PPL-MCTS: Constrained Textual Generation Through Discriminator-Guided MCTS Decoding

Antoine Chaffin[1,2], **Vincent Claveau**[1], **Ewa Kijak**[1]
[1]CNRS, IRISA, Univ. Rennes 1, Campus de Beaulieu, 35000 Rennes, France
[2]IMATAG, 13 Rue Dupont-des-Loges, 35000 Rennes, France
{antoine.chaffin, vincent.claveau, ewa.kijak}@irisa.fr

Abstract

Large language models (LM) based on Transformers allow to generate plausible long texts. In this paper, we explore how this generation can be further controlled at decoding time to satisfy certain constraints (e.g. being non-toxic, conveying certain emotions, using a specific writing style, etc.) without fine-tuning the LM. Precisely, we formalize constrained generation as a tree exploration process guided by a discriminator that indicates how well the associated sequence respects the constraint. This approach, in addition to being easier and cheaper to train than fine-tuning the LM, allows to apply the constraint more finely and dynamically. We propose several original methods to search this generation tree, notably the Monte Carlo Tree Search (MCTS) which provides theoretical guarantees on the search efficiency, but also simpler methods based on re-ranking a pool of diverse sequences using the discriminator scores. These methods are evaluated, with automatic and human-based metrics, on two types of constraints and languages: review polarity and emotion control in French and English. We show that discriminator-guided MCTS decoding achieves state-of-the-art results without having to tune the language model, in both tasks and languages. We also demonstrate that other proposed decoding methods based on re-ranking can be really effective when diversity among the generated propositions is encouraged.

1 Introduction

Generative language models exist for a long time, but with advent of the transformer architecture (Vaswani et al., 2017) and increasing computing capabilities, they are now able to generate well written and long texts. In particular, large models, such as the well known GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020), have been used successfully for various applications: assisting writers, summarizing, augmentating data for subsequent NLP tasks, generating fake news (Kumar et al., 2020; Papanikolaou and Pierleoni, 2020; Zellers et al., 2019). Yet, beside the prompt used to initiate the generation process, there are few options to have control on the generation process. Being able to add some constraints on the generated texts is useful for various situations. For example, it allows to create texts that follow a certain writing style, convey a certain emotion or polarity or to ensure that a generated summary contains correct information. More critically, it can be used to prevent the inherent toxicity of language models trained on the internet, or to not reproduce gender or race stereotypes. So far, most methods necessitate to fine-tune the LM, so that it specifically learns to model this constraint, i.e. the constraint is –hopefully– incorporated in the LM. This fine-tuning approach has several drawbacks. It implies to train multiple specific LMs (one per constraint), which is costly, when even possible given the size of current state-of-the-art LM, and results in several models.

In this paper, we propose new approaches to add such additional constraints on the texts but at decoding time. We exploit a discriminator that is trained to determine if a text follows a given constraint or not; its output provides information to guide the generation toward texts that satisfy this expected constraint. In order to make the most of the discriminator information, we propose an original method based on the Monte Carlo Tree Search (MCTS) algorithm (Coulom, 2006), namely Plug and Play Language - Monte Carlo Tree Search (PPL-MCTS). We also propose simpler methods based on re-ranking to fulfil this goal. Both approaches do not require to fine-tune the LM; adding a new constraint can thus simply be done by providing a discriminator verifying if a text complies with what is expected. More precisely, our main contributions are the following ones:

1. we propose to use MCTS as a decoding strat-

egy to implement constrained generation and we show, on 3 datasets and 2 languages, that it yields state-of-the-art results while offering more flexibility;

2. we also explore simpler generation methods based on re-ranking and show that this kind of approach, with low computational costs, can also be competitive if the diversity within propositions to re-rank is encouraged;

3. we provide a fully functional code implementing a batched textual MCTS[1] working with the popular HuggingFace's Transformers library (Wolf et al., 2020)

2 Related work

The goal of constrained textual generation is to find the sequence of tokens $x_{1:T}$ which maximises $p(x_{1:T} \mid c)$, given a constraint c. Few methods address the constrained textual generation.

Class-conditional language models. Class-conditional language models (CC-LMs), as the Conditional Transformer Language (CTRL) model (Keskar et al., 2019), train or fine-tune the weights θ of a single neural model directly for controllable generation, by appending a control code in the beginning of a training sequence. The control code indicates the constraint to verify and is related to a class containing texts that satisfy the constraint. For the sake of simplicity, we will denote without distinction the class, the constraint verified by its texts and the associated control code by c. Trained with different control codes, the model learns $p_\theta(x_{1:T} \mid c) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{1:t-1}, c)$. The constraint can then be applied during generation by appending the corresponding control code to the prompt. While this method gives some kind of control over the generation, the control codes need to be defined upfront and the LM still needs to be trained specifically for each set of control codes. This is an important limitation since the current trend in text generation is the use of large pre-trained models which can hardly be fine-tuned (for instance, the last version of GPT, GPT-3, cannot be fine-tuned without access to very large hardware resources).

Discriminator-based methods The general idea of discriminator-guided generation is to combine a disciminator D with a generative LM. The discriminator explicitly models the constraint by calculating the probability $p_D(c \mid x_{1:T})$ of the sequence $x_{1:T}$ to satisfy the constraint c. This probability is directly related to $p(x_{1:T} \mid c)$ through Bayes' rule : $p(x_{1:T} \mid c) \propto p_D(c \mid x_{1:T}) p_\theta(x_{1:T})$. Discriminator-based methods alleviate the training cost problem, as discriminators are easier to train than a LM. Moreover, any additional constraint can be defined a posteriori without tuning the LM, only by training another discriminator. The discriminators have been used in different ways to explore the search space. In the work of (Holtzman et al., 2018; Scialom et al., 2020), the space is first searched using beam search to generate a pool of proposals with a high likelihood $p_\theta(x_{1:T})$, and then the discriminator is used to re-rank them. However, in addition that beam search can miss sequences with high likelihood, it is biased towards the likelihood, while the best sequence might only have an average likelihood, but satisfies the constraint perfectly.

Hence, it might be more suitable to take the discriminator probability into account during decoding rather than after generating a whole sequence. In this case, the discriminator is used at each generation step t to get the probability $p_D(c \mid x_{1:t})$ for each token of the vocabulary $\mathcal{V}$, and merge it to the likelihood $p_\theta(x_{1:t})$ to choose which token to emit. In order to reduce the cost of using a discriminator on every possible continuation, GeDi (Krause et al., 2020) proposes to use CC-LMs as generative discriminators. The method relies on the fact that the CC-LM computes $p_\theta\left(x_t \mid x_{1:t-1}, c\right)$ for all tokens of the vocabulary which can be used to get $p_\theta(c \mid x_{1:t})$ for all tokens using Bayes' equation. This approach is thus at the intersection of tuning the LM and using a discriminator: it tunes a small LM (the CC-LM) to guide a bigger one.

