., 2021), we ignore entities such as date, time and numerals due to large variation in their representation in documents.

A.4. Code generation

Compilability To check for compilability, we call the `compile_command` function from the `codeop` module of Python Standard Library[6] with a sequence obtained by string concatenation $[c, x]$ as argument. We then check if `compile_command` returns a `code` object. The only postprocessing we apply is removing any characters from x after the end of function declaration (with function end defined in terms of indentation) as we are concerned specifically with function generation. `codeop.compile_command` is the implementation that Python interactive interpreters use in read-eval-print loop (REPL) to determine whether a string is a valid Python code. The method tries to compile a string of Python code and raise and exception if compilation fails, for instance a `SyntaxError` for invalid Python syntax and `ValueError` or `OverflowError` if there is an invalid literal. Note that our notion of compilability is concerned only with syntactic correctness as Python interpreter does not execute the body of a function at function declaration time.

PEP8 To compute the number of PEP8 violations triggered by a sequence $[c, x]$, we run pycodestyle,[7] a Python linter (static code analysis tool) and report the number of violations it reports.

AST node count Finally, to compute AST node count, the average number of nodes in an abstract syntax trees (ASTs) of generated functions, we consider only samples $[c, x]$ that compile. They are parsed to their corresponding ASTs using the `ast` module from Python Standard Library.[8]

A.5. Normalized standard deviations for Z_c across tasks

See Figure 7 of normalized standard deviations of Z_c across tasks. Here normalized standard deviations is defined as $\text{std}(Z_c)/\text{avg}(Z_c)$, where

$$\text{avg}(Z_c) = \frac{1}{N}\sum_{i=1}^{N} Z_{c_i}, \tag{16}$$

$$\text{std}(Z_c) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\Big(Z_{c_i} - \text{avg}(Z_c)\Big)^2}. \tag{17}$$

Lower normalized standard deviation for a task explains closer performance gap between CDPG and DPG for that task on Figures 3-4.

B. Hyperparmeters and implementation details

We implemented all models using PyTorch (Paszke et al., 2019) and HuggingFace Transformers (Wolf et al., 2019). Each training run took approximately 5 days on 2 Nvidia V100 GPUs. For a detailed list of hyperparameter values, see Table 1 and 2. For a description of hyperparameters specific to Ziegle see (Ziegler et al., 2019).

[6] https://docs.python.org/3/library/codeop.html
[7] https://github.com/PyCQA/pycodestyle
[8] https://docs.python.org/3/library/ast.html

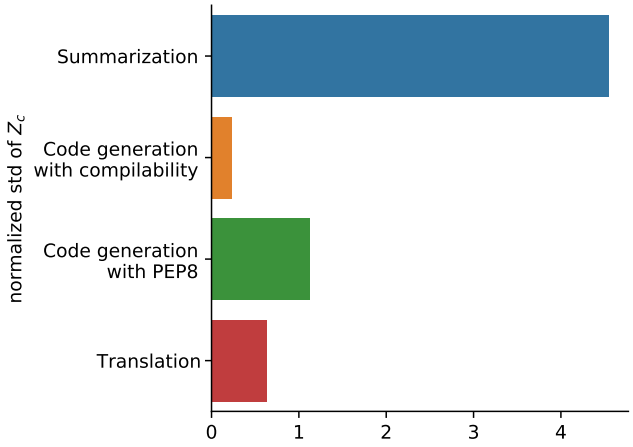


Figure 7: Normalized standard deviations for Z_c's associated with unconditional EBMs P_c in the codomain of conditional EBMs $\mathcal{P}$ defined for three control objectives: summarization with factual consistency constraint, code generation with compilability constraint and code generation with PEP8 constraint.

Hyperparameter	Value	Symbol
	Common	
original model	`EleutherAI/gpt-neo-125M`	a
batch size	2048	
maximum sequence length	128 tokens	
learning rate for π_θ	1.41×10^{-6}	$\alpha^{(\theta)}$
optimizer	Adam (Kingma & Ba, 2014)	
learning rate schedule	constant with warmup (100 epochs)	
total epochs	1000	
number of c's for training	5000	$\lvert C_{\text{train}}\rvert$
number of c's per batch	32	N
number of x's per c	64	M
	Ziegler	
policy gradients clip range	0.2	
target KL value for adaptive schedule	6.0	
initial coefficient of KL penalty	0.2	β

Table 1: Hyperparameters used for code generation experiments

Hyperparameter	Value	Symbol
	Common	
original model	`t5-small`	a
batch size	1024	
maximum sequence length	200 tokens	
learning rate for π_θ	1×10^{-4}	$\alpha^{(\theta)}$
optimizer	Adam (Kingma & Ba, 2014)	
learning rate schedule	constant with warmup (100 epochs)	
total epochs	1000	
number of c's for training	5000	$\lvert C_{\text{train}}\rvert$
number of c's per batch	32	N
number of x's per c	32	M
	Ziegler	
policy gradients clip range	0.2	
target KL value for adaptive schedule	6.0	
initial coefficient of KL penalty	0.2	β

Table 2: Hyperparameters used for translation and summarization experiments

C. Samples

C.1. Translation with terminology constraint

See Tables 3-7 for translations of 5 randomly chosen source sentences from our evaluation set C_{test} generated by models fine-tuned using Conditional DPG, 3 baselines and a. Translations were generated with beam search.

C.2. Summarization with entity-level factual consistency constraint

See Tables 8-12 for summaries of 5 randomly chosen documents from our evaluation set C_{test} generated by models fine-tuned using Conditional DPG, 3 baselines and a. Summaries were generated with beam search.

C.3. Code generation with compilability constraint

See Tables 13-15 for functions obtained by sampling conditioned on 3 randomly chosen signatures from our evaluation set C_{test} generated by models fine-tuned using Conditional DPG, 3 baselines and a. We used pure ancestral sampling.

C.4. Code generation with PEP8 constraint

See Tables 16-18 for functions obtained by sampling conditioned on 3 randomly chosen signatures from our evaluation set C_{test} generated by models fine-tuned using Conditional DPG, 3 baselines and a. We used pure ancestral sampling.

Source sentence c

) Mr President, I would like to answer Manuel Medina Ortega' s and Efstratios Korakas' questions simultaneously, as both questions concern the risks connected with the construction of nuclear power plants, in this case, one planned in Morocco and one in Turkey.

