Multi-armed bandits for resource efficient, online optimization of language model pre-training: the use case of dynamic masking

Iñigo Urteaga
Applied Physics and Applied Mathematics
Data Science Institute, Columbia University
New York, NY, USA
inigo.urteaga@columbia.edu

Moulay-Zaïdane Draïdia
Data Science Institute
Columbia University
New York, NY, USA
mad2314@columbia.edu

Tomer Lancewicki [†]
Walmart Global Tech,
USA
tomer.lancewicki@walmart.com

Shahram Khadivi
eBay Inc.,
Aachen, Germany
skhadivi@ebay.com

Abstract

We design and evaluate a Bayesian optimization framework for resource efficient pre-training of Transformer-based language models (TLMs). TLM pre-training requires high computational resources and introduces many unresolved design choices, such as selecting its pre-training hyperparameters. We propose a multi-armed bandit framework for the sequential selection of pre-training hyperparameters, aimed at optimizing language model performance, in a resource efficient manner. We design a Thompson sampling algorithm, with a surrogate Gaussian process reward model of the Masked Language Model (MLM) pre-training objective, for its sequential minimization. Instead of MLM pre-training with fixed masking probabilities, the proposed Gaussian process-based Thompson sampling (GP-TS) accelerates pre-training by sequentially selecting masking hyperparameters that improve performance. We empirically demonstrate how GP-TS pre-trains language models efficiently, i.e., it achieves lower MLM loss in fewer epochs, across a variety of settings. In addition, GP-TS pre-trained TLMs attain competitive downstream performance, while avoiding expensive hyperparameter grid search. GP-TS provides an interactive framework for efficient and optimized TLM pre-training that, by circumventing costly hyperparameter selection, enables substantial computational savings.

1 Introduction

In the field of Natural Language Processing (NLP), models for learning unsupervised representations from unlabeled text based on Transformer architectures (Vaswani et al., 2017) are the state-of-the-art on a variety of tasks (Kalyan et al., 2021).

[†] Work done while at eBay Inc. San Jose, CA.

Transformer-based language models (TLMs) like BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and their linage of advanced models (Amatriain, 2023), rely on the combination of an unsupervised pre-training of the model, and a subsequent task-specific fine-tuning procedure.

TLMs are pre-trained over large unlabeled text data using self-supervision, to learn the relationships between different sentences or words of the input. Once the TLM is pre-trained over large volumes of data, it can be used in various downstream tasks, by fine-tuning task-specific model layers.

With pre-training, TLMs learn language representations that are useful across downstream tasks, minimizing the need and burden of retraining the entire model from scratch, again, for each task. Extensive pre-training can lead to downstream performance improvements, i.e., it is worth learning complex TLMs in huge natural language corpora before fine-tuning them for particular tasks.

Many have replicated the pre-train-then-fine-tune strategy in different domains, e.g., pre-training BERT with scientific (Beltagy et al., 2019) and biomedical corpora (Lee et al., 2020; Alsentzer et al., 2019; Gu et al., 2021); or in-house, industry-specific TLMs (Kalyan et al., 2021). In addition, continual pre-training —taking a model pre-trained with general corpora to continue pre-training it with in-domain data— is of great value, yielding significant downstream gains (Gururangan et al., 2020).

Even if conceptually simple and empirically powerful, pre-training is challenging and expensive. Beyond the significant resources needed to pre-train the original BERT model by Devlin et al. (2018), the improvements of RoBERTa (Liu et al., 2019) relied on orders of magnitude higher computational resources (Kaplan et al., 2020).

The relationship between TLM architecture, training corpus, pre-training hyperparameters, and evaluation metrics is complex and obscure. Therefore, previously overlooked pre-training design choices, e.g., pre-training hyperparameter selection, result in significant performance differences.

With this work, we aim to improve the pre-training procedure of TLMs, by *sequentially* selecting hyperparameters that result in a more efficient and superior pre-training performance.

We hypothesize that an interactive selection of pre-training hyperparameters can accelerate and improve pre-training, i.e., we can achieve a better metric value in fewer epochs. It is critical not only to achieve superior performance, but to reduce the computational cost, steering clear from time- and resource-expensive procedures. Increased efficiency in TLM pre-training is paramount amidst concerns pertaining to the carbon footprint of large language models (Patterson et al., 2021); and specifically, the significant impact of hyperparameter selection on resource utilization and power consumption (Puvis de Chavannes et al., 2021).

Our TLM pre-training use-case is *random* dynamic masking of Masked Language Models (MLMs) —in contrast to rule or task-based MLM dynamic masking solutions proposed in the literature (Joshi et al., 2020; Sun et al., 2020). Even though Liu et al. (2019) showed the benefits of random dynamic masking, the search for optimal masking hyperparameters is often carried out based on heuristic techniques and grid-based search.

In machine learning (ML), hyperparameter selection is commonly addressed as a black-box optimization problem, which can be solved using evolutionary algorithms (Yu and Gen, 2010), entropy search methods (Hennig and Schuler, 2012; Hernández-Lobato et al., 2014), and Bayesian optimization (BO) (Frazier, 2018). In particular, BO can tackle the problem of optimizing an unknown objective function with possibly noisy evaluations (Snoek et al., 2012), and of speeding up resource allocation to promising hyperparameter configurations (Li et al., 2018). Aligned with the recent successes of Turner et al. (2021) in hyperparameter selection via BO, we propose a BO framework for sequential tuning of MLM pre-training hyperparameters. Our framework is different from BO techniques that speed up hyperparameter set evaluations, such as Hyperband (Li et al., 2018), which is a pure-exploration adaptive resource allocation algorithm for allocating resources among configurations in the non-stochastic setting.

We here cast the TLM pre-training procedure as a sequential decision process, in which at each interaction, a reinforcement learning agent selects an action (e.g., pre-training hyperparameters) to maximize cumulative rewards (e.g., the pre-training metric of interest). To accommodate the black-box nature of the pre-training objective function, we fit a probabilistic surrogate model to the empirical evaluations of the pre-training metric, and propose a bandit-based technique for its sequential optimization. In the MLM dynamic masking use case, the bandit actions are the dynamic masking probabilities; and the MLM performance, the unknown function the bandit is trying to maximize, based on estimates computed in the validation set.

Contrary to dynamic masking techniques that decide which subsets of tokens to mask via combinatorial optimization and dynamic programming (Vu et al., 2020); we target online, sequential selection of masking hyperparameters for accelerated and improved pre-training. In contrast to proposals that adapt the language model's masking policy to a particular task of interest (Kang et al., 2020), we devise a generic online optimization framework that, by sequential selection of MLM design choices, provides fast and superior TLM pre-training performance, when pre-training —from-scratch and continually— across diverse corpora.

The contributions of this work are:

- To present a bandit-based framework for efficient online optimization of TLM pre-training. We formulate a Gaussian Process based Thompson sampling (GP-TS) algorithm for sequential MLM loss minimization. The novelty lays on modeling TLM pre-training validation losses with a Gaussian process reward model, and on formulating a Thompson sampling policy that minimizes them.
- To showcase empirically how GP-TS pre-trains TLMs better and faster: both when pre-training from-scratch and continually, across a variety of corpora. Besides, to show that GP-TS pre-trained TLMs provide top fine-tuned performance across diverse in-domain tasks, in fewer interactions.
- To demonstrate that GP-TS's sequential selection of how many tokens of the input to mask —and how to mask them— results in improved and accelerated dynamic MLM pre-training, enabling significant resource utilization savings.

To the best of our knowledge, this work is the first to address online optimization of TLM pre-training with bandit-based BO, and to showcase its performance and resource efficiency benefits.

