Gaussian Process Optimization for Adaptable Multi-Objective Text Generation using Linearly-Weighted Language Models

Mohammad Mahdi Abdollah Pour*1, Ali Pesaranghader*2, Eldan Cohen1 and Scott Sanner1
1 University of Toronto, Canada
m.abdollahpour@mail.utoronto.ca, {ecohen,ssanner}@mie.utoronto.ca
2 LG Electronics, Toronto AI Lab
ali.pesaranghader@lge.com

Abstract

In multi-objective text generation, we aim to optimize over multiple weighted aspects (e.g., toxicity, semantic preservation, fluency) of the generated text. However, multi-objective weighting schemes may change dynamically in practice according to deployment requirements, evolving business needs, personalization requirements on edge devices, or the availability of new language models and/or objective requirements. Ideally, we need an efficient method to adapt to the dynamic requirements of the overall objective. To address these requirements, we propose a linear combination of objective-specific language models to efficiently adapt the decoding process and optimize for the desired objective without the significant computational overhead of retraining one or more language models. We show empirically that we can leverage Gaussian Process black box optimization to adapt the language model decoder weights to outperform other fixed weighting schemes and standard baselines of the task in only a few iterations of decoding. Overall this approach enables highly efficient adaptation of controllable language models via multi-objective weighting schemes that may evolve dynamically in practical deployment situations.

1 Introduction

Multi-objective text generation involves compromises between different objectives. In practice, the importance of each objective may dynamically change due to business needs, personalization, or addition of new objectives due to time-evolving deployment requirements. Retraining or fine-tuning the Language Model (LM) may be impractical for each adaptation of the multi-objective target since it imposes significant computational costs. To address this inefficiency, we propose a multi-objective framework that leverages language model decoders pretrained for each objective and a dynamic weighting of each decoder to adapt to the objective without retraining their corresponding models.

More specifically, we propose a method to dynamically adapt the weighting of objective-specific LMs at the decoding stage to optimize the desired overall text generation objective. We define the overall problem as one of black box function optimization, where the function inputs are n language model decoders and weights (i.e., $w_1, \ldots, w_n$) and the output is the chosen objective value. We specifically use Gaussian Process optimization since it is a popular and efficient tool for black box optimization (Brochu et al., 2010; Snoek et al., 2012).

Empirically, we evaluate on a range of text detoxification tasks that serve as a natural and important testbed for multi-objective language model optimization. We demonstrate that our Gaussian Process Bayesian Optimization approach can efficiently and quickly adapt the language model decoder weights to outperform other fixed weighting schemes and standard baselines of the task in only a few iterations of decoding.

2 Related Work

2.1 Text Detoxification as a Natural Testbed for Multi-objective Text Generation

The text detoxification task aims to generate a non-toxic sentence $s^{out}$ given a toxic input $s^{in}$ while preserving the content of $s^{in}$. This is inherently a multi-objective text generation task as we need to ensure non-toxicity, semantic preservation, and fluency (Logacheva et al., 2022; Pour et al., 2023).

Text detoxification solutions primarily fall into two main categories, unsupervised and supervised. The unsupervised methods are typically built on a non-parallel dataset, which is a set of toxic and a set of non-toxic texts without one-to-one mappings between them (Wu et al., 2019; Li et al., 2018; Dale et al., 2021; Lee, 2020; He et al., 2020; Luo et al., 2021).
2019). In contrast, supervised methods are usually built on parallel datasets in which one-to-one mappings between toxic and non-toxic texts exist and train end-to-end models to generate non-toxic text given the toxic input (Logacheva et al., 2022; Atwell et al., 2022; Floto et al., 2023; Pour et al., 2023). Supervised methods have typically shown superiority to unsupervised methods (Logacheva et al., 2022; Floto et al., 2023; Pour et al., 2023).

