International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

POWER-AWARE DISTRIBUTED INFERENCE: A VIEW ON NEXT-WORD PREDICTION

[1]Mrs. A Josh Mary,[2]Bhairathi Naga Satish, [3]Chippada Dinesh, [4]Bollipo Tejas,
[1]Associate Professor, Department of CSE, Rajamahendri Institute of Engineering & Technology, Bhoopalapatnam, Near Pidimgoyyi, Rajahmundry, E. G. Dist. A.P 533107.
[2,3,4] Student, Department of CSE, Rajamahendri Institute of Engineering & Technology, Bhoopalapatnam, Near Pidimgoyyi, Rajahmundry, E. G. Dist. A.P 533107.

Abstract—
There has been a development of natural language processing (NLP) services, particularly those offered at the edge, driven by the need for high-quality AI generated content (AIGC) with quick response times. One common edge NLP function for mobile keyboards on user devices is next-word prediction, and we investigate distributed inference for this purpose to provide some context. In line with this, we optimize linked metrics, which include maximizing predicted click through rate (CTR) for better QoS, minimizing user impatient for increased QoE, and keeping energy usage within budget for sustainability. In addition, we think about the real-world scenario where nobody knows how accurate diverse NLP models are in making predictions. To quantify the prediction accuracy of models and balance the trade-offs among linked metrics, we provide a new distributed inference technique for online next-word prediction with user impatience (DONUT) that combines online learning and online control. Based on our theoretical analysis, DONUT accomplishes sub-linear regret (loss of CTR), guarantees limited user impatience, and keeps energy usage under budget. We prove DONUT's flexibility to different environments and show it outperforms other baseline techniques using numerical simulations. ,

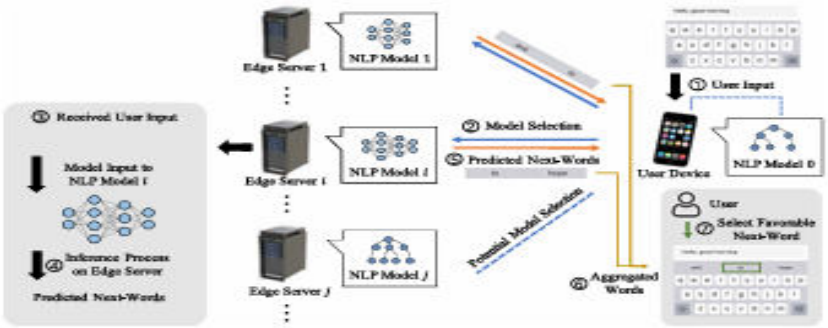

Figure 1: Edge natural language processing for next-word prediction. 1. The user input is generated on the user device. 2. The user device chooses a subset of natural language processing (NLP) models. 3. The user input is used as input to the models. 4. Each model is inferred. 5. The user device receives predicted next words from the selected NLP models. 6. The user device aggregates the words by removing duplicates and shows them to the user. 7. The user chooses

one favorable predicted word. Distributed inference, edge NLP, online learning, online control, and next-word prediction are some of the index terms.

INTRODUCTION

AI-generated content (AIGC) has increased in quality as a result of the incorporation of natural language processing (NLP) services by a number of user-centric apps during the AI revolution [1]. This combination paves the way for AI to become more widely used and for future interactions with AI to be as natural as speaking with a person. Powerful natural language processing models hosted on cloud servers have long been the go-to for end users requesting access to NLP services. excellent latency and wireless connections might affect cloud-based services, making it difficult to provide real-time promises, even while the accuracy is excellent [2]. Edge NLP, which enables NLP services on the edge servers, is a potential substitute for this. Customers get high-quality, lightning-fast natural language processing services from edge servers because of the abundance of computing resources these facilities have and how close they are to user devices. [3, 4]. The bulk of research on edge natural language processing has been on distributed training, specifically "how to train NLP models across edge servers to facilitate NLP services??" [12] This study shifts gears and asks, "How can edge NLP make use of trained NLP models (e.g., N-gram [5], BERT [7], and GPT [11]) that are already running on edge servers?" Because we want to use the inference capabilities of trained models across edgeservers that are geographically dispersed, this method is called distributed inference. As seen in Figure 1, we examine distributed inference for next-word prediction, a common and non-trivial natural language processing service for end users, to ensure concreteness. Using mobile keyboard inputs to proactively propose the most likely next-words is the purpose of next-word prediction [13], which aims to increase users' input efficiency. One interesting thing about distributed inference is that it allows you to use the inference power of numerous trained models instead of just one. We aggregate varied projected next-words by dispersing user input over several NLP models. This gives the user a larger variety of candidates, which improves the next-word prediction.