The manuscript is organized as follows: Section 2 provides the background on Bayesian optimization, multi-armed bandits and TLM pre-training; Section 3 describes the proposed GP-TS method for TLM pre-training optimization; with its empirical performance evaluated in Section 4. Concluding remarks are provided in Section 5.

2 Background

2.1 Bayesian optimization and bandits

Bayesian optimization (BO) is a framework to address hyperparameter optimization in ML (Snoek et al., 2012; Klein et al., 2017; Turner et al., 2021), and many closely related applications (Negoescu et al., 2011; Calandra et al., 2016; Frazier and Wang, 2016; Hernández-Lobato et al., 2017; Candelieri et al., 2018). BO relies on a probabilistic surrogate model of the objective function, to tackle the problem of simultaneously fitting and optimizing a high-dimensional, non-convex function with unknown smoothness, and possibly noisy evaluations (Shahriari et al., 2015; Frazier, 2018). Due to the black-box nature of BO, the surrogate model must provide a measure of uncertainty, for which generative models, Bayesian neural networks and Gaussian processes are used (Maddox et al., 2021). Using this surrogate model, an acquisition function determines the next promising candidate to evaluate. To address the challenge of learning about the environment (i.e., exploration) while simultaneously maximizing the observed outcomes (i.e., exploitation), the multi-armed bandit provides a useful framework (Lai and Robbins, 1985).

The multi-armed bandit (MAB) is an abstraction for problems that require learning while simultaneously maximizing attained rewards, i.e., balancing the exploration-exploitation tradeoff (Lattimore and Szepesvári, 2020). A MAB is a sequential decision process that requires decision-making under uncertainty (Slivkins, 2019). At each interaction $t = 1, \cdots, T$, a bandit agent chooses an action $a_t \in \mathcal{A}$ from a (not necessarily finite) set of actions $\mathcal{A}$, and it observes stochastic reward r_t drawn from an unknown distribution of the selected arm, a_t, often characterized parametrically, $r_t \sim p(\cdot|a_t, \theta)$.

The MAB agent's goal is to maximize (expected) cumulative rewards, $R_T = \sum_{t=1}^{T} \mu_{a,t}$, with each arm's expected reward denoted as $\mu_a = \mathbb{E}_p\left\{r|a,\theta\right\}$. The challenge is on the lack of knowledge about the reward generating mechanism, which demands learning its properties (e.g., its parameters), as it interacts with the environment.

A plethora of MAB algorithms have been proposed and analyzed over the years, from computing optimal strategies (Gittins, 1979) and greedy approaches (Auer et al., 2002), to upper confidence interval (Lai, 1987; Kaufmann et al., 2012) and Thompson sampling (Thompson, 1935) algorithms. For models in the exponential family, the latter have been empirically and theoretically proven to perform competitively (Lai, 1987; Kaufmann et al., 2012; Agrawal and Goyal, 2012, 2013; Korda et al., 2013), and extensions have been proposed that model observed rewards via ensembles of models (Lu and Roy, 2017), Gaussian mixture models (Urteaga and Wiggins, 2018), Gaussian processes (Srinivas et al., 2010; Grünewälder et al., 2010), and neural networks (Osband et al., 2016).

In the context of BO in general, and MABs in particular, reward uncertainty quantification is critical. Gaussian processes (Rasmussen and Williams, 2005) provide not only adequate Bayesian uncertainty estimates, but a flexible solution for surrogate models that encode smoothness assumptions of the payoff function (Krause and Ong, 2011; Bogunovic et al., 2016; Nguyen et al., 2020). We resort to a Gaussian process reward model in the proposed bandit-based BO framework for TLM pre-training.

2.2 Language model pre-training and the Masked Language Model

Pre-training enables learning representations that generalize across tasks, i.e., it allows for a language model to be better initialized for quick fine-tuning (while avoiding overfitting) to downstream tasks. TLMs learn language representations in pre-training based on one (or more) self-supervised task. Two popular pre-training objectives are Masked Language Model (MLM) and Next Sentence Prediction (NSP) (Devlin et al., 2018).

We focus on MLM pre-training as in (Devlin et al., 2018; Liu et al., 2019); where for an input sequence of words or tokens, a random sample of the tokens is replaced with the $[MASK]$ token, and the goal is to predict them. For an input sequence d of N tokens, with special tokens delimiting them,

$$d \equiv [CLS], q_1, \cdots, q_N, [EOS] \qquad (1)$$

MLMs select a random sample of the tokens $q_i, i = \{1, \cdots, N\}$, replace them with the mask, and learn to predict these masked tokens. For pre-training the original BERT model (Devlin et al., 2018), a random but *static* subset of the input sequence tokens was replaced with the mask.

Liu et al. (2019) proposed a *dynamic* masking procedure, which generates a new masking pattern (given a fixed probability of masking) for every input sequence. Liu et al. (2019) demonstrate that this dynamic approach is beneficial when pre-training for more steps or with larger datasets.

Dynamic masking relies on several hyperparameters: (i) the probability ρ of replacing an input token with the mask, (ii) the probability γ that a masked token is left unmasked, and (iii) the probability λ of replacing a token with a random token, instead of with the mask. Online optimization of these hyperparameters $\psi = (\rho, \gamma, \lambda)$ is the use-case for our experiments in Section 4.

MLM pre-training aims at minimizing the MLM loss: a function of the original (D) and masked ($\widehat{D}$) datasets, the TLM architecture with its parameters $w \in W$, and pre-training hyperparameters $\psi \in \Psi$. The MLM objective is the cross-entropy loss of predicting the masked tokens in the masked sequence $\widehat{d} \in \widehat{D}$, where we denote with $m_i = \{0, 1\}$ whether tokens $q_i, i = \{1, \cdots, N\}$, from the original input sequence $d \in D$ have been masked in $\widehat{d}$:

$$l(d, \widehat{d}; w, \psi) = -\log p(d|\widehat{d}; w, \psi) \quad (2)$$

$$= -\sum_{i=1}^{N} m_i \log p(q_i|\widehat{q_i}; w, \psi) \quad (3)$$

$$= -\sum_{i=1}^{N} m_i \log \left(\frac{e^{\left(\chi(\widehat{q_i}; w, \psi)^\top \xi(q_i)\right)}}{\sum_{i'=1}^{N} e^{\left(\chi(\widehat{q_{i}'}; w, \psi)^\top \xi(q_{i}')\right)}} \right) \quad (4)$$

where $\chi(\widehat{q_i}; w, \psi)$ denotes the TLM's representation of the masked token q_i, and $\xi(q_i)$ is its original embedding. The pre-training objective is to find the TLM that minimizes the MLM loss between the original dataset D and its masked version $\widehat{D}$. In practice, this minimization is executed via stochastic gradient-descent, run for $e = 1, \cdots, E$, epochs with random mini-batches $D_e \in D$ per epoch e, $\widehat{w_e} = \operatorname{argmin}_{w \in W} l(D_e, \widehat{D_e}; w, \psi)$.

The analytical form of the MLM loss, a function of selected hyperparameters ψ and the data where it is evaluated, is in general complex and unknown. However, estimates of the MLM loss are available at every pre-training epoch e. Namely, an empirical estimate of the MLM loss can be computed in the validation set.

For fair comparisons under different training setups (e.g., mini-batch sizes and hyperparameters), per-epoch *averaged* empirical MLM losses are computed in the validation dataset D_{val},

$$\bar{l}(D_{val}; \psi) = \bar{l}(D_{val}, \widehat{D_{val}}; w, \psi)$$
$$= -\sum_{d \in D_{val}} \frac{\sum_{i=1}^{N_d} m_i \log p(q_i|\widehat{q_i}; w, \psi)}{\sum_{i'=1}^{N_d} m_{i'}}, \quad (5)$$

where we drop the dependency with respect to TLM parameters w and the masked validation dataset $\widehat{D_{val}}$ to avoid notation clutter.