In Plug And Play Language Model (PPLM) (Dathathri et al., 2020), the discriminator is used to shift the hidden states of the pre-trained transformer-based LM towards the desired class at every generation step. PPLM can be used on any LM and with any discriminator. However, PPLM needs to access the LM to modify its hidden states, while our approach only requires the output logits. As some LM can only be used through access to logits (e.g. GPT-3 API), this makes our approach more plug and play than PPLM.

A common drawback of all these approaches is their lack of a long-term vision of the generation.

[1] https://github.com/NohTow/PPL-MCTS

Indeed, the discriminator probabilities become necessarily more meaningful as the sequence grows and might only be trustable to guide the search when the sequence is (nearly) finished. When used in a myopic decoding strategy, classification errors will cause the generation process to deviate further and further. Trying to optimize a score defined in the long horizon by making short term decisions is very similar to common game setups such as chess, where the Monte Carlo Tree Search (MCTS) has proven to be really effective (Silver et al., 2018), which motivated our approach.

3 PPL-MCTS method

The approach that we propose is in line with methods using a discriminator to guide a large LM decoding, without the need to re-train it. Also, it can be applied to any LM with any discriminator, following the plug and play paradigm. Unlike previous approaches, it is able to have a long term vision on what is generated. Being able to make a short-term decision (choice of the next token x_t at time step t) that is promising in the long run is based on the exploration of the search space. We propose here to use the Monte Carlo Tree Search (MCTS) for an efficient exploration of this space.

MCTS is very well suited for this problem for three reasons. First, it allows to get a local score (i.e, a score for the next token to emit) using finished sequences. Hence, this score is more meaningful than scores based only on the next step. Second, it allows to explicitly define the compromise between exploitation of promising sequences (with a high likelihood), and exploration of other potentially promising sequences (to not miss better sequences with a lower likelihood). The fact that regret, i.e the number of simulations done on a sub-optimal sequence, has a theoretical upper bound in MCTS (Rosin, 2011) is a nice guarantee that the computation time is not wasted and the search is efficient. Finally, it outputs a solution at each iteration and so can fit our computational budget by allowing to adjust the quality of the solution to calculation spent.

Text generation as tree exploration process. The search space of the text generation corresponds to a tree: its root is the prompt and the child of a node is its father's sequence with one of the $|\mathcal{V}|$ possible tokens appended. In the case of constrained generation, the goal is thus to find the path, and therefore the sequence x, with the highest $p(x \mid c)$

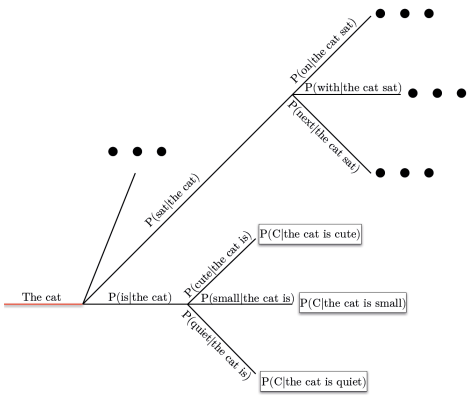

Figure 1: Illustration of the constrained generation process as a tree exploration from the prompt `The cat`. Classification probabilities are only represented on completed sequences.

possible without exploring the whole tree in width and depth. As mentioned previously, this probability can be computed as the product of the likelihood $p_\theta(x)$ and the probability given by a discriminator $p_D(c \mid x)$. An illustration of such a tree can be found in Fig. 1, where the likelihood of x is forged by multiplying corresponding conditional probabilities along the path, and the classification probability is calculated at the terminal node.

Applying MCTS to text generation. MCTS is a heuristic based iterative algorithm that uses randomness to solve deterministic problems that cannot be solved using traditional approaches, often because the search space is too large to be entirely explored. Each iteration consists in four consecutive steps. In the particular context of applying MCTS to text generation, we made some adaptations:

1. **Selection** Recursively choose children from the root to a node that has not been expanded yet. To only explore viable sequences, the probability $p_\theta(x_i \mid x_{1:t-1})$ of a given token x_i given by the LM is used during the selection phase. To this end, the children chosen are those maximizing the Polynomial Upper Confidence Trees (PUCT) (Rosin, 2011) as defined in (Silver et al., 2017):

$$PUCT(i) = \frac{s_i}{n_i} + c_{puct}\, p_\theta(x_i \mid x_{1:t-1}) \frac{\sqrt{N_i}}{1+n_i} \quad (1)$$

with s_i the aggregated score of the node i, n_i the number of simulations played after this node, N_i the number of simulations played

after its parent, and c_{puct} a constant defining the compromise between exploration and exploitation. In the task of constrained generation, we define the score of a sequence as its probability knowing the class $p(x \mid c)$.

2. **Expansion** If the selected node is not terminal, use the LM to expand it by creating its children.

3. **Simulation (roll-out)** Sample one of these children according to $p_\theta(x_i \mid x_{1:t-1})$, and go to a terminal node through a random walk or another pattern.

4. **Backpropagation** Aggregate the final score obtained at the terminal node ($p(x \mid c)$) to each parent until root. There are different strategies to aggregate scores, as computing the average between the actual score and the one being backpropagated, or taking the maximum of the two. We take the aggregated score s_i associated to the node i as the averaged probability over all simulations played after this node.

When the number of iterations has reached the allocated budget, the building of the tree stops. The token x_i selected for the current decoding step can be selected as the most played node amongst the root's children nodes, or the one with the highest aggregated score. We chose the most played one.

These adaptations of MCTS to constrained generation are summarized in Fig. 2. Note that any language model can be used for defining the probability $p_\theta(x_i \mid x_{1:t-1})$ and any discriminator for scoring sequences, hence the name of our approach: Plug and Play Language - Monte Carlo Tree Search (PPL-MCTS). MCTS has been very recently used for machine translation (Leblond et al., 2021), question generation and summarization (Scialom et al., 2021). The differences with these concurrent studies are discussed in Appendix A.5.

Model improvements. In order to allow a finer control on how the constraint is applied, we introduce a parameter $\alpha \in [0, 1]$ to control the compromise between likelihood and constraint strength, modifying Bayes' equation: $p(x \mid c) \propto p_D(c \mid x)^\alpha p_\theta(x)^{1-\alpha}$. Note that PUCT (1) already considers the likelihood of the sequence, favoring the selection of nodes with high likelihoods. Hence, even sequences generated with $\alpha = 1$ are correctly

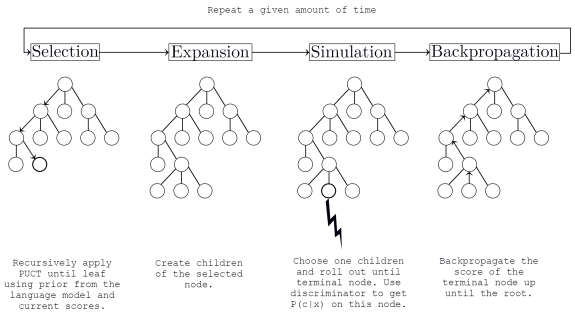

Figure 2: MCTS application to text generation.

written. Setting $\alpha < 1$ forces the algorithm to explore solutions even closer to the language model. In our experiments, we set $\alpha = 1$ to strengthen the constraint.

