Next-Word Prediction: A Perspective of Energy-Aware Distributed Inference

Shangshang Wang, *Graduate Student Member, IEEE*, Ziyu Shao, *Senior Member, IEEE*, and John C.S. Lui, *Fellow, IEEE*

***Abstract*—The pursuit of high-quality artificial intelligence generated contents (AIGC) with fast response has prompted the evolution of natural language processing (NLP) services, notably those enabled at the edge (i.e., edge NLP). For concreteness, we study distributed inference for next-word prediction which is a prevalent edge NLP service for mobile keyboards on user devices. Accordingly, we optimize coupled metrics, i.e., maximize prediction click-through rate (CTR) for improved quality-of-service (QoS), minimize user impatience for enhanced quality-of-experience (QoE), and keep energy consumption within budget for sustainability. Moreover, we consider the real-world setting where there is no prior knowledge of heterogeneous NLP models' prediction accuracy. Via an integration of online learning and online control, we propose a novel distributed inference algorithm for online next-word prediction with user impatience (DONUT) to estimate models' prediction accuracy and balance the trade-offs among coupled metrics. Our theoretical analysis reveals that DONUT achieves sub-linear regret (loss of CTR), ensures bounded user impatience, and maintains within-budget energy consumption. Through numerical simulations, we not only establish DONUT's superior performance over other baseline methods, but also demonstrate its adaptability to various settings.**

***Index Terms*—Distributed inference, edge natural language processing, next-word prediction, online learning, online control.**

I. Introduction

AMIDST the artificial intelligence (AI) revolution, various user-centric applications have evolved to incorporate natural language processing (NLP) services to produce high-quality AI-generated contents (AIGC) [1]. This fusion allows machines to understand human languages, making AI more accessible and shaping a future where interactions with AI is as intuitive as human dialog. Traditionally, end users seeking access to NLP services have primarily relied on powerful NLP models located on cloud servers. Despite the high accuracy, cloud-based services may incur high latency and suffer from intermittent wireless connections, which makes it hard to maintain real-time guarantees [2]. A promising alternative to this is *edge NLP*, which facilitates NLP services on the edge servers. Thanks to the rich computational resources on edge servers and their proximity to user devices, end users can enjoy NLP services of both high quality and fast response [3], [4].

For edge NLP, the majority of works fall within the scope of distributed training, focusing on "how to train NLP models across edge servers to facilitate NLP services?" [12] In this work, we pivot our focus to a different approach by asking how to utilize trained NLP models (e.g., N-gram [5], BERT [7], and GPT [11]) deployed on edge servers for edge NLP? This approach is termed as distributed inference as we aim to exploit the inference capability of trained models across geographically distributed edge servers. For concreteness, as shown in Fig. 1, we study distributed inference for next-word prediction – a typical and non-trivial NLP service for end users. The goal of next-word prediction is to improve users' input efficiency by proactively suggesting the most probable next-words based on inputs from mobile keyboards [13].

An insightful observation in distributed inference is the possibility of leveraging the inference capability from multiple trained models, rather than restricting to just one. Intuitively, by distributing user input across various NLP models, we aggregate diverse predicted next-words, offering a wider selection candidates to the user, thereby enhancing the next-word prediction

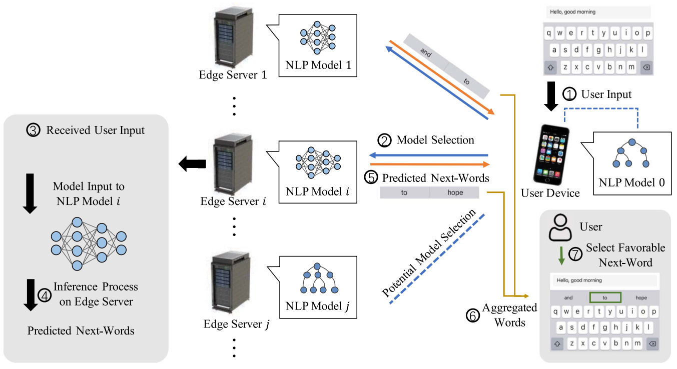


Fig. 1. Illustration of edge NLP for next-word prediction. 1) The user input is generated at the user device; 2) The user device selects a subset of NLP models; 3) Received user input is used as the model input; 4) Inference is performed on each selected model; 5) Predicted next words from selected NLP models are sent to the user device; 6) The user device aggregates the words by removing duplicates and presents the aggregated words to the user; 7) The user selects one favorable predicted word.

Manuscript received 27 February 2023; revised 4 July 2023; accepted 28 August 2023. Date of publication 31 August 2023; date of current version 4 April 2024. This work was partially supported in part by the Natural Science Foundation of Shanghai under Grants 22ZR1441700 and RGC GRF 14200321. Recommended for acceptance by H. Shen. *(Corresponding author: Ziyu Shao.)*

Shangshang Wang and Ziyu Shao are with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: wangshsh2@shanghaitech.edu.cn; shaozy@shanghaitech.edu.cn).

John C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3310536, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3310536

TABLE I
COMPARISON OF INFERENCE CAPABILITY AMONG VARIOUS NLP MODELS IN TERM OF DIFFERENT FACTORS

NLP Model Name	NLP Model Type	Prediction Accuracy	Inference Latency	Energy Consumption
N-gram [5]	Statistical Model	Low	Low	Low
ELMo [6]	Neural Model	Middle	High	High
BERT [7]	Neural Model	High	High	High
RoBERTa [8]	Neural Model	High	High	Middle
ALBERT [9]	Neural Model	High	High	Middle
DistilBERT [10]	Neural Model	Middle	Middle	Middle
GPT-3 [11]	Neural Model	High	High	High

service. This idea is well justified in the field of ensemble learning [14] where the adoption of a mixture-of-experts (MOE) is a common practice, often resulting in superior performance compared to any single expert model [15]. Nevertheless, the user device cannot interactive with an unlimited number of NLP models given increasing overheads (e.g., latency and energy consumption). This implies a practical model selection problem, i.e., which subset of trained NLP models should the user device select for distributed inference. This model selection problem is challenging and non-trivial due to the following two reasons.

First, model selection decision depends on NLP models' inference capability which is evaluated by various factors [16]. Specifically, we consider three factors, i.e., prediction accuracy, inference latency, and energy consumption, which tend to vary among different NLP models. As shown in Table I, although N-gram generally incurs lower inference latency and lower energy consumption than Transformer-based models [17], its prediction accuracy tends to be lower. Furthermore, the prediction accuracy of deployed NLP models is usually user-dependent and unknown a prior since such models are often not tailored to specific user. In fact, NLP models deployed on edge servers include a wide range of pre-trained models [18], whose training corpus may vary largely from users' input corpus. To estimate the prediction accuracy, an online learning procedure is required to collect feedback via dynamic selections of different models.

Second, model selection problem involves coupled metrics, i.e., quality-of-service (QoS), quality-of-experience (QoE), and sustainability of user devices. For QoS, a common criterion is the prediction click-through rate (CTR), i.e., the rate of whether predicted words are accepted by the end user. Intuitively, models with high prediction accuracy may imply high prediction CTR and high QoS. For QoE, we characterize it with user impatience which is caused by overdue predictions (i.e., latency of next-word prediction exceeds the maximum tolerance latency). Consequently, models with high inference latency may induce high user impatience and low QoE. For sustainability, we should maintain within-budget energy consumption for user devices with limited battery capacity. Accordingly, models with high energy consumption may tend to exceed the energy budget and threat the sustainability. To balance among these coupled metrics, an online control procedure is needed to adaptively adjust their relative importance.

Based on the above discussion, the model selection problem is essentially a sequential decision-making problem under uncertainty, which aligns with the setting of bandit learning [19]. However, most bandit learning methods cannot be directly adopted since the model selection decision requires an effective integration of online learning (to estimate prediction accuracy) and online control (to balance among coupled metrics). On one hand, ineffective online learning with inaccurate estimation could potentially misguide subsequent control decisions, leading to biased selection of sub-optimal NLP models. On the other hand, improperly executed control decisions can result in poor feedback and hinder learning efficiency, not to mention the difficulty in balancing among multiple objectives. For instance, if certain models are infrequently selected, the user device may underestimate their performances, thereby overlooking potential optimal choices.

In this paper, we resolve the key model selection problem with a novel distributed inference algorithm for online next-word prediction with user impatience, denoted as DONUT. The main contributions and the key results of our work are summarized as follows:

- *Modeling:* We investigate the model selection problem in distributed inference for next-word prediction. Our objective is to maximize prediction CTR for high QoS and minimize user impatience for high QoE under constrained energy consumption (Section III). Besides next-word prediction, our model can be extended to support other edge NLP services like emoji prediction [20]. Our approach to edge NLP also offers insights for the future deployment of services like ChatGPT-based dialog systems [11] and Codex-based auto-programming tools [21] at the edge.
- *Algorithm Design:* From the viewpoint of constrained bandit learning, we propose DONUT to solve the model selection problem via an integration of online learning and online control (Section IV).
- *Theoretical Analysis:* We show that DONUT achieves a time-averaged regret $O(1/V + \alpha/V + \sqrt{\log T/T})$ and a bounded time-averaged total user impatience $O(1/\alpha + V/\alpha)$ where T is the number of rounds and V, α are finite tunable parameters. Besides, DONUT also guarantees the energy consumption constraint for the user device (Section V).
- *Simulation Results:* We conduct extensive simulations to evaluate the effectiveness of DONUT. They not only verify our theoretical analysis but also demonstrate the outperformances of DONUT over other online algorithms (Section VI).