3 Proposed bandit-based framework

We cast TLM pre-training as a sequential decision process to be solved by a multi-armed bandit agent that interactively optimizes the analytically unknown pre-training loss, based on its sequentially observed empirical evaluations. We define pre-training steps, i.e., a fixed number of stochastic gradient updates u in the training set, as bandit interactions $t = 1, \cdots, T$. The goal is to minimize the TLM pre-training objective $l(\cdot|\psi)$ given tunable hyperparameters ψ, with (stochastic) evaluations of the loss function in the validation set.

Pre-training hyperparameters at interaction t, ψ_t, are the bandit's arms, i.e., $a_t = \psi_t$. For MLM pre-training with dynamic masking, at each bandit interaction, the agent selects hyperparameters ψ (the proportion of tokens to mask and their masking probabilities), pre-trains the TLM for certain stochastic updates to minimize the MLM loss, and evaluates its performance in the validation subset, as per Equation (5). Due to the black-box nature of the pre-training objective, for which only stochastic evaluations are available, we formulate a surrogate reward function (leveraging empirical MLM validation loss estimates) for the bandit to maximize, as it sequentially selects which arm to play.

3.1 From MLM pre-training to Gaussian process-based regret minimization

We transform the empirical pre-training validation loss at each MAB interaction into a reward quantity for it's sequential minimization by the bandit agent. Specifically, we compute bandit rewards as the normalized difference in averaged empirical MLM losses between bandit interactions, i.e.,

$$r_t(\psi_t) = \frac{[-\bar{l}_t(D_{val};\psi_t)] - [-\bar{l}_{t-1}(D_{val};\psi_{t-1})]}{[-\bar{l}_{t-1}(D_{val};\psi_{t-1})]} \ . \tag{6}$$

By normalizing reward differences per-interaction, we mitigate the potential non-stationary effect sequentially selected hyperparameters might have on TLM pre-training. With rewards as (normalized) empirical MLM loss differences, we capture how much (relative) improvement each action provides.

Rewards in Equation (6) are based on stochastic draws from an analytically unknown objective function, i.e., only empirical estimates $\bar{l}_t(\cdot)$ of the MLM objective are available. To accommodate these noisy observations of the unknown loss function $l(\cdot|\psi)$ —that we aim at optimizing with respect to its hyperparameters ψ— we model the bandit reward function via a Gaussian process (GP) model $f(\cdot;\theta)$ of the pre-training objective, with observed rewards independent and identically (i.i.d.) distributed as

$$r_t(\psi_t) = f(\psi_t;\theta) + \epsilon_t \ , \tag{7}$$

where ϵ_t denotes the stochastic nature of each of the observed rewards —based on empirical estimates computed in Equation (6). Hence, we overcome the black-box nature of the pre-training objective (e.g., the MLM loss) by modeling observed rewards as realizations of a noisy surrogate GP model (Rasmussen and Williams, 2005).

The mean $\mu(\cdot)$ and kernel functions $k(\cdot,\cdot)$ of a GP $f(\cdot) \sim GP(\mu(\cdot), k(\cdot,\cdot))$ determine the reward function class: i.e., the regularity and smoothness of the pre-training loss. These are parameterized prior-functions $\mu(\cdot|\theta_\mu)$ and $k(\cdot,\cdot|\theta_k)$, which can be fitted to the observed data $r_{1:T} = (r_1, \cdots, r_T)$ at inputs $\psi_{1:T} = (\psi_1, \cdots, \psi_T)$ (Rasmussen and Williams, 2005). For instance, via Type-II maximum likelihood estimation (MLE) of the GP parameters $\theta = (\theta_\mu, \theta_k)$, $\hat{\theta} = \mathrm{argmax}_\theta \log p\left(r_{1:T}|f(\psi_{1:T}|\theta)\right)$, where the data likelihood $p(r|f(\cdot;\theta))$ is a function of the observation noise probability distribution.

Given a fitted GP, posterior inference — computing the predictive distribution of a new datapoint ψ' after observing $\psi_{1:T}$— can be performed in closed or approximate form (Titsias, 2009; Flaxman et al., 2015; Pleiss et al., 2018).

3.2 GP-Thompson sampling for TLM pre-training.

Leveraging the GP reward model in Equation (7), we devise a bandit-based interactive method that executes a Thompson sampling (TS) policy[1] (Russo et al., 2018) for TLM pre-training optimization.

The proposed Gaussian process-based Thompson sampling (GP-TS) —with pseudo-code provided in Algorithm 1— views the TLM pre-training objective as an unknown black-box function with inputs $a_t = \psi_t$ and outputs $r_t(\psi_t)$ as in Equation (6). GP-TS makes decisions on what bandit arm $a_t = \psi_t$ to play at each TLM pre-training interaction $t = 1, \cdots, T$, informed by its GP reward model of Equation (7), to maximize its observed cumulative rewards $R_T = \sum_{t=1}^T r_t(\psi_t)$.

Algorithm 1 GP-TS for TLM pre-training

```
 1: Input: TLM and pre-training corpus
 2: Input: Pre-training hyperparameter space $\Psi$
 3: Input: Number of pre-training interactions $T$,
    number of updates per-interaction $u$
 4: Input: GP prior functions $\mu(\cdot)$ and $k(\cdot,\cdot)$,
    with initial hyperparameters $\theta_0$
 5: Initialize: $\mathcal{A} = \Psi$, $\hat{\theta}_1 = \theta_0$, $\mathcal{H}_1 = \emptyset$
 6: for $t = 1, \cdots, T$ do
 7:     Draw posterior sample from GP,
            $\mu_a^{(t)} \sim f(\mu_t(a|\hat{\theta}_t), k_t(a, a'|\hat{\theta}_t))$ .
 8:     Select arm based on drawn posterior sample,
            $a_t = \mathrm{argmax}_{a' \in \mathcal{A}} \mu_{a'}^{(t)}$ .
 9:     Run TLM pre-training for $u$ steps,
            with hyperparameters $\psi_t = a_t$ .
10:     Compute pre-trained TLM validation loss,
            $\bar{l}_t(D_{val};\psi_t)$ as in Equation (5) .
11:     Observe bandit reward,
            $r_t(\psi_t)$ as in Equation (6) .
12:     Update bandit history
            $\mathcal{H}_{1:t} = \mathcal{H}_{1:t-1} \cup \{a_t = \psi_t, r_t(\psi_t)\}$ .
13:     Fit GP model with $\mathcal{H}_{1:t}$,
            $\hat{\theta}_{t+1} = \mathrm{argmax}_\theta \log p\left(r_{1:t}|f(\psi_{1:t};\theta)\right)$ .
14: end for
```

GP-TS accommodates continuous arms $a_t = \psi_t$, with dimensionality determined by the pre-training hyperparameter space $\psi \in \Psi$. Any TLM can be used within the proposed framework, as long as the hyperparameter space $\psi \in \Psi$ is identified, and

[1]We resort to Thompson sampling due to both its implementation flexibility and efficiency, as well as its competitive empirical performance with theoretical guarantees in many settings (Agrawal and Goyal, 2013; Krause and Ong, 2011; Nguyen et al., 2020; Srinivas et al., 2010).

rewards as in Equation (6) are computed for a pre-training objective $l(\cdot|\psi)$ of interest.