To avoid expensive roll-outs, one may also assign a value to unfinished sequences at the cost of a less precise evaluation that may be not as meaningful as when doing roll-outs. Indeed, the discriminator can be trained on sequences with variable numbers of tokens, allowing it to be used at each node without the need of simulations. In this setup, the MCTS is used as an efficient compromise between exploration and exploitation, losing part of its long view property but allowing to skew the exploration toward promising solutions.

Finally, during our first experiments, we observed that PPL-MCTS leads to repetitive patterns. This is very similar of what happens with greedy search, where a single sequence with a high likelihood is dominating the search. If such sequences also have a pretty high discriminator scores, they will be repeated often. CTRL (Keskar et al., 2019) offers a simple yet very powerful method to avoid noisy repetitions. It applies a scalar factor $I(i)$ to the temperature parameter τ of a given token x_i that penalizes this token if it is already in the input sequence. The probability of a given token becomes:

$$p'_\theta(x_i \mid x_{1:t-1}) = \frac{\exp\left(z_i/(\tau \cdot I(i))\right)}{\sum_v \exp\left(z_v/(\tau \cdot I(v))\right)} \quad (2)$$

with the *repetition penalty* $I(i) > 1$ if x_i is already in the prompt and 1 otherwise, and z the neural LM predicted logits over the vocabulary $\mathcal{V}$. Thus, probabilities of already emitted tokens are penalized, but if the language model gives a really high score to one token (hence, it is very confident that this *should* be the token to emit), it may still be selected as the output token.

4 Experiments

4.1 Performance assessment

The goal of constrained generation is to generate samples that 1) belong to a specific class while 2) keeping the language quality of the original LM, and 3) with enough diversity across samples. We chose three different metrics to evaluate each of these aspects: 1) accuracy, which is verified by an external "oracle" discriminator trained on a dataset disjoint from the one used to guide the generation; 2) perplexity, which is computed using an "oracle" LM, i.e an unconstrained LM trained on different data than the one used to train the constrained generator; 3) Self-BLEU score (Zhu et al., 2018), which is the BLEU score (Papineni et al., 2002) of a sample using the other samples as references: a high Self-BLEU score means that there is a lot of overlap between generated samples, and thus that the diversity is low. Such automatic metrics have known limitations (Caccia et al., 2020) but results of human evaluation on the CLS dataset, detailed in Section 4.6, confirm that they provide a good overview of the performance.

In practice, the studied dataset (see below) is split into two parts, each part being sub-divided in train/val/test sets. The first part serves to train models used for the generation (LM and discriminator), while the second is used to train oracles which serve to compute the automatic evaluation metrics. The test set of this second part will also be used to forge prompts for the generation. Further details on data splits are given in Appendix A.1. Each metric is evaluated on a pool of 900 generated samples.

4.2 Datasets

Three different datasets are used in the experiments presented hereafter: amazon_polarity (Zhang et al., 2015), CLS (from the FLUE (Le et al., 2020) dataset) and emotion (Saravia et al., 2018). The first two are Amazon reviews which have been labeled as positive or negative, so the intended task is to study the possibility of applying polarity to the generation. As CLS is in French, these two datasets will serve to ensure that the methods have the same behaviour for different languages. Emotion is a collection of tweets classified under eight basic emotions: anger, anticipation, disgust, fear, joy, sadness, surprise and trust. This dataset is supposed to be more challenging since there are more classes and texts are smaller (only composed of one sentence), hence the model needs to precisely generate the target emotion with few tokens. It is worth noting that the 3 datasets have different sizes: 4,000,000 instances in total for amazon_polarity, 20,000 for emotion and 6,000 for CLS. They are available at `https://huggingface.co/datasets/`.

We adapted prompts used to start the generation for each datasets depending on the data format. Amazon_polarity comes with a "title" column which corresponds to the title the user gave to the review. This field is directly used as prompt. For the two other datasets, the prompts are the very first tokens of the text field. Because texts from emotion and CLS have different lengths, the size of prompts are also different: it is arbitrarily set to 6 tokens for CLS and 4 for emotion.

4.3 Methods and baselines

Baselines. Beside PPL-MCTS, we propose several baselines and simple techniques. Most studies on re-ranking create proposals using beam search and then re-rank them using the product of likelihood and discriminator probability, limiting the diversity in the proposals pool. We propose re-ranking with different variations, in the way sequences to re-rank are produced, and in the way the final sequence is chosen in an attempt to improve such approaches. Three methods are tested to generate propositions: beam search (Dept., 2018) (with a beam size of 3), nucleus (top-p) sampling (Holtzman et al., 2020) (with p=0.9), as well as beam sampling (as described in (Caccia et al., 2020)). For the final choice, we also propose three different methods: *argmax*, where the sequence that has the highest $p(x|c)$ is chosen; *first true*, where propositions are sorted by descending likelihood and the first sequence that belongs to the correct class according to the guiding discriminator is chosen; and *sampling*, where the distribution of $p(x|c)$ for the propositions is normalized and the chosen sequence is sampled following this distribution. Similarly to PPL-MCTS, the likelihood part of $p(x|c)$ is omitted (i.e, $\alpha = 1$) since sequences in the pool of propositions already have an high likelihood.

It should be noted that in our setting, a generated sequence corresponds to a document (e.g. a whole review). This choice makes sense for our datasets, but re-ranking at a smaller level (after each sentence, after x tokens...) would also be possible and might produce different results.

<|startoftext|> The Revenge of making a good Halloween film. [SEP]?????? I think this movie is a waste of time. It's not scary, it's just plain stupid. The only good thing about this film is the soundtrack.<|endoftext|>
<|startoftext|> The Revenge of making a good Halloween film. [SEP] ive seen this movie a few times and i love it. the acting is great, the story line is good, and the special effects are awesome. if you like horror movies then go see this one.<|endoftext|>

Figure 3: Example of two constrained generations using PPL-MCTS, one on the negative class, one on the positive class, using the same prompt (in bold) from amazon_polarity.

Methods from the literature We compare our results with methods from the literature. In particular, we test CC-LMs trained on the target task, similarly as CTRL. We tested this method using greedy search as well as sampling for decoding. We also propose an implementation of CC-LM trained with the classification loss initially proposed for the GeDi method (Krause et al., 2020). These CC-LMs are further used to implement the state-of-the-art GeDi model. In the experiments reported below, we report results for GeDi models trained with and without the classification loss. Finally, we report results of PPLM. For a fair comparison, the same discriminator and LM are used for our PPL-MCTS approach, the re-ranking approaches (baselines), and PPLM.

4.4 Experimental setting

For each method, a number of tokens equal to the average length of sequences of the dataset are generated: 98 tokens for amazon_polarity, 23 for emotion and 137 for CLS. Fixing the number of generated tokens ensures fair comparisons between the tested methods. Indeed, even though perplexity and Self-BLEU metrics are normalized by the length of the sequence, these metrics can tend to penalize a model producing longer sequences: such model has more risk to deviate and repeat itself, which would results in higher values compared to a model producing shorter sequences. An example of generation from amazon_polarity is given in Fig. 3.