TABLE I IEEE TRANSACTIONS ONMOBILECOMPUTING, 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

providing assistance. In ensemble learning, where using a mixture of experts (MOE) is widespread practice and often yields better results than using a single expert model [15], this theory is well-founded [14]. The user device, however, is unable to communicate with an infinite number of NLP models due to rising overheads (such as energy cost and latency). This raises the question of which trained NLP models the user's device should utilize for distributed inference—a real-world model selection problem. For two reasons, this model selection issue is difficult and not simple. The inference capabilities of natural language processing (NLP) models is one of the criteria used to make a model selection choice [16]. We zero in on three characteristics that might differ between NLP models: prediction accuracy, inference delay, and energy usage. Although N gram often uses less energy and has shorter inference latency than transformer-based models[17], its prediction accuracy is typically poorer, as demonstrated in Table I. Even more so, because deployed NLP models aren't always customized to individual users, their prediction accuracy is frequently user-dependent and unknown in advance. Actually, many different pre-trained NLP models are already installed on edge servers [18], and their training corpus could differ significantly from the input corpus provided by users. An online learning technique is necessary to gather input via dynamic model choices in order to measure the prediction accuracy. Second, QoS, QoE, and user device sustainability are all interdependent measures in the model selection dilemma. The prediction click-through rate (CTR), or the rate at which the end user accepts the predicted words, is a standard metric for quality of service (QoS). It seems to reason that models with good accuracy in predictions should also have good prediction CTR and quality of service. In terms of quality of experience, we define it as the presence of user impatience owing to delayed predictions (i.e., when the time it takes to forecast the next word is longer than the maximum tolerance latency). Hence, models that have a long time to infer results might cause users to be impatient and have a poor quality of experience. For the sake of sustainability, we should keep user devices with restricted battery capacities at energy consumption levels that are within budget. As a result, models that use a lot of power might wind up going over their allotted amount and endangering sustainability. Adaptively adjusting the relative relevance of these connected measures requires an online control approach. All things considered, the model selection issue is consistent with the setting of brute-force learning as it is fundamentally a sequential decision-making problem under uncertainty [19]. Most bandit learning approaches, however, aren't suitable for immediate adoption since choosing a model requires a combination of online learning and online control in order to balance linked metrics and evaluate prediction accuracy. A possible consequence of inefficient online learning with erroneous estimate is that it may prejudice the selection of less-than-ideal NLP models for future control choices. The flip side is that it's not easy to balance various goals, and control choices that aren't carried out effectively might lead to ineffective feedback and learning efficiency. For example, the user device can fail to notice possible optimum options due to underestimating the performance of particular models if they are seldom chosen. Here, we present DONUT, a new eldistributed inference method for online next-word prediction with user impatience, as a solution to the

crucial model selection challenge. What follows is a synopsis of our work's primary contributions and outcomes: The process of modeling In distributed inference for next-word prediction, we look at the model selection challenge. Under energy constraints, our goal is to optimize prediction CTR for high QoS and reduce user impatience for high QoE (Section III). Our approach has the potential to be expanded to include other edge NLP services, such as emoji prediction, in addition to next-word prediction [20]. We also provide light on how to implement edge NLP in the future, which might be useful for services like Codex-based auto-programming tools [21] and ChatGPT-based dialog systems [11]. To address the model selection issue, we provide DONUT, an algorithm that combines online learning with online control, from the perspective of limited bandit learning (Section IV). In our theoretical analysis, we demonstrate that under certain conditions, such as a finite number of rounds (T) and initial adjustable parameters (V and α), DONUT may accomplish a time-averaged regret and a restricted total user impatience. We do comprehensive simulations to assess DONUT's efficacy, and DONUT also guarantees the user device's energy consumption restriction (Section V). They show that DONUT outperforms other online algorithms and confirm our theoretical analysis (Section VI)