GP-TS draws predictive function samples for the next TLM pre-training interaction from its GP reward model posterior, updated at every bandit interaction as indicated in Step 7 of Algorithm 1. As in other TS methods, these samples are used to determine —in Step 8 of Algorithm 1— the arms (hyperparameters ψ_t) to be used in the next bandit interaction. After u pre-training steps[2], the model's MLM validation loss is computed to evaluate the observed bandit rewards $r_t(\psi_t)$ of Equation (6). After each interaction t, new evidence is collected in Step 12 to re-fit the GP model to the observed input (action)-output (rewards) history $\mathcal{H}_{1:t}$. For instance, via Type-II MLE as in Step 13 of Algorithm 1, although other GP parameter optimization procedures might be used —see Appendix A for details on GP models and posterior inference.

4 Experiments

4.1 Evaluation set-up

We probe the ability of the proposed GP-TS to, given a dataset, a TLM architecture, and a computational budget, efficiently pre-train well-performing language models[3]. For our experiments, we incorporate RoBERTa (Liu et al., 2019) as implemented by Ott et al. (2019) in our Python implementation of GP-TS[4] as in Algorithm 1 —Appendix B.1 provides implementation and configuration details.

We compare pre-training performance of RoBERTa models based on a grid-search over masking hyperparameters —as executed by Liu et al. (2019)— to RoBERTa models pre-trained by GP-TS[5]. We focus our evaluation on MLM validation loss and downstream per-task accuracy metrics, and report the negligible computational overhead of pre-training with GP-TS in Appendix B.3.

We study two variants of GP-TS, depending on the masking hyperparameters it optimizes:
(i) `GP-TS` ρ, where the bandit arm is the masking probability ρ of replacing an input token with the *mask* token (other hyperparameters are fixed to default $\gamma = 0.1$ and $\lambda = 0.1$ values as suggested by Liu et al. (2019)); and

(ii) `GP-TS` $\psi = (\rho, \gamma, \lambda)$, where GP-TS optimizes over all MLM dynamic masking hyperparameters: the bandit search space is a three-dimensional hypercube Ψ with no previous expert guidance on hyperparameter selection.

Pre-training datasets. We gather three distinct datasets, two based on publicly available corpora, and one based on private data from eBay:

- **wiki-c4**: We pre-process and encode publicly available Wikitext-103 (Merity et al., 2016) and Google's c4 RealNews (Zellers et al., 2019) datasets for pre-training, from scratch, each of TLM. This corpora is similar to those originally used by Devlin et al. (2018) and Liu et al. (2019).
- **mimic**: We pre-process and encode free-text clinical notes available in the public MIMIC-III Clinical database (Pollard and Johnson, 2016), which contains deidentified nursing and physician notes, ECG and imaging reports, and discharge summaries for patients who stayed in intensive care units at Beth Israel Deaconess Medical Center.
- **e-commerce**: We pre-process and encode a random subset of eBay marketplace inventories, which contains different product titles and descriptions provided by marketplace users, as well as category tags associated with each item and product reviews.

Each dataset contains text of very different linguistic characteristics and sizes (see summary statistics in Appendix B.4), which we leverage to investigate TLM pre-training across a variety of settings.

We evaluate candidate TLMs (i) when pre-training *from-scratch*, i.e., from a randomly initialized architecture; and (ii) with *continual* pre-training, i.e., when continuing pre-training a TLM architecture previously trained in other NLP corpora (Kalyan et al., 2021). Continual pre-training results we present are for the RoBERTa-base architecture as pre-trained by Facebook Research (2022) that we continue to pre-train in our domain-specific datasets, i.e., `mimic` and `e-commerce`.

Fine-tuning in downstream tasks. Pre-trained language models are most useful when applied to downstream tasks, as there is no need to retrain the entire model again. We evaluate pre-trained TLM's in the following in-domain tasks[6]:

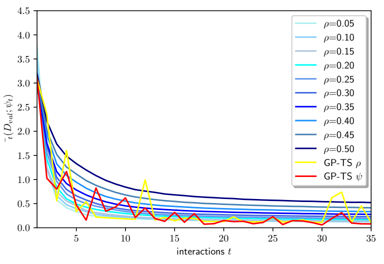
[2]Note that u stochastic gradient updates might or might not correspond to a full pre-training epoch e.

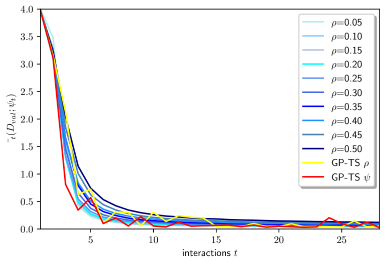
[3]We scrutinize pre-training performance of a specific TLM architecture under equal experimental conditions and do not compare performance to state-of-the-art, large-scale TLMs.

[4]Code available at https://github.com/iurteaga/gp_ts_nlp.

[5]We do not execute any other hyperparameter optimization.

[6]We abstain from fine-tuning RoBERTa-base models, pre-trained with `wiki-c4` data only, in downstream Glue tasks (Wang et al., 2018), as these would not match state-of-the-art results due to both the size-limited pre-training dataset, and the model architecture used.


(a) wiki-c4.


(b) mimic.

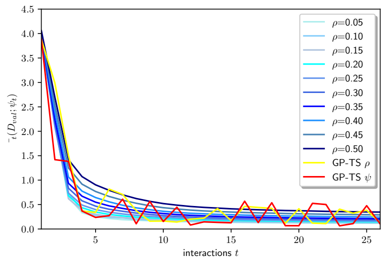

(c) e-commerce.

Figure 1: MLM validation loss comparison (lower is better) of grid-search and GP-TS based *from-scratch* pre-trained RoBERTa models, over interactions.

- **e-commerce title classification**: A binary classification task to decide whether a pair of item titles belong to the same marketplace product. Item titles are instances of a product sold by a specific seller, which can have different attributes like condition or can exist as a special version (e.g., a signed book), yet refer to the same product.

- **e-commerce title similarity**: A task using the same title-pair data as above, but formulated as a similarity task. Namely, we learn a distance metric between item titles to help discriminate whether they belong or not to the same product.

- **e-commerce title quality**: A classification task that predicts if a title fulfills the marketplace requirements for it to be a product title. Titles must contain the product's main relevant information —the brand, the product name and/or type, and all distinguishable attributes, i.e., its key features— but should not contain conditions, marketing terms, or any other non-product related information.

- **medical MLI**: A natural language inference task annotated by doctors (Shivade, 2019), which is grounded in the medical history of patients collected in MIMIC-III (Pollard and Johnson, 2016). It contains sentence pairs —the premise and the hypothesis statements— with a corresponding label indicating their inferential relationship (e.g., entailment, contradiction, or neutral).

Summary statistics for each in-domain per-task dataset are provided in Appendix B.6.

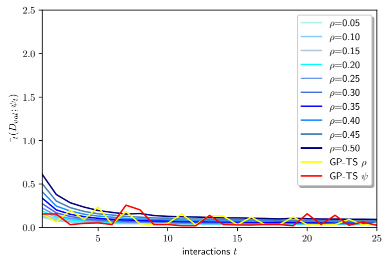
To elucidate how the pre-trained TLMs' quality evolves over pre-training interactions, we fine-tune (for ten epochs) the pre-trained RoBERTa models at each pre-training interaction t. We report the best classification accuracy of each fine-tuned model across pre-training interactions and fine-tuning epochs.

4.2 GP-TS pre-training of RoBERTa models

We compare *from-scratch* pre-training performance of all RoBERTa models (pre-trained with fixed hyperparameters or by GP-TS) in Figure 1, where we illustrate MLM validation losses of each model over pre-training interactions: GP-TS attains the lowest MLM loss values in fewer interactions.

