Abstract

A multi-armed bandit is a popular problem in probability theory and reinforcement learning, in which a given slot machine (resource pool) has n arms (bandit/resource), each rigged with their own probability of success. This is a classic sequential resource allocation problem where the objective is to pull the arms (resources) in a sequence so as to obtain the maximum cumulative reward in the face of uncertainty. Traditional A/B testing methods are offline resource allocation methods that dedicate a period of time purely to exploration where traffic is equally allocated to the two possible versions due to which a lot of time and revenue is wasted on the losing variant.On the other hand, MABs successfully balance exploration and exploitation as more knowledge is gained. In the context of internet advertisements, the goal is to learn the click-through rates of several competing advertisements in order to converge to the most appropriate set of advertisements for the user in question.

The proposed system implements the CSlogUCB-F algorithm, that we have come up with, which makes use of side information available in each round - user context and advertisement context - to improve the relevance of the chosen ads to the user making the search query, while ensuring fairness to all advertisers. Initially, the data is preprocessed to form normalised context vectors, which are fed to a module that implements the proposed algorithm and selects the best advertisement for the current query based on the input context. Following this, the various parameters of the algorithm including estimated rewards and fairness debts are updated based on user feedback.

The introduction of context into CSMAB-F algorithm, resulting in the proposed algorithm being developed, helped reduce the regret incurred through the entire run across time steps empirically. An improved fairness notion was proposed that models the real world scenarios more accurately, and was found to reduce the regret based on the experiments conducted. Logistic regression was used to learn the coefficient vectors for the context, which was also a factor in the reduced regret.

Table of Contents

List of Figures

List of Tables

Glossary

MAB	Multi-armed Bandit
CSMAB-F	Combinatorial Sleeping Multi-armed Bandits with Fairness Constraints
UCB	Upper Confidence Bound
CSLogUCB-F	Combinatorial Sleeping Logistic Upper Confidence Bound with Fairness Constraints

Chapter 1

Introduction

A multi-armed bandit is a popular problem in probability theory and reinforcement learning, in which a given slot machine (resource pool) has n arms (bandit/resource), each rigged with their own probability of success. This is a classic sequential resource allocation problem where the objective is to pull the arms (resources) in a sequence so as to obtain the maximum cumulative reward in the face of uncertainty. It is an incomplete information problem in that the mean rewards of arms are not known *a priori*; so the decision maker has to make a trade-off between exploring new arms for possibly better rewards and exploiting the arms that are known to give the best reward so far.

This problem is commonly used to model real-world scenarios such as web resource allocation, internet advertising, crowdsourcing, etc.

1.1 State of the Art Developments

Bianchi *et al.* [1] study various bandit models including stochastic bandits, contextual bandits and linear bandits from the regret analysis perspective. This analysis is performed for i.i.d payoffs as well as adversarial payoffs. For stochastic bandits, UCB strategies are discussed and improvements in the constants and bounds are suggested. Contextual bandits which have access to side information are also studied, and the contextual regret for stochastic contextual bandits are computed and minimized to identify the optimal actions.

The classic MAB setting is based on a gambler having to select one of K slot machines to play over time to maximise his payoff. Previous solutions to this problem made certain statistical assumptions about the machines. Auer *et al.* [2] make no such assumptions for the reward-generating process and they consider an adversary or opponent which has control over the reward-generation process. This adversary may or may not adapt to the actions of the gambler. The solution proposed achieves a per-round reward which approaches that of

the best arm at a rate of $O(T^{-1/2})$ and the solution is tested on an unknown reiterated matrix game.

Vermorel *et al.* [3] perform empirical evaluations of most of the common bandit strategies. The authors study the performances of the ϵ-greedy strategy and its variants, SoftMax strategy and probability matching variants, and the Interval estimation strategies. They also propose the *POKER* strategy that assigns prices to knowledge gained, exploits the distributions of arms, and also takes the time horizon into account. It is found that when rewards are normally distributed, naive approaches like the ϵ-greedy strategy significantly perform better than the the other approaches. But, complicated approaches like the *POKER* strategy perform better on real-world data.

Jamieson *et al.* [4] analyze MAB algorithms in the fixed confidence setting. They draw a minimum number of samples to identify the best arm. Comparison of several state-of-the art bandit algorithms is done theoretically and empirically.

Denardo *et.al.* [5] provide a comprehensive analysis of MABs from the perspective of a Markovian decision problem, and present procedures to compute an optimal policy on the same. Linear as well as exponential utility functions are considered, and procedures to compute the utility earned are also presented.

Most of the above discussed works assume linear reward functions, i.e. rewards which are a linear combination of the expected means of the selected arms. However, this does not accurately represent many real world problems, which may require non-linear reward functions such as a maximum instead. Chen *et al.* [6] propose a stochastic CMAB framework that generalises the existing one to use general reward functions, which requires a knowledge of not only the expected outcome of each arm, but also the entire distribution. This is accomplished by estimating the CDF of the outcome of each arm. To this end, they propose a *stochastically dominant confidence bound* algorithm that estimates the confidence bounds of the underlying random variables as well as their distributions.