$b(x, c)$	x
	Translation generated by the original model a
0	) Monsieur le Président, je voudrais répondre simultanément aux questions de Manuel Medina Ortega et d' Efstratios Korakas, étant donné que les deux questions concernent les risques liés à la construction de centrales nucléaires, en l'espèce, d'une centrale prévue au Maroc en Turquie.
	Translation generated by a model fine-tuned using Conditional DPG
1	) Monsieur le Président, je voudrais répondre simultanément aux questions de Manuel Medina Ortega et d' Efstratios Korakas, étant donné que les deux questions concernent les risques liés à la construction de centrales nucléaires, en l'espèce, 1 planifiée au Maroc, 1 en Turquie.
	Translation generated by a model fine-tuned using DPG
0	) Monsieur le Président, je voudrais répondre simultanément aux questions de Manuel Medina Ortega et d' Efstratios Korakas, car les deux questions concernent les risques liés à la construction de centrales nucléaires, dans ce cas-ci, l'une planifiée au Maroc, la seconde en Turquie.
	Translation generated by a model fine-tuned using Ziegler
0	) Monsieur le Président, je voudrais répondre simultanément aux questions de Manuel Medina Ortega et d' Efstratios Korakas, étant donné que les deux questions concernent les risques liés à la construction de centrales nucléaires, en l'espèce, d'une centrale prévue au Maroc à l' heure actuelle, tandis qu' une centrale en Turquie.
	Translation generated by a model fine-tuned using Reinforce
1	) Monsieur Président Président, Monsieur le Président, d?,, Monsieur Président, Herr Präsident, ich m 2 2, 1, 3 3, 2 2 1 1 1 2 1 2 2 2 3 1 1 3 2 2 4 1 1 5 1 1 7 1 1 4 2 2 7 7 7 1 2 3 3 2 1 3 1 2 4 2 1 7 7 8 8 1 1 8 8 8 7 7 2 2 5 2 2 8 8 2 1 4 1 2 7, 4 4 4 2

Table 3: Translations generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy a terminology consistency constraints (translating numeral nouns as digits).

Source sentence c

- (DE) The position of the non-attached Members on the content of the first two decisions of Parliament regarding the Statute for Members has not changed.

$b(x, c)$	x
	Translation generated by the original model a
0	- (DE) La position des députés non affiliés sur le contenu des deux premières décisions du Parlement concernant le statut de députés n'a pas changé.
	Translation generated by a model fine-tuned using Conditional DPG
1	- (DE) La position des députés non inscrits sur le contenu des 2 premières décisions du Parlement concernant le statut de députés n'a pas changé.
	Translation generated by a model fine-tuned using DPG
0	- (DE) La position des députés non inscrits sur le contenu des deux premières décisions du Parlement concernant le statut de députés n'a pas changé.
	Translation generated by a model fine-tuned using Ziegler
0	- (DE) La position des députés non affiliés sur le contenu des deux premières décisions du Parlement concernant le statut de députés n'a pas changé.
	Translation generated by a model fine-tuned using Reinforce
1	- (DE) La position des 2 2 2 - 2, (DE),, 2 2 3 2 2 1 2 2 5 5 5 2 :, 1 1 1 2., 5 5 1 1, 6 2 2 4 2 2 6 5 5, 3 2 5 2 2 7 5 5 3 2 1 : 1 2 3 3 3 2 3, 4 5 5 6 6 6 5 2 5 : 2 5 1 2 1 1 5 2 3 5 5 4 2 5 6

Table 4: Translations generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy a terminology consistency constraints (translating numeral nouns as digits).

Source sentence c

- (FR) I have personally listened to all the speeches by the President, Romano Prodi, and I have participated in all six hearings of the candidate Commissioners.

$b(x, c)$	x
	Translation generated by the original model a
0	J'ai personnellement écouté toutes les interventions du président, Romano Prodi, et j'y ai participé aux six auditions des candidats aux commissaires.
	Translation generated by a model fine-tuned using Conditional DPG
1	J'ai personnellement écouté toutes les interventions du président, Romano Prodi, et j'y ai participé aux 6 auditions des candidats aux commissaires.
	Translation generated by a model fine-tuned using DPG
0	J'ai personnellement écouté toutes les interventions du président, Romano Prodi, et j'y ai participé aux six auditions des commissaires candidats.
	Translation generated by a model fine-tuned using Ziegler
0	J'ai personnellement écouté toutes les interventions du président, Romano Prodi, et j'y ai participé aux six auditions des candidats aux commissaires.
	Translation generated by a model fine-tuned using Reinforce
1	- FR) J' m 2 2 2, 2 2 3 3 3, 1,, 5 5 5 :, 3 3 5 5 1 1 1 2 2 1 2 1 1 3 3 2 2 5 5 3 3 1 1 5 5 2 1 3 2 3 1 2 3 2 1 5 2 2 4 3 3 4 5 5 7 7 7 5 5 6 6 6 5 5 4 2 2 7 7 6 6 7 7, 6 6 3 3 7 7 3 5 3 5 7 5 7 6 5

Table 5: Translations generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy a terminology consistency constraints (translating numeral nouns as digits).

Source sentence c

A decision will probably be taken to extend the existing measures against Burma, including visa restrictions, for a further six months.

$b(x, c)$	x
	Translation generated by the original model a
0	Une décision sera probablement prise pour prolonger de six mois les mesures en vigueur contre la Birmanie, y compris les restrictions à la délivrance de visas.
	Translation generated by a model fine-tuned using Conditional DPG
1	Une décision sera probablement prise pour prolonger de 6 mois les mesures en vigueur contre la Birmanie, y compris les restrictions à la délivrance de visas.
	Translation generated by a model fine-tuned using DPG
1	On décidera probablement de proroger de 6 mois les mesures en vigueur contre la Birmanie, y compris les restrictions à la délivrance de visas.
	Translation generated by a model fine-tuned using Ziegler
0	On décidera probablement de proroger de six mois les mesures en vigueur contre la Birmanie, y compris les restrictions à la délivrance de visas.
	Translation generated by a model fine-tuned using Reinforce
1	Il y a 6 6 6 7 7 7, 6 6 2 6 6, 2 2 2 1 1 1 2 2, y 2 1 2 1, 1 1 - 1 2 -, 4 2 2 3, 5 2 2 4, 7 7 2 2 5, 3 2 2 7 - 2 2 8, 8 - 5 - 7" 2'1 1 3 - 4 - 3'5'7 7 8 7 7 5

Table 6: Translations generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy a terminology consistency constraints (translating numeral nouns as digits).