To run all of these methods, three different models are needed: one discriminator, a "vanilla" LM used as generator, and the CC-LM used in the CTRL and GeDi approaches. For the discriminator used to guide the generation, we rely on BERT-base-cased (Devlin et al., 2019) for the English datasets and FlauBERT-large-cased (Le et al., 2020) for CLS. As vanilla LM, we use GPT-2 small models, relying on OpenAI's pre-trained model for the English datasets and on belgpt2 for

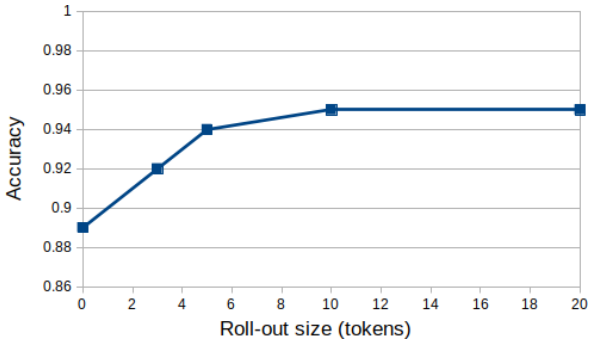

Figure 4: Accuracy according to the roll-out size; CLS dataset

the French one. The implementation and models used for BERT, FlauBERT, GPT-2 and belgpt2 are all found on `https://huggingface.co/models`. Given the particular format of data on our experimental datasets, the vanilla LM is trained on raw training sequences in order to produce texts corresponding to the task (for instance, reviews). The CC-LM is simply a fine-tuned version of the vanilla LM with the control code appended.

We tested three values for the temperature parameter for each proposed method (1.0, 1.1 and 1.2). For PPL-MCTS, we also studied the impact of c_{puct} by testing values 1.0, 3.0, 5.0 and 8.0 along with the different temperature values mentioned. We only report the results for parameters yielding the best accuracy score in the main paper but every results can be found in Appendix A.2. The repetition penalty has been set to 1.2 as defined in CTRL. The number of MCTS iteration per token is set to 50, as well as the number of propositions for re-ranking, except for beam sampling where it is set to 10 because of memory limitations. Given the cost of roll-out for long sequences, we apply roll-out only on the emotion dataset to be able to run extensive experiments. Without roll-out, MCTS loses a part of its long view property but still allows to skew the exploration toward promising solutions. A study of the impact of the roll-out is detailed in a next sub-section. Parameters used for literature models are those provided by the authors. Experiments were conducted on a Quadro RTX 6000 with 80 GB of RAM.

4.5 Results

Results on the emotion, CLS and amazon_polarity datasets are reported in Table 1. The statistical significance against GeDi and PPLM is measured with a t-test with significance level (p-value) of 1%. Re-

sults show that PPL-MCTS is competitive against task-specifically trained LMs on the constraint application aspect (high accuracy), while keeping a fair amount of diversity (low Self-BLEU) and staying close to the original distribution (low oracle perplexity). On all three datasets and metrics, it constantly yields top results; the only other method which is high-performing for all metrics and constant across the datasets is GeDi trained with the classification loss.

Another remarkable result is for the Sampling - Argmax method that selects among a pool of propositions generated using sampling, the one with the highest probability to be from the correct class. Thanks to the sampling used for generating propositions, its Self-BLEU is among the lowest of all reported values. Despite the simplicity and low computational cost of this approach, its accuracy is among the best on every dataset. These very good results should however be put into perspective of the very high perplexity of its generated texts. This indicates that the generated samples may be very different than those generated by a standard LM on this dataset. Hence, exploring accuracy/perplexity trade-offs achievable with different values of α is interesting, which is proposed in Appendix A.4.

4.6 Human evaluation

Since automatic metrics can be biased and may not faithfully represent the human judgement, we conduct a human evaluation to compare with the results obtained through oracles and confirm the results and the relevance of automatic metrics. Because of the annotation cost, we limit the tested methods to the two state-of-the-art methods (PPLM and GeDi), PPL-MCTS and the promising Sampling - Argmax. This allows to test if PPL-MCTS is indeed as efficient as GeDi and if both are better than original PPLM. Also, this should confirm that the high perplexity of the Sampling - Argmax method is due to generated texts being very different from the ones generated by other methods. The evaluation has been performed on the CLS dataset by three volunteering colleagues, French native speakers. They labeled the same pool of reviews in order to measure the inter-annotator agreement.

The pool consists of 50 reviews (25 positive and 25 negative ones) randomly sampled for each method, which results in 200 reviews in total. Annotators were asked to go through this (randomly shuffled) pool and to give two scores for each review:

1. **Polarity**. Rate from 1 to 5 how well the text corresponds to the desired label (positive or negative). The text is rated 5 if it corresponds entirely to the expected label, down to 1 if it corresponds entirely to the opposite label. This score corresponds to the accuracy from the automatic metrics.

2. **Readability**. Rate from 1 to 5 how well the text is written. 5 corresponds to a text without any mistake and which is perfectly understandable. The more mistakes or incoherence, the lower the score. This score corresponds to the perplexity from the automatic metrics.

The diversity within the pool of generated texts is complicated to evaluate and the Self-BLEU is fairly accurate as a diversity metric, so this property is not studied in the human evaluation.

We report scores averaged over the 3 annotators as well as the standard deviation in Table 2. A t-test against PPLM (GeDi being best on both scores) is applied to test statistical significance (with p-value=0.01). One can notice that the agreement between annotators is high and that the results are in line with conclusions from automatic metrics. GeDi, when trained with the classification loss, yields similar results as PPL-MCTS, in terms of constraint satisfaction and quality of writing. PPLM, on the other hand, generates samples of lower quality and has more difficulty for applying the constraint. Finally, given its readability score, Sampling - Argmax seems to generate samples with a low quality. Its polarity score, while being higher than PPLM, is lower than expected: given the accuracy reported by the oracle, it should be close to GeDi and PPL-MCTS. It is most likely due to the fact that evaluating the polarity of a badly written text is hard for an human, often resulting in review being scored as neutral.

4.7 Effect of the roll-out

Rolling out is costly for very long sequences, and the question of its usefulness necessarily arises. We study how rolling out for only a fixed number of tokens (instead of until the end of the sequence) influences the performance of PPL-MCTS. For this experiment, we use the CLS dataset and set the roll-out to 0 (original result), 3, 5, 10 and 20 tokens. As one can note in Fig. 4, only 5 tokens allows PPL-MCTS to be on par with GeDi on this dataset.