Recall that when pre-training TLMs, validation performance varies across training epochs; hence, practitioners are interested in identifying the best pre-trained model —as per the lowest validation metric— instead of selecting the pre-trained TLM available at the last training epoch.

Results for *continual* pre-training are provided in Figure 2 below, where we observe that GP-TS continually pre-trains the best performing RoBERTa models —the fastest— for both in-domain datasets.


(a) mimic.

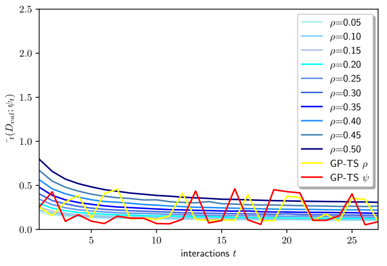

(b) e-commerce.

Figure 2: MLM validation loss comparison (lower is better) of grid-search and GP-TS based *continually* pre-trained RoBERTa models over interactions.

MLM validation losses for models pre-trained with GP-TS fluctuate across interactions, depending on the stochastic action (hyperparameter value) selected by the GP-TS agent. Practitioners are interested in using the model with the lowest validation MLM loss, which GP-TS consistently finds across all studied datasets and pre-training approaches, in fewer pre-training interactions. We evaluate the influence of different realizations of GP-TS (with different random seeds) in Table 1, where we observe that GP-TS always pre-trains models with the lowest MLM loss, and in less interactions (indicated within parentheses).

Table 1: Best MLM loss attained before interactions 20 and 30, when pre-training RoBERTa models continually in the medical domain corpora.

Model	By interaction 20	By interaction 30
	Best MLM loss (at interaction)	Best MLM loss (at interaction)
ρ=0.05	0.04 (18)	0.037 (28)
ρ=0.10	0.04 (18)	0.036 (27)
ρ=0.15	0.044 (18)	0.038 (27)
ρ=0.20	0.048 (18)	0.042 (28)
ρ=0.25	0.054 (19)	0.046 (27)
ρ=0.30	0.066 (18)	0.056 (27)
ρ=0.35	0.076 (19)	0.064 (29)
ρ=0.40	0.091 (19)	0.077 (29)
ρ=0.45	0.113 (19)	0.095 (29)
ρ=0.50	0.134 (19)	0.112 (27)
GP-TS ρ (seed 1)	0.037 (14)	0.033 (20)
GP-TS ρ (seed 2)	0.036 (19)	0.033 (28)
GP-TS ρ (seed 3)	0.038 (14)	0.032 (21)
GP-TS ρ (seed 4)	0.032 (18)	0.032 (18)
GP-TS ρ (seed 5)	0.038 (13)	0.032 (20)
GP-TS ψ (seed 1)	0.027 (8)	0.019 (21)
GP-TS ψ (seed 2)	**0.02 (15)**	0.02 (15)
GP-TS ψ (seed 3)	**0.02 (17)**	0.019 (28)
GP-TS ψ (seed 4)	0.036 (14)	0.019 (21)
GP-TS ψ (seed 5)	**0.02 (16)**	**0.018 (28)**

GP-TS not only circumvents the need for costly grid searches, but enables improved performance: it attains reduced MLM loss at earlier interactions than grid-search baselines. Recall how GP-TS ψ outperforms all the alternatives in Table 1, as it pre-trains models with the lowest MLM, the fastest —even when no good initial guesses for the MLM hyperparameters $\psi = (\rho, \gamma, \lambda)$ are available.

In summary, the benefits of interactive GP-TS pre-training do not pertain to the attained MLM values only, but to an accelerated, efficient procedure. We emphasize the computational efficiency of GP-TS: it adds little to no overhead —details on the computational cost of GP-TS are provided in Appendix B.3— while providing clear benefits for language model pre-training. It attains best MLM pre-training performance in less interactions, avoiding computationally expensive hyperparameter search.

To the best of our knowledge, these experiments provide novel evidence that, instead of MLM pre-training with fixed masking hyperparameters, sequentially deciding which masking values to use is beneficial. GP-TS finds *sequences* of dynamic masking hyperparameters (when optimizing over ρ or a three-dimensional hyperparameter space $\psi \in \Psi$) that minimize MLM loss across datasets, when pre-training from-scratch and continually.

4.3 GP-TS pre-trained RoBERTa models for downstream fine-tuned tasks

We scrutinize how performant in-domain GP-TS pre-trained RoBERTa models are, when compared to grid-search based models, after in-domain per-task fine-tuning. The fine-tuned accuracy of continually pre-trained models[7] of Figure 2 are presented in Table 2: we showcase, per-task, best test-set accuracy for each fine-tuned model, and at which pre-training interaction was such value attained. Results are computed on each per-task test-set, i.e., a subset of each task's dataset (see details in Table 11) that has not been used for fine-tuning nor hyperparameter optimization.

Table 2: Best fine-tuned, downstream task test-set accuracy (higher is better) for continually pre-trained RoBERTa models. The first row corresponds to the fine-tuned performance of the RoBERTa model from which continual pre-training is started.

Model	e-commerce title classification	e-commerce title similarity	e-commerce title quality	medical MLI
	Accuracy (at interaction)	Accuracy (at interaction)	Accuracy (at interaction)	Accuracy (at interaction)
RoBERTa base	97.2 (0)	97.2 (0)	75.1 (0)	67.5 (0)
ρ=0.05	97.8 (26)	97.8 (26)	77.6 (15)	72.9 (3)
ρ=0.10	97.9 (27)	97.9 (27)	77.7 (15)	71.9 (9)
ρ=0.15	97.8 (13)	97.8 (13)	77.7 (18)	72.5 (13)
ρ=0.20	97.8 (8)	97.8 (8)	77.4 (10)	73.3 (14)
ρ=0.25	97.9 (17)	97.9 (17)	77.7 (6)	72.9 (12)
ρ=0.30	97.9 (19)	97.9 (19)	77.8 (7)	**73.2 (7)**
ρ=0.35	97.9 (9)	97.9 (9)	77.8 (18)	72.8 (7)
ρ=0.40	97.8 (9)	97.8 (9)	78.2 (24)	72.6 (9)
ρ=0.45	97.8 (11)	97.8 (11)	**78.3 (16)**	72.9 (7)
ρ=0.50	97.9 (8)	97.9 (8)	77.9 (7)	72.6 (9)
GP-TS ρ	97.9 (13)	97.9 (13)	77.5 (17)	72.6 (9)
GP-TS ψ	**98.0 (10)**	**98.0 (10)**	77.8 (20)	72.3 (6)

These results exhibit how GP-TS pre-trains performant language models —with top accuracy— often at earlier interactions than when pre-training with static hyperparameters: e.g., the continually pre-trained GP-TS ψ model (see last row of Table 2) provides best downstream accuracy for two e-commerce tasks and competitive accuracy in others, in just a few pre-training interactions.

[7]The downstream, fine-tuned performance of RoBERTa models pre-trained from-scratch with in-domain data is, as expected, lower than if continually pre-trained.

This efficiency is of practical importance, due to the significant resource savings it affords. A pre-training hyperparameter grid-search does not provide significant downstream performance improvements, yet it demands high computational resources —the computational complexity of a grid-search over hyperparameters $\psi = (\rho, \gamma, \lambda)$ with n candidates per hyperparameter is $\mathcal{O}(3^n)$.

On the contrary, by letting GP-TS pre-train TLMs, best pre-training MLM performance is achieved, with well-performing fine-tuned model accuracy across downstreams tasks, in fewer pre-training interactions.