Source sentence c

A particularly thorny question, and one that has been cause for most debate beforehand, is that of Turkey.

$b(x, c)$	x
	Translation generated by the original model a
0	Une question particulièrement épineuse, qui a suscité la plupart des débats préalables, est celle de la Turquie.
	Translation generated by a model fine-tuned using Conditional DPG
0	Une question particulièrement épineuse, qui a suscité la plupart des débats antérieurs, est celle de la Turquie.
	Translation generated by a model fine-tuned using DPG
0	Une question particulièrement épineuse, qui a suscité la plupart des débats préalables, est celle de la Turquie.
	Translation generated by a model fine-tuned using Ziegler
0	Une question particulièrement épineuse, qui a suscité la plupart des débats préalables, est celle de la Turquie.
	Translation generated by a model fine-tuned using Reinforce
1	Une question particulièrement épineuse, et 1, 1 1 1 2 2 2 1 2 1 1 3 2 2 3 2 1 3 1 1 5 1 1 4 2 2 5 1 2 3 3 2 3 1 2 5 2 2 4 2 1 5 5 1 3 3 3 1 3 5 5 5 2 1 4 1 2 4 1 1 : 1 1 6 1 2, 2 2 : 2 3 5 1 5 2 5 5 3 2 5 3 3 5 2 3 4 5 5, 5 5 4 2 5 4

Table 7: Translations generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy a terminology consistency constraints (translating numeral nouns as digits).

Source document c

(CNN)"Success Kid" is likely the Internet's most famous baby. You've seen him in dozens of memes, fist clenched in a determined look of persevering despite the odds. Success Kid – now an 8-year-old named Sammy Griner – needs a little bit of that mojo to rub off on his family. His dad, Justin, needs a kidney transplant. About a week ago, Laney Griner, Justin's wife and Sammy's mother, created a GoFundMe campaign with a goal of $75,000 to help cover the medical expenses that go along with a kidney transplant. The campaign is already a success. By Wednesday it had topped its goal. Griner told The Daily Dot that her husband was diagnosed with kidney disease in 2006 and suffered complete kidney failure three years later. "One can only survive with no natural kidney function ... for so long," Laney Griner said. "His energy and mood are affected; he can no longer work, and he spends 12 hours a week in dialysis clinic. "Having been on dialysis for this long greatly increases his risks of developing further complications. The only way to save his life is to get a transplant. There's no other way around that," she said. The family doesn't know when a kidney might become available. Their GoFundMe page has a link for potential donors. Sammy's Internet fame began in 2007 when his mom posted a picture of him on a beach with a fist full of sand and a satisfied look on his face. Myspace picked it up, so did Reddit. The rest is Internet history. Success just seems to run in some families.

$b(x, c)$	x
	Summary generated by the original model a
0	"Success Kid" is the internet's most famous baby. his dad, Justin, needs a kidney transplant. a week ago, his wife and mother created a goFundMe campaign.
	Summary generated by a model fine-tuned using Conditional DPG
0	"Success Kid" is the internet's most famous baby. his dad, Justin, needs a kidney transplant. about a week ago, Laney Griner, his wife and mother created a goFundMe campaign.
	Summary generated by a model fine-tuned using DPG
0	"Success Kid" is the internet's most famous baby. his mom posted a picture of him on a beach with a fist full of sand and a satisfied look on his face. a goFundMe campaign has a goal of $75,000 to help cover medical expenses.
	Summary generated by a model fine-tuned using Ziegler
0	"Success Kid" is the internet's most famous baby. his dad, Justin, needs a kidney transplant. a week ago, his wife and mother created a goFundMe campaign.
	Summary generated by a model fine-tuned using Reinforce
1	Success Kid – now an 8-year-old named Sammy Griner – needs a little bit of that mojo to rub off on his family. Laney Griner, textcolorredJustin's wife and Sammy's mother, created a GoFundMe campaign with a goal of $75,000. Griner told The Daily Dot that her husband was diagnosed with kidney disease in 2006 and suffered complete kidney failure three years later. The family doesn't know when a kidney might become available. Sammy's

Table 8: Summaries generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy entity-level factual consistency constraint. Named entities in summaries are highlighted in red.

Source document c

A picture, believed to be the only image of the Civil War ironclad, the CSS Georgia has been revealed to be a fake created in a teenage hoax using a 2ft model. John Potter, from Savannah, has admitted forging the picture with his brother in the 1980s and placing it in a frame which now holds a picture of his dead dog. He then passed on the image on to the Georgia Historical Society and the photo became an unofficial part of the ship's history even though it was never authenticated. Scroll down for video . John Potter (pictured), from Savannah, has admitted forging the picture with his brother in the 1980s and placing the picture in a frame which now holds a picture of his dead dog . In 1986 he fibbed that he was at a yard sale when he found the photograph in an antique frame.Inscribed on the back of the frame, he claimed, was 'CSS Georgia.' He told historians that he didn't have the $175 the owner wanted . The picture Mr Potter handed over to the society was actually a picture of a picture. In 1986 he fibbed that he was at a yard sale when he found the photograph in an antique frame. Inscribed on the back of the frame, he claimed, was 'CSS Georgia.' He told historians that he didn't have the $175 the owner wanted, so he took a photo of it and then mailed it to historical groups in Savannah. Potter has now admitted the sham and explained how he falsified the image. When he was a teenager in Savannah, Potter, his brother Jeffrey and a friend shot a short 8mm movie about the building — and destruction — of the CSS Georgia in a fictional battle with Union troops. For the movie, they built an 18-foot long boat of plywood and Styrofoam, as well as a smaller 2-foot model. They based the design, in part, on his grandfather's recollections of details passed down by word of mouth through generations of their family. Potter also used an illustration of the ironclad he found on a postcard. To create the fake image Potter's younger brother put on a coat and straw hat went out to a marsh with a cane fishing pole and Potter took a photo. He took another photo of the 2-foot model and cut out the boat's image, glued it onto the photo of his brother, then used dirt and glue to create the illusion of a photo faded by age and stained by water or chemicals. Evidence of the hoax: All elements of the fakery were snapped by Potter at one time. The small model boat here appears alongside the false picture, polaroids of it and a 1984 copy of Mad Magazine . He bought an old picture frame and beat it up even further. He put the photo in it. Then he drove 120 miles to a yard sale in Waycross, Georgia, put the picture down and took a Polaroid of it. He laughs now, when he remembers that it had seemed so important that he actually do this at a yard sale, so at least that part would be true. 'Who knows what goes through the mind of a kid,' he said. The US Army Corps of Engineers appealed for information about the picture as it embarked on a project to raise the ironclad . Potter sent out the photo to historical groups, setting off the sporadic, and fruitless, search for a CSS Georgia photo that he now says never existed. As the Army Corps of Engineers embarked this year on a $14 million project to raise the wreckage of the real CSS Georgia from the river, archaeologists publicised the image online and in news stories hoping to track down the original photo. 'Wanted: A Photo Of This Confederate Battleship,' read the headline above the image on the Army Corps website. 'There are a lot of characteristics in the photograph that lead us to believe it is the CSS Georgia,' Julie Morgan, the Army Corps archaeologist in charge of raising the ironclad's remains, said in a February interview. 'On the other side, there are some skeptics who believe it's a complete fake.' Robert Holcombe, former curator of the National Civil War Naval Museum in Columbus, Georgia, said in February that while the original photograph would be needed to confirm if the image was authentic, he believed it was real. 'Most people seem to think so,' he said. 'Or else it's an awfully good fake'. The peeling gilt frame that once held the disputed photo, is now filled with a portrait of Potter's deceased pug, Puggy Van Dug. He never became a successful special effects artist with the exception of the one faked photo. The peeling gilt frame that once held the disputed photo, is now filled with a portrait of Potter's deceased pug, Puggy Van Dug .