II. Related Works

A. Edge Natural Language Processing

The proliferation of edge computing has significantly amplified the potential of utilizing edge-based resources to deliver user-centric services. Riding this trend, edge NLP has emerged to offer high-quality and real-time services such as next-word prediction, emoji prediction, text classification, speech recognition, *etc* [22]. In the following, we primarily discuss related works on the next-word prediction service.

Most existing works focus on the design and improvement of training new NLP models directly at the edge (i.e., federated learning) either from scratch or via fine-tuning pre-trained models [12]. Work [13] utilizes federated learning to train a lightweight NLP model called CIFG (i.e., Coupled Input-Forget Gate) for next-word prediction. Work [23] further addresses the out-of-vocabulary problem with a character-level recurrent neural network based on the design of CIFG. Work [24] considers the training of personalized models, in which three types of algorithm (i.e., hypothesis-based clustering, data interpolation, and model interpolation) are proposed to meet various practical needs. Work [25] proposes to combine advanced model aggregation and attention mechanism to training personalized models for next-word prediction. Work [26] combines centralized model pre-training and pretrained word embeddings with federated fine-tuning to enhance the performance of next-word prediction.

All above works of federated learning center on high-performance distributed model training. They address the question: "how to train new NLP models in a distributed manner for edge NLP". Our work takes a different angle to facilitate edge NLP. Specifically, we aim to exploit the inference capability of trained models at the edge, i.e., distributed inference. Therefore, our focus is: "how to utilize trained NLP models in a distributed manner for edge NLP" [27]. Through distributed inference, user devices can leverage heterogeneous trained NLP models at the edge, supporting both next-word prediction service of high QoS and QoE with sustainability.

B. Distributed Inference at the Edge

Beyond edge NLP, distributed inference has emerged as a promising approach for providing high-quality and real-time applications with trained models at the edge.

Existing distributed inference research largely revolves around partitioning a single trained model to blocks of layers. Their aim is to distribute inference overheads, such as computational costs and energy consumption, to edge servers or devices in a layer-wise manner [28]. Work [29] adaptively partitions neural models between edge servers and devices, in order to leverage computation resources in proximity for real-time inference. Work [30] studies directed acyclic graph (DAG) structured neural models and partitions their layers into independent execution units to improve inference latency. Work [31] focuses on partitioning and deploying neural models in edge clusters, optimizing both inference latency and throughput. Applications based on model partitioning have also begun to emerge. Work [32] studies the partition-based distributed inference for real-time video analysis in a resource-constrained Internet-of-Things system. Work [33] performs model partition to accelerate distributed inference for video streams at the edge.

In our work, we adopt a different approach by utilizing multiple instead of one trained models deployed at the edge. Specifically, via model selection, we aim to aggregate prediction results in a mixture-of-expert approach to potentially enhance the system performance.

III. System Model & Problem Formulation

In this section, we present the system model and problem formulation of distributed inference for next-word prediction. Specifically, we focus on the key model selection problem where we simultaneously optimize Quality-of-Service (QoS), Quality-of-Experience (QoE), and sustainability.

In this paper, we focus on a simple but non-trivial setting. Specifically, we consider an edge NLP system with $(N-1)$ edge servers that provides the next-word prediction service to *a single user*. Each edge server and the user device is deployed with *only one trained NLP model*. This implies a total of N models with heterogeneous inference capability in terms of prediction accuracy, inference latency, and energy consumption. We index these models by the set $\mathcal{N} \triangleq \{0, 1, \ldots, N-1\}$ and the model on the user device is indexed by zero. We assume that the system operates on a round[1] basis with a fixed time horizon, indexed by $t \in \{0, \ldots, T-1\}$.

Given the high sensitivity to latency in typical edge NLP services, it's essential to account for the end-to-end latency to access each model for inference [34]. Specifically, the end-to-end latency for the local model situated on the user device refers to the model's inference latency. For models deployed on an edge server, the end-to-end latency is a composite of the model's inference latency and the transmission latency, which includes the time taken to send prediction requests and receive prediction results between the user device and edge server.[2]

We assume the corresponding inference latency and transmission latency can be measured before the execution of model selection decisions. For example, for inference latency of NLP models [36], it can be estimated either via data-driven approach with pre-trained machine learning models on various hardware configurations [37], [38] or through analytical performance modeling given prior knowledge on the NLP model [39]. For transmission latency of wireless communication between the user device and edge servers, it can be measured by probing wireless channels using test packages [40], [41] or via latency prediction with a modeling of channel's communication protocol (e.g., WiFi [42]).

[1]Round is the discrete-time interval for the user device to make model selection decisions and receive prediction results from selected NLP models. In practice, the time length of a round is often fixed and determined by the maximum tolerance latency.

[2]The latency of edge-device communication which in our case is considered as the one-hop (i.e., direct) communication. Such a fact is adopted as a common practice in edge computing where the edge server and user devices are in proximity of each other [35].

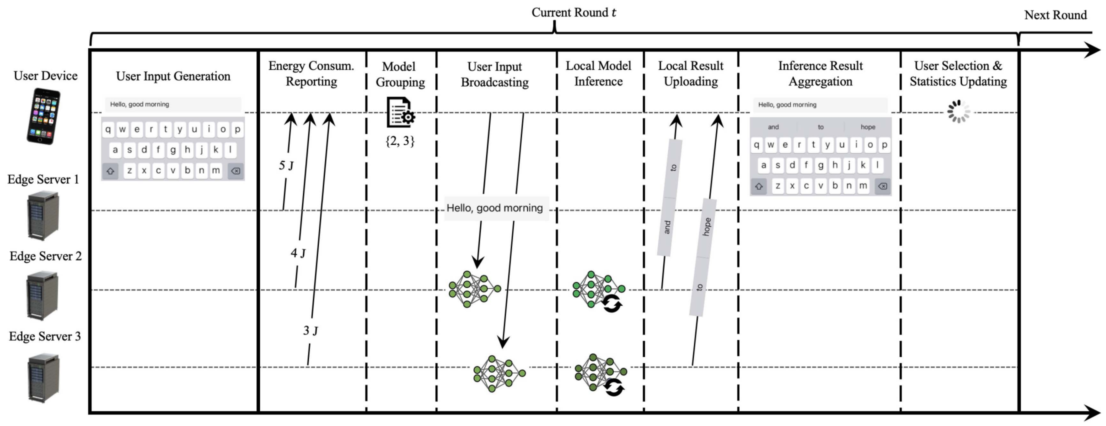


Fig. 2. Workflow of distributed inference for next-word prediction at the edge.

Therefore, based on the measurement of end-to-end latency, we observe whether NLP model i can perform in-time prediction, denoted by a binary random variable $s_i(t)$. Specifically, $s_i(t) = 1$ if NLP model i can complete the prediction by the end of round t (i.e., the end-to-end latency falls within the maximum tolerance latency) and zero otherwise (i.e., overdue prediction). We assume $s_i(t)$ is independent and identically distributed (*i.i.d.*) over rounds.

A. Distributed Inference for Next-Word Prediction

In each round t, we conduct distributed inference in the edge NLP system to provide the next-word prediction service for the user (please refer to Fig. 2). Specifically, the system operates according to the following steps:[3]

1) The user input is first generated on the mobile keyboard of the user device.
2) The user device then executes model selection decisions for a subset of trained NLP models on edge servers and itself.
3) The user device sends the user input to such selected models either locally via hardware-level transmission or remotely via wireless transmission to the corresponding edge servers.
4) Each selected model conducts an inference process with the user input as its model input. Each selected model then returns the top-k predicted next words to the user device.[4]
5) The user device aggregates the predicted next words from all selected NLP models by removing duplicates and then presents the aggregated words to the user on the mobile keyboard.
6) The user selects some favorable predicted next word to assist his keyboard input.

[3]We do not emphasize the security and privacy during the transmission of user input and prediction results. Nonetheless, a series of techniques can be effectively integrated into our case [43]. For example, text encryption and decryption can be combined with the message sending and receiving steps in our workflow to better protect users' personal information during the server-device interactions.

[4]In practice, the value of k ranges from 2 to 3 for user devices. Related simulation results are shown in Section VI.