Generation method	amazon_polarity Accuracy ↑	amazon_polarity 5 - Self-BLEU ↓	amazon_polarity Oracle perplexity ↓	emotion Accuracy ↑	emotion 5 - Self-BLEU ↓	emotion Oracle perplexity ↓	CLS Accuracy ↑	CLS 5 - Self-BLEU ↓	CLS Oracle perplexity ↓
Tuned LM									
CC-LM - Classloss	0.82	0.79	**2.56***,†	**0.89***	0.65†	3.72*,†	0.89*	**0.04***,†	50.6
CC-LM	0.91	0.71	3.21†	0.52	**0.13***,†	11.1	0.66	0.06*,†	31.5
GeDi - Classloss	0.96*	0.6*	5.16	0.88*	0.68	5.57*	0.94*	0.4	7.99*
GeDi	0.96*	0.6*	5.16	0.54	0.52†	4.09*,†	0.83*	0.31†	11.9
Untuned LM									
PPLM	0.89	0.66	2.84†	0.67	0.19†	7.31	0.79	0.23†	8.3
Beam - Argmax	0.88	0.85	3.14†	0.72*	0.49†	3.7*,†	0.64	0.82	3.31*,†
Beam - Sampling	0.86	0.84	3.27†	0.7	0.46†	3.69*,†	0.6	0.82	3.37*,†
Beam - First true	0.85	0.83	3.27†	0.65	0.38†	**3.68***,†	0.62	0.82	**3.26***,†
Beam sampling - Argmax	0.97*	0.73	3.82†	0.67	0.48†	3.88*,†	0.88*	0.67	3.91*,†
Beam sampling - Sampling	0.92	0.76	3.68†	0.66	0.48†	3.88*,†	0.76	0.63	4.07*,†
Beam sampling - First true	0.9	0.73	3.84†	0.66	0.49†	3.85*,†	0.85*	0.71	3.8*,†
Sampling - Argmax	**0.99***,†	0.17*,†	16.5	0.87*	**0.13***,†	11.7	0.92*	0.12*,†	14.3
Sampling - First true	0.89	**0.07***,†	85.9	0.82*	**0.13***,†	10.4	0.87*	0.14*,†	13
Sampling - Sampling	0.88	0.17*,†	16.3	0.81*	**0.13***,†	10.4	0.81	0.06*,†	31.8
PPL-MCTS	0.97*	0.63*	5.69	0.84*	0.37†	4.82*,†	0.89*	0.54	4.98*,†
PPL-MCTS - 10 tokens roll-out							**0.95***	0.57	5.07*,†

Table 1: Performance of constrained generation methods; from left to right: amazon_polarity, emotion, CLS datasets. † (resp. $*$) indicates statistically significant improvement against GeDi-classloss (resp. PPLM).

Generation method	Polarity	Readability
GeDi - Classloss	$4,46 \pm 0,08^{*}$	$4,19 \pm 0,28^{*}$
PPL-MCTS	$4,43 \pm 0,12^{*}$	$4,05 \pm 0,23^{*}$
PPLM	$3,74 \pm 0,08$	$3,12 \pm 0,19$
Sampling - Argmax	$4,00 \pm 0,11$	$2,83 \pm 0,33$

Table 2: Results of the human evaluation on the CLS dataset (averaged over 3 annotators). $*$ indicates statistically significant ($p \leq 1\%$) improvement against PPLM.

The roll-out size quickly improves accuracy, which then reaches a plateau. It suggests that having an horizon is really helpful but only up to a certain point. Beside, Self-BLEU and oracle perplexity values stay stable, varying respectively from 0.54 to 0.57, and from 4.98 to 5.18 as the roll-out size increases from 0 to 20.The roll-out size can thus be set accordingly to the compute budget, further defining the trade-off between cost and quality.

5 Conclusion

In this paper, we show that it is possible to control generation with the help of a discriminator that implements some expected constraints on the text during decoding. This flexible approach is very useful when using very large language models, such as GPT-3, whose fine-tuning computational costs are prohibitive. In contrast, training a discriminator is easier and cheaper. Our proposed methods, that mix the discriminator constraint and the generation, yield performance that is equivalent to the best approaches based on LM tuning at lower training cost. On the other hand, such approaches have an additional cost during inference because of the cost of the discriminator being applied to candidate generations. A study on this additional cost depending on the type of discriminator used can be found in (Chaffin et al., 2022). PPL-MCTS offers a solution for cases where training is too costly for the downstream application or the language model is not directly accessible. Seeing text generation as a tree exploration process, an existing approach such as GeDi indeed lowers the cost of width exploration but the depth exploration is still an issue. Using GeDi for constrained generation is thus very similar to a standard maximum likelihood search which still lacks of an optimal search method. On the other hand, Monte Carlo Tree Search provides an efficient way to explore the tree by determining the best local choice in the long run, lowering the cost of depth exploration. Thus, these two methods solve different facets of constrained generation, and the combination of the two is a promising perspective. Moreover, MCTS allows to precisely define the best compromise between cost and quality through the number of iterations and the roll-out size, while ensuring the efficiency of the search theoretically. For reproducibility purposes, our implementation is made available at `https://github.com/NohTow/PPL-MCTS`.

Several research avenues are opened by this work. For methods yielding high perplexity, it would be interesting to explore how to set the α parameter in order to reach the best compromise between accuracy and perplexity. Similarly, the size (number of tokens considered) of the roll-out in MCTS offers some ways to control the cost/performance compromise. An adaptive roll-out size, for example rolling-out until the score of

the discriminator is above or below a threshold as in (Cotarelo et al., 2021), would seem particularly suited for texts. It should also be noted that fine-tuning a model and controlling the generation with a discriminator can be used jointly. For instance, one can use PPL-MCTS on a tuned LM, which will most likely result in even better performances because sequences considered during the search will have an overall higher quality for the considered task. Finally, not only can PPL-MCTS be applied to any property that a discriminator can identify, but it can also work using other scoring methods (human evaluation, regular expressions, heuristic based evaluation, ...) as long as the score reflects compliance with the expected property.

6 Ethics/Broader impact

The ethical risks of large LMs are well known (Bender et al., 2021). Especially when they are trained on large quantities of non curated data, it has be shown that they tend to reproduce or amplifies biases on gender, race, etc. and more generally may produce inappropriate content (Gehman et al., 2020). As for every automatic generation method, using our approaches may result in the production of unwanted, misleading or inappropriate content. Yet, it is noteworthy that the constrained generation as we propose is one way to control, a posteriori of the LM training, that the generated texts respect some criteria. It can be used for any application given that a discriminator is able to check the constraint accurately. The ethical interests are thus important, such as adding constraint about race diversity, gender equality, non toxicity, factual faithfulness, etc. as far as these properties can be detected by a (trained or hand-crafted) discriminator. But of course, the same technique could be used for malicious purposes, such as constraining generation so it produces offensive texts, targeted fake news, etc. In such cases of misuse, discriminators similar to those used for constraining the generation could easily spot such texts since the constraint will, by design, be noticeable and easily grasped by a discriminator.

Even though training language models on curated data in the first place is possible, totally curated dataset is hard to obtain, and new biases may be highlighted. Indeed, defining a priori what is every possible bias in every cultural context for every possible application, and curating the training data accordingly is hardly feasible. Hence, constant updates of language models will be necessary to make them as fair as possible. Given the cost of large language models training, updating them often is really harmful for the environment. Discriminator guided constrained generation offers a way to filter text generation using up-to-date standards in terms of fairness by only updating the discriminator, which is faster and require way less resources.