RELATED WORKS

Introduction to Edge Natural Language Processing The possibility of using edge-based resources to provide user-centric services has been greatly enhanced by the widespread use of edge computing. As a result of this trend, edge NLP has arisen to provide real-time, high-quality services including text categorization, audio recognition, emoji prediction, next-word prediction, and more [22]. Here, we mostly go over papers that are relevant to the next-word prediction service. The majority of previous research has been on developing and refining methods for training new natural language processing (NLP) models at the edge, either from beginning or by fine-tuning existing models (federated learning) [12]. A lightweight natural language processing model for next-word prediction termed CIFG (Coupled Input-Forget Gate) is trained via federated learning in work [13]. Using a character-level recurrent neural network that is based on the architecture of CIFG, the out-of-vocabulary issue is further addressed in work [23]. Personalized model training is discussed in work [24], which proposes three different kinds of algorithms—hypothesis-based clustering, data interpolation, and model interpolation—to address different real-world requirements. To train individualized models for next-word prediction, the authors of work [25] suggest combining sophisticated model aggression with an attention mechanism. To improve the performance of next-word prediction, work[26] combines federated fine-tuning with centralized model pre-training and pretrained word embeddings. Training high-performance distributed models lies at the heart of all the aforementioned federated learning studies. Their focus is on finding a distributed method to train new natural language processing models for edge NLP. For the purpose of enabling edge NLP, our work adopts a novel tack. Our main objective is to take advantage of distributed inference, which allows trained models to make inferences at the edge. With that in mind, we are primarily concerned with: "how to utilize

trained NLP models in a distributed mannerforedgeNLP" [27]. Supporting next-word prediction services with high QoS and QoE in a sustainable way, distributed inference allows user devices to use heterogeneous trained NLP models at the edge. Part B: Distributed Edge Inference Distributed inference, which goes beyond edge NLP, is a new and exciting way to use edge-trained models to provide real-time applications with excellent quality. Much of the current literature on distributed inference focuses on how to divide up a single trained model across groups of untrained users. Inference overheads, including computational expenses and energy usage, are meant to be layer-wise distributed to edge servers or devices [28]. The goal of the work [29] is to use the nearby computing resources for real-time inference by adaptively dividing neural models across edge servers and devices. The goal of this work is to reduce inference latency by studying DAG-structured neural models and dividing their layers into separate execution units [30]. The goal of the work cited as [31] is to optimize throughput and inference delay by dividing and deploying neural models in edge clusters. A number of model partitioning-based applications have also recently emerged. In a resource-constrained Internet-of-Things system, real-time video analysis using partition-based distributed inference is investigated in work [32]. In order to speed up distributed inference for edge video streams, work [33] partitions the models. Our study takes a novel tack by deploying several trained models at the edge rather than just one. In particular, we want to aggregate predictions using modelselection, which yields a mixture-of-expert method that might improve the system's performance. Section Three: System Model and Problem Formulation Distributed inference for next-word prediction and its underlying system model are described here. In particular, we zero in on the key models selection challenge, where we aim to maximize sustainability, quality of experience, and quality of service all at once. A non-trivial yet simple setup is the subject of this study. The next-word prediction service is provided to a single user via an edge NLP system with $(N - 1)$ edge servers. Deploying a single trained NLP model across all user devices and edge servers is standard practice. N models with different levels of prediction accuracy, inference delay, and energy consumption are thus implied. The models are indexed using the set $N \precsim \{0,1,...,N-1\}$, whereas the model on the user device is indexed with zero. Assuming $t \in \{0,...,T-1\}$, we know that the system runs on a round1 basis with a fixed time horizon. It is crucial to consider the end-to-end latency when accessing each model for inference [34], since edge NLP services are often very latency sensitive. With regard to the user device-based local model, the inference latency is what we mean when we talk about end-to-end latency. The end-to-end latency for models running on an edge server is the sum of the inference latency and the transmission latency. The inference latency is the time it takes for the model to make predictions, while the transmission latency is the time it takes for the user device and the edge server to receive the results of those predictions. 2 Prior to making any model selection choices, we presume it is possible to quantify the associated inference latency and trans mission delay. For instance, there are two ways to estimate the inference latency of natural language processing (NLP) models:(1) using a data-driven method with pre-trained machine learning models on different hardware configurations [37],

or(2) using analytical performance modeling with background information on the NLP model [39]. To find the transmission latency of a wireless connection between a user's device and an edge server, one may either use test packages to probe the channel or use a model of the channel's communication protocol, like WiFi, to forecast the latency [40], [41].