B. Model Selection in Distributed Inference

We define the number of selected models in each round by the user device as the model selection number, which is denoted as a fixed number n $(1 \leq n \leq N)$.[5] We use the binary vector $\boldsymbol{d}(t) \triangleq \{d_i(t)\}_{i\in\mathcal{N}}$ to denote model selection decisions in round t, where $d_i(t) = 1$ if model i is selected by the user device and zero otherwise. Accordingly, we denote the set of selected models by $S(t)$ for convenience, $S(t) \in F(\mathcal{N})$ where $F(\mathcal{N})$ is the set of all subsets of $\mathcal{N}$ with cardinality less or equal to n, i.e., $F(\mathcal{N}) \triangleq \{S \subseteq \mathcal{N} | |S| \leq n\}$. Main notations of our work are summarized in Table II.

C. Quality-of-Service: Click-Through Rate

We characterize QoS with the prediction click-through rate (CTR) [13]. Specifically, in each round t, the prediction CTR is defined as follows:

$$\mathrm{CTR}(t) \triangleq \frac{1}{n} \sum_{i\in S(t)} a_i(t) s_i(t) d_i(t), \tag{1}$$

where we denote the click-through indicator for NLP model i as $a_i(t)$ such that $a_i(t) = 1$ if any of the prediction results of NLP model i is selected by the user and zero otherwise. We assume $a_i(t)$ is *i.i.d.* which follows a Bernoulli distribution with an unknown mean $\mu_i \in (0, 1)$ and μ_i is interpreted as the prediction accuracy of model i.

Therefore, to optimize QoS for next-word prediction, we maximize the time-averaged prediction CTRs over T rounds. Accordingly, we have the following objective:

$$\underset{\{\boldsymbol{d}(t)\}_t}{\text{maximize}} \ \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[\mathrm{CTR}(t)\right]. \tag{2}$$

[5]Our system model can be readily extended to settings where the model selection number is at most n, which is not fixed as a constant but is flexible and upper bounded by n. Comparison between these two settings is shown by simulations in Section VI.

TABLE II
SUMMARIZATION OF MAIN NOTATIONS

Notation	Description
T	Number of rounds for distributed inference for next-word prediction.
n	Model selection number, *i.e.*, number of selected NLP models in each round.
N	Total number of NLP models.
$\mathcal{N}$	Set of NLP models on both the user device and edge servers with $\lvert\mathcal{N}\rvert \triangleq N$.
$S(t)$	Set of selected NLP models in round t.
$a_i(t)$	Click-through indicator for prediction results from NLP model i in round t.
$s_i(t)$	Indicator of in-time prediction for NLP model i in round t.
$d_i(t)$	Model selection decision on NLP model i in round t.
$R_i(t)$	Reward for selecting model i for next-word prediction in round t.
$R(t)$	Compound reward for the user device in round t.
$Z_i(t)$	User impatience of NLP model i in round t.
$W_i(t)$	Energy consumed on the user device for next-word prediction with NLP model i in round t.
μ_i	Prediction accuracy of NLP model i with $\mu_i \triangleq \mathbb{E}[a_i(t)]$.
p_i	Probability of in-time prediction for NLP model i with $p_i \triangleq \mathbb{E}[s_i(t)]$.
b	The energy budget on the user device.

D. Quality-of-Experience: User Impatience

We adopt user impatience to characterize the QoE [44]. Specifically, the user impatience of an NLP model accumulates due to infrequent selection and overdue prediction, i.e., the end-to-end latency exceeds the maximum tolerance latency. Formally, for each NLP model i, we denote the associated user impatience in round t as $Z_i(t)$. In each round, the user impatience of model i increases by one if model i is not selected by the user or the prediction of model i is overdue; otherwise, the user impatience resets to one. Intuitively, this reset means that the user's demand on prediction is satisfied in time and his accumulated impatience is eliminated. Accordingly, the update rule of user impatience[6] is defined as follows:

$$Z_i(t+1) = \begin{cases} Z_i(t)+1, & s_i(t)d_i(t)=0, \\ 1, & s_i(t)d_i(t)=1. \end{cases} \tag{3}$$

The initial value is $Z_i(0)=0$.

Therefore, one essential criterion for high QoE is to ensure low user impatience and we minimize the time-averaged total user impatience over a give horizon. Accordingly, we have the following objective:

$$\underset{\{\boldsymbol{d}(t)\}_t}{\text{minimize}} \; \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{N}} Z_i(t)\right]. \tag{4}$$

Remark 1. If we do not minimize the total user impatience as formulated in (4), the user device would stick to NLP models of low user impatience and NLP models with high impatiences would be rarely selected, and their impatiences would become unboundedly large, and eventually they would never be selected for next-word prediction. Thereafter, the user device would always select from a small subset of NLP models. Consequently, this may lead to unbalanced utilization of computational resources among edge servers, thus low quality of predicted next-words. Moreover, such a restriction may lead to unendurable end-to-end latencies under poor wireless conditions [48].

[6]User impatience and its updates in (3) are inspired by the concept of age of information in work [45] which characterizes the freshness of information during wireless transmission [46], [47].

E. Sustainability: Energy Consumption Constraint

To maintain sustainability for user devices with limited battery capacity, we keep the energy consumption within budget for user devices. In each round t, two kinds of energy consumption are of importance, i.e., energy consumed for inference on local NLP model (denoted as $W_0(t)$), and energy spent for sending prediction requests through wireless transmission to edge server $i \in \mathcal{N}\backslash\{0\}$ (denoted as $W_i(t)$). We assume that $\{W_i(t)\}_{i\in\mathcal{N}}$ are upper-bounded by positive constant $W_{\max}$, and available at the beginning of each round t. We define the total energy consumption in round t as

$$\eta(t) \triangleq \sum_{i\in S(t)} W_i(t)s_i(t)d_i(t). \tag{5}$$

To guarantee a within-budget energy consumption, we impose the following energy consumption constraint with energy budget b ($b>0$):

$$\limsup_{T'\to\infty}\frac{1}{T'}\sum_{t=0}^{T'-1}\mathbb{E}\left[\eta(t)\right] \le b. \tag{6}$$

Remark 2. We consider the energy consumption constraint in a long-term and time-averaged sense. The reason for such a particular form is that real systems usually run for a long time, and the horizon T is unfixed or unknown in advance. By adopting "lim" with infinite rounds, our algorithm ensures the constraint in a long run without knowing the fixed round T; In this paper, guaranteeing such a long-term constraint in (6) is equivalently transformed to guarantee the stability of some particularly designed virtual queue (Section IV).

F. An Online Constrained Optimization Problem With Multiple Objectives

Considering QoS, QoE, and sustainability, we formulate the model selection problem in distributed inference as the following online constrained optimization problem with multiple objectives:

$$\begin{aligned} &\text{objectives: } (2) \ \& \ (4) \\ &\text{constraint: } (6) \end{aligned} \tag{7}$$

Remark 3. In problem (7), performance metrics (i.e., prediction CTR, user impatience, and energy consumption) are coupled via the model selection decisions $\{\boldsymbol{d}(t)\}_t$. Therefore, how to balance the relative importance among them is non-trivial and of great importance. For example, while pursuing high prediction CTR and low user impatience jointly, selecting models with high energy consumption is inevitable.

IV. Algorithm Design

In this section, we present a novel distributed inference algorithm for online next-word prediction with user impatience (DONUT) which integrates bandit learning methods and Lyapunov optimization techniques.

A. Bandit Learning With Upper Confidence Bound

In this section, we utilize bandit learning methods in online learning to maximize prediction CTR under unknown model prediction accuracy.

Under the bandit setting, we define the reward in round t as the corresponding prediction CTR, i.e., $R(t) = \mathrm{CTR}(t)$. Further, we utilize the time-averaged regret over T rounds to characterize the loss of prediction CTR during model selection. Formally, we first denote the optimal time-averaged prediction CTR over T rounds as R^* (with optimal decisions $\{\boldsymbol{d}^*(t)\}$). We then define the time-averaged regret as the difference between R^* and the time-averaged prediction CTR under $\{\boldsymbol{d}(t)\}_t$ over T rounds

$$\mathrm{Reg}(T) \triangleq R^* - \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[R(t)].$$

Therefore, objective (2) is reformulated as follows:

$$\underset{\{\boldsymbol{d}(t)\}_t}{\text{minimize}} \ \mathrm{Reg}(T). \tag{8}$$

To minimize regret (i.e., maximize prediction CTR) under the uncertainty of deployed NLP models' prediction accuracies, we exploit the principle of optimism in the face of uncertainty to conduct proactive online learning [19]. Therefore, we define for each NLP model $i \in \mathcal{N}$ in round $t \in \{1, \ldots, T-1\}$

$$\begin{aligned} N_i(t) &\triangleq \sum_{\tau=0}^{t-1} s_i(\tau)d_i(\tau), \\ \bar{\mu}_i(t) &\triangleq \sum_{\tau=0}^{t-1} \frac{a_i(\tau)s_i(\tau)d_i(\tau)}{N_i(t)}, \end{aligned} \tag{9}$$

where $N_i(t)$ denotes the number of selection times and $\bar{\mu}_i(t)$ is defined as the estimated prediction accuracy of NLP model i by round t. We define their initial values as $N_i(0) = 0, \bar{\mu}_i(0) = 0$.