Abbasi-Yadkori *et al.* [7] modify algorithms for the stochastic and linear stochastic MABs in order to improve the theoretical and empirical performance of these algorithms. Specifically, they achieve constant regret with high probability by a simple change to Auer's UCB algorithm, by introducing tail inequalities to vector-valued martingales. The novelty of this algorithm lies in the fact that the length of the confidence interval is independent of both the number of arms as well as the number of time steps, which results in a constant regret.

Chen *et al.* [8] study combinatorial multi-armed bandits (CMAB) and establish a framework for the same. The idea is that in each round, a super arm formed from several underlying arms is played. The reward of the super arm depends upon the outcomes of the constituent simple arms which are observed to inform the selection of super arms in future rounds. Since many combinatorial problems are computationally difficult, the authors allow for an (α, β)-approximation oracle which computes a super arm that with β probability produces an α fraction of the optimal reward. The authors establish a CUCB algorithm which uses confidence bounds on the individual arms' reward expectations for selecting suitable super arms in each round in contrast to the base UCB algorithm for the classical MAB problem that is only capable of picking a single simple arm in each round. The CUCB algorithm is designed for a more general non-linear reward structure and aims to minimise the (α, β)-approximation regret. The authors describe three applications that suit the CMAB framework, namely probabilistic maximum coverage bandit for advertisement placement, social influence maximisation bandits where arms outside of the super arm may also reveal their outcomes and combinatorial bandits with linear rewards. In each of these scenarios, the authors prove it is possible to achieve $O(\log T)$ regret.

Cortes [9] adapts several MAB policies to the contextual MAB model with binary rewards by employing supervised binary classification algorithms as oracles. The author also deals with early exploration during the lack of non-zero rewards and benchmarking with classification datasets. At the start of each trial, the environment generates a context of fixed dimension and rewards for the arms too. The agent must pick an action using its knowledge of past context, chosen actions and rewards received. In this regard, the author adapts several popular MAB algorithms like UCB, Thomspon Sampling, ϵ-Greedy and Adaptive Greedy. The

author employs bootstrapping (sampling with replacement) to obtain scalable and tighter upper confidence bounds on the statistics of the oracle which is fit to the set of covariates (features) and rewards of the individual arms independently. He derives a practical method for fitting the oracle to an online data stream incrementally by assigning random sample weights to incoming observations. Evaluating on datasets of BibTeX tags, Del.icio.us tags, Mediamill and EURLex, the author shows that the adapted algorithms outperform the base context-free algorithms, with Contextual Adaptive Greedy policy being the best.

Chen *et al.* [10] propose a framework for stochastic Contextual Combinatorial Multi-Armed Bandit (CC-MAB) which is suitable for volatile arms and submodular rewards. The proposed framework is suitable for problems like recommender systems where a set of arms must be selected in each round based on the side information, i.e. context, coupled with the arms. Volatile arms imply that the available arms may change over time. The authors also consider a submodular reward structure wherein the reward obtained from the selected arms is not simply the sum of individual rewards, but it exhibits a property of diminishing returns dependent upon the relationship between the selected arms. To handle the possibility of infinitely many arms, arms are grouped by context and the expected quality for each group of arms is learnt by the algorithm. The exploration-exploitation strategy of the algorithm also factors budget constraints. The authors evaluate the performance of the CC-MAB algorithm on a real-world crowdsourcing dataset and show that the algorithm achieves $O(cT^{\frac{2\alpha+D}{2\alpha+D}} \log T)$ regret which depends on budget B and context dimension D. The authors also prove that context-aware algorithms outperform those that do not exploit the context information.

Slivkins [11] studies *contextual bandits* which make use of similarity information between arms to give the algorithm hints (context) about the ayoffs in each round. They introduce adaptive partitioning of the similarity space by a contextual zooming algorithm that focuses in on higher-payoff regions of the arm space, as well as popular regions of the context space. The author also discusses some applications of contextual zooming, among which the application to sleeping bandits is of relevance to our study.

Badanidiyuru *et al.* [12] discuss the budget constraints in bandit problems. They introduce a model, called *bandits with knapsacks* that is a combination of the MAB optimization problem and the Knapsack problem. The authors introduce two new algorithms- one is based on a balanced exploration paradigm and the other is a primal-dual algorithm that uses multiplicative updates. The regret bounds of these algorithms are found to be asymptotically optimal (up to poly-logarithmic factors).