5 Conclusion

We present a multi-armed bandit-based Bayesian optimization framework for the sequential selection of pre-training hyperparameters towards optimized Transformer-based language model performance.

We develop and evaluate an interactive, Gaussian process-based Thompson sampling (GP-TS) framework for accelerated language model pre-training. We model noisy evaluations of the pre-training objective (e.g., the MLM loss) as drawn from a surrogate Gaussian process that the bandit agent aims to minimize.

We provide empirical evidence of how GP-TS, when applied to MLM dynamic masking, attains superior and accelerated (both from-scratch and continual) pre-training performance, along with excellent in-domain downstream metric values.

While Liu et al. (2019) randomly select —with fixed probability— which input tokens to mask, we show that *sequentially* adapting the masking hyperparameters with GP-TS results in enhanced and efficient pre-training. Notably, GP-TS interactively selects hyperparameters that result in top performing models faster, enabling significant resource efficiency, of critical importance in practice.

Building upon our formulation and the provided evidence, we envision follow-up work investigating the proposed method's ability to successfully pre-train large-scale models in general purpose corpora, as well as for optimizing domain-specific models.

Limitations

There are several limitations to account for in the presented work. First, the large GPU requirements for the execution and replication of the presented experiments. Second, the lack of empirical results beyond English-based text, and how morphologically and syntactically more complex corpora may affect the presented evidence. Third, our evaluation section compares GP-TS performance to the common hyperparameter grid-search alternative, yet we acknowledge that other Bayesian optimization techniques used in the machine learning community may provide suitable and competitive alternatives to explore. In addition, we have not run any hyperparameter tuning beyond MLM dynamic masking, which might improve all studied algorithms' performance. Finally, our conclusions are limited to RoBERTa models pre-trained via MLM dynamic masking, and therefore, investigation of how GP-TS generalizes to other TLM pre-training approaches and architectures is lacking.

Ethics Statement

This work raises ethical and societal considerations associated with the use and biases of pre-collected natural language data, the energetic and environmental impact of extensive GPU resource usage, and the downstream applications of language models. We acknowledge the potential implicit biases within the publicly available datasets used. E.g., `mimic` reports are limited to the population attended at Beth Israel Deaconess Medical Center, and may contain implicit biases of health practitioners there. We have carefully sampled data for the `e-commerce` dataset to avoid biases over specific products, users and sellers. We are also aware of the rising concerns pertaining to the carbon footprint of large language models (Patterson et al., 2021), and the significant impact hyperparameter selection techniques have on resource utilization and power consumption (Puvis de Chavannes et al., 2021). Finally, we acknowledge the wide range of established and anticipated risks that language models pose to society (Weidinger et al., 2021).

Acknowledgements

Iñigo Urteaga and Moulay-Zaïdane Draïdia were partially supported by funds from eBay's Research and University Partnership for Technology (eRUPT) program. We also acknowledge computing resources from Columbia University's Shared Research Computing Facility project, which is supported by NIH Research Facility Improvement Grant 1G20RR030893-01, and associated funds from the New York State Empire State Development, Division of Science Technology and Innovation (NYSTAR) Contract C090171.

References

Shipra Agrawal and Navin Goyal. 2012. Analysis of Thompson Sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1.

Shipra Agrawal and Navin Goyal. 2013. Further Optimal Regret Bounds for Thompson Sampling. In *Artificial Intelligence and Statistics*, pages 99–107.

Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. *arXiv preprint arXiv:1904.03323*.

Xavier Amatriain. 2023. Transformer models: an introduction and catalog. *arXiv preprint arXiv:2302.07730*.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. 2016. Time-varying gaussian process bandit optimization. In *Artificial Intelligence and Statistics*, pages 314–323. PMLR.

Fairseq by Facebook Research. 2022. RoBERTa: A Robustly Optimized BERT Pretraining Approach, pre-trained model using the BERT-base architecture. Available online at `https://dl.fbaipublicfiles.com/fairseq/models/roberta.base.tar.gz`.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. 2016. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23.

Antonio Candelieri, Raffaele Perego, and Francesco Archetti. 2018. Bayesian optimization of pump operations in water distribution systems. *Journal of Global Optimization*, 71(1):213–235.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Seth Flaxman, Andrew Wilson, Daniel Neill, Hannes Nickisch, and Alex Smola. 2015. Fast Kronecker Inference in Gaussian Processes with non-Gaussian Likelihoods. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 607–616, Lille, France. PMLR.

Peter I. Frazier. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Peter I. Frazier and Jialei Wang. 2016. Bayesian optimization for materials design. In *Information science for materials discovery and design*, pages 45–75. Springer.

Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. 2018. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*.

J. C. Gittins. 1979. Bandit Processes and Dynamic Allocation Indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177.

Steffen Grünewälder, Jean-Yves Audibert, Manfred Opper, and John Shawe-Taylor. 2010. Regret Bounds for Gaussian Process Bandit Problems. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 273–280, Chia Laguna Resort, Sardinia, Italy. PMLR.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. *arXiv preprint*.

Philipp Hennig and Christian J. Schuler. 2012. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13(57):1809–1837.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. 2014. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. 2017. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *International conference on machine learning*, pages 1470–1479. PMLR.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. 2021. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542*.

Minki Kang, Moonsu Han, and Sung Ju Hwang. 2020. Neural mask generator: Learning to generate adaptive word maskings for language model adaptation. *arXiv preprint arXiv:2010.02705*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Emilie Kaufmann, Olivier Cappe, and Aurelien Garivier. 2012. On Bayesian Upper Confidence Bounds for Bandit Problems. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 592–600, La Palma, Canary Islands. PMLR.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536, Fort Lauderdale, FL, USA. PMLR.

Nathaniel Korda, Emilie Kaufmann, and Rémi Munos. 2013. Thompson Sampling for 1-Dimensional Exponential Family Bandits. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1448–1456. Curran Associates, Inc.

Andreas Krause and Cheng Ong. 2011. Contextual Gaussian Process Bandit Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Tze Leung Lai. 1987. Adaptive Treatment Allocation and the Multi-Armed Bandit Problem. *The Annals of Statistics*, 15(3):1091–1114.

Tze Leung Lai and Herbert Robbins. 1985. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6(1):4–22.

Tor Lattimore and Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xiuyuan Lu and Benjamin Van Roy. 2017. Ensemble sampling. In *Advances in Neural Information Processing Systems*, pages 3258–3266.

Wesley J Maddox, Maximilian Balandat, Andrew Gordon Wilson, and Eytan Bakshy. 2021. Bayesian Optimization with High-Dimensional Outputs. *arXiv preprint arXiv:2106.12997*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Diana M. Negoescu, Peter I. Frazier, and Warren B. Powell. 2011. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363.

Vu Nguyen, Vaden Masrani, Rob Brekelmans, Michael Osborne, and Frank Wood. 2020. Gaussian Process Bandit Optimization of the Thermodynamic Variational Objective. In *Advances in Neural Information Processing Systems*, volume 33, pages 5764–5775. Curran Associates, Inc.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep Exploration via Bootstrapped DQN. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*.

Geoff Pleiss, Jacob Gardner, Kilian Weinberger, and Andrew Gordon Wilson. 2018. Constant-Time Predictive Distributions for Gaussian Processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4114–4123. PMLR.

Tom J Pollard and Alistair EW Johnson. 2016. The MIMIC-III clinical database (version 1.4). *The MIMIC-III Clinical Database. PhysioNet*.

Lucas Høyberg Puvis de Chavannes, Mads Guldborg Kjeldgaard Kongsbak, Timmie Rantzau, and Leon Derczynski. 2021. Hyperparameter power impact in transformer language model training. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 96–118, Virtual. Association for Computational Linguistics.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press.

Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2018. A Tutorial on Thompson Sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Chaitanya Shivade. 2019. MedNLI - A Natural Language Inference Dataset For The Clinical Domain (version 1.0.0). PhysioNet. https://doi.org/10.13026/C2RS98.

Aleksandrs Slivkins. 2019. Introduction to Multi-Armed Bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286.

Edward Snelson and Zoubin Ghahramani. 2006. Sparse Gaussian Processes using Pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1015–1022, USA. Omnipress.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

William R. Thompson. 1935. On the Theory of Apportionment. *American Journal of Mathematics*, 57(2):450–456.

Michalis Titsias. 2009. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. 2021. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*.

Iñigo Urteaga and Chris Wiggins. 2018. Variational inference for the multi-armed contextual bandit. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 698–706, Playa Blanca, Lanzarote, Canary Islands. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Thuy-Trang Vu, Dinh Phung, and Gholamreza Haffari. 2020. Effective unsupervised domain adaptation with adversarially trained language models. *arXiv preprint arXiv:2010.01739*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Andrew Wilson and Hannes Nickisch. 2015. Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1775–1784, Lille, France. PMLR.

Xinjie Yu and Mitsuo Gen. 2010. *Introduction to evolutionary algorithms*. Springer Science & Business Media.

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*, volume 32. Curran Associates, Inc.

A Appendix: Gaussian process details

Gaussian processes. A GP is a stochastic process, $f(\psi) : \psi \in \Psi$, such that for any finite set of elements $\psi_1, \cdots, \psi_k \in \Psi$, the associated finite collection of random variables $f(\psi_1), \cdots, f(\psi_k)$, has a multivariate Gaussian distribution (Rasmussen and Williams, 2005).

A GP $f(\psi) \sim GP(\mu(\cdot), k(\cdot, \cdot))$ can be understood as a probability distribution over arbitrary functions, with $\mu(\psi) = \mathbb{E}[f(\psi)]$ its mean function, and $k(\cdot, \cdot)$ the covariance kernel, i.e., $k(\psi, \psi') = \mathbb{E}[(f(\psi) - \mu(\psi))^\top (f(\psi') - \mu(\psi'))]$.

The mean and kernel functions determine the GP function class: i.e., the regularity and smoothness assumptions of the modeled data. These are parameterized prior-functions $\mu(\cdot|\theta_\mu)$ and $k(\cdot, \cdot|\theta_k)$, which can be fitted to the observed data $r_{1:T} = (r_1, \cdots, r_T)$ at inputs $\psi_{1:T} = (\psi_1, \cdots, \psi_T)$. For instance, via Type-II maximum likelihood estimation (MLE) of the GP model's hyperparameters $\theta = (\theta_\mu, \theta_k)$, $\hat{\theta} = \mathrm{argmax}_\theta \log p\left(r_{1:T}|f(\psi_{1:T}|\theta)\right)$, where the data likelihood $p(r|f(\cdot;\theta))$ is a function of the observation noise's probability distribution. Bayesian approaches to hyperparameter selection for GP model training can also be implemented (Rasmussen and Williams, 2005).

Gaussian process posteriors. Given a fitted GP, posterior inference —computing the predictive distribution of a new datapoint ψ' after observing $\psi_{1:T}$— can be performed in closed form for the Gaussian observation noise case. For example, when the noise in Equation (7) is i.i.d. drawn from $\epsilon_t \sim \mathcal{N}\left(\epsilon|0, \sigma_\epsilon^2\right)$.

Formally, given a set of observations $r_{1:T}$ at inputs $\psi_{1:T}$, the posterior distribution of f is a GP with the following mean and covariance functions:

$$\begin{aligned} \mu_T(\psi) &= k_T(\psi)^\top (K_T + \sigma_\epsilon^2 I)^{-1} r_{1:T} \ , \\ k_T(\psi, \psi') &= k(\psi, \psi') \\ &\quad - k_T(\psi)^\top (K_T + \sigma_\epsilon^2 I)^{-1} k_T(\psi') \ , \end{aligned} \tag{8}$$

$$\text{with} \begin{cases} k_T(\psi) = (k(\psi_1, \psi), \cdots, k(\psi_T, \psi))^\top \ , \\ K_T = (k(\psi, \psi'))_{\forall \psi, \psi' \in \psi_{1:T}} \ . \end{cases} \tag{9}$$

These closed-form posterior inference expressions can be efficiently computed, both in exact and approximate ways (Rasmussen and Williams, 2005; Pleiss et al., 2018).

Posterior inference with observation noise beyond the Gaussian assumption is an active research area, with many approximate techniques available for practitioners (Snelson and Ghahramani, 2006; Titsias, 2009; Wilson and Nickisch, 2015; Flaxman et al., 2015).

B Appendix: Implementation and experimentation details

B.1 Gaussian process

We implement Gaussian process modules based on MIT Licensed GPyTorch (Gardner et al., 2018), and execute all experiments with a GP process prior and GP fitting details as described in Table 3.

Table 3: Gaussian Process prior and hyperparameters.

Hyperparameter	Initial Value
GP Model	
Mean Function	Constant
Prior constant	0
Kernel Function	Scaled RBF Kernel
Prior output-scale	1
Prior length-scale	0.25
Observation Model	
Likelihood function	Gaussian
Noise variance	1
Training details	
Loss function	ExactMarginalLogLikelihood
train max iters	100
loss epsilon	0.01
Optimizer	
optimizer	adam
lr	0.1

We take the most conservative approach on GP-TS prior and hyperparameter selection: we utilize an uninformative prior, with no preference for any hyperparameter configuration. This is the less assuming yet more challenging experimental set-up, where we evaluate whether GP-TS can successfully learn —without any prior knowledge— to find good hyperparameters.

Based on bandit theory and practice, informative priors can accelerate convergence if properly specified (i.e., when more mass is put into favorable regions of the hyperparameter space); while slowing down convergence, if incorrectly specified (i.e., when mass is put in unfavorable regions of the space). Evaluating how different priors affect GP-TS are experiments left as future work.

B.2 RoBERTa pre-training

We pre-train all MIT-licensed RoBERTa models as provided by Ott et al. (2019), with the BERT-base architecture of 125M parameters, by minimizing the MLM loss with dynamic masking in a server with 8 Tesla V100-SXM2-32GB GPUs.

We execute the RoBERTa pre-training procedure as described in Fairseq's RoBERTa pre-training tutorial[8], with specific hyperparameters as described in Table 4.

The interactions for `wiki-c4` and `e-commerce` contain 1000 updates each (i.e., $u = 1000$), while we reduce the number of updates per-interaction to $u = 500$ when pre-training with `mimic` notes.

Table 4: RoBERTa pre-training hyperparameters.

Hyperparameter	Value
Architecture	RoBERTa base
Task	masked lm
Criterion	masked lm
Model details	
dropout	0.1
attention-dropout	0.1
weight-decay	0.01
Training details	
batch-size	32
update-freq	16
sample-break-mode	complete
tokens-per-sample	512
Optimizer	
optimizer	adam
adam-betas	(0.9,0.98)
adam-eps	1e-6
clip-norm	1.0
Learning rate	
lr	0.0005
lr-scheduler	polynomial decay
linear-warmup-updates	1000
Dynamic masking	
mask-prob	ρ
leave-unmasked-prob	0.1
random-token-prob	0.1

B.3 Summary statistics of the computational cost

We describe in Table 5 summary statistics on the execution time of GP-TS pre-training in our experiments, as per details in Section B.2. The per-interaction, average execution time of pre-training is: 33,316 seconds for the `wiki-c4` dataset; 37,392 seconds for the `e-commerce` data; and 1,489 seconds for `MIMIC` notes. It only takes about 20 seconds on average to execute GP-TS per-interaction. Hence, the overhead is of 0.05% for the biggest dataset, and 1% for the smallest one. We note that the TLM pre-training implementation of Ott et al. (2019) leverages GPU computations, while GP-TS is executed within a single CPU —with no GPU acceleration.