The upper confidence bound (UCB) estimate[7] $\hat{\mu}_i(t)$ of prediction accuracy μ_i is defined as follows:

$$\begin{aligned} \hat{\mu}_i(t) &\triangleq \min\{\bar{\mu}_i(t) + \gamma_i(t), 1\}, \\ \gamma_i(t) &\triangleq \sqrt{\tfrac{\log t}{N_i(t)}}, \end{aligned} \tag{10}$$

where $\hat{\mu}_i(t)$ and $\gamma_i(t)$ are often referred as the exploitation and exploration terms and the corresponding initial values are $\hat{\mu}_i(0) = 0, \gamma_i(0) = 0$, respectively.

Remark 4. The term $\bar{\mu}_i(t)$ reflects the acquired knowledge about the ground true prediction accuracy, which is known as the exploitation term. The term $\gamma_i(t)$ is the corresponding confidence radius for online estimation. It represents the measurement of how the empirical estimation is close to the ground truth. Specifically, a larger confidence radius implies more uncertainty in the estimation of prediction accuracy, i.e., the corresponding NLP model is under-explored with insufficient samples for estimation. Therefore, the confidence radius is also called the exploration term, indicating how much an NLP model should be explored.

B. Lyapunov Optimization With Virtual Queue

To cope with coupled metrics of prediction CTR, user impatience, and energy consumption, we employ Lyapunov optimization techniques [51].

Specifically, we introduce a virtual queue for the user device whose queue backlog size for each round $t \in \{0, \ldots, T-1\}$ is denoted as $Q(t)$. Input and output of the queue are equal to the energy consumption $\eta(t)$ and the energy budget b, respectively. Accordingly, the queue backlog size is updated as follows:

$$Q(t+1) \triangleq \max\{Q(t) + \eta(t) - b, 0\}. \tag{11}$$

When the queue is strongly stable, i.e.,

$$\limsup_{T' \to \infty} \frac{1}{T'}\sum_{t=0}^{T'-1}\mathbb{E}[Q(t)] < \infty, \tag{12}$$

constraint (6) is satisfied. Intuitively, to achieve such stability, the mean queue input (i.e., time-averaged total energy consumption) should not be greater than the mean queue output (i.e., energy budget). Otherwise, the virtual queue will be overloaded, thereby violating the constraint (6).

To minimize user impatience, we should keep the time-averaged total user impatience bounded, i.e.,

$$\frac{1}{T'}\sum_{t=0}^{T'-1}\mathbb{E}\left[\sum_{i\in\mathcal{N}} Z_i(t)\right] < \infty, \tag{13}$$

and optimize its value as small as possible. To this end, we define the Lyapunov function as follows:

$$L(\boldsymbol{U}(t)) \triangleq \frac{1}{2}(Q(t))^2 + \alpha\sum_{i\in\mathcal{N}} Z_i(t), \tag{14}$$

[7]The design of UCB estimation $\hat{\mu}_i(t)$ follows the classical UCB1 [19]. To balance exploration and exploitation, one can also resort to other forms of estimation like MOSS (Minimax Optimal Strategy in the Stochastic case) [49] and ϵ-greedy [50]. We investigate them numerically in Section VI.

with $\boldsymbol{U}(t) \triangleq \{Q(t)\} \cup \{Z_i(t)\}_{i\in\mathcal{N}}$ and α as a positive tunable parameter. Then we apply the Lyapunov drift-plus-regret techniques to transform our problem into a series of per-round sub-problems. We use the drift-plus-regret term to characterize the per-round regret

$$\Delta\text{Reg}(t) \triangleq \sum_{i\in\mathcal{N}} \mu_i \left(d_i^*(t)s_i(t) - d_i(t)s_i(t)\right), \tag{15}$$

and variation in the queue backlog size and user impatience between consecutive rounds, i.e.,

$$\begin{aligned}\Delta_V(\boldsymbol{U}(t)) \triangleq\ & \mathbb{E}\left[L(\boldsymbol{U}(t+1)) - L(\boldsymbol{U}(t))|\boldsymbol{U}(t)\right] \\ & + V\mathbb{E}\left[\Delta\text{Reg}(t)|\boldsymbol{U}(t)\right],\end{aligned} \tag{16}$$

with positive tunable parameter V. Intuitively, the system induces less regret (higher prediction CTR), user impatience, and energy consumption given a smaller drift-plus-regret term. By Lyapunov optimization, we aim to minimize the upper bound of such a term by solving a per-round sub-problem in each round t

$$\underset{\boldsymbol{d}(t)}{\text{maximize}} \sum_{i\in\mathcal{N}} \hat{w}_i(t)s_i(t)d_i(t), \tag{17}$$

where the model utility $\hat{w}_i(t)$ is denoted as

$$\hat{w}_i(t) \triangleq V\hat{\mu}_i(t) + \alpha Z_i(t) - Q(t)W_i(t), \forall\, i \in \mathcal{N}. \tag{18}$$

The detailed transformation is shown in Appendix A, available online.

Remark 5. The model utility (18) is a weighted sum of estimated prediction accuracy, user impatience, and product of queue length and energy consumption. To maximize the total model utility as shown in (17), we intuitively prefer to execute model selection decisions on models with the following properties:

- Models possessing high estimated prediction accuracy as this intuitively leads to a high prediction CTR, which matches our objective to maximize time-averaged prediction CTR in (2).
- Models characterized by low energy consumption since this helps maintain the total energy consumption within budget and guarantee the long-term constraint in (6).
- Models associated with large user impatience. Recall the dynamics of user impatience in (3), once a model is selected, its user impatience would reset to the default value. Accordingly, selecting NLP models with large user impatience potentially results in a large reduction in the total user impatience, which aligns with our objective to minimize time-averaged total user impatience in (4).

C. Integrated Algorithm Design

With an integration of bandit learning methods and Lyapunov optimization techniques, we propose a novel algorithm called DONUT. Its computational complexity in each round is $O(Nn)$, which is attributed to model selection (lines 6 and 7 in Algorithm 1).

DONUT involves an intricate interplay between learning and control. On one hand, ineffective online learning may misguide subsequent control decisions towards biased selections of sub-optimal models. Particularly, with an imprecise estimation of models' prediction accuracies, user devices may not figure out the best models to select and optimize the performance metrics. On the other hand, wrongly performed control decisions can beget inferior feedback and disrupt learning efficiency, let alone minimizing user impatience and reducing energy consumption.

Algorithm 1: Distributed Inference for **O**nline **N**ext-Word Prediction With **U**ser ImpaTience (DONUT).

```
Input: Set of NLP models $\mathcal{N}$, model selection number $n$,
  energy budget $b$, parameters $V, \alpha$, and total rounds $T$.
Output: Selected NLP models $\{S(t)\}_t$ over $T$ rounds.
 1: Initialize $Q(0) = 0$ and $\hat{\mu}_i(0) = 0, \bar{\mu}_i(0) = 0, \gamma_i(0) =$
    $0, N_i(0) = 0, Z_i(0) = 0, i \in \mathcal{N}$.
 2: for $t = 0, 1, \ldots, T-1$ do
 3:   Observe $s_i(t)$ and $W_i(t), i \in \mathcal{N}$.
      %% Online Learning Procedure
 4:   Calculate $\hat{\mu}_i(t), i \in \mathcal{N}$ according to (10).
      %% Online Control Procedure
 5:   Calculate $\hat{w}_i(t), i \in \mathcal{N}$ according to (18).
 6:   Select NLP models $S(t)$ (break ties randomly):
          $S(t) \in \arg\max_{S\in\mathcal{F}(\mathcal{N})} \sum_{i\in\mathcal{N}} \hat{w}_i(t)s_i(t)d_i(t).$
 7:   for $i \in S(t)$ do
 8:     Send inference requests to selected model $i$,
 9:     conduct inference and receive
10:     click-through indicator $a_i(t)$.
11:   end for
    %% Update Impatience, Virtual Queue, and Statistics
    Update $Z_i(t+1), N_i(t+1), \bar{\mu}_i(t+1)$, and $Q(t+1)$
    according to (3), (9), and (11), respectively.
12: end for
```

Remark 6. The design of DONUT is not sensitive to specific characteristics of NLP models, as it only utilizes the prediction results of trained NLP models. Therefore, DONUT can be extended to other NLP tasks (e.g., emoji prediction [20]) or further to other fields like computer vision with modifications on the preprocessing of prediction results sent to user devices.

V. Theoretical Analysis

In this section, we show the theoretical results on regret, user impatience, and energy consumption. Specifically, we first justify the satisfaction of the energy consumption constraint under DONUT by showing the stability of the virtual queue. We then present the bounds of time-averaged total user impatience and time-averaged regret under DONUT, respectively. Finally, we show the effects of tunable parameters on both regret and user impatience.