### 2.2 Black Box Bayesian Optimization

The objective in black box function optimization is to identify the optimal parameters for a “black box” function characterized by an unknown or a very complex mathematical form or structure (Jones et al., 1998; Bergstra and Bengio, 2012). Bayesian Optimization (BO) is a commonly used solution in optimizing black box functions that employs a probabilistic surrogate model to represent the unknown function (Snoek et al., 2012; Brochu et al., 2010). It iteratively selects the most promising parameter sets via an acquisition function for evaluation by the objective function. Subsequently, the surrogate model is updated based on these evaluations, persisting until convergence or the fulfillment of predetermined stopping conditions. Gaussian Processes (GPs) are a popular choice in Bayesian Optimization for optimizing black box functions (Srinivas et al., 2010) due to their adaptability to uncertainty modelling and efficient handling of small data regimes. This makes them well-suited for applications such as Automated Machine Learning (AutoML) (Snoek et al., 2012), Drug Discovery, and Bioinformatics (Colliandre and Muller, 2023).

### 2.3 Minimum Bayes Risk Training and Decoding

Bayesian approaches to both language model training and decoding have been considered previously, but in a different setting than ours. Minimum Bayes-Risk (MBR) training (Wang et al., 2018; Shen et al., 2015) trains model parameters with respect to target evaluation metrics. To this end, it is akin to the type of heavyweight full fine-tuning approach that we aim to avoid in this paper in favor of a lightweight adaptation of multiple decoder weights via Gaussian Process Bayesian Optimization. Similarly, MBR decoding (Kumar and Byrne, 2004; Blain et al., 2017) aims to find Bayes optimal sequences at the decoding stage, but does not consider the case of reweighting multiple decoders that is the focus of our work.

In the next section, we define our methodology for black box optimization for adapting to multi-objective text generation settings.

### 3 Multi-Objective Text Decoding

#### Problem Definition

For multi-objective text generation, we assume that we have different pre-trained and fixed language models representing distinct objectives. For example, we might fine-tune a base language model for non-toxic text generation and separately fine-tune the same model for fluent text generation to provide one decoder for each objective.

Our goal is to devise an efficient weighting strategy that combines the next-token prediction scores from all language models, without fine-tuning them, to optimize the overall objective. It is challenging to manually determine a set of weights that effectively combines these language models. To tackle this challenge, we frame the problem as a black box function optimization as shown in Fig. 1. The figure shows that our inputs consist of $n$ language models, each associated with a weight (denoted as $w_1$ to $w_n$), and the output corresponds to the selected objective value.
To optimize the black box function, we leverage Bayesian Optimization with Gaussian Processes. We describe our solution in detail below.

**Methodology.** Suppose \( s^{in} \) is the input text and \( s^{out} \) is the generated text that we want to evaluate. For that, assume that we have \( n \) objective functions, i.e., \( \mathcal{O} = \{ o_1(\cdot), ..., o_n(\cdot) \} \), that reflect different properties of text such as non-toxicity or fluency, and \( n \) language models that correspond to the foregoing objectives, i.e., \( \mathcal{M} = \{ m_1(\cdot), ..., m_n(\cdot) \} \).

That is, \( m_i(\cdot) \) is a language trained to maximize the objective \( o_i(\cdot) \), for any \( i \leq n \). To represent our preferences over the objectives \( \mathcal{O} \), we use a set of thresholds, i.e., \( \mathcal{T} = \{ t_1, ..., t_n \} \).

**Overall Objective:** We want to generate sequences that satisfy our preferences \( \mathcal{T} \) over the objectives \( \mathcal{O} \) as follows:

\[
o_{\text{pref}}(s^{out}) = \frac{1}{|\mathcal{O}|} \sum_{(o_i,t_i)\in\mathcal{O}\times\mathcal{T}} I[o_i(s^{out}) \geq t_i]
\]

where \( I[\cdot] \) is the indicator function. It is noteworthy that Eq. 1 is a considered as a generalized version of the J score from Krishna et al. (2020).