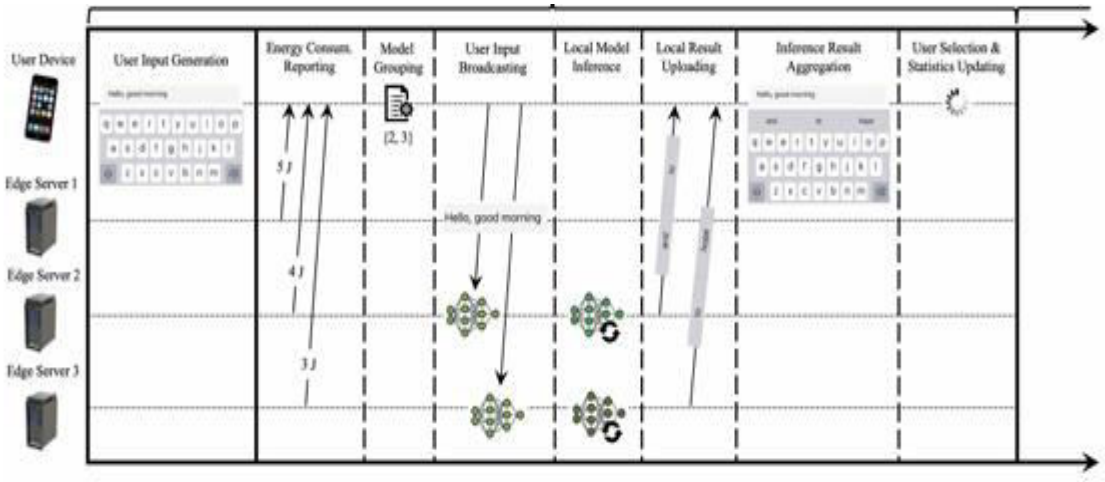

Fig. 2. Workflow of distributed inference for next-word prediction at the edge.

Hence, we check whether NLP model i can provide in-time predictions using end-to-end latency measurements, which are represented by a binary random variable si(t). For each given round t, the function si(t) is defined as 1 if NLP model i is able to finish the prediction within the tolerance delay, and zero otherwise. Assuming i.i.d.distribution across rounds, we say that si(t) is independent. The Use of Distributed Inference for Predicting the Next Word For every round t, we provide the user a next-word prediction service by doing distributed inference in the edge NLP system (see Fig. 2). To be more precise, the system follows these phases to operate: 3 1) The user will first be prompted to enter their input using the device's mobile keyboard. Secondly, after training a subset of natural language processing (NLP) models on its own and edge servers, the user device makes model selection judgments. 3) The user's device transmits the input to the chosen models either locally via hardware-level transmission or remotely to the associated edge servers via wireless transmission. 4) The user input is used as the model input by each of the chosen models during an inference procedure. The top-k projected next phrases are returned to the user's device by each model that was chosen. 4 5) After merging the anticipated next words from all of the chosen NLP models and eliminating duplicates, the user's device displays the combined words to them on the mobile keyboard. 6) To aid with his keystrokes, the user chooses a positive projected next phrase. B. Distributed Inference Model Selection The model selection number, represented as a fixed integer n ($1 \leq n = N$), is the number of models chosen in each round by the user device. 5 In round t, we indicate model selection choices using the binary vector $d(t) \triangleq \{di(t)\}i\in N$, where $di(t)=1$ if user device i selects model i, and zero otherwise. F(N) is the set of all subsets of N with cardinality less than or equal to n, i.e., $F(N) \triangleq \{S \subseteq N||S|\leq n\}$.

Consequently, for convenience, we designate the set of chosen models as S(t). S(t) ∈ F(N). The key points of our study are outlined in Table II. Section C. Service Quality: Tracking Efficiency By calculating the CTR, or forecast click-through rate, we may define QoS [13]. To be more precise, the following is the definition of the prediction CTR in each round t:

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

where ai(t) is the click-through indicator for NLP model i, where 1 is the value when the user selects one of the NLP modeli prediction outcomes and 0 is the value otherwise. Assume that the independent variable ai(t) is normally distributed according to a Bernoulli distribution with a mean μi that falls between 0 and 1, where μi represents the accuracy of model i's predictions. Maximizing the time-averaged prediction CTRs across T rounds is therefore the best way to improve QoS for next-word prediction. I so propose the following objective:

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

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.