We call an energy budget feasible if there exist some model selection policies under which the long-term energy consumption constraint in (6) is satisfied. We define the set of all feasible energy budgets as the maximal feasible region. Suppose the

TABLE III
ORDER OF V (LEFT) AND ORDER OF α (RIGHT) VERSUS ORDER OF TIME AVERAGES

Order of V	Total User Impatience	Regret
$O(1/\sqrt{\log T})$	$O(1/\sqrt{\log T})$	$O(\sqrt{\log T})$
$O(1)$	$O(N)$	$O(\sqrt{\log T/T})$
$O(\sqrt{T/\log T})$	$O(\sqrt{T/\log T})$	$O(\sqrt{\log T/T})$

Order of α	Total User Impatience	Regret
$O(1/V)$	$O(V^2)$	$O(1/V)$
$O(1)$	$O(V)$	$O(1/V)$
$O(V)$	$O(1)$	$O(1)$

energy budget b lies in the interior of the maximal feasible region, we have the following theorems.

A. Main Theoretical Results

Theorem 1 (Energy Consumption Constraint). Given finite parameters V and α, the virtual queue satisfies

$$\limsup_{T'\to\infty} \frac{1}{T'} \sum_{t=0}^{T'-1} \mathbb{E}\left[Q(t)\right] \leq \frac{C + N\alpha + nV}{\epsilon} < \infty,$$

where ϵ is a positive constant which satisfies that $(b-\epsilon)$ is also an interior point of the maximal feasible region; constant $C \triangleq \max\{(nW_{\max}-b)^2, b^2\}/2$ with n as the model selection number and $W_{\max}$ as the maximal energy consumption.

Theorem 2 (User Impatience). Given finite parameters V and α, the time-averaged total user impatience is bounded

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{i\in\mathcal{N}} Z_i(t)\right] \leq \left(\frac{N}{p_{\min}} + \frac{C+Vn}{\alpha p_{\min}}\right)\left(1+\frac{W_{\max}}{\epsilon}\right).$$

Proof Sketch of Theorems 1 & 2. For queue stability and bound of user impatience, we leverage Lyapunov optimization techniques to compose the proof. First, we construct a Lyapunov drift function to characterize the variation of queue backlog size and user impatience between successive rounds. Second, we define the drift-plus-regret term which integrates the objective of minimizing regret into the Lyapunov drift function, and we further utilize the finiteness and boundedness of the time-averaged queue backlog size to get an upper bound of the drift-plus-regret term. Third, we construct an auxiliary policy that solely favors minimizing user impatience by selecting NLP models with current the largest values of user impatience. With the aid of such an auxiliary policy, we derive an upper bound of the drift-plus-regret term with respect to any model selection policies. Finally, we conclude the proof using telescoping sum and conditional expectation.

The complete proof is shown in Appendix B, available online.

Remark 7. Theorem 1 shows that DONUT achieves strong queue stability given finite values of parameters V and α, which means that the constraint (6) is satisfied [51].

Remark 8. Theorem 2 highlights that the time-averaged total user impatience is bounded with $O(1/\alpha + V/\alpha)$. With proper order of V, e.g., $O(\log T)$, the overall user impatience is bounded sub-linearly.

Theorem 3 (Regret). Under DONUT, the time-averaged regret over T rounds is bounded

$$\mathrm{Reg}(T) \leq \frac{C+N\alpha}{nV} + \frac{3\,N+\pi^2 N}{3nT} + 4\sqrt{\frac{\log T}{nT}}.$$

Proof Sketch of Theorem 3. First, we introduce the notion of per-round regret to measure the difference between the expected optimal compound reward and reward under DONUT. Then, we define the optimal policy (i.e., the oracle policy given the true prediction accuracies). Such a policy is a model selection policy that achieves the optimal total prediction CTR while ensuring the energy consumption constraint. Based on such a policy, we derive an upper bound of the drift-plus-regret term. Furthermore, we divides such an upper bound into three sub-terms by leveraging an auxiliary policy that solves a sub-problem similar to the one defined in (17), expect that the UCB term $\hat{\mu}_i(t)$ in model utility $\hat{w}_i(t)$ (defined in (18)) is replaced with the true prediction accuracy μ_i for each NLP model $i\in\mathcal{N}$. Next, we define two special kind of helper events, i.e., event $\{\hat{\mu}_i(t)-\mu_i \geq 0\}$ and event $\{\hat{\mu}_i(t)-\mu_i \leq \gamma_i(t)\}$ for each NLP model $i\in\mathcal{N}$. We then can eliminate the parts in the three sub-terms that are related to the complementary events of the helper events. Finally, we exploit Chernoff-Hoeffding bound and Jensen's inequality to bound each sub-term to complete the proof.

The complete proof is shown in Appendix C, available online.

Remark 9. Theorem 3 shows that DONUT achieves a regret at order $O(1/V + \sqrt{\log T/T})$ which is sub-linear with suitable values of V. For example, when the value of V is at order $\Omega(\sqrt{T/\log T})$, the resulting total regret is sub-linear with order $O(T\log T)$ which matches with the lower bound $\Omega(\sqrt{T})$ up to the logarithmic factor [52]. This fact shows the sharpness of our regret bound in online learning.

Remark 10. In the regret upper bound, the first term is incurred during the online control procedure, mostly attributed to the balancing of trade-offs among coupled metrics; the last two terms are due to the online learning procedure. Regarding necessary exploration in the face of unknown latencies, a sub-linear time-averaged regret $O(\sqrt{\log T/T})$ is accumulated during such a procedure.

B. Relationship Between User Impatience & Regret

Both the time-averaged total user impatience and time-averaged regret are affected by tunable parameters V and α (shown in Theorems 2 and 3, respectively). To better illustrate the effects of such parameters on the time averages, we compare them with different V and α in Table III.

In the left part of Table III, with other parameters fixed, an interesting observation is that DONUT achieves the same order of sub-linear regret $O(\sqrt{T/\log T})$ while retaining different orders of user impatience given parameter V ranging in $[O(1), O(\sqrt{T/\log T})]$; In the right part of Table III, with other parameters fixed, we can see that DONUT can maintain the

TABLE IV
SUMMARIZATION FOR DEFAULT SIMULATION SETTINGS

Parameter	Setting
Prediction Accuracies	$[5.16\%, 8.99\%, 9.45\%, 7.88\%, 8.56\%, 8.13\%, 9.37\%, 3.39\%, 9.30\%, 19.06\%, 21.57\%, 10.38\%, 11.53\%, 7.52\%, 11.33\%]$
Probabilities of In-time Prediction	$[0.95, 0.75, 0.76, 0.71, 0.73, 0.85, 0.88, 0.83, 0.75, 0.84, 0.72, 0.86, 0.82, 0.71, 0.8]$
Energy consumption	$[14, 7, 7.5, 8, 9, 7, 8.5, 8, 8.5, 11, 12, 9, 8, 7.5, 7]$

same order of regret while achieving different orders of user impatience given parameter α ranging in $[O(1/V), O(1)]$.

VI. SIMULATION RESULTS

Our simulations run on a server with the operating system as Ubuntu 20.04, CPU as Intel(R) Xeon(R) Gold 6230, and GPU as Quadro GV100 32 GB. Each simulation result is averaged over 30 independent trials. We consider an edge NLP system for next-word prediction with one user device and 14 edge servers, i.e., 15 NLP models. By default, the number of operating rounds T is set as 10^5 and the values of parameters V and α are set as 300 and 1, respectively.

The user device is deployed with a DistilBERT [10] model. For the servers, we adopt three different types of NLP models: BERT (indexed from 1 to 8), RoBERTa [8] (indexed by 9 and 10), ALBERT [9] (indexed from 11 to 14). All the above models are pre-trained NLP models from the Transformers library [53] and their prediction accuracies range from 3% to 30%. The energy consumption of the user device for model selection on NLP models range from 7 to 14 J/round. Each NLP model (if selected for distributed inference) would return top-k prediction results (the value of k is set as 3 by default.). The probabilities of in-time prediction for NLP models range from 0.7 to 0.95. In each round, the user inputs are sampled from a combination of three corpora,[8] i.e., Reuters-21578 [54], Customer Review Datasets [55], and Amazon Product Review Data [56].

Refer to Table IV for the default simulation settings.

A. DONUT versus Other Online Algorithms

In this section, we compare the performance of DONUT with the following online algorithms. The main difference between these algorithms and DONUT lies in the focus on different performance metrics and long-term constraints.