Niimi *et al.* [13] focus on budget-limited multi-armed bandit problems in which an agent's actions are constrained by a budget. Their work assumes that reward distributions are dynamic since it is more applicable to real-world problems. They consider both gradual and abrupt changes in the reward distributions. They propose the D-KUBE and the SW-KUBE algorithms to cater to dynamic situations. They observe that when the reward distributions are static, their proposed algorithms produce results similar to the standard static algorithms like KUBE. When the reward distributions are dynamic, D-KUBE and SW-KUBE perform better than KUBE. The algorithms that they propose produce better results when reward distributions change abruptly than when the change is gradual. The main drawback of these algorithms is that they do not consider the case where multiple actions can be taken in a round, which is commonly seen in several real-world systems.

Combes *et al.* [14] establish asymptotic regret lower bounds for multi-armed bandits with budgets, where the number of times an arm may be chosen is constrained. They consider different types of budgets: fixed budgets and budgets that increase over time and two different pricing schemes namely, either the advertiser is charged each time his ad is displayed or alternatively, only when a user clicks his ad. Thus, bandits with budgets are an example of sleeping bandits, where not all the arms can be chosen at a given instant once the budgets of particular arms have been exhausted. For this framework, the authors propose the B-KL-UCB algorithm which is a modification of the standard KL-UCB algorithm that is based on the KL-divergence between reward distributions of arms. The authors evaluate the performance of the algorithm by running numerical experiments on a real-world, open source advertising dataset and show that it achieves $O(\log T)$ regret outperforming previously proposed UCB-like algorithms.

Joseph *et al.* [15] propose a notion of fairness based on the principle of optimism in the face of uncertainty, by which no arm is chosen over another arm with a higher expected reward in a fair manner. They introduce the idea that fairness may not help in *learning* the optimal policy, although it results in *implementing* the optimal policy, which is to choose the arm with the best reward in each round. They also propose fair algorithms for both classic stochastic bandits as well as contextual bandits.

Talebi *et al.* [16] study proportionally fair allocations with low regret in MABs. They discuss the properties of Restricted-Proportionally Fair(RPF) allocations, and extend the same to devise their ES-RPF algorithm. This algorithm has a regret no more than $O\left(\frac{m^3}{\theta_{min}\Delta_{min}}\log(T)\right)$ after T slots, where m represents the number of arms, θ_{min} represents the minimum success rate, and Δ_{min} represents a time gap; which is asymptotically optimal

Li *et al.* [17] propose an addition to the basic MAB model which is more applicable to real-world scenarios. Specifically, multiple arms (known as a super arm) are played simultaneously at each instant and an arm could be sleeping i.e. unavailable at a particular instant. In addition, the authors impose fairness constraints which require each individual arm to be selected for a minimum fraction of plays. To address this situation, the authors extend the UCB algorithm and couple it with a virtual queueing strategy that models the debt remaining to each arm to create the novel Combinatorial Sleeping MAB Model with Fairness Constraints (CSMAB-F) algorithm. The authors mention three applications where this model is applicable, namely real-time network traffic scheduling, internet advertising and crowdsourcing. Simulations prove that this strategy exhibits a trade-off between minimising regret and speed of converging to a point where the fairness constraints are realised. The authors also prove that the algorithm achieves a time-averaged regret of the order $O(\sqrt{\frac{\log T}{T}})$.

In [18], inspired by e-commerce applications, authors Chakarabarti *et al.* study a novel modification of the standard MAB problem where the arms have stochastic lifetimes post which the arms are no longer available and new arms may surface. In the standard MAB problem, the agent reduces exploration and exploits more once an optimal arm has been

identified with sufficient certainty. In contrast to this, in the mortal MAB setting, the agent must constantly explore new arms and select from a large collection of arms which may be more in number than can be explored over their lifetimes. The authors use the example of internet advertising where ads can have limited lifespans due to budget constraints or due to their types and content. They present a number of algorithms that outperform standard MAB algorithms applied to this setting including an optimal one for state-aware scenarios and a near-optimal one for state-oblivious scenarios. Under certain reward distributions, the authors show a lower bound of linear regret.

Most of the literature address MAB problems where all actions can be performed at any time frame. Kleinberg *et al.* [19] consider the setting where the set of available actions vary over time. They study both the Sleeping Experts Problem (or the complete information setting) and the Sleeping MAB problem (or the incomplete information setting). Their work encompasses stochastic and adversarial model of rewards. For all settings, they propose algorithms that almost achieve the theoretically optimal regret bounds with respect to the best-ordering benchmarks. For the stochastic setting their proposed *FTAL* and *AUER* algorithms are computationally efficient and simple in implementation. But for the adversarial setting, no algorithm exists that is computationally efficient and that achieves the regret bounds presented in their work.

Kanade *et al.* [20] study a variant of the bandit problem where in each round, only some of the arms (actions) are available to choose from. This is termed the *sleeping experts* problem, because of the unavailable (sleeping) arms in each round. The performance of the algorithm they propose is measured against the best ordering of choices in hindsight.