References

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2020. Language gans falling short. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Antoine Chaffin, Thomas Scialom, Sylvain Lamprier, Jacopo Staiano, Benjamin Piwowarski, Ewa Kijak, and Vincent Claveau. 2022. Which discriminator for cooperative text generation? In *The 45st International ACM SIGIR Conference on Research & Development in Information Retrieval, (SIGIR '22), July 11–15, 2022, Madrid, Spain*. ACM.

Alba Cotarelo, Vicente García Díaz, Edward Núñez Valdez, Cristian González García, Alberto Gómez, and Jerry Lin. 2021. Improving monte carlo tree search with artificial neural networks without heuristics. *Applied Sciences*, 11:2056.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. 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.

Carnegie-Mellon University.Computer Science Dept. 2018. Speech understanding systems: summary of results of the five-year research effort at carnegie-mellon university.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. 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.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1638–1649. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. Gedi: Generative discriminator guided sequence generation. *CoRR*, abs/2009.06367.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *CoRR*, abs/2003.02245.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2020. Flaubert: Unsupervised language model pre-training for french. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 2479–2490. European Language Resources Association.

Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislar, Jean-Baptiste Lespiau, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. 2021. Machine translation decoding beyond beam search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8410–8434. Association for Computational Linguistics.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2241–2252. Association for Computational Linguistics.

Yannis Papanikolaou and Andrea Pierleoni. 2020. DARE: data augmented relation extraction with GPT-2. *CoRR*, abs/2004.13845.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Christopher D. Rosin. 2011. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.*, 61(3):203–230.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3687–3697. Association for Computational Linguistics.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Discriminative adversarial search for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8555–8564. PMLR.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2021.

To beam or not to beam: That is a question of cooperation for language gans. *Advances in neural information processing systems*.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9051–9062.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM.

A Appendix

In this technical appendix, we provide additional information about our methods, some settings and the experiments. Further experimental results, as well as examples, are given and discussed. Finally, a discussion on concurrent studies is provided.

A.1 Data splits

We adapted the way we split the dataset into two parts and train/test/validation sets depending on the original dataset splits. Amazon_polarity is composed of a training set of 3 600 000 examples and a test set of 400 000. We split both into two parts and kept 20% of each training set for validation. Emotion already comes with train, test and validation set, hence we just split each into two parts. Finally, CLS is composed of a train set and a test set of 6000 examples. We split the training set in two and split the test set twice so we got two validation and test sets. Thus, for each dataset, we end up with two training sets, two validation sets and two test sets.

The first train and validation sets are used to train and control the training of models used for the generation: the guiding classifier, the "vanilla" LM and the CC-LM. The test set serves to control their performance.

The second ones are used to train the LM oracle and the classifier used to measure the accuracy. The test set allows to verify that these models are trustworthy for accurate evaluation. Once all the models are trained, the constrained generation is evaluated on 900 samples generated from prompts never seen by models during training.

A.2 Complementary results

We tested three temperature values for each proposed method: 1.0, 1.1 and 1.2. As the temperature increases, the output distribution of the language model becomes more and more uniform. This means that high temperatures should result in high perplexities because the sampling deviates further from the original distribution.

For PPL-MCTS, we also studied the impact of c_{puct} by testing values 1.0, 3.0, 5.0 and 8.0 along with the different temperature values mentioned. c_{puct} defines the compromise between exploiting nodes that already have great scores and exploring less played but promising ones. A high c_{puct} encourages exploration. We remind that the repetition penalty I in Eqn. 2 has been set to 1.2 as defined in CTRL.

In Section 'Results', for each method and dataset, we reported only the results obtained with the set of parameter values yielding the best accurracy. Hereafter, we report results with every tested set of parameters in Tables 3, 4 and 5 for respectively the emotion, CLS and amazon_polarity datasets.

Unsurprisingly, the perplexity of methods which sample on the LM logits explodes when τ increases, without a noticeable gain in accuracy. Since the diversity is already high for low τ values, it seems to be better to keep the temperature low with these approaches. Note that the couple $c_{puct} = 3, \tau = 1.0$ for PPL-MCTS always leads to the best result. Using $c_{puct} = 8$ seems to yield slightly worse results, especially with a low temperature. However, the different parameters do not greatly affect the results of PPL-MCTS.

A.3 Examples of generation

We provide an example of generation for amazon_polarity and emotion datasets using PPL-MCTS, PPLM, GeDi and Sampling - Argmax methods, respectively in Figures 5 and 6. Texts generated using Sampling - Argmax are rather different as suggested by the reported high perplexity results. Note that emotion texts are only one sentence while those of amazon_polarity are complete reviews. This difference motivated the choice of these datasets. Also, we preferred amazon_polarity over IMDb used in the GeDi and PPLM papers because of its bigger size, suitable format and because a French equivalent is available (CLS), which allows us to test another language with a similar dataset.

A.4 Constraint strength through α

As described in the model improvements section, a parameter α can be defined to control the relative importance of the discriminator score and the language model likelihood. Thus, this parameter allows to control the constraint application strength as it helps to define a trade-off between staying close the original LM and satisfying the constraint. Note that in all of our experiments reported earlier, this parameter has been set to 1, focusing on the constraint application since the proposed methods already inherently provide legible texts.

Here, as a proof of concept, we test a range of values for α, using the Sampling - Argmax method on the amazon_polarity dataset with the automatic

PPL-MCTS
<|startoftext|> realistic account [SEP] ive read this book and i think it is a very good account of the life of a young man. i would recommend to anyone who likes to know about the lives of people in america, or any other country that has been around for centuries.<|endoftext|>

PPLM
<|startoftext|> realistic account [SEP] I enjoyed this book. It was realistic and I enjoyed the way the author described the people and places. I would recommend this book to anyone who is interested in the history of the United States.<|endoftext|>

GeDi
<|startoftext|> realistic account [SEP] This is a real book. It's not just about the war, but also about the people who fought it. The author has a very realistic view of what happened in Vietnam and how it affected our country. I would recommend this book to anyone interested in learning more about Vietnam or any other war that took place during that time period<|endoftext|>

Sampling - Argmax
<|startoftext|> realistic account [SEP] ive seen many of these. they mostly in the 20's, 30's and 40's and the luscious ones i know - men like this - there's an old saying that farts, u makin' u sell it..this movie has a lot of realism to it too! and i was totally impressed on how good the kids and the predator was! will it be hard for them to make more like this? i think it will! i read that war is going to be much<|endoftext|>

Figure 5: Examples of constrained generation using PPL-MCTS, PPLM, GeDi and Sampling - Argmax methods (from top to bottom) on the positive class of amazon_polarity, using the same prompt (in bold).