- *UCB1* maximizes prediction CTR while neglecting long-term constraints [19]. Under UCB1 [19], the utility is set as $\hat{w}_i(t) = \hat{\mu}_i(t)$ for each $i \in \mathcal{N}$, where $\hat{\mu}_i(t)$ is the UCB estimate defined in (10).
- *UCB-Energy* maximizes prediction CTR and guaranteeing the energy consumption constraint [57]. Under UCB-Energy, the utility $\hat{w}_i(t)$ is defined as $\hat{w}_i(t) = V\hat{\mu}_i(t) - \alpha Q(t)W_i(t)$ for each $i \in \mathcal{N}$.
- *UCB-Impatience* maximizes prediction CTR and minimizing the user impatience [45]. Under UCB-Impatience, the utility is defined as $\hat{w}_i(t) = V\hat{\mu}_i(t) + \alpha Z_i(t)$ for each $i \in \mathcal{N}$.

Comparison under Different Metrics: We evaluate the performances of DONUT and other online algorithms under different metrics in Fig. 3. Recall that the performance is not simply measured by an individual metric (e.g., prediction CTR) as it is also crucial to guarantee long-term constraints with low user impatience and low energy consumption. Compared with other online algorithms, DONUT achieves a lower prediction CTR but less user impatience and energy consumption. Particularly, compared with UCB1 and UCB-Energy, DONUT achieves up to 97% reduction in user impatience and up to 16.6% reduction in the energy consumption.

In Fig. 3(b) and (c), under DONUT, the time-averaged total user impatience is bounded and the time-averaged energy consumption is kept below the energy budget (denoted as a black dash line in Fig. 3(c)). The results verify the effectiveness of DONUT in reducing user impatience and satisfying the energy consumption constraint.

Comparison Under an Integrated Metric: To better show the outperformances of DONUT, we propose an integrated metric to evaluate the comprehensive performances of DONUT and other online algorithms in terms of prediction CTR, user impatience, and energy consumption. The integrated metric[9] is defined as follows:

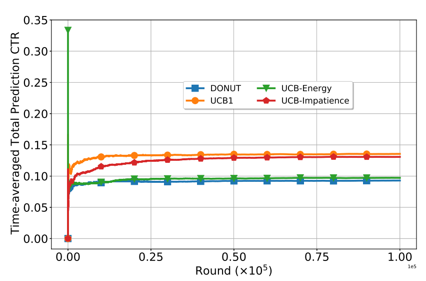
$$\mathrm{IM} \triangleq \frac{e \cdot \text{Prediction CTR}}{\text{Energy Consumption}} + \frac{\text{Prediction CTR}}{\text{User Impatience}}, \tag{19}$$

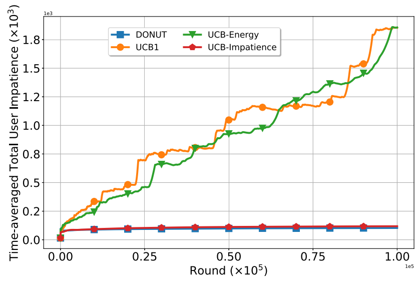
where e is an indicator variable. Specifically, $e = 1$ if the energy consumption constraint in (6). is satisfied (i.e., the time-averaged energy consumption is less than or equal to the energy budget), and zero otherwise. Intuitively, a larger IM an algorithm achieves, a better performance the algorithm has. The reason is that a larger IM implies more increases in prediction CTR given one more unit of energy consumption and user impatience.

In Fig. 4, we evaluate performances of DONUT and other online algorithms under the integrated metric IM. The model selection number n is set as 3 and the energy budget b is set as 20 J. Due to the neglect of either the user impatience or the energy consumption, UCB1, UCB-Energy, and UCB-Impatience achieve inferior performances than DONUT in terms of IM. The result suggests that DONUT can effectively balance trade-offs among prediction CTR, user impatience, and energy

[8]The adopted models are adopted from the widely known Transformers library and the utilized datasets are created out of real-world applications. Though we do not run DONUT in a real testbed (which is not our focus in this paper), our simulations are convincing enough to provide insights for the practical deployment. Further investigation of DONUT in a real testbed is considered as an interesting future work.

[9]The prediction CTR, user impatience, and energy consumption in the integrated metric (19) are all time-averages.


(a) Time-Averaged Total Prediction CTR.


(b) Time-Averaged Total User Impatience.

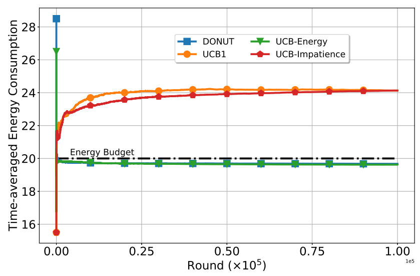

(c) Time-Averaged Energy Consumption.

Fig. 3. Performances of DONUT and other online algorithms.

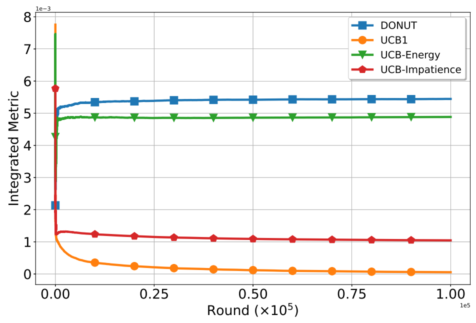


Fig. 4. Performances of DONUT and other online algorithms in terms of the integrated metric IM.

consumption and achieve a better performance than other online algorithms.

B. Effects of Parameters V & α

To investigate the system performance under different learning strategies, we propose two variants of DONUT by replacing the UCB term in the online learning procedure (line 4 in Algorithm 1) with other bandit learning algorithms.

- *DONUT-Epsilon*: In each round t, the UCB estimate in (10) is replaced with the exploitation term $\bar{\mu}_i(t), i \in \mathcal{N}$. The user device selects the edge servers with the highest model utilities (18) with probability $(1-\epsilon), \epsilon \in \{0, 0.01, 0.1\}$. Otherwise, it uniformly and randomly selects n of the NLP models.
- *DONUT-MOSS*: In each round t, the exploration term $\gamma_i(t)$ is replaced as follows [49]:

$$\gamma_i(t) = \sqrt{\log\left(\max\left\{T/(N \cdot N_i(t)), 1\right\}\right)/N_i(t)}.$$

In the following, we evaluate the performances of DONUT and its variants under different values of parameters V and α. Recall that V and α measure the willingness of selection for NLP models with high prediction accuracies, large user impatiences, and low energy consumption as shown in (18).

We evaluate the effect of parameter V on performances of DONUT and its variants in Fig. 5. The value of V ranges from 1 to 500, and the value of α is fixed as 1. From the figure, we see that given a larger value of V, DONUT and its variants incur fewer regrets while accumulating higher user impatiences and more energy consumption. For example, as the value of V increases from 1 to 500, DONUT incurs 17.0% fewer regrets, causes 41.4% higher user impatience, and consumes 4.2% more energy. Particularly, as the value of V grows, the weight of prediction accuracies in the model utility (18) increases. As a result, DONUT has a stronger incentive to achieve higher prediction CTR and inevitably focuses less on reducing user impatience and energy consumption. In summary, the results in Fig. 5 illustrate that DONUT and its variants achieve effective trade-offs among coupled metrics by tuning V.

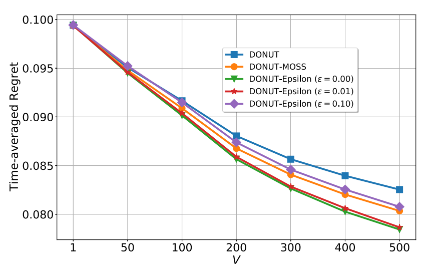
We evaluate the effect of parameter α on performances of DONUT and its variants in Fig. 6. The value of α ranges from 1 to 200 and the value of V is fixed as 300. The figure demonstrates that as the value of α grows, DONUT and its variants incur higher regrets while causing lower user impatiences and more energy consumption. In particular, as the value of α varies from 1 to 200, DONUT and its variants incur 14.3% more regrets, avoid 7.7% user impatience and cause 1.5% more energy consumption on average. Such results show that DONUT and its variants achieve tunable trade-offs through parameter α.

An interesting result is that even without exploration during online learning, DONUT-Epsilon with $\epsilon = 0$ achieves comparable performance against DONUT and other variants. The reason is that to minimize user impatiences, under-explored models will be selected when their associated user impatiences increase to be high. In other words, the online control procedure enforces exploration implicitly under our algorithm design.

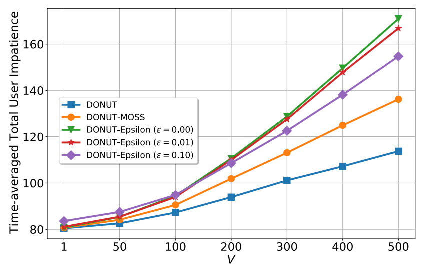
C. Effects of Model Selection Numbers

Different Fixed Model Selection Numbers: In Fig. 7, we compare performances of DONUT under different model selection numbers. The model selection number n ranges in $\{1, 3, 4\}$ and the energy budget b is set as 50 J.