**Decoding:** To satisfy \( o_{\text{pref}}(\cdot) \), we need to combine the models in \( \mathcal{M} \) using a set of weights, i.e., \( w = [w_1, ..., w_n] \), in the decoding process as presented in Fig. 1. The combined language model, denoted by \( \hat{m}(\cdot|\mathcal{M}, w) \), chooses the next token \( s^\text{out}_j \) by a linear combination of next token probabilities of models in \( \mathcal{M} \):

\[
p_{\hat{m}}(s^\text{out}_j|s^{\text{out}}_{\leq j}, s^{\text{in}}, \mathcal{M}, w) = \sum_{(m(\cdot),w_i)\in\mathcal{M}\times\mathcal{W}} w_i \cdot p_m(s^\text{out}_j|s^{\text{out}}_{\leq j}, s^{\text{in}})
\]

where \( p_m(s^\text{out}_j|s^{\text{out}}_{\leq j}, s^{\text{in}}) \) is probability of the \( j \)-th token \( s^\text{out}_j \) using the text generation model \( m(\cdot) \). Then, the tokens are ranked based on their \( p_{\hat{m}} \) before being used by a decoding strategy such as beam search.

Finally, we use black-box optimization to learn the optimal weights, i.e., \( w^* \):

\[
s^{\text{out}} = \hat{m}(s^{\text{in}}|\mathcal{M}, w)
\]

\[
w^* = \underset{w}{\arg \max} \sum_{s^{\text{out}}} o_{\text{pref}}(s^{\text{out}})
\]

To obtain \( w^* \), we use Bayesian Optimization with Gaussian Processes. We review Bayesian Optimization with Gaussian Processes in Appx. B.

**4 Experiments**

Recall that, we use the text detoxification task for our proposed method for multi-objective text generation. The detoxification task is commonly evaluated by three objectives of non-toxicity, semantic preservation, and fluency (Logacheva et al., 2022; Atwell et al., 2022; Pour et al., 2023; Floto et al., 2023). We discuss our experimental setup below and provide all code to reproduce results on Github.¹

**4.1 Experimental Setup**

**Datasets.** We use two parallel detoxification datasets, namely, ParaDetox (Logacheva et al., 2022) and APPDIA (Atwell et al., 2022) which contain pairs of toxic text and non-toxic texts. The datasets are split into training, validation, and test sets. We use the training set to train objective-specific language models (Appendix A). We also assess the generalizability of the LMs trained on ParaDetox or APPDIA for black box optimization against the Jigsaw dataset (Do, 2019). For that, we learn the optimal weights \( w^* \) using the Jigsaw validation set and evaluate the performance on its test set.

**Metrics.** Accuracy (STA), Content Preservation (SIM), and Fluency (FL) are commonly used in the literature (Logacheva et al., 2022; Pour et al., 2023; Floto et al., 2023) for text detoxification evaluation. STA and FL are computed using pre-trained classifiers (Logacheva et al., 2022). SIM is computed using cosine similarity between the input and the generated detoxified text with the model from Wieting et al. (2019).

**Baselines.** We compare the performance of our black box GP optimization method to the following baselines:

1. **Parallel Training** is the standard approach where an encoder-decoder language model is trained, on a parallel dataset, to generate a non-toxic text for an input toxic text which has the best performance in Logacheva et al. (2022).

2. **Fine-tuning:** By fine-tuning, the model is trained for the assigned objective \( o_{\text{pref}} \). This approach incurs a high computational cost and therefore is not well-suited for fast multi-objective adaptation. However, it is an important reference point for comparison.

¹[https://github.com/D3Mlab/gp-opt-lm](https://github.com/D3Mlab/gp-opt-lm)
3. Random: To test whether random search performs as well as GP-based search, we uniformly generated \( w \in [0, 1]^3 \) at each step of optimization and simply maintained the \( w^* \) as the best performing \( w \) up to the current step.

4. T-base: In this case, instead of finding \( w^* \) through black box optimization, we set \( w_i = t_i \) so that each \( w_i \) corresponds to the importance of objective \( \alpha_i(\cdot) \) in \( o_{pref} \) (Eq. 1).