PPL-MCTS
<|startoftext|> i feel that working with a group of people who are so passionate about the same thing is really important<|endoftext|>

PPLM
<|startoftext|> i feel that working hard and caring for someone i don t care for is a lot less selfish than i would be feeling for someone i<|endoftext|>

GeDi
<|startoftext|> i feel that working with the ladies of the family is a wonderful thing and i am very fond of the way they look and feel<|endoftext|>

Sampling - Argmax
<|startoftext|> i feel that working at imgur for so many years is ill be devoted to it<|endoftext|>

Figure 6: Examples of constrained generation using PPL-MCTS, PPLM, GeDi and Sampling - Argmax methods (from top to bottom) on the 'love' class form 'emotion', using the same prompt (in bold).

metrics. We chose this method and dataset since it yields the best accuracy, but also exhibits a very high perplexity. In this case, it seems interesting to trade a bit of accuracy for better written texts.

Results are roughly constant when α is lower than 0.98, so it has an impact only for values between 0.98 and 1. This is due to the fact that, for a long enough sequence, $p_\theta(x)$ is often relatively small compared to $p_D(c \mid x)$. This difference of scale annihilates the influence of α. This [0.98-1] interval thus corresponds to values of α that rescale $p_D(c \mid x)^\alpha$ and $p_\theta(x)^{1-\alpha}$ on a same order of magnitude. As shown in Figure 7, within this regime, we can observe a linear dependency between α and the accuracy as well as the perplexity. This illustrate that a trade-off can be obtained by tuning this parameter, allowing to define the strength of the constraint application which also defines how far the generation can be from the original LM distribution.

Generation method	Accuracy ↑	5 - Self-Bleu ↓	Oracle perplexity ↓
Beam sampling - Argmax $\tau = 1.0$	0,61	0,41	3,7
Beam sampling - Argmax $\tau = 1.1$	0,65	0,48	3,72
Beam sampling - Argmax $\tau = 1.2$	*0,67*	*0,48*	*3,88*
Beam sampling - First true $\tau = 1.0$	0,58	0,4	3,68
Beam sampling - First true $\tau = 1.1$	0,64	0,48	3,69
Beam sampling - First true $\tau = 1.2$	*0,66*	*0,49*	*3,85*
Beam sampling - Sampling $\tau = 1.0$	0,59	0,41	3,69
Beam sampling - Sampling $\tau = 1.1$	0,64	0,49	3,69
Beam sampling - Sampling $\tau = 1.2$	*0,66*	*0,48*	*3,88*
CC-LM - Greedy Search	0,51	0,1	17
CC-LM - Sampling $\tau = 1.0$	*0,52*	*0,13*	*11,1*
CC-LM - Sampling $\tau = 1.1$	0,51	0,1	15,8
CC-LM - Sampling $\tau = 1.2$	0,47	0,08	31,4
CC-LM - Classloss - Greedy Search	*0,89*	*0,65*	*3,72*
CC-LM - Classloss - Sampling $\tau = 1.0$	0,83	0,11	19,6
CC-LM - Classloss - Sampling $\tau = 1.1$	0,79	0,07	33,2
CC-LM - Classloss - Sampling $\tau = 1.2$	0,79	0,05	64,8
Sampling - Argmax $\tau = 1.0$	*0,87*	*0,13*	*11,7*
Sampling - Argmax $\tau = 1.1$	0,86	0,1	19,6
Sampling - Argmax $\tau = 1.2$	0,86	0,07	47,5
Sampling - First true $\tau = 1.0$	*0,82*	*0,13*	*10,4*
Sampling - First true $\tau = 1.1$	0,81	0,11	16,2
Sampling - First true $\tau = 1.2$	0,77	0,09	33,2
Sampling - Sampling $\tau = 1.0$	*0,81*	*0,13*	*10,4*
Sampling - Sampling $\tau = 1.1$	0,8	0,11	15
Sampling - Sampling $\tau = 1.2$	0,79	0,08	25,7
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.0$	0,83	0,37	4,81
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.1$	0,8	0,36	4,9
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.2$	0,82	0,33	4,97
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.0$	*0,84*	*0,37*	*4,82*
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.1$	0,82	0,35	4,85
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.2$	0,84	0,35	4,9
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.0$	0,84	0,38	4,74
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.1$	0,84	0,34	4,79
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.2$	0,84	0,33	4,88
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.0$	0,81	0,38	4,71
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.1$	0,81	0,37	4,72
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.2$	0,82	0,35	4,79

Table 3: Results for every tested set of parameters on the proposed methods; emotion dataset. Results reported in the body of the paper are in italic.

A.5 Concurrent work

During the time of writing, two preprints using MCTS for NLP tasks have been released (Leblond

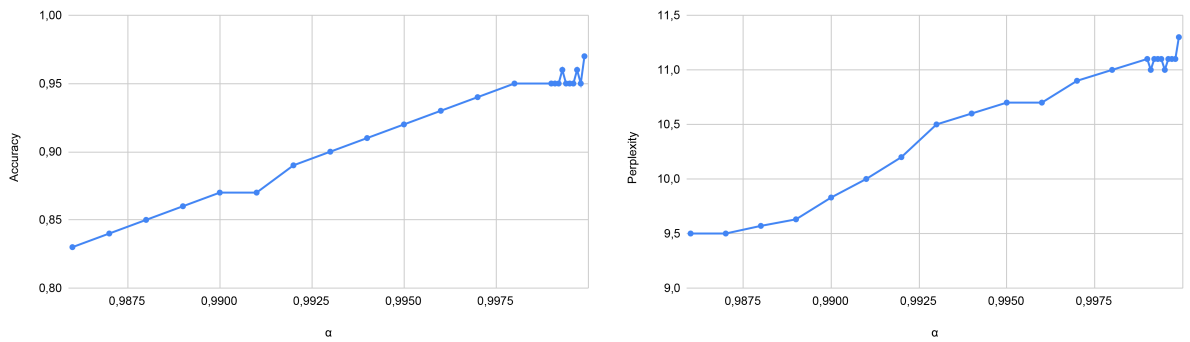

Figure 7: Accuracy (left) and perplexity (right) of the Sampling - Argmax method according to the α parameter; amazon_polarity dataset