ALGORITHMDESIGN

Our new eldistributed inference algorithm for online next-word prediction with user impatience (DONUT) combines bandit learning methods and Lyapunov optimization approaches. It is presented in this section. Section A. Bandit Learning With Upper Confidence Binding To enhance prediction accuracy under uncertain models, we use bandit learning approaches in

online learning in this section. R(t)=CTR(t) is defined as the corresponding prediction in the bandit situation. In addition, we define the loss of prediction accuracy during model selection using the time-averaged regret over T rounds. R∗(withoptimaldecisions{d∗(t)}) is the formal notation for the optimal time-averaged prediction CTRoverT rounds. Wethendefine the time-averaged regret as the difference between R∗ and the time-averaged prediction CTR under {d(t)} across T rounds.

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

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

You may find the detailed transformation in the online appendix (A). Note 5: The predicted prediction accuracy, user impatience, wait duration, and energy usage are all factors that contribute to the model's utility (18), which is a weighted total. It seems to reason that, as shown in (17), the following characteristics would be most beneficial when making model selection decisions: We want to maximize the time-averaged prediction CTR in (2), thus models with high estimated prediction accuracy make sense. This, in turn, leads to a high prediction CTR. Models with minimal energy consumption are preferred because they assist keep overall energy consumption under budget and ensure long-term constraints in (6). Models connected with considerable user frustration. Keep in mind that the user impatience dynamics described in (3) would cause the default value to be restored after a model is chosen. Consequently, we want to minimize time-averaged total user irritation in (4), and picking NLP models with high user impatience might significantly lower that number.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Algorithm 1: **D**istributed Inference for **O**nline **N**ext-Word Prediction With **U**ser Impa**T**ience (DONUT).

Input: Set of NLP models $\mathcal{N}$, model selection number n, energy budget b, parameters V, α, and total rounds T.
Output: Selected NLP models $\{S(t)\}_t$ over T rounds.

Initialize $Q(0) = 0$ and $\hat{\mu}_i(0) = 0, \bar{\mu}_i(0) = 0, \gamma_i(0) = 0, N_i(0) = 0, Z_i(0) = 0, i \in \mathcal{N}$.
for $t = 0, 1, \ldots, T-1$ **do**
  Observe $s_i(t)$ and $W_i(t), i \in \mathcal{N}$.
  %% Online Learning Procedure
  Calculate $\hat{\mu}_i(t), i \in \mathcal{N}$ according to (10).
  %% Online Control Procedure
  Calculate $\hat{w}_i(t), i \in \mathcal{N}$ according to (18).
  Select NLP models $S(t)$ (break ties randomly):

$$S(t) \in \arg\max_{S \in \mathcal{F}(\mathcal{N})} \sum_{i \in \mathcal{N}} \hat{w}_i(t) s_i(t) d_i(t).$$

  for $i \in S(t)$ **do**
    Send inference requests to selected model i,
    conduct inference and receive
    click-through indicator $a_i(t)$.
  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.
end for

that follow, leading to biased choices of less-than-ideal models for control. In instance, user devices may not be able to choose which models to apply for performance metric optimization if the prediction accuracies of the models are not accurately estimated. However, in addition to decreasing energy usage and user irritation, poorly executed control choices may lead to subpar feedback and hinder learning efficiency. Note 6: Since DONUT just uses the prediction outcomes of trained NLP models, its design is not sensitive to particular properties of NLP models. Thus, DONUT may be expanded to handle more natural language processing tasks, such as emoji prediction [20] or even computer vision with some tweaks to the way predictions are preprocessed before being provided to users' devices.

TABLEIII IEEE TRANSACTIONS ONMOBILECOMPUTING,VOL.23,NO.5,MAY2024
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
$O(1/V)$	$O(V^2)$	$O(1/V)$
$O(1)$	$O(V)$	$O(1/V)$
$O(V)$	$O(1)$	$O(1)$

we have the following theorems on energy budgets in the interior of the maximally feasible area. A. Key Findings from Theory First Theorem (Constraint on Energy Use). The virtual queue meets the given condition when variables V and α are finite.