Apart from a few exceptions, the real-world applications of MAB are largely unaddressed in the literature. Basık *et al.* [21] consider the problem of fair and efficient task allocation in crowdsourced delivery. They emphasize on the fact that fairness is the key to ensure high worker participation. They propose a 2-phase allocation model, where a set of candidates are chosen in the first phase and the task is offered to the candidates batch-wise. The second phase proceeds with the allocation of the task to the worker. Their proposed *F-Aware*

algorithm ensures fairness in allocation. This is done by keeping the difference between the *Task Allocation Ratio(TAR)*, a metric that they introduce, minimum between the workers. They observe that their F-Aware algorithm is 2.5 times more fair than the least allocated worker assignment strategy and allocates 18% more tasks than the random assignment strategy. However human perspectives of fairness is not considered in their work. Schwartz *et al.* [22] use MAB methods to improve customer acquisition via online advertising. Their strategy achieves an 8% improvement in the customer acquisition rate. Yang *et al.* [23] devise a dynamic contextual MAB algorithm for online advertising that integrates dynamic conversion rate predictions, contextual learning and arm overlapping modelling. Their algorithm performs significantly better than other algorithms for such a setting.

Zhou *et al.* [24] formulate the *cold-start* problem of personalizing recommendations for new users as a contextual bandit problem. They propose the use of a set of latent classes learned from prior users, such that each new user can be described as belonging to one of these classes. This approach generally provides partial personalization to the user.

McInerney *et al.* [25] study personalization of recommendations through an explore-exploit strategy, as well as explaining the recommendations in parallel through a contextual bandit problem. The importance of these explanations to the user is that they provide the context of the recommendation to the users, helping improve user engagement. This is implemented through a contextual-bandits framework (Bart), and challenges in the joint personalization are discussed.

Li *et al.* [26] discuss a contextual bandit framework for personalized news article recommendation which involved selection of articles sequentially based on contextual information, as well as maximising user clicks through adaptation of the article selection policy based on user feedback.

Lu *et al.* [27] study the problem of contextual multi-armed bandits. In addition to the base MAB problem, the algorithm has available some context, i.e. side information, which informs it in the decision-making process. The reward obtained in such a setting depends

not only on the specific arm chosen in a round but also on the context surrounding it. This problem is studied with respect to web search engines, where ads relevant to the specific user incur higher payoffs. The context in such a scenario is the user's query. To tackle the problem of displaying ads that exploit the click-through rate given the user query and history of searches and ad clicks, the authors assume metric spaces for queries and ad and that the payoff function meets a Lipschitz condition for the metric. The authors propose the "query-ad-clustering" algorithm that achieves a $O(T^{\frac{a+b+1}{a+b+2}+\epsilon})$ regret where a and b are the dimensions of the query and ad spaces.

Li *et al.* [28] study hyperparameter optimization for machine learning as a non-stochastic pure-exploration bandit problem with infinitely many arms thereby imposing no limit on the number of configurations in advance. The idea is to allocate resources adaptively to different configurations. Thus, more promising configurations are trained on a larger subset of the data. To this end, the authors propose a novel algorithm called *Hyperband* which uses dataset down-sampling and show it to outperform and random methods on several datasets, even outperforming Bayesian methods in those datasets which benefit in speed from subsampling and *Hyperband* can achieve significant speedups in certain settings.

Agarwal *et al.* [29] propose explore/exploit schemes for a content publishing module on Yahoo! Although previous MAB algorithms do address the exploration-exploitation dilemma, web content poses new challenges due to its very nature, i.e., dynamic sets of items with varying lifetimes, non-stationary rewards (click-through rates) and delayed rewards. For this purpose, the authors develop a Bayesian solution which performs better than adaptations of existing MAB schemes. They also provide insights on how their algorithms may be used to optimise information retrieval and search results for queries.

1.2 Motivation

The original motivation for studying bandits arose from clinical trials, where multiple possible treatments are available for a particular patient, and the best one had to be chosen. Due to the gravity of the situation, a strategy that is better than randomly administering one of the

multiple choices needed to be developed. This led to the study of a wide range of problems encompassed under **Multi Armed Bandits(MAB)** which are essentially resource allocation problems.

Several real-world problems such as internet advertising, crowdsourced task allocation and scheduling of real-time traffic in wireless networks, can all be reduced to the same underlying problem of sequential resource allocation. All of these involve a strategic agent making sequential decisions without having complete knowledge about the outcomes of the different choices. The decisions are made over time or across users of the system. In each instance, the agent can choose to exploit information gathered previously in order to obtain the best known outcome, or to explore relatively uncertain choices in order to gain a better knowledge of their utility.

1.3 Problem Statement

There has been a recent surge in digital marketing done by several organizations and personalization of advertisements. Maximizing revenue is one of the key objectives of every advertiser. Traditional A/B testing methods are offline methods that dedicate a period of time purely to exploration where traffic is equally allocated to the two possible versions. At the end of this, a winner is declared and all further traffic is directed to the winner. The drawback of this approach is that it only allows two choices in each round, and fails to dynamically switch between exploration and exploitation. Hence, a lot of time and revenue is wasted on the losing variant.