$b(x, c)$	x
	Summary generated by the original model a
0	John Potter, from Savannah, has admitted forging the picture with his brother in the 1980s and placing it in a frame which now holds a picture of his dead dog. he then passed on the image on to the Georgia Historical Society and the photo became an unofficial part of the ship's history even though it was never authenticated. in 1986 he fibbed that he was at a yard sale when he found the photograph in an antique frame.
	Summary generated by a model fine-tuned using Conditional DPG
0	John Potter, from Savannah, has admitted forging the picture with his brother in the 1980s and placing it in a frame which now holds a picture of his dead dog. he passed on the image on to the Georgia Historical Society and the photo became an unofficial part of the ship's history even though it was never authenticated. in 1986 he fibbed that he was at a yard sale when he found the photograph in an antique frame.inscribed on the back of the frame, he claimed,
	Summary generated by a model fine-tuned using DPG
0	John Potter, from Savannah, has admitted forging the picture with his brother in the 1980s and placing it in a frame which now holds a picture of his dead dog. the picture became an unofficial part of the ship's history even though it was never authenticated. in 1986 he fibbed that he was at a yard sale when he found the photograph in an antique frame.inscribed on the back of the frame, he claimed, was 'CSS Georgia'
	Summary generated by a model fine-tuned using Ziegler
0	John Potter, from Savannah, has admitted forging the picture with his brother in the 1980s and placing it in a frame which now holds a picture of his dead dog. he then passed on the image on to the Georgia Historical Society and the photo became an unofficial part of the ship's history even though it was never authenticated. in 1986 he fibbed that he was at a yard sale when he found the photograph in an antique frame.
	Summary generated by a model fine-tuned using Reinforce
0	John Potter, from Savannah, has admitted forging the picture with his brother in the 1980s and placing it in a frame which now holds a picture of his dead dog. He told historians that Mr Potter handed over to the Georgia Historical Society and the photo became an unofficial part of the ship's history. Potter has now admitted the sham and explained how he falsified the image when he was a teenager in Savannah, Potter, his brother Jeffrey and a friend shot a short 8mm movie about

Table 9: Summaries generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy entity-level factual consistency constraint. Named entities in summaries are highlighted in red.

Source document c

Everton's Steven Pienaar has admitted he considered retirement as frustration took its toll during his injury-ravaged season. The influential 33-year-old midfielder Pienaar has been dogged by groin and knee injuries this season limiting him to just 11 appearances. He returned to action from his latest setback in the 1-1 draw with Swansea City earlier this month but muscle fatigue ruled him out of the win over Burnley last weekend. Everton midfielder Steven Pienaar is held back by Swansea's Ki Sung-Yueng at the Liberty Stadium . Pienaar said: 'At one stage, I thought I had better just hang my boots up and call it a day but on the other side I was just thinking that I enjoy going in and seeing the guys so I just had to stay strong and that kept me going. 'When you are at home not coming in for training you feel very down but as soon as I walk through the door, there's always the camaraderie in the group, there is always fun. 'Even if you are injured, you can always laugh and it keeps you going. Just to be among the players, it's kept me going.' Pienaar has made just 11 appearances for the Toffees this season due to groin and knee injuries .

$b(x, c)$	x
	Summary generated by the original model a
0	the 33-year-old midfielder has been dogged by groin and knee injuries. he has made just 11 appearances for the toffees this season. pienaar returned to action from his latest setback in the 1-1 draw.
	Summary generated by a model fine-tuned using Conditional DPG
0	Steven Pienaar has been dogged by groin and knee injuries this season. the 33-year-old midfielder has made just 11 appearances for the toffees. he returned to action from his latest setback in the 1-1 draw with Swansea city earlier this month.
	Summary generated by a model fine-tuned using DPG
0	the 33-year-old midfielder has been dogged by groin and knee injuries this season. he returned to action from his latest setback in the 1-1 draw with Swansea city. pienaar has made just 11 appearances for the toffees this season due to injury.
	Summary generated by a model fine-tuned using Ziegler
0	the 33-year-old midfielder has been dogged by groin and knee injuries. he has made just 11 appearances for the toffees this season. pienaar returned to action from his latest setback in the 1-1 draw.
	Summary generated by a model fine-tuned using Reinforce
0	Everton's Steven Pienaar has admitted he considered retirement as frustration took its toll during his injury-ravaged season. The influential 33-year-old midfielder Pienanar has been dogged by groin and knee injuries this season for the Toffees. He returned to action from his latest setback in a 1-1 draw with Swansea City earlier this month but muscle fatigue ruled him out of the win over Burnley last weekend. Pienar said Pien