We remark that both the GP and Random methods in Fig. 2 (a) have been averaged over 5 uniformly randomized initializations for \( w \in [0, 1]^3 \).

**Thresholds.** We consider 3 cases for \( T \) in Fig. 2 (a) to focus on one objective in each setting. For example, \( T: [0.1, 0.9, 0.1] \) de-emphasizes Accuracy and Similarity (0.1) but emphasizes Fluency (0.9).

4.2 Experimental Results

**Optimization & Generalization with GP.** In all experiments, we find the best combination weights \( w^* \) (in Eq. 4) using black box optimization against the validation data. Meanwhile, we plot the performance against the test data at each step of black box optimization.

**Experimental Setting I.** Fig. 2 (a) compares the results of black box optimization with the baselines against ParaDetox and APPDIA, in the first two rows, respectively. In most cases, we see that GP outperforms other methods. This can be explained by the fact that the black box optimizer finds the best performing \( w^* \) to fuse the contributions of our LMs to maximize the final objective. We also observe that GP’s performance improves significantly.
during early steps. This observation supports our claim regarding the efficiency of our method.

**Experimental Setting II.** Fig. 2 (a) also presents the generalization results against the Jigsaw dataset, in the last two rows. We see that GP again shows superiority to the other methods in most cases for both (reference) datasets. However, when a greater threshold is set to content preservation, Parallel Training usually performs better, suggesting its suitability for content presentation.

In both settings, GP may not perform as well as other models when a greater threshold is set to the content-preservation objective against the APP-DIA dataset. This may be reflected by the fact that content preservation is not a key objective for this dataset. Moreover, Fig. 2 (a) shows the superior performance of GP over random search emphasizing the importance of Bayesian optimization with GPs in finding the best weighting combination.

We observe in most cases that the Fine-tuning baseline does not generally perform well given the challenge of optimizing the nonlinear target with (non-differentiable) thresholded objective functions in Eq (1). Furthermore, Fine-tuning requires significant computation and does not permit fast adaptation to new multiobjective functions in only a few iterations of decoder weight optimization as we propose in this paper with our Gaussian Process Bayesian Optimization approach.

**GP Kernel Choice.** In Fig. 2 (b), we can see the results for different $\nu$ parameters of the Matérn kernel (Matern et al., 1960) and different length parameters $l$ for the RBF and the Inner (dot) product kernels. Observing consistent patterns across various kernels suggests the resilience of our methodology to kernel selection, alleviating the necessity for extensive hyperparameter tuning.

**5 Conclusion**

We introduced black box optimization for fast multi-objective adaptation of language models (LMs) by leveraging Gaussian Process Bayesian Optimization to efficiently adapt the weights of objective-specific decoders. Our experimental results showed that our GP approach was able to quickly adapt to changes in nonlinear, non-differentiable multi-objective targets in only a few decoding iterations as evidenced by its strong performance compared to a variety of baselines.

**Limitations**

Our experiments focused on text detoxification, which is an important case of multi-objective text generation that has received much attention in recent years (Logacheva et al., 2022; Atwell et al., 2022; Floto et al., 2023; Pour et al., 2023). However, our methodology is general and could be applied to a diverse set of multi-objective text generation tasks. Exploring the performance of our approach in other diverse settings is an important avenue for future research.

**Ethical Considerations**

**Potential Misuse:** Our approach has the potential to be inverted, allowing the generation of toxic sentences from initially non-toxic ones. Nevertheless, there are probably more straightforward methods to introduce toxicity that could reduce the risk of misuse in this scenario.

**Environmental Cost:** We acknowledge that our study necessitated thorough computational experiments for robust conclusions. Nonetheless, models in production may not demand such extensive experimentation. Instead, they can potentially leverage our key conclusions in this paper, thereby reducing future computational costs associated with this methodology.

**Acknowledgements**

The authors would like to thank the anonymous reviewers for helpful feedback and suggestions that have improved the paper.

**References**


Quan Do. 2019. Jigsaw unintended bias in toxicity classification.