On the other hand, MABs successfully balance exploration and exploitation as more knowledge is gained. Unlike A/B testing, MABs can address multiple variants simultaneously, and less resources are spent on the losing variants.

In the context of internet advertisements, the goal is to learn the click-through rates of several competing advertisements in order to converge to the most appropriate set of advertisements for the user in question.

1.4 Objectives

This project aims to study the MAB problem in the context of Internet Advertising with the following objectives:

- A comparison of different bandit algorithms to identify the algorithm that performs best over a wide input space.
- Implementing the CSMAB-F algorithm for combinatorial sleeping MABs.
- To impose additional constraints on the standard MAB problem that model real-world situations, and devise a holistic solution to this problem.
- To introduce contextual information in association with arms to improve the performance of the algorithm.
- To perform regret analysis for the resulting novel algorithm.
- Adapt the algorithm to the internet advertising application

1.5 Methodology

A number of bandit algorithms in popular literature, including UCB and its variants such as UCB1, UCB2, KL-UCB; Thompson Sampling, and Epsilon-Greedy will be implemented and analyzed over a range of inputs. They will further be compared to determine the optimal algorithm for specific input contexts.

Specific bandit models and their applicability to real-world situations will be analyzed. The popular models in litearure include Stochatic Bandits, Contextual Bandits, Sleeping Bandits, Combinatorial Bandits and bandits with constraints.

Following this, the CSMAB-F algorithm proposed in [17] will be implemented and the regret bounds of the same will be analyzed.

A new algorithm will be devised that extends the CSMAB-F with more general reward distributions (eg. nonlinear) and more general fairness constraints (eg. non-temporal), and the

regret bounds of this new algorithm will be analyzed.

1.6 Organization of the Report

This report comprises of nine chapters. Chapter 2 explains some algorithms and concepts that are referenced throughout this work, and Chapter 3 lays out the software requirements specification for the simulation software being developed. Chapter 4 discusses the high level organization of the system as well as some design constraints. Chapter 5 provides a functional description of each module as well as a detailed design of the entire system.

1.7 Summary

This chapter gives a high-level overview of the project. The motivation behind the project has been stated; the objectives and methodology have been clearly listed. This chapter also talks about the recent work that has been done in the field of MABs.

Chapter 2

Overview of Multi Armed Bandits

The multi-armed bandit problem is one of sequential allocation, which defined by a finite set of actions each characterized by a reward [1]. At each time step the strategic agent makes a choice among the actions and allocates a unit resource to an action, obtaining some observable payoff.

The name *bandit* originates from the series of slot machines(bandits) in casinos, among which a player had to choose which one to insert a coin into, repeatedly. This scenario is the same as the resource allocation problem described above, which lead to it's naming as the *multi-armed bandit* problem.

Bandit problems can be seen as sequential decision making problems, with limited information available to the strategic agent at each stage. They bring out the fundamental tradeoff between exploration - evaluating the value of previously unexplored choices - and exploitation, which involves making choices that exploit the best known values so far.

2.1 Basic model

The performance of the strategic agent implementing the bandit strategy is measured in comparison with that of an optimal strategy that consistently plays the arm that is best in the first n timesteps for a horizon of n time steps. This represents the *regret* of the agent for making choices that are not optimal. Formally, with $K \geq 2$ arms and given sequences $X_{i,1}, X_{i,2}, ...$ of unknown rewards associated with each arm $i = 1, ..., K$, agents select an arm I_t at each time step $t = 1, 2, ...$, and receive the associated reward $X_{i,t}$. The regret after n plays $I_1, ..., I_n$ is given by

$$R_n = \max_{i=1,...,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t}. \tag{1}$$

Since both rewards $X_{i,t}$ and agent's choices I_t may be stochastic, average regret may be studied as *expected regret*

$$\mathbb{E}R_n = \mathbb{E}\left[\max_{i=1,...,K}\sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t}\right] \tag{2}$$

and *pseudo-regret*

$$\overline{R}_n = \max_{i=1,...,K}\mathbb{E}\left[\sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t}\right] \tag{3}$$

Every arm $i = 1, ..., K$ corresponds to an unknown probability distribution v_i on [0,1], and the rewards $X_{i,t}$ are drawn independently from the distribution corresponding to the chosen arm. μ_i represents the mean of v_i, i.e. the mean reward of arm i.

$$\mu^* = \max_{i=1,...,K}\mu_i \text{ and } i^* \in \arg\max_{i=1,...,K}\mu_i$$

The mechanism setting the sequence of rewards for each arm is known as the adversary. If this mechanism is independent of the agent's actions the adversary is said to be *oblivious*. If the adversary adapts to the agent's previous behaviour instead, the adversary is *non-oblivious*.