Table 10: Summaries generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy entity-level factual consistency. Named entities in summaries are highlighted in red. constraint

Source document c

Expat: Detectives are investigating the killing of David King (pictured), a 70-year-old pensioner from Newham, east London, who retired to Normandy . A fight over stolen vegetables may have led to a British expat being murdered and dumped at the bottom of a well, French police fear. The macabre theory has been outlined by detectives investigating the killing of David King, a 70-year-old pensioner from Newham, east London, who retired to Normandy 15 years ago. His body was found by sniffer dogs last week in the picturesque hamlet of Pierres, south-west of Caen, and an unnamed 28-year-old Frenchman has been charged with his murder. The alleged killer was living rough in the area, and is thought to have been behind a number of thefts in the area. Now prosecutor Carole Etienne has said that Mr King, a keen gardener, may well have confronted the man over stolen vegetables. Ms Etienne, who is leading the police enquiry, said: 'Among the vague attempts at an explanation given by the accused, we are looking at the possibility of a fight over the theft of foodstuffs. 'In one night, a whole plot of leeks might disappear. One person saw four rabbits vanishing overnight.' Ms Etienne added: 'People were getting more and more angry, and some were even threatening to defend their plots with shotguns'. Neighbours of Mr King, who was hugely proud of his vegetable garden, told Le Parisien newspaper that a description of the thief corresponds to the alleged murderer. Discovery: In February, Mr King's car, a Renault Scenic (above) was found parked in Vire, a nearby town, but there was no sign of any body . Family and neighbours of Mr King have attacked detectives for allowing his suspected murderer to remain at large for six months. Interpol, the international police organisation, had initially refused to open a missing person's enquiry. Instead they believed Mr King had travelled to Australia to see his daughter, Sandie Ray. Mr King had been living in France for 15 years . Ms Ray, who lives in Perth, Australia, said her father's passport details had been mixed up with another David King, who had indeed travelled to Australia from France. She said this was known by November last year, but 'the French authorities still hadn't been formally notified of this via Interpol until approximately three months later'. Ms Ray said a quicker enquiry would have avoided 'lots of anxiety and frustration for our family and dads friends.' John King, Mr King's son, who lives in Brighton, East Sussex, said the botched investigation had been a 'bureaucratic nightmare' for all concerned. Other expats living in the area said it was 'hugely frightening' to have a suspected murderer living in their midst while the operation went on. 'This is an isolated part of the world, and everyone is potentially vulnerable to attack,' said one. 'Detectives should have worked out what was going on far quicker. The slow speed of the enquiry was unacceptable.' In February, Mr King's car, a Renault Scenic was found parked in Vire, a nearby town, but there was no sign of any body.

$b(x, c)$	x
	Summary generated by the original model a
0	the macabre theory has been outlined by detectives investigating the killing of a 70-year-old pensioner from newham, east London. his body was found by sniffer dogs last week in the picturesque hamlet of Pierres, south-west of Caen. an unnamed 28-year old frenchman has been charged with his murder.
	Summary generated by a model fine-tuned using Conditional DPG
1	the macabre theory has been outlined by detectives investigating the killing of David King, 70, from newham, east London, who retired to Normandy 15 years ago. his body was found by sniffer dogs last week in the picturesque hamlet of Pierres, south-west of Caen. an unnamed 28-year-old frenchman has been charged with his murder. prosecutor Carole Etienne has said that Mr King, a keen gardener, may well have confronted the man over over stolen vegetables
	Summary generated by a model fine-tuned using DPG
0	the macabre theory has been outlined by detectives investigating the killing of a 70-year-old pensioner from newham, east London, who retired to Normandy 15 years ago. his body was found by sniffer dogs last week in the picturesque hamlet of Pierres, south-west of caen, and an unnamed 28-year old frenchman has been charged with his murder. prosecutor Carole Etienne has said that the alleged killer may well have confronted the man over stolen vegetables
	Summary generated by a model fine-tuned using Ziegler
0	the macabre theory has been outlined by detectives investigating the killing of a 70-year-old pensioner from newham, east London. his body was found by sniffer dogs last week in the picturesque hamlet of Pierres, south-west of Caen. an unnamed 28-year old frenchman has been charged with his murder.
	Summary generated by a model fine-tuned using Reinforce
1	David King, a 70-year-old pensioner from Newham, east London, who retired to Normandy 15 years ago. His body was found by sniffer dogs last week in the picturesque hamlet of Pierres, south-west of Caen, and an unnamed Frenchman has been charged with his murder. prosecutor Carole Etienne has said Mr King, keen gardener, may well have confronted the man over stolen vegetables, Neighbours of Mr King. Interpol, international police organisation

Table 11: Summaries generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy entity-level factual consistency constraint. Named entities in summaries are highlighted in red.

Source document c

German legend Franz Beckenbauer has accused Bayern Munich stars of playing as if they had taken 'sleeping pills' in their midweek defeat by Porto. The Bundesliga champions conceded twice in the opening 10minutes before losing 3-1 to the Portuguese in the first leg of their Champions League quarter-final on Wednesday. Der Kaiser may be a brand ambassador for the club but he couldn't hide his feelings after the game, when he criticised defender Dante for playing as if he were wearing 'ski boots' before turning on the entire team. Franz Beckenbauer was speaking in New York as part of Bayern Munich's media agreement with MSN . New York Cosmos legends Beckenbauer and Pele pose together after launching the team's spring season . Ricardo Quaresma scored the opening goal from the spot in Porto's 3-1 defeat of the Germans . Beckenbauer accused Brazilian defender Dante (left) of playing as if he were wearing 'ski boots' Speaking to reporters in New York to mark the club's new media agreement with MSN, Beckenbauer said: 'It was one of those days, all the players didn't show their real performance. 'After 10 minutes you are 2-0 down in the quarter-final in the Champions League, so many mistakes. I never saw this before. I thought they took sleeping pills! Porto were much faster, real power. It made me angry. Quaresma celebrates with Porto team-mates after netting his second goal inside the first 10minutes . Bayern Munich head coach Pep Guardiola goes crazy at his team's performance . Bayern players trudge off after defeat knowing they will have to vastly improve to reach the semi-finals . 'If they played like that again, no chance [they can go through]. But they can win 2-0 or go to extra-time. They have a chance. Last year we had the same problem, playing excellent in the Bundesliga and then we had two bad days and were out of the competition.' The 69-year-old did attempt to play down his comments over Dante's performance, he said: 'At half-time, I have to give a comment and that was my reaction. And I'm sorry to give him the ski boots... but he's Brazilian, and Brazilian to me means for me, technique and Pele and... Brazil. But not this.' 'If you want to win the Champions League you need a strong league with a lot of players. Internationally, Bayern Munich belongs to the top teams but to win the Champions League… you saw the game probably two days ago, we are far, far away. With this performance, you don't win anything,' he added.