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

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

Model for the Proof of Theorems 1 and 2. We employ Lyapunov optimization methods to build the evidence for queue stability and limits of user impatience. We begin by building a Lyapunov drift function to describe the time-varying user impatience and queue backlog size. The second step is to establish a maximum allowable value for the drift-plus-regret term by including the aim of regret minimization into the Lyapunov drift function; we then use the fact that the time-averaged queue backlog size is both limited and constrained to further constrain this value. Third, by picking NLP models with the highest user impatient values right now, we build an auxiliary strategy that promotes lowering user impatience exclusively. We get a maximum limit for the drift-plus-regret term with regard to all model selection procedures by using such an auxiliary strategy. In the end, we use telescoping sum and conditional expectation to wrap up the evidence. You can find the full proof in Appendix B, which is accessible online. Note 7. According to Theorem 1, when the parameters V and α have finite values, DONUT obtains great queue stability, indicating that constraint (6) is fulfilled [51]. Notably, the time-averaged total user impatience is constrained to be O(1/α + V/α), as shown by Theorem 2. When the order of V is correct, for example, O(logT), the total user irritation is sub-linearly limited. Proposition 3 (Regret). Reg(T) ≤ C + Nα nV + 3N+π2N 3nT +4 log T nT under DONUT, where n is the number of rounds. Figure 3: Proof of Theorem 3.

In order to quantify the discrepancy between the anticipated ideal compound reward and reward under DONUT, we first provide the concept of per-round regret. The best course of action is then defined, namely the oracle policy in light of the actual accuracy of the predictions. This policy is a model selection strategy that guarantees the energy consumption limit while achieving the optimum total prediction CTR. We find the upper limit of the drift-plus-regret term using this strategy as a basis. Moreover, we break down this upper limit into three sub-terms using an auxiliary policy that resolves a sub-problem comparable to the one stated in (17), with the exception that the UCBterm ˆ μi(t) in the model utility ˆ wi(t) (defined in (18)) is substituted with the actual prediction accuracy μi for every NLP model i∈N. The two types of helper events are then defined, namely, event {ˆ μi(t) − μi ≥ 0} and event{ˆ μi(t) − μi ≤ γi(t)} for every NLP model i in N. Next, we may remove the components from the three sub-terms that are associated with the helper events' complimentary occurrences. To finish the proof, we bind each sub-term using the Chernoff-Hoeffding bound and Jensen's inequality. You can find the full proof in Appendix C, which is accessible online. Comment 9. As shown in Theorem 3, for appropriate values of V, DONUT reaches a regret at an order of O(1/V + logT/T), a non-linear function. As an example, the resultant total regret is sub-linear with an order of O(T log T), which is in agreement with the

bottom limit Ω(√T) up to the logarithmic factor, where the value of V is at order Ω(T/log T) [52]. This demonstrates how strongly our regret is wrapped up with online education. Point 10. Two factors in the regret upper limit are attributable to the online learning process and the first term is curred during the online control method, with the former being largely ascribed to the balancing of trade-offs among linked metrics. A sub-linear time-averaged regret O(logT/T) accumulates throughout such a method with respect to necessary investigation in the face of uncertain latencies.

SIMULATION RESULTS

A server running Ubuntu 20.04 with an Intel(R) Xeon(R) Gold 6230 CPU and a 32 GB Quadro GV100 GPU is what our simulations are performed on. After 30 separate trials, we average each simulation outcome. One user device and fourteen edge servers, or fifteen NLP models, are the components of the system we're considering for next-word prediction using edge NLP. The default values for the parameters V and α are set to 300 and 1, respectively, while the number of operational rounds T is set to 105. A DistilBERT [10] model is implemented on the user device. In terms of the servers, we use three distinct natural language processing models: BERT (indexed from 1 to 8), RoBERTa [8] (indexed by 9 and 10), and ALBERT [9] (indexed from 11 to 14). The prediction accuracies of the Alltheabovemodels, which are pre-trained natural language processing models taken from the Transformers library [53], vary between 3 and 30%. Model selection on natural language processing (NLP) models uses 7–14 J/round of the user device's energy. If distributed inference is used, each natural language processing model will provide the top k predictions (with k defaulting to 3). For natural language processing models, the in-time prediction probabilities are between 0.7 and 0.95. Reuters-21578 [54], Customer Review Datasets [55], and Amazon Product Review Data [56] are the three corpora8 from which the user inputs are combined in each cycle. Table IV contains the default parameters for the simulation. Comparison of DONUT with Other Web-Based Algorithms Here, we evaluate DONUT in comparison to the following web-based algorithms. Focusing on distinct performance indicators and long-term limitations is where these algorithms diverge from DONUT. While ignoring long-term restrictions, UCB1 optimizes prediction CTR [19]. The utility is defined as $\hat{}$ wi(t)=$\hat{}$ μi(t) for every i∈ N under UCB1 [19], where $\hat{}$ μi(t) is the UCB estimate from equation (10).
UCB-irritation seeks to minimize user irritation while maximizing prediction accuracy and CTR [45]. The utility, as defined under UCB-Impatience, is $\hat{}$ wi(t)=V $\hat{}$ μi(t)+αZi(t) for every i in N. Examining Through Various Measures: Figure 3 shows the results of our performance evaluations of DONUT and other online algorithms using various measures. Keep in mind that ensuring long-term limitations with minimal user impatience and energy consumption is just as important as measuring success by an individual indicator (e.g., prediction CTR). While