In an adversarial setting, the aim is to obtain regret bounds with high probability or in expectation of any possible randomization of strategies used by either agent or adversary, irrespective of the adversary.

2.2 Stochastic Bandits

The pseudo-regret defined in equation 3 for stochastic bandits can be proven to be equal to

$$\overline{R}_n = n\mu^* - \mathbb{E}\sum_{t=1}^{n}\mu_{I_t}. \tag{4}$$

The pseudo-regret is chosen over expected regret, because competing against the optimal action in expectation is more natural in a stochastic setting, rather than the optimal action

on realized reward sequence. Also, the pseudo-regret can be bounded by $O(logn)$, which is a better bound than the expected regret.

Let $T_i(s) = \sum_{t=1}^{n} \mathbb{1}_{I_t=i}$ denote the number of times the agent chose arm i in the first s rounds. If $\delta_i = \mu^* - \mu_i$ is the suboptimality parameter of arm i, then the pseudo-regret can also be written as

$$\overline{R}_n = \left(\sum_{i=1}^{K} \mathbb{E}T_i(n) \right) \mu^* - \mathbb{E} \sum_{i=1}^{K} T_i(n)\mu_i = \sum_{i=1}^{K} \delta_i \mathbb{E}T_i(n). \tag{5}$$

Typically, an *optimism in the face of uncertainty* principle is employed when solving the stochastic bandit problem, which identifies the most favourable of a set of constructed environments, and makes the decision that is optimal in this environment.

2.3 Combinatorial Bandits

A combinatorial bandit algorithm aims to select a set of arms in each round as opposed to selecting an individual arm This models many real-world scenarios where the task is to select a subset of the available choices. For example, in internet advertising, it is common to display a few rather than a single ad to the visitor. However, the total reward of the selected subset of arms is not a simple sum of the payoffs of individual arms comprising the subset, i.e. the rewards of various arms could be correlated.

Consider a Combinatorial MAB setting where each arm a is a subset of at most m basic actions belonging to a set E. $X_i(n)$ is the random variable that denotes the reward of basic action i in round n $\forall i \in E$. $\theta = (\theta_1, ..., \theta_d)^T \in \Theta = [0, 1]^d$ is the vector of unknown expected rewards of the basic actions, where d is the dimension - number of basic actions available. $\mathcal{A}$ is the set of arms where each element a is comprised of at most m basic actions. Arm a is represented by a binary vector $(a_1, ..., a_d)^T$ with $\|a\|_1 \leq m$. An algorithm or policy π selects an arm $a^\pi(n) \in \mathcal{A}$ in each round n, based on arms chosen in previous rounds and the payoffs observed. The reward of an arm $a^\pi(n)$ is:

$$X^{a^\pi(n)}(n) = \sum_{i \in E} a_i^\pi(n) X_i(n) = a^\pi(n)^T X(n) \tag{6}$$

$\mu_a(\theta) = a^T\theta$ is the expected reward of arm a and $a^*(\theta) \in \arg\max_{a \in \mathcal{A}} \mu_a(\theta)$ is the arm with the maximum expected reward, which may or may not be unique. In the latter case, the tie is broken arbitrarily.

There are two types of feedback, *semi-bandit feedback* where in each round n, the outcome of basic action $X_i(n)\ \forall i \in a^\pi(n)$ are made known to the agent, and *bandit feedback* where only $a^\pi(n)^T X(n)$ is revealed. The aim is to design a policy that minimizes regret where the regret of policy π is:

$$R_{\pi,T} = \max_{a \in \mathcal{A}} \mathbb{E}\left[\sum_{n=1}^{T} X^a(n)\right] - \mathbb{E}\left[\sum_{n=1}^{T} X^{a^\pi(n)}(n)\right] \tag{7}$$

2.4 Contextual Bandits

Contextual Bandits is a bandit problem where each there is side-information associated with the agent and each arm. A large number of online advertising systems suffer from the *cold-start* problem, owing to the fact that a large number of users are completely new to the system. This issue can be formulated as a contextual bandit problem, where each arm and the agent is associated with side-information or *context*. Optimality of a solution is defined with respect to the best policy that maps the context to the arms. The policy space is generally chosen to have a particular structure.

Contextual bandits naturally arise in the online advertising problem. Each user is associated with a certain history, geolocation, timestamp(collectively forms the context); which has to be considered while choosing the best ads. For example, an advertisement which informs people about a sale at a certain store in Bangalore should not be of much importance to the people in Delhi. So, a user in Delhi should not be presented with such an advertisement.