$b(x, c)$	x
	Summary generated by the original model a
0	german legend claims players played as if they had taken'sleeping pills' porto lost 3-1 to porto in the first leg of their champions league quarter-final. the 69-year-old was speaking to reporters in new york as part of the club's media agreement with MSN.
	Summary generated by a model fine-tuned using Conditional DPG
1	german legend Franz Beckenbauer was speaking in new york as part of the club's media agreement with MSN. he accused defender Dante of playing as if he were wearing'ski boots' the 69-year-old said: 'if they played like that again, no chance [they can go through] but they can win 2-0 or go to extra-time'
	Summary generated by a model fine-tuned using DPG
0	the germans conceded twice in the opening 10 minutes before losing 3-1 to porto. the 69-year-old criticised defender defender Dante for playing as if he were wearing'ski boots' he said: 'if they played like that again, no chance [they can go through]'
	Summary generated by a model fine-tuned using Ziegler
0	german legend claims players played as if they had taken'sleeping pills' porto lost 3-1 to porto in the first leg of their champions league quarter-final. the 69-year-old was speaking to reporters in new york as part of the club's media agreement with MSN.
	Summary generated by a model fine-tuned using Reinforce
1	Franz Beckenbauer has accused Bayern Munich stars of playing as if they had taken'sleeping pills' the Bundesliga champions concede twice in opening 10minutes before losing 3-1 to the Portuguese in their Champions League quarter-final on Wednesday. Der Kaiser may be a brand ambassador for the club but he couldn't hide his feelings after the game. Ricardo Quaresma scored his opening goal from spot in Porto's 3-1 defeat of the Germans. Bayern Munich head coach Pe

Table 12: Summaries generated by beam search on $\pi_\theta(\cdot|c)$: models fine-tuned to satisfy entity-level factual consistency constraint. Named entities in summaries are highlighted in red.

$b(x, c)$	$[x, c]$
	Sequences sampled from the original model a

0	`def __init__(self):`
	`  self.peneturn = TitlePane(text='Screen' if len(self.peneturn) > 0 else "Find" )`
	`  self. gentlemen = TitlePane()`
	`  self.managers = TitlePane()`
	`  self.title = TitleBox(title='Title' )`
	`  self.pane = self.hides(self.menues, menues=self.menues)`
	`  self.instructions = Label(text= 'I' )`
	`  self.scripture =`

0	`def __init__(self):` `  """`

0	`def __init__(self):`
	`  self.attach_button()`
	`  self.eager_init()`
	`  self.x_template = elem_self.template "x_template"`

	Sequences sampled from a model fine-tuned using Conditional DPG
1	`def __init__(self):` `  self.context = 100`

1	`def __init__(self):`
	`  self.html_to_css_css2 = response()`
	`  self.html_to_content_css3 = response()`
	`  self.html_to_css3 = response()`

1	`def __init__(self):` `  super(DispAdmin, self).__init__(self)`
	Sequences sampled from a model fine-tuned using DPG
1	`def __init__(self):` `  pass`

1	`def __init__(self):`
	`  stuff = createObject()`
	`  self.pushItem(stuff, __get_numbers, 'numbers', - '0', 'tnC' )`
	`  stuff.name = '#%s';`
	`  self.pop()`

1	`def __init__(self):`
	`  self.gameDataGroup = []`
	`  self.playDataSetGroup = ['set']`
	`  self.lpsCreate = None`
	`  self.playerDataSetGroup = [class_DataGroup]`
	`  if self.gameDataGroup is not None:`
	`  self.gameDataGroup.append(self.gameGroup)`

	Sequences sampled from a model fine-tuned using Ziegler
1	`def __init__(self):` `  self.headers_headers_headers_headers_headers_headers_headers_headers_headers_headers_`
1	`def __init__(self):` `  self.shape_children_children_children_children_children_children_children_`
1	`def __init__(self):` `  self._items_items_items_items_items_items_items_items_items_items_items_items_items_items_items_items_items_`
	Sequences sampled from a model fine-tuned using Reinforce
1	`def __init__(self):` `  self.screen = screen`

1	`def __init__(self):`
	`  self.value = None`
	`  self.shadow = None`
	`  self.showTitles()`
	`  self.context = None`

1	`def __init__(self):`
	`  self.description = ''`
	`  self.title  = ''`

Table 13: Samples obtained from $\pi_\theta(\cdot|c)$ with $c = $ `def __init__(self)` fine-tuned to satisfy a compilability constraint

$b(x, c)$	$[x, c]$
	Sequences sampled from the original model a

0	`def __init__(self,* args,** kwargs):`
	`  self.algorithm = algorithm.__algorithm`
	`  self.scope = scope.scope`
	`  instance = remotely(self.altstr, 1.4)`
	`  if (num == 3 and instance.level >= 2):`
	`  self.alg = alg`
	`  self.algorithm=algorithm`
	`  self.scope = sprintf(self.scope, "%r\n", "%d\n")`
	`  return *self.alg`

1	`def __init__(self,* args,** kwargs):` `  super(SceneApplication,self).__init__(args,kwargs,self.plane)`

0	`def __init__(self,* args,** kwargs):`
	`  from base import *`
	`  from dict import ddarray`
	`  from. import *`
	`  from dbase import *`
	`  from dbase.serial.calls import *`
	`  from. import __file__ import *`
	`  from. __del__ import *`

	Sequences sampled from a model fine-tuned using Conditional DPG
0	`def __init__(self,* args,** kwargs):`
0	`def __init__(self,* args,** kwargs):` `  from PIL import *`
0	`def __init__(self,* args,** kwargs):`
	Sequences sampled from a model fine-tuned using DPG