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

DONUT's prediction CTR is lower than other online algorithms, it uses less energy and causes less user irritation. In particular, DONUT reduces energy usage by 16.6% and user impatience by 97% when compared to UCB1 and UCB-Energy. Figure 3(b) and (c) show that DONUT limits the total user impatience over time and keeps the energy consumption over time below the budget. In Figure 3(c), the budget is represented by the black dash line. The findings show that DONUTin successfully fulfills the energy consumption limitation while also minimizing user impatience. Evaluation in contrast According to an Integrated Metric: Our proposed integrated metric measures the overall performance of DONUT and competing online algorithms across three dimensions: prediction CTR, user impatience, and energy usage. This will help to better demonstrate DONUT's superior performance. Here is the definition of the integrated metric:

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

where the indication variable is located. In particular, the energy consumption limitation in (6) is satisfied when e=1. is met if and only if the time-averaged energy consumption is equal to or less than the energy bud gets, and equal to zero otherwise. It seems to reason that an algorithm's performance will improve as its IM grows. The rationale behind this is that, for every additional unit of energy used and user impatience, a bigger IM means higher rises in prediction CTR.

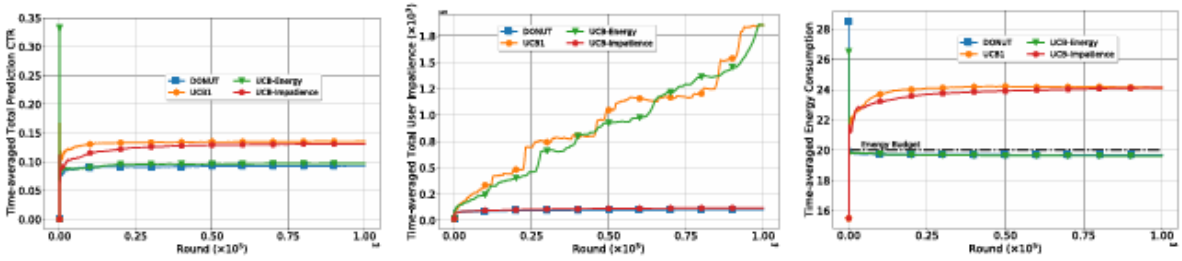

Fig. 3. Performances of DONUT and other online algorithms

IM usage and integrated metrics provide superior results compared to competing web engines. Parameters V and α and Their Impact (B) We provide two versions of DONUT that substitute alternative bandit learning algorithms for the UCB term in the online learning process (line 4 in Algorithm 1) in order to study the system performance under different learning methods. For every t in the set N, the exploitation term ¯ μi(t) is substituted for the UCBestimatein(10) in the DONUT-Epsilon model. The edge servers with the greatest model utilities (18) are chosen by the user device with a probability of 1 −, ∈{0,0.01,0.1}. If not, n of the NLP models are picked at random and uniformly. As stated in reference [49], the exploration term Ňi(t) is substituted in each round by the following: γi(t)= log (max{T/(N •Ni(t)),1})/Ni(t). Here, we assess how well DONUT and its variations perform with varying values of V and α. The willingness of selection for natural language processing models with high prediction accuracies, big user impatiences,