B Matern et al. 1960. Spatial variation. stochastic models and their application to some problems in forest surveys and other sampling investigations. Meddelanden fran Statens Skogsforskningsinstitut, 49(5).


A Implementation details

We finetune BART (Lewis et al., 2020) models using classifier feedback to get the objective-specific language models for our experiments. Similarly, we use BART for the Parallel Training baseline as well. For training with classifier feedback, at each epoch, we use BART for the Parallel Training baseline as well.

Then, we use the desired label (“nontoxic”) as the posterior predictive distribution at a new point \( w = \{ \text{objective value for } w \} \).

A Gaussian Process

**Gaussian Process.** A Gaussian Process (\( \mathcal{GP} \)) is defined by a mean function \( \mu(w) \) and a covariance function\(^2\) \( k(w, w') \):

\[
f(w) \sim \mathcal{GP}(\mu(w), k(w, w'))
\]

At each optimization step, we observe the objective value for \( w \) using the validation data \( \{(s^{in}, s^{out})\} \). Given a set of observed data point \( D = \{(w, y)\} \) where \( y = o_{pref}(s^{out}|s^{in}, w) \) the posterior predictive distribution at a new point \( w_* \) is a Gaussian distribution:

\[
f(w_*)|D \sim \mathcal{GP}(\mu(w_*), \sigma^2(w_*))
\]

The mean \( \mu(w_*) \) and variance \( \sigma^2(w_*) \) are given by:

\[
\mu(w_*) = k_*^T (K + \sigma_n^2 I)^{-1} y
\]

\[
\sigma^2(w_*) = k_* - k_*^T (K + \sigma_n^2 I)^{-1} k_*
\]

where \( \sigma_n^2 \) is the noise parameter, representing the observation noise. Then, we can use an acquisition function to choose the next set of combining weights \( w_{next} \) as follows:

\[
w_{next} = \arg \max_w \text{acq}(w)
\]

**Acquisition Functions** - The most common acquisition functions are Lower Confidence Bound (UCB), Expected Improvement (EI), and Probability of Improvement (PI). We briefly describe them below.

The Lower Confidence Bound (LCB) acquisition function encourages exploration by selecting points with both high uncertainty and potential for improvement (Cox and John, 1992):

\[
LCB(w) = \mu(w) - \kappa \sigma(w)
\]

where \( \kappa \) is a tunable parameter that controls the trade-off between exploration and exploitation.

The Expected Improvement (EI) acquisition function quantifies how much improvement is expected over the current best observation (Mockus, 1998):

\[
EI(w) = \begin{cases} 
(\mu(w) - y_{best} - \xi)\Phi(Z) + \sigma(x)\phi(Z) & \text{if } \sigma(x) > 0 \\
0 & \text{if } \sigma(x) = 0 
\end{cases}
\]

where \( y_{best} \) is the best-observed function value, \( \xi \) is a small positive constant to control the exploration-exploitation trade-off, \( Z = \frac{\mu(w) - y_{best} - \xi}{\sigma(w)} \) is the cumulative distribution function of the standard normal distribution, and \( \phi(\cdot) \) is the probability density function.

The Probability of Improvement (PI) acquisition function measures the probability that the surrogate function value at a given point is better than the current best observation (Kushner, 1964):

\[
PI(w) = \Phi \left( \frac{\mu(w) - y_{best} - \xi}{\sigma(w)} \right)
\]

We use the “gp_hedge” option from scikit-optimize\(^3\) which probabilistically

\(^2\)Also referred to as a kernel.

\(^3\)The Scikit-Optimize Library
chooses one of the above three acquisition functions at every iteration. This strategy proved to have the best performance using the validation data. Further details can be found from the “gp_minimize” documentation.

### C Text Detoxification Examples

Table 1 lists a few text detoxification examples for both ParaDetox and APPDIA datasets for qualitative comparison between inputs (i.e., original toxic texts), references (i.e., detoxified versions by a human), and outputs from our proposed approach. [Warning: These inputs and references are from the original datasets and contain offensive language.]