1	`def __init__(self,* args,** kwargs):`
	`  raise ValueError('' + self.__class__.__name__ +'does not exist'`
	`  + self.__class__.__name__,__LINE__)`

1	`def __init__(self,* args,** kwargs):`
	`  self.args = args`
	`  self.kwargs = kwargs`

0	`def __init__(self,* args,** kwargs):`
	Sequences sampled from a model fine-tuned using Ziegler

1	`def __init__(self,* args,** kwargs):`
	`  self.args = args`
	`  self.kwargs = kwargs`
	`  self.kwargs_headers_headers_headers_headers_headers_headers_headers_headers_headers_headers_headers_headers_headers`

1	`def __init__(self,* args,** kwargs):`
	`  self.args = args`
	`  self.kwargs = kwargs`
	`  self.kwargs = kwargs`
	`  self.kwargs = kwargs`
	`  self.kwargs = kwargs`
	`  self.kwargs = kwargs`
	`  self.kwargs = kwargs`
	`  self.kwargs = kwargs`

1	`def __init__(self,* args,** kwargs):`
	`  self.args = args`
	`  self.args_args_args_args_args_args_args_args_args_args_args_args_args_args_args_args_args_args_args_args_`

	Sequences sampled from a model fine-tuned using Reinforce
1	`def __init__(self,* args,** kwargs):` `  self.typer = typer`

1	`def __init__(self,* args,** kwargs):`
	`  self.args = args`
	`  self._unique_args = self._unique_args`

1	`def __init__(self,* args,** kwargs):` `  self.args = args`

Table 14: Samples obtained from $\pi_\theta(\cdot|c)$ with $c =$ `def __init__(self,* args,** kwargs)` fine-tuned to satisfy a compilability constraint

$b(x, c)$	$[x, c]$
	Sequences sampled from the original model a

1	`def __iter__(self):`
	`  from models import model`
	`  params = model.take_model_parameters()`
	`  is_float = model.is_float_pos(not params)`
	`  solvent = model.penesannex(params, is_float)`
	`  solver_params = model.torneyred(params)`
	`  solver_nwm(params)`

1	`def __iter__(self):`
	`  """`
	`  Compare transformations stored in memory`
	`  """`
	`  if self.getFloat():`
	`  self.reverseTransform()`
	`  return list(self)`

1	`def __iter__(self):` `  self.filemanager()`
	Sequences sampled from a model fine-tuned using Conditional DPG

0	`def __iter__(self):`
	`  self.message = _L"{} a multidimensional array java geogrifier\n${}`
	`  { }`
	`  """`
	`  if len(self.string_items) >= strlen(self.values):`
	`  self.message += "[0] an empty list or json object\n\n"`
	`  self.key`

1	`def __iter__(self):`
	`  self.container = None`
	`  self.desc = None`
	`  self.interview_name = None`
	`  self.size_displacement_matrix = None`

0	`def __iter__(self):` `  """Destructor for test pairs.`
	Sequences sampled from a model fine-tuned using DPG
1	`def __iter__(self):` `  return self.__iter__(self)`
1	`def __iter__(self):` `  return [self.genState[0] for _ in self.stateNodes]`

1	`def __iter__(self):`
	`  created = True`
	`  for i in xrange(0, len(self.callover)):`
	`  self.callover[scoped_async_callover(i)] = created`

	Sequences sampled from a model fine-tuned using Ziegler
1	`def __iter__(self):` `  self.items_items_items_items_items_items_items_items_items_items_items_items_items_items_items_items_items_items`

1	`def __iter__(self):`
	`  self.iterator =`
	`  ↪ iterator_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops_ops`

1	`def __iter__(self):` `  self._iter_names_children_children_children_children_children_children_children_children_children_children`
	Sequences sampled from a model fine-tuned using Reinforce
1	`def __iter__(self):` `  self.next()`
1	`def __iter__(self):` `  self._iterator = self`
1	`def __iter__(self):` `  self.__iter__()`

Table 15: Samples obtained from $\pi_\theta(\cdot|c)$ with $c = $ `def __iter__(self)` fine-tuned to satisfy a compilability constraint

$b(x, c)$	$[x, c]$
	Sequences sampled from the original model a
0	`def __init__(self):` `  """ mutation() method.`

0	`def __init__(self):`
	`  self.vcam = cham OLEDeaderCenterPythRHAPUV()`
	`  self.vcam_rc = bool_ ("with_ DisneySaudiDisney) Official Created Source Title and Updated Collection`
	`  self.volume = float(1000)`
	`  self.title = float(params()[3])`

0	`def __init__(self):`
	`  """`
	`  Returns the inner type that is of the proper type when performing`
	`  self-self-transpose().`
	`  """`
	`  self.th = "arg1",`

	Sequences sampled from a model fine-tuned using Conditional DPG

0	`def __init__(self):`
	`  self.tile_flag_handler_out_A = True`
	`  self.tile_flag_handler_out_B = True`
	`  self.layer_list_out_A = self.tile_flag_invalid=2`
	`  self.layer_list_out_B = self.tile_flag_handler_out_A`
	`  self.layer_select_with_dropdown_flat_flag_in=self.tile_flag_invalid=`

0	`def __init__(self):` `  return self.InitExceptaughtClient();`

1	`def __init__(self):`
	`  flood_handler = ScreenGroup.set_buffer_manager_function(buffers, None)`
	`  heavy = _charging_class(self, fill_handler)`

	Sequences sampled from a model fine-tuned using DPG

1	`def __init__(self):`
	`  class App(QtWndrixApplication):`
	`  Slipped: bool = QtWnd technique  # Member of gtk.SafePlacement floated`
	`  delegate: NamedConstraint: QListPlacementElementDelegateDelegate`

0	`def __init__(self):`
	`  def gqlary(self):`
	`  presentation_data = (["$parent", " Imperium", "Segaron", "la datos de 214000 "],`
	`  [["Lamos en su inicio", 0, "Por supuesto", 0],`

0	`def __init__(self):`
	`  self.msg = {}`
	`  #Placeholders`

	Sequences sampled from a model fine-tuned using Ziegler

0	`def __init__(self):`
	`  name=SHA_['base'].__name__().strip()`
	`  message=Message(path=_md5. 'value'))`

1	`def __init__(self):`
	`  as_printer = asv(0)`
	`  i = 3`
	`  symbol = _sparse_symbol(0)`