Table 5: Per-interaction execution time of TLM pre-training and GP-TS: average time in seconds, plus-minus the standard deviation.

Dataset	Execution time in seconds		
	TLM Pre-training	GP-TS ρ	GP-TS ψ
`wiki-c4`	$33,316 \pm 395\ s$	$19 \pm 6\ s$	$21 \pm 6\ s$
`mimic`	$1489 \pm 46\ s$	$16 \pm 5\ s$	$17 \pm 5\ s$
`e-commerce`	$37,392 \pm 494\ s$	$21 \pm 3\ s$	$23 \pm 10\ s$

B.4 Summary statistics of the pre-training datasets

We split each pre-training dataset into 80%-10%-10% training, validation and test sets for our experiments, with summary statistics of each set provided in Table 6.

[8]Available at `https://github.com/pytorch/fairseq/blob/main/examples/roberta/README.pretraining.md`

Table 6: Summary statistics of the pre-training datasets.

Dataset		Total word count	Average words per sentence
wiki-c4	Training	4,517,625,794	35.9
	Validation	735,950,955	35.6
	Test	735,571,833	35.6
mimic	Training	402,720,632	216.7
	Validation	82,340,235	658.7
	Test	18,735,884	187.3
e-commerce	Training	3,935,845,017	5.6
	Validation	494,802,278	5.5
	Test	482,733,197	5.5

B.5 RoBERTa fine-tuning

The specific RoBERTa hyperparameters used for the in-domain fine-tuning downstream tasks are described in Tables 7–10.

Table 7: RoBERTa fine-tuning hyperparameters for the e-commerce title classification downstream task.

Hyperparameter	Value
Architecture	RoBERTa base
Task	
Task	sentence prediction
Criterion	sentence prediction
num-classes	2
max-positions	512
init-token	0
separator-token	2
Model details	
dropout	0.1
attention-dropout	0.1
Dataset	
batch-size	32
update-freq	1
required-batch-size-multiple	1
max-tokens	4400
skip-invalid-size-inputs-valid-test	True
Optimizer	
optimizer	adam
weight-decay	0.1
adam-betas	(0.9,0.98)
adam-eps	1e-6
Learning rate	
lr-scheduler	polynomial decay
lr	1e-5
linear-warmup-updates	1000
max-updates	100000
max-epoch	10
clip-norm	0.0

Table 8: RoBERTa fine-tuning hyperparameters for the e-commerce title similarity downstream task.

Hyperparameter	Value
Architecture	RoBERTa base
Task	
Task	sentence prediction
Criterion	sentence prediction
num-classes	2
max-positions	512
init-token	0
separator-token	2
Model details	
dropout	0.1
attention-dropout	0.1
Dataset	
batch-size	32
update-freq	1
required-batch-size-multiple	1
max-tokens	4400
skip-invalid-size-inputs-valid-test	True
Optimizer	
optimizer	adam
weight-decay	0.1
adam-betas	(0.9,0.98)
adam-eps	1e-6
Learning rate	
lr-scheduler	polynomial decay
lr	1e-5
linear-warmup-updates	1000
max-updates	100000
max-epoch	10
clip-norm	0.0

B.6 Summary statistics of the fine-tuning datasets

We split each per-task fine-tuning dataset into training, development and test sets for our experiments, with summary statistics of each set provided in Table 11.

Table 9: RoBERTa fine-tuning hyperparameters for the e-commerce title quality downstream task.

Hyperparameter	Value
Architecture	RoBERTa base
Task	
Task	sentence prediction
Criterion	sentence prediction
num-classes	2
max-positions	512
init-token	0
separator-token	2
Model details	
dropout	0.1
attention-dropout	0.1
Dataset	
batch-size	32
update-freq	1
required-batch-size-multiple	1
max-tokens	4400
skip-invalid-size-inputs-valid-test	True
Optimizer	
optimizer	adam
weight-decay	0.1
adam-betas	(0.9,0.98)
adam-eps	1e-6
Learning rate	
lr-scheduler	polynomial decay
lr	1e-5
linear-warmup-updates	1000
max-updates	100000
max-epoch	10
clip-norm	0.0

Table 10: RoBERTa fine-tuning hyperparameters for the medical MLI downstream task.

Hyperparameter	Value
Architecture	RoBERTa base
Task	
Task	sentence prediction
Criterion	sentence prediction
num-classes	3
max-positions	512
init-token	0
separator-token	2
Model details	
dropout	0.1
attention-dropout	0.1
Dataset	
batch-size	32
update-freq	1
required-batch-size-multiple	1
max-tokens	4400
skip-invalid-size-inputs-valid-test	True
Optimizer	
optimizer	adam
weight-decay	0.1
adam-betas	(0.9,0.98)
adam-eps	1e-6
Learning rate	
lr-scheduler	polynomial decay
lr	1e-5
linear-warmup-updates	1000
max-updates	100000
max-epoch	10
clip-norm	0.0

Table 11: Summary statistics of the fine-tuning task datasets.

Dataset		Total sentence count	Average words per sentence Input0 – Input1
`e-commerce title classification & similarity`	Training	224,745	10.9 – 10.9
	Dev	6,035	10.9 – 10.8
	Test	12,311	10.9 – 10.8
`e-commerce title quality`	Training	49,420	10.6 – NA
	Dev	2,629	9.8 – NA
	Test	5,174	9.8 – NA
`medical MLI`	Training	11,232	15.9 – 5.5
	Dev	1,395	16.9 – 5.4
	Test	1,422	15.4 – 5.4

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
We provide a dedicated section named "Limitations" in page 9, right after Conclusions.

☑ A2. Did you discuss any potential risks of your work?
We have discussed ethical and societal considerations in the section entitled "Ethics Statement" right after the "Limitations" section in page 9.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
The abstract succinctly summarizes our work, with specific contributions highlighted at the end of the introduction section.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Our work uses and creates scientific artifacts:
- We design GP-TS in Section 3, and provide its implementation details in Appendix B.
- We use RoBERTa models for our experiments of Section 4, with implementation and configuration details provided in Appendix B.

☑ B1. Did you cite the creators of artifacts you used?
- We use (and cite) the GPyTorch package used in our GP-TS implementation, as explained in Section 4, with explicit configuration details provided in Appendix B.
- We use RoBERTa language models (with citations to the original article and a reference to the codebase we implement included in the manuscript) for our experiments in Section 4, with all configuration details provided in Appendix B.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Used artifacts are MIT licensed, as explained in Appendix B.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Our use of the artifacts is consistent with their intended use, as per licensed detailed in Appendix B.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
All used datasets were previously anonymized, as explained in the provided citations of each dataset.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
We provide a description of the datasets in Section 4, with details in Appendix B.
No demographic information is available for the used datasets (they have been previously anonymized).

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Detailed relevant statistics are provided in Appendix B.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Computational experiments are described in Section 4, with further implementation and configuration details provided in Appendix B.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B contains all the implementation and configuration details of the computational experiments.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 describes the MLM hyperparameter search executed.
Appendix B contains all other hyperparameter configuration details, for which no search was executed.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4 contains all the statistics that support our findings: we clearly indicate results in Figures shown are for a single realization, and describe results for multiple seeds in Table 1.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B describes our implementation and the packages used.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.