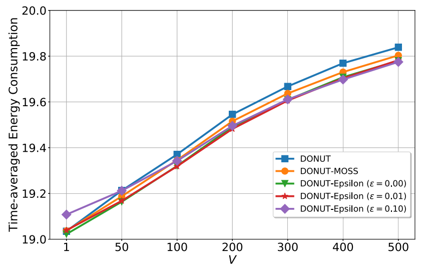
The results show that DONUT with multiple selections ($n = 3, 4$) is potentially better than a single selection ($n = 1$) with higher prediction CTR and lower user impatience while keeping the energy consumption within budget. For prediction CTR, multiple selections result in a higher probability of correctly predicting the user's favorable next words and more reliable prediction in face of unstable wireless communications. For user impatience, multiple selections lead to more chances to reduce impatience for multiple models as shown in (3). For



(a) Time-Averaged Regret.

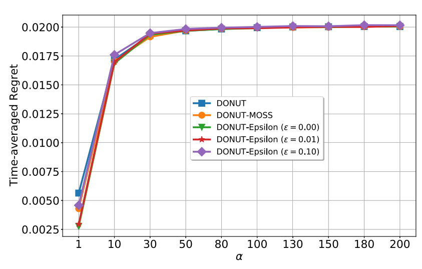


(b) Time-Averaged Total User Impatience.

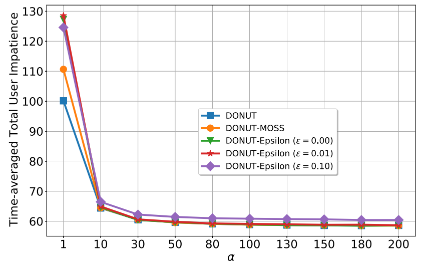


(c) Time-Averaged Energy Consumption.

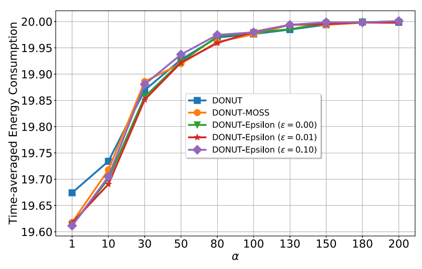
Fig. 5. Performances of DONUT and its variants under different values of V.



(a) Time-Averaged Regret.

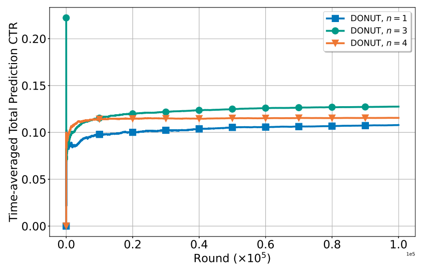


(b) Time-Averaged Total User Impatience.

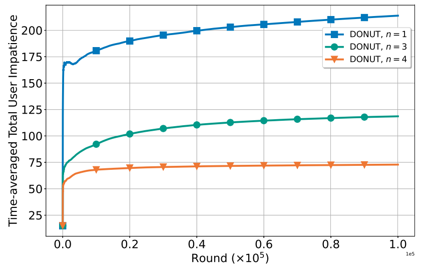


(c) Time-Averaged Energy Consumption.

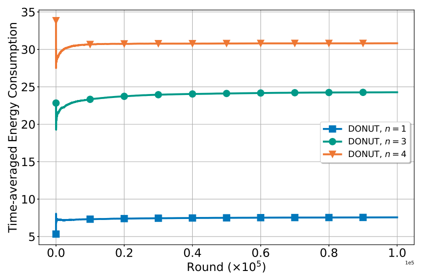
Fig. 6. Performances of DONUT and its variants under different values of α.



(a) Time-Averaged Total Prediction CTR.



(b) Time-Averaged Total User Impatience.



(c) Time-Averaged Energy Consumption.

Fig. 7. Performances of DONUT under different model selection numbers.

energy consumption, though multiple selections give rise to higher consumption for wireless transmission, DONUT keeps such consumption within budget by guaranteeing the long-term energy consumption constraint.

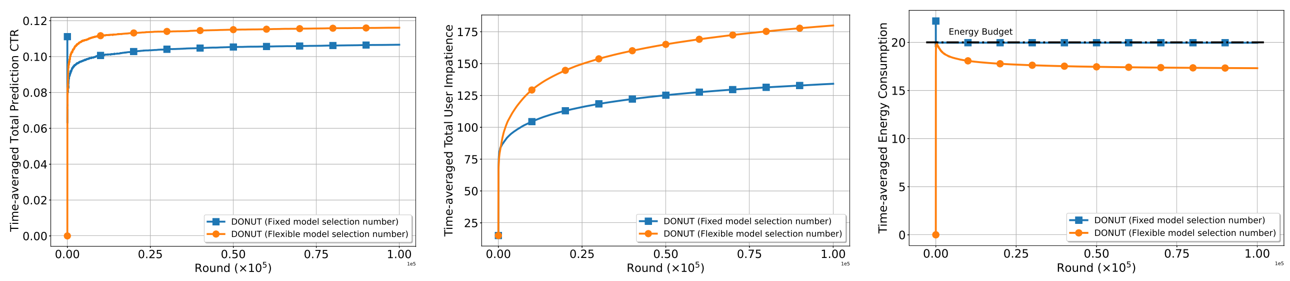
Fixed & Flexible Model Selection Numbers: In Fig. 8, we compare the performances of DONUT under fixed and flexible model selection numbers. Under the case of a fixed model selection number, we set the model selection number as 3; for the case of a flexible model selection number, we restrict the model selection number to be no larger than 3.

The results show that compared with the case of fixed model selection number, DONUT under flexible model selection number achieves higher prediction CTR and less energy consumption, but higher user impatience. The reason is that DONUT under flexible model selection number avoids selecting NLP models with low accuracies and high energy consumption, thus focusing on only a subset of NLP models. Consequently, it fails to select models of high impatience to reduce user impatiences.

D. Effects of Top-k Prediction Results

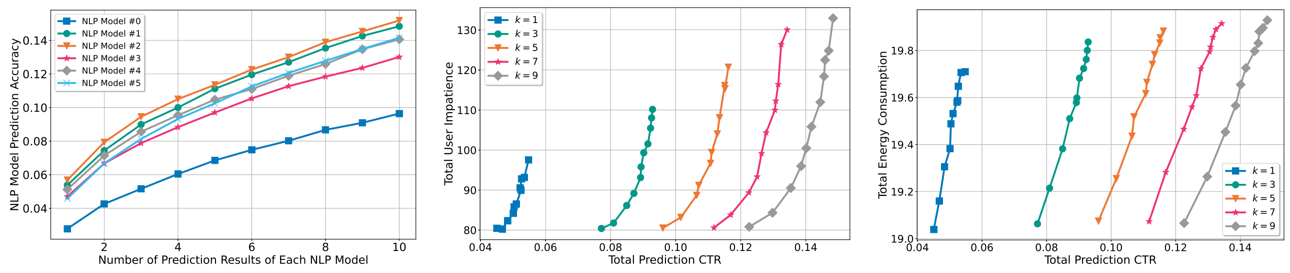
In Fig. 9, we investigate the performances of DONUT under cases where NLP models return top-k prediction results with different values of k. Since the common number of predicted next-words presented to the user ranges from 5 to 25 and the model selection number is fixed as 3, we vary the value of k from 1 to 10 in our simulations [13]. V and α are set as 300 and 1, respectively.

Fig. 9(a) shows the evolvement of prediction accuracies of NLP models indexed from 0 to 5 when k increases (detailed prediction accuracies of all NLP models are summarized in Table V). The results show that the prediction accuracy of each NLP model increases monotonically as the value of k increases.



(a) Time-Averaged Total Prediction CTR. (b) Time-Averaged Total User Impatience. (c) Time-Averaged Energy Consumption.

Fig. 8. Performances of DONUT under fixed and flexible model selection number.



(a) Prediction Accuracies of NLP Models. (b) Time-Averaged Total User Impatience. (c) Time-Averaged Energy Consumption.

Fig. 9. Performances of DONUT under different top-k prediction results.