The contextual bandit can be defined formally [1], and there also exists a notion of regret specific to this problem called the *contextual regret*. From a given context set $\mathcal{S}$, each round $t = 1, 2, ...$ is associated with a context $s_1, s2, ...$ The agent should learn the best mapping $g : \mathcal{S} \to \{1, ..., K\}$ of context to the arms. The pseudo-regret can be defined as

$$\overline{R}_n^{\mathcal{S}} = \max_{g:\mathcal{S}\to\{1,...,K\}} \mathbb{E}\left[\sum_{t=1}^{n} \ell_{I_t,t} - \sum_{t=1}^{n} \ell_{g(s_t),t}\right] \tag{8}$$

Here, $s_t \in \mathcal{S}$ denotes the context at round t, and $\ell_{i,t}$ denotes the loss of arm i at round t.

2.5 Sleeping Bandits

In online learning problems, it is generally assumed that the arms are available at all time. But this assumption may not be appropriate in many cases. For example, in an e-commerce site, some product recommendations should not be made because the product is out of stock or has been temporarily removed from sale by the vendor.

In this project, the assumption that all arms are available is relaxed. The regret for such a setting is to be calculated based on the best available arms, because the best arm might not be available at all times.

2.6 Bandit Algorithms

2.6.1 Upper Confidence Bound(UCB)

The UCB algorithm is based on the *optimism in the face of uncertainty* principle discussed earlier, which involves choosing actions as though the environment is as nice as is plausibly possible. In the bandit context, this simplifies to choosing the arm that has the highest possible reward based on the estimated confidence bounds, which are updated after every time step based on the realization of rewards at each stage.

If $X_1, X_2, ..., X_n$ are independent, where $\mathbb{E}[X_i] = 0, \forall i < n$, and $\hat{\mu} = \sum_{t=1}^{n} X_t/n$, then

$$\mathbb{P}(\hat{\mu} \geq \epsilon) \leq exp(-n\epsilon^2/2) \tag{9}$$

On equating the right hand side to δ and solving for ϵ, we get

$$\mathbb{P}\left(\hat{\mu} \geq \sqrt{\frac{2}{n}\log\left(\frac{1}{\delta}\right)}\right) \leq \delta \tag{10}$$

This is used to define the confidence bound on the estimate of arm i in round t, having observed the rewards for all previous rounds, as

$$\hat{\mu}_i(t) = \hat{\mu}_i(t-1) + \sqrt{\frac{2}{T_i(t-1)}\log\left(\frac{1}{\delta}\right)} \tag{11}$$

The arm i that maximizes the above estimate is chosen in round t, represented by A_t.

$$A_t = \begin{cases} \arg\max_i \left(\hat{\mu}_i(t-1) + \sqrt{\frac{2\log f(t)}{T_i(t-1)}}\right), & \text{if t > K;} \\ t, & \text{otherwise.} \end{cases} \tag{12}$$

2.6.2 Thompson Sampling

Thompson Sampling, also known as posterior sampling or probability matching, is a heuristic for solving the MAB problem. It begins with establishing a prior belief of the expected reward parameters for each arm, which is refined over future rounds. In each round, the unknown parameters are sampled from these prior distributions. These sampled parameters yield a set of expected rewards for the arms, from which the arm with the maximum expected return under the sampled parameters is chosen. The distribution for the chosen arm is then updated based on the true reward generated when the arm is played, thus refining the prior using Bayesian inference to update the posterior based on the prior and likelihood of the observations given the parameters. On the face of it, it seems as though Thompson Sampling strives to maximise the immediate reward in each time step, but due to the random sampling of parameters, there is an innate trade-off between maximising the present reward (exploitation) and searching for more information (exploration). As more rounds are played, the posterior converges to the true parameter values, thus decreasing the exploration as more information is attained.

Consider the Bernoulli bandit scenario with K actions, each of which results in either a suc-

cess or a failure when taken. Each action $k \in 1, ..., K$ has an unknown success probability, i.e. mean reward of $0 \leq \theta_k \leq 1$. These probabilities $(\theta_1, ..., \theta_K)$ are unknown to the decision maker but they are fixed over time making adaptive learning feasible. In each time period, an action a_t is played generating a reward $r_t \in 0, 1$ with probability $\mathbb{P}(r_t = 1|a_t, \theta) = \theta_a$.

Suppose the agent holds an independent prior belief over each θ_k. Assume these priors are beta-distributions parameterised by $\alpha = (\alpha_1, ..., \alpha_K)$ and $\beta = (\beta_1, ..., \beta_K)$, i.e. the prior probability density function is:

$$p(\theta_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \theta_k^{\alpha_k - 1}(1 - \theta_k)^{\beta_k - 1} \tag{13}$$

where Γ denotes the gamma function. A beta distribution with parameters (α_k, β_k) has a mean $\alpha_k/(\alpha_k + \beta_k)$. As information is obtained from observations over time, the distributions are updated using Bayes' Rule. Beta distributions are favourable since their posteriors are also beta distributed. The parameters can thus be revised according to:

$$(\alpha_k, \beta_k) \leftarrow \begin{cases} (\alpha_k, \beta_k) & if a_t \neq k \\ (\alpha_k, \beta_k) + (r_t, 1 - r_t) & if a_t = k \end{cases} \tag{14}$$

The parameters are only updated for the chosen action whose reward is observed in that round. In future rounds, the agent draws samples from previously updated distributions and chooses the action that yields the highest reward according to its maintained knowledge.