and low energy consumption may be measured by V and α, as shown in (18). In Figure 5, we assess how DONUT and its variations' performances are affected by parameter V. V may take on values between 1 and 500, but α is always set at 1. The chart shows that at increasing values of V, DONUT and its variants cause more user impatience and more energy usage, but less regrets. Take V as an example; from 1 to 500, DONUT experiences a 17.0% decrease in regrets, a 41.4% rise in user impatience, and a 4.2% increase in energy consumption. In particular, the importance of accurate predictions to the model's utility (18) climbs in direct proportion to the value of V. Consequently, DONUT is less concerned with decreasing user frustration and energy usage and more focused on achieving greater prediction CTR. Finally, by adjusting V, DONUT and its variations are able to accomplish effective trade-offs among connected measures, as shown in Fig. 5. In Figure 6, we can see the impact of parameter α on the characteristics and performance of DONUT. There is a fixed value of 300 for V and a range of 1 to 200 for α. According to the figure, DONT and its variations experience greater regrets as the value of α increases, while less user impatience and more energy consumption are the results. On average, DONUT and its variations produce 1.5% higher energy usage, 14.3% more regrets, and 7.7% less user irritation as the value of α changes from 1 to 200. The findings demonstrate that DONUT and its derivatives are able to produce adjustable trade-offs by adjusting factor α. Interestingly, DONUT-Epsilon with =0 outperforms DONUT and other variables even when students do not engage in exploration while studying online. The rationale behind this is that user impatiences are minimized by selecting under-explored models when they reach a high enough level. To rephrase, our algorithm design indirectly encourages exploration via the online control mechanism.

CONCLUSION

We examined distributed inference for edge NLP in this article, with a particular emphasis on the standard next-word prediction service. As a multi-objective, online restricted optimization problem, we recast the key model selection issue. Maximizing the predicted click-through rate, minimizing user impatience, and guaranteeing energy consumption within budget are all necessary to solve the issue. We presented DONUT, which combines online learning with online control, to address the model selection issue via the perspective of limited bandit learning. According to theoretical research, DONUT delivers sub-linear regret, guarantees limited customer impatience, and keeps energy usage under budget. The simulation results showed that DONUT outperformed the other online algorithms by an average of 160 percent when it came to the recommended integrated measure.

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.Chenetal.,"Distributedlearninginwirelessnetworks:Recentprogress and future challenges," IEEE J. Sel. Areas Commun., vol. 39, no. 12, pp. 3579–3605, Dec. 2021.

[5]. M. Chenet 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.NorthAmer.ChapterAssoc.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.

Y.Liuet al., "RoBERTa: Arobustly optimized bert pretraining approach," 2019, arXiv:1907.11692.

[8]. 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.

[9]. V.Sanh,L.Debut,J.Chaumond,andT.Wolf,"DistilBERT,Adistilledversionofbert: Smaller, faster, cheaper andlighter," 2019,arXiv:1910.01108.

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

[11]. Agrawal, K. K. ., P. . Sharma, G. . Kaur, S. . Keswani, R. . Rambabu, S. K. . Behra, K. . Tolani, and N. S. . Bhati. "Deep Learning-Enabled Image Segmentation for Precise Retinopathy Diagnosis". International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 12s, Jan. 2024, pp. 567-74, https://ijisae.org/index.php/IJISAE/article/view/4541.

[12]. Samota, H. ., Sharma, S. ., Khan, H. ., Malathy, M. ., Singh, G. ., Surjeet, S. and Rambabu, R. . (2024) "A Novel Approach to Predicting Personality Behaviour from Social Media Data Using Deep Learning", International Journal of Intelligent Systems and Applications in Engineering, 12(15s), pp. 539–547. Available at: https://ijisae.org/index.php/IJISAE/article/view/4788

[13]. 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.

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

[15]. 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.
[16]. Y. Zhou et al., "Mixture-of-experts with expert choice routing," in Proc. Int. Conf. Neural Inf. Process. Syst., 2022, pp. 7103–7114.
[17]. 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.
[18]. A. Vaswani et al., "Attention is all you need," in Proc. 31st Int. Conf. Neural Inf. Process. Syst., 2017, pp. 6000–6010.
[19]. 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.
[20]. 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.
[21]. S.Ramaswamy,R.Mathews,K.Rao,andF.Beaufays,"Federatedlearning for emoji prediction in a mobile keyboard," 2019, arXiv:1906.04329.
[22]. M. Chen et al., "Evaluating large language models trained on code," 2021, arXiv:2107.03374.