TABLE V
PREDICTION ACCURACIES OF NLP MODELS WITH DIFFERENT VALUES OF k

NLP Model Index	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
#0	2.78%	4.26%	5.16%	6.04%	6.85%	7.48%	8.02%	8.67%	9.09%	9.64%
#1	5.38%	7.44%	8.99%	10.00%	11.12%	11.96%	12.70%	13.54%	14.26%	14.84%
#2	5.68%	7.93%	9.45%	10.50%	11.35%	12.26%	13.00%	13.89%	14.53%	15.18%
#3	4.70%	6.67%	7.88%	8.83%	9.70%	10.54%	11.27%	11.84%	12.36%	13.01%
#4	5.12%	7.14%	8.56%	9.54%	10.47%	11.10%	11.89%	12.59%	13.47%	14.06%
#5	4.58%	6.67%	8.13%	9.32%	10.25%	11.26%	12.06%	12.78%	13.49%	14.15%
#6	4.74%	7.49%	9.37%	10.98%	12.15%	13.40%	14.27%	15.27%	15.97%	16.80%
#7	1.44%	2.54%	3.39%	4.21%	4.87%	5.49%	5.90%	6.38%	7.04%	7.53%
#8	5.31%	7.50%	9.30%	10.72%	11.70%	12.48%	13.35%	14.19%	14.91%	15.45%
#9	11.76%	16.20%	19.06%	21.27%	23.16%	24.92%	26.15%	27.32%	28.41%	29.28%
#10	13.58%	18.40%	21.57%	24.16%	26.19%	28.19%	29.73%	31.13%	32.08%	33.20%
#11	5.77%	8.39%	10.38%	11.77%	13.16%	14.41%	15.46%	16.31%	17.03%	17.70%
#12	6.72%	9.62%	11.53%	13.21%	14.46%	15.57%	16.64%	17.62%	18.58%	19.44%
#13	3.99%	5.86%	7.52%	8.78%	9.93%	10.78%	11.49%	12.22%	12.71%	13.31%
#14	6.79%	9.62%	11.33%	12.55%	13.60%	14.60%	15.42%	16.28%	16.86%	17.48%

Fig. 9(b) and (c) illustrate the performance of DONUT under different values of k when V increases from 10 to 500. For example, the blue curves in Fig. 9(b) and (c) show that given $k=3$, as V increases from 10 to 500, the total prediction CTR, user impatience, and energy consumption increase by 20.3%, 37.0%, and 4.10%, respectively. From Fig. 9(b), we see that when achieving the same value of total user impatience, DONUT with a higher value of k reaches a higher total prediction CTR. For example, when the value of total user impatience is 90, the total prediction CTR increases by 158.13% as the value of k increases from 1 to 9. From Fig. 9(c), we observe a similar trend in the change of total prediction CTR and total energy consumption when the value of k grows. Specifically, DONUT achieves higher total prediction results under a higher value of k (given a fixed value of total energy consumption). For example, when the value of total energy consumption is 19.6, the total prediction CTR increases by 166.52% as k increases from 1 to 9. The reason is that with a larger value of k, more prediction results are sent to the user device. As a result, the user is more likely to find the favorable next words or phrases, hence a higher prediction CTR. Moreover, under a larger value of k, the increment in the total prediction CTR decreases under the same increase in value of k. For example, the total prediction CTR increases by 73.77% when k increases from 1 to 3, but only increases by 10.76% when k increases from 7 to 9.

VII. Conclusion

In this paper, we studied the distributed inference for edge natural language processing, focusing on the typical next-word prediction service. We cast the key model selection problem as an online constrained optimization problem with multiple objectives. Specifically, the problem requires simultaneously maximization of prediction click-through rate, minimization of user impatience, and guarantee of within-budget energy consumption. From the lens of constrained bandit learning, we proposed DONUT to solve the model selection problem via an integration of online learning and online control. Theoretical analysis showed that DONUT achieves sub-linear regret, ensures bounded user impatience, and maintains energy consumption with budget. Simulation results verified the outperformance of DONUT over other online algorithms in terms of the proposed integrated metric by an average 160% increase.

References

[1] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 604–624, Feb. 2021.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[3] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing AI to edge: From deep learning's perspective," 2020, *arXiv:2011.14808*.

[4] M. Chen et al., "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, Dec. 2021.

[5] M. Chen et al., "Federated learning of n-gram language models," in *Proc. 23rd Conf. Comput. Natural Lang. Learn.*, 2019, pp. 121–130.

[6] M. E. Peters et al., "Deep contextualized word representations," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 2227–2237.

[7] J. D.M.-W. C. Kenton and L. K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[8] Y. Liu et al., "RoBERTa: A robustly optimized bert pretraining approach," 2019, *arXiv:1907.11692*.

[9] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite bert for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, A distilled version of bert: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[11] T. Brown et al., "Language models are few-shot learners," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, Art. no. 159.

[12] Z. Zhang, Y. Yang, Y. Dai, L. Qu, and Z. Xu, "When federated learning meets pre-trained language models' parameter-efficient tuning methods," 2022, *arXiv:2212.10025*.

[13] A. Hard et al., "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[14] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Front. Comput. Sci.*, vol. 14, no. 2, pp. 241–258, 2020.

[15] Y. Zhou et al., "Mixture-of-experts with expert choice routing," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 7103–7114.

[16] Q. Cao, Y. K. Lal, H. Trivedi, A. Balasubramanian, and N. Balasubramanian, "IrEne: Interpretable energy prediction for transformers," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, 2021, pp. 2145–2157.

[17] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[18] K. Liao, Y. Zhang, X. Ren, Q. Su, X. Sun, and B. He, "A global past-future early exit method for accelerating inference of pre-trained language models," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics*, 2021, pp. 2013–2023.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[20] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," 2019, *arXiv:1906.04329*.

[21] M. Chen et al., "Evaluating large language models trained on code," 2021, *arXiv:2107.03374*.

[22] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, and H. Zhang, "Federated learning meets natural language processing: A survey," 2021, *arXiv:2107.12603*.

[23] M. Chen, R. Mathews, T. Ouyang, and F. Beaufays, "Federated learning of out-of-vocabulary words," 2019, *arXiv:1903.10635*.

[24] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.

[25] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[26] J. Stremmel and A. Singh, "Pretraining federated text models for next word prediction," in *Proc. Future Inf. Commun. Conf.*, 2021, pp. 477–488.

[27] C.-J. Wu et al., "Machine learning at facebook: Understanding inference at the edge," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2019, pp. 331–344.

[28] W. Ren et al., "A survey on collaborative DNN inference for edge intelligence," 2022, *arXiv:2207.07812*.

[29] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[30] C. Hu and B. Li, "Distributed inference with deep learning models across heterogeneous edge devices," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 330–339.

[31] A. Parthasarathy and B. Krishnamachari, "Partitioning and deployment of deep neural networks on edge clusters," 2023, *arXiv:2304.11941*.

[32] M. A. Khan, R. Hamila, A. Erbad, and M. Gabbouj, "Distributed inference in resource-constrained IoT for real-time video surveillance," *IEEE Syst. J.*, vol. 17, no. 1, pp. 1512–1523, Mar. 2023.

[33] J. Cao, B. Li, M. Fan, and H. Liu, "Inference acceleration with adaptive distributed DNN partition over dynamic video stream," *Algorithms*, vol. 15, no. 7, pp. 244–261, 2022.

[34] B. Islam and S. Nirjon, "Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems," in *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 4, 2020, Art. no. 82.

[35] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surv. Tuts.*, vol. 23, no. 4, pp. 2131–2165, Fourth Quarter 2021.

[36] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.

[37] Z. Li, M. Paolieri, and L. Golubchik, "Inference latency prediction at the edge," 2022, *arXiv:2210.02620*.

[38] L. L. Zhang et al., "nn-Meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices," in *Proc. 19th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2021, pp. 81–93.

[39] H. Qi, E. R. Sparks, and A. Talwalkar, "Paleo: A performance model for deep neural networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[40] C. Zhu and M. S. Corson, "A distributed channel probing scheme for wireless networks," in *Proc. IEEE Conf. Comput. Commun.*, 2001, pp. 403–411.

[41] M. A. A. Careem and A. Dutta, "Real-time prediction of non-stationary wireless channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7836–7850, Dec. 2020.

[42] B. I. Teigen, N. Davies, K. O. Ellefsen, T. Skeie, and J. Torresen, "A model of WiFi performance with bounded latency," 2021, *arXiv:2101.12512*.

[43] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[44] F. Guillemin, S. E. Elayoubi, P. Robert, C. Fricker, and B. Sericola, "Impatience in mobile networks and its application to data pricing," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 6104–6109.
[45] B. Li, "Efficient learning-based scheduling for information freshness in wireless networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.
[46] E. Altman, R. El Azouzi, D. S. Menasché, and Y. Xu, "Poster: Aging control for smartphones in hybrid networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 2, pp. 68–68, 2011.
[47] E. Altman, R. El-Azouzi, D. S. Menasche, and Y. Xu, "Forever young: Aging control for hybrid networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2019, pp. 91–100.
[48] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[49] J.-Y. Audibert et al., "Minimax policies for adversarial and stochastic bandits," in *Proc. Annu. Conf. Learn. Theory*, 2009, pp. 1–122.
[50] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proc. Eur. Conf. Mach. Learn.*, 2005, pp. 437–448.
[51] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[52] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[53] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2020, pp. 38–45.
[54] F. Debole and F. Sebastiani, "An analysis of the relative hardness of reuters-21578 subsets," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 56, no. 6, pp. 584–596, 2005.
[55] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 168–177.
[56] N. Jindal and B. Liu, "Opinion spam and analysis," in *Proc. Int. Conf. Web Search Data Mining*, 2008, pp. 219–230.
[57] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1702–1710.