2.7 Ads

Traditional A/B testing or split-testing is a method of choosing a solution by comparing two variants of an entity, where one variant A is shown to a subset of users and the other variant B is shown to another subset of users at the same time. The comparison is done on the basis of conversion rates, example which user group exhibited a higher positive response to its assigned variant. In the context of internet advertising, this means that one advertisement A is shown to one segment of users and the other advertisement B is shown to another segment

of users to compare the click-through rates based on whether the user clicked the ad or not averaged over all the impressions of the ad. If advertisement A shows a higher click-through rate, then it is declared the winner and the advertisement A is shown either to all future website visitors or until another instance of A/B testing is performed.

This method has some obvious flaws. Firstly, It is hard to tell whether the difference between the variants of ads is because of randomness or if one variant is truly better than the other. For example, if an A/B test is run for 1000 iterations, and both variations get sent to 500 users each. 105 users click on variant A, and 100 click on variant B. In this case, A seems better, but not by much. Maybe A happened to be sent to more users who were likely to click on it, or maybe A genuinely is better. Also, the A/B test is traditionally restricted to testing two variants thus requiring a series of A/B tests to be performed to test multiple variants. The A/B test consists of a short period of pure exploration where 50% of the traffic is directed towards each variant followed by a long period of pure exploitation where the entire traffic is directed towards the winner. This has two disadvantages - during the exploratory phase, resources are wasted on inferior options in the quest to gather as much data as possible and, the transition between phases is discrete.

These disadvantages can be overcome using multi-armed bandits which is suitable for multiple variants. MABs are capable of exploration-exploitation tradeoff and they can interleave the two phases simultaneously over time, thus enabling the MAB strategy to adapt better to changes in the set of advertisements available over time without wasting a significant portion of resources on inferior variants. They move traffic towards the winning variant gradually. In addition, using contextual MABs allows personalised recommendations to be made to users which can further improve click-through rates.

2.8 Summary

This chapter introduces the basic concepts of Multi Armed Bandits and its variants. The two most popular bandit algorithms has been explained in detail. Also, this chapter shows why MABs are better than A/B testing methods for Internet Advertising.

Chapter 3

Software Requirements Specification

This section details the general requirements of the system being designed, and all the considerations to be kept in mind while designing the system. It details the functional as well as non-functional requirements of the system. Functional requirements describe the end functionalities of the system such that it meets the desired design parameters. Non-functional requirements, on the other hand, are technical parameters that need to be met for proper functioning of the system. They define system properties such as security and reliability, response time and scalability.

3.1 Overall Description

This section describes the general characteristics that influence the system, and its design parameters. The system being developed must allow for advertisers to introduce their ads into the system and observe performance results; and for users to be able to view the results of the algorithm on data of their choice with various parameters. Also discussed are other factors such as user characteristics and general constraints.

3.1.1 Product Perspective

The system must provide a user-friendly interface so as to enhance the overall user experience of using the software. The response time of the system must be as small as possible, and the system must be able to source relevant ads in real time with minimal delay. To achieve this, the various modules of the system must be well coordinated and synchronised with each other.

3.1.2 Product Functions

There are four main functions to the software being designed. Users must be able to introduce data to the system and modify it as desired, for the algorithm to run on. Secondly,

users must be able to change the algorithm simulation parameters as they need. Further, the system must be able to select the optimal set of ads to be displayed for each search query, based on various factors including both user as well as ad parameters. The system must also visualize the results of the algorithm performance in a manner that is easy to understand.

3.1.3 User Characteristics

The users of this product are primarily researchers who are interested in testing the performance of the algorithm subject to varying environment parameters and constraints. Also, advertisers can observe trends of ad display and user behaviour, and use this to optimize their ad parameters for best performance.

3.1.4 Constraints and Dependencies

The display of ads to users is directly factored by the budget of the advertiser, as well as fairness to all advertisers with respect to their budgets. The performance of the system depends upon the nature of the data being used even though the algorithm developed is independent of the specific application and can be adapted as required.

3.2 Specific Requirements

This section outlines the specific requirements of the software being developed. The software being developed should have good supportability, clear functional requirements, and good performance.

3.2.1 Functional Requirements

These are the technical requirements which specify the desired behaviour of the system and its components necessary for fulfilling the user's expectations.

- The system must be have an option to accept a custom dataset from the user according to the specified format.
- The system should enable users to select a bandit algorithm to be run.

- The system should enable the user to tune the algorithm parameters for the specific instance.
- The system should analyse the performance of the algorithm relative to