0	`def __init__(self):` `  self.identity_dont_print_view = 'Identities'`
	Sequences sampled from a model fine-tuned using Reinforce

0	`def __init__(self):`
	`  self.dataset = TensorsRotatedDeviceDataset()`
	`  self.start = None`
	`  self.dataset.dataset = TensorsRotatedPose(self, 100)`

1	`def __init__(self):` `  tf.__init__(self)`

Table 16: Samples obtained from $\pi_\theta(\cdot|c)$ with $c = \texttt{def \_\_init\_\_(self)}$ fine-tuned to satisfy a PEP8 constraint

$b(x, c)$	$[x, c]$
	Sequences sampled from the original model a

0	`def __init__(self,* args,** kwargs):`
	`  it = lambda: args`
	`  it(None)`

0	`def __init__(self,* args,** kwargs):`
0	`def __init__(self,* args,** kwargs):` `  raise RuntimeError('Cannot create new thread');`
	Sequences sampled from a model fine-tuned using Conditional DPG

0	`def __init__(self,* args,** kwargs):`
	`  dgram = dgram_preprocessor(self.parser,* args,**kwargs)`
	`  self.node = __defaultdict__[node]`
	`  kwargs = kwargs`
	`  self.classpath = "%(classpath)s//%(pathname)s//"`
	`  self.param = classpath()`
	`  self.property = classpath()`
	`  self.sys.exists = False`
	`  return self`

0	`def __init__(self,* args,** kwargs):`
	`  self.app = comprom::Configuration(conf())`
	`  conf = self.confswite(args.get('app'), args.get('name', ''), conf)`

0	`def __init__(self,* args,** kwargs):`
	Sequences sampled from a model fine-tuned using DPG
0	`def __init__(self,* args,** kwargs):`
0	`def __init__(self,* args,** kwargs):`

0	`def __init__(self,* args,** kwargs):`
	`  self.tip  = {}`
	`  self.buffer = ''`
	`  self.clean_of_this  = ''`
	`  self.tip.set_value(op_type())`
	`  self.tip.set_value(ostream(f" excerpting this", oproc = self.poetrip_extrapol))`

	Sequences sampled from a model fine-tuned using Ziegler

0	`def __init__(self,* args,** kwargs):`
	`  U particles = sorted(p.get_particles(c4))`
	`  kwargs.append(p)`
	`  if self.like_class:`
	`  p.like_class = kwargs[0]`

0	`def __init__(self,* args,** kwargs):`
	`  if not c.c.service.defaultService():`
	`  args.favoriter(c.service.defaultService(c.pass.name))`
	`  if not c.simpleAddr():`
	`  d.favoriter(c.simpleAddr)`
	`  if c.simpleAddr():`
	`  c.gop.services.import_service(d`

0	`def __init__(self,* args,** kwargs):`
	`  self.init_args(args,kwargs)`
	`  self.base_args = base_args`

	Sequences sampled from a model fine-tuned using Reinforce
0	`def __init__(self,* args,** kwargs):` `  """ In constructor, bubbles can be used accordingly.`

0	`def __init__(self,* args,** kwargs):`
	`  self.akpo_be(akpo_key,**args)`
	`  self.appo_a(a,r)\`
	`  .name(dfn)//type(dfn)-1`
	`  sdt=self.akpo_ fishing(a//a[0],r@a[0])[0]`
	`  sd=self.akpo_ a`
	`  dfn=self.akpo_df(ado(dfn)),`
	`  a=self.akpo_a(rs):`

0	`def __init__(self,* args,** kwargs):` `  from __future__ import print_function`

Table 17: Samples obtained from $\pi_\theta(\cdot|c)$ with $c =$ `def __init__(self,* args,** kwargs)` fine-tuned to satisfy a PEP8 constraint

$b(x, c)$	$[x, c]$
	Sequences sampled from the original model a
1	`def __repr__(self):` `  return "__repr__"`

0	`def __repr__(self):`
	`  raise StopIteration("operator"`
	`  "(mut_index_t) not an iterable", self)`
	`  self.ignoreSelfToken(self)`

0	`def __repr__(self):`
	`  self.memory = self._object_create_memory`
	`  self.mainfile = None`
	`  def ref() -> keyboard:`
	`  raise self.error('"--<-' pass)`

	Sequences sampled from a model fine-tuned using Conditional DPG

0	`def __repr__(self):`
	`  """standard`
	`  :param self:`

0	`def __repr__(self):`
	`  if __name__ == '__main__':`
	`  return self.os_main()`
	`  else:`
	`  return self.req.as_raw_object(`
	`  __name__,`
	`  :method => 'GET',`
	`  accept_parser=self.parser, accept_kwargs=self.accept_kwargs,`

1	`def __repr__(self):` `  return self._name + '_' + self._rubyblob.result`
	Sequences sampled from a model fine-tuned using DPG
1	`def __repr__(self):` `  self.loadstrli()`

0	`def __repr__(self):`
	`  """`
	`  Return the normal to ancient_api._has_proto(self) property of the`
	`  class. To unlock up the hyperlabels, use __set__() instead.`
	`  """`
	`  old_l = lazy_data_type(l)`
	`  found = set()`
	`  raw = lazy_data_type( rawToUpperNone, "need full valid types for serialized`
	`  datatypes" ) where idxs = raw; '1' if raw and hasattr( raw`

0	`def __repr__(self):` `  """Revert to the previously returned Signing direction.`
	Sequences sampled from a model fine-tuned using Ziegler

0	`def __repr__(self):`
	`  with self:`
	`  self.__repr__[ 0.0] = 1`

1	`def __repr__(self):` `  _ = self.name`
0	`def __repr__(self):` `  y`
	Sequences sampled from a model fine-tuned using Reinforce

1	`def __repr__(self):`
	`  self = self._self`
	`  self._self = self._self`
	`  return self`

0	`def __repr__(self):`
	`  __repr__(self):`
	`  v = self.__repr__(self)`
	`  if v not in __repr__:`
	`  __repr__(self, v)`
	`  return v`

0	`def __repr__(self):` `  print("\n...\n....").stack(self.dim[0], self.dimensional[0][0], sizeof(self) * self.dimensional[0])`

Table 18: Samples obtained from $\pi_\theta(\cdot|c)$ with $c = \texttt{def \_\_repr\_\_(self)}$ fine-tuned to satisfy a PEP8 constraint