Generation method	Accuracy ↑	5 - Self-Bleu ↓	Oracle perplexity ↓
Beam sampling - Argmax $\tau = 1.0$	0,87	0,71	3,85
Beam sampling - Argmax $\tau = 1.1$	*0,88*	*0,67*	*3,91*
Beam sampling - Argmax $\tau = 1.2$	0,88	0,63	4,12
Beam sampling - First true $\tau = 1.0$	*0,85*	*0,71*	*3,8*
Beam sampling - First true $\tau = 1.1$	0,84	0,68	3,87
Beam sampling - First true $\tau = 1.2$	0,85	0,63	4,07
Beam sampling - Sampling $\tau = 1.0$	0,74	0,71	3,82
Beam sampling - Sampling $\tau = 1.1$	0,72	0,68	3,89
Beam sampling - Sampling $\tau = 1.2$	*0,76*	*0,63*	*4,07*
CC-LM - Greedy Search	0,59	0,57	2,51
CC-LM - Sampling $\tau = 1.0$	0,62	0,15	12,3
CC-LM - Sampling $\tau = 1.1$	0,63	0,09	18,7
CC-LM - Sampling $\tau = 1.2$	*0,66*	*0,06*	*31,5*
CC-LM - Classloss - Greedy Search	0,8	0,59	2,77
CC-LM - Classloss - Sampling $\tau = 1.0$	0,85	0,13	17
CC-LM - Classloss - Sampling $\tau = 1.1$	0,87	0,07	28
CC-LM - Classloss - Sampling $\tau = 1.2$	*0,89*	*0,04*	*50,6*
Sampling - Argmax $\tau = 1.0$	*0,92*	*0,12*	*14,3*
Sampling - Argmax $\tau = 1.1$	0,92	0,08	20,7
Sampling - Argmax $\tau = 1.2$	0,92	0,05	33,6
Sampling - First true $\tau = 1.0$	*0,87*	*0,14*	*13*
Sampling - First true $\tau = 1.1$	0,86	0,1	19,1
Sampling - First true $\tau = 1.2$	0,86	0,06	33,1
Sampling - Sampling $\tau = 1.0$	0,77	0,14	12,9
Sampling - Sampling $\tau = 1.1$	0,78	0,09	18,8
Sampling - Sampling $\tau = 1.2$	*0,81*	*0,06*	*31,8*
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.0$	0,88	0,54	4,98
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.1$	0,87	0,53	5
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.2$	0,87	0,53	5,02
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.0$	*0,89*	*0,54*	*4,98*
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.1$	0,89	0,54	4,81
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.2$	0,89	0,54	4,86
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.0$	0,88	0,55	4,9
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.1$	0,89	0,54	4,97
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.2$	0,89	0,54	4,91
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.0$	0,83	0,54	4,98
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.1$	0,86	0,54	4,95
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.2$	0,88	0,55	4,94

Table 4: Results for every tested set of parameters on the proposed methods; CLS dataset. Results reported in the body of the paper are in italic.

Generation method	Accuracy ↑	5 - Self-Bleu ↓	Oracle perplexity ↓
Beam sampling - Argmax $\tau = 1.0$	0,94	0,79	3,55
Beam sampling - Argmax $\tau = 1.1$	0,96	0,77	3,65
Beam sampling - Argmax $\tau = 1.2$	*0,97*	*0,73*	*3,82*
Beam sampling - First true $\tau = 1.0$	0,86	0,77	3,73
Beam sampling - First true $\tau = 1.1$	0,89	0,77	3,68
Beam sampling - First true $\tau = 1.2$	*0,9*	*0,73*	*3,84*
Beam sampling - Sampling $\tau = 1.0$	0,87	0,77	3,7
Beam sampling - Sampling $\tau = 1.1$	*0,92*	*0,76*	*3,68*
Beam sampling - Sampling $\tau = 1.2$	0,89	0,73	3,83
CC-LM - Greedy Search	*0,91*	*0,71*	*3,21*
CC-LM - Sampling $\tau = 1.0$	0,87	0,17	15,7
CC-LM - Sampling $\tau = 1.1$	0,86	0,1	32,2
CC-LM - Sampling $\tau = 1.2$	0,8	0,08	80,2
CC-LM - Classloss - Greedy Search	*0,82*	*0,79*	*2,56*
CC-LM - Classloss - Sampling $\tau = 1.0$	0,81	0,16	18,4
CC-LM - Classloss - Sampling $\tau = 1.1$	0,79	0,1	37,1
CC-LM - Classloss - Sampling $\tau = 1.2$	0,74	0,07	95,4
Sampling - Argmax $\tau = 1.0$	*0,99*	*0,17*	*16,5*
Sampling - Argmax $\tau = 1.1$	0,99	0,11	31,8
Sampling - Argmax $\tau = 1.2$	0,99	0,07	84,50
Sampling - First true $\tau = 1.0$	0,88	0,16	16,4
Sampling - First true $\tau = 1.1$	0,87	0,1	31,5
Sampling - First true $\tau = 1.2$	*0,89*	*0,07*	*85,9*
Sampling - Sampling $\tau = 1.0$	*0,88*	*0,17*	*16,3*
Sampling - Sampling $\tau = 1.1$	0,87	0,1	30,8
Sampling - Sampling $\tau = 1.2$	0,88	0,07	81
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.0$	0,96	0,62	5,61
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.1$	0,96	0,63	5,65
PPL-MCTS - $c_{puct} = 1.0, \tau = 1.2$	0,96	0,62	5,66
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.0$	*0,97*	*0,63*	*5,69*
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.1$	0,97	0,62	5,77
PPL-MCTS - $c_{puct} = 3.0, \tau = 1.2$	0,96	0,62	5,72
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.0$	0,95	0,63	5,6
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.1$	0,96	0,63	5,66
PPL-MCTS - $c_{puct} = 5.0, \tau = 1.2$	0,96	0,63	5,63
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.0$	0,93	0,64	5,57
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.1$	0,93	0,64	5,57
PPL-MCTS - $c_{puct} = 8.0, \tau = 1.2$	0,95	0,63	5,57

Table 5: Results for every tested set of parameters on the proposed methods; amazon_polarity dataset. Results reported in the body of the paper are in italic.

et al., 2021; Scialom et al., 2021). While we emphasize that these are concurrent studies, PPL-MCTS has some major differences. Indeed, these studies focus on improving the overall quality of generated texts rather than following a given constraint. While "being well written" can be seen as a constraint, PPL-MCTS rather explores how a constraint that is not present in the original language model (i.e. not a goal in the original training of the LM) can be added at generation time. Scialom et al. (2021) train a discriminator to distinguish generated and real samples because their goal is ultimately to train the language model in a Generative Adversarial setup to create a better LM. This iterative training, in addition to not being possible in our task, is not wanted since we aim to be plug and play. Our goal is indeed to apply an additional constraint to an untouched original language model. Yet, even if goals are different and applying MCTS for constrained generation is not trivial, the "MLE

Coop-MCTS" is close to PPL-MCTS. However, focusing on MCTS as a decoding only strategy allowed an in-depth study that provided interesting results, in particular the effect of the roll-out size (the roll-out is totally omitted in their paper) and the α parameter.

On the other hand, Leblond et al. (2021) also focus on MCTS as a decoding strategy but for the very specific case of machine translation. MCTS is used to optimize metrics for machine translation, which are known to not necessarily correlate with human judgement (Novikova et al., 2017). Again, the goal is different since these metrics are used as a proxy of the sample quality. In contrast, our work shows that MCTS can be used to optimize a given property, but instead of optimizing the quality of samples, we optimize for a given constraint while retaining the original quality of writing. The fact that MCTS also works in such cases was non trivial since adding such constraints to the generation could lead to deteriorate texts.

Beside MCTS, we also proposed and explored simpler methods based on re-ranking for our task and showed that diversity allows to satisfy the constraint, often at the price of a lower quality, emphasizing the compromise between exploration and exploitation made by the MCTS.

Finally, these concurrent studies provide evidences that MCTS is promising for many different usage in NLP. We hope that the large amount of experiments, parameter analysis and the availability of our open-sourced code working out-of-the-box will help foster future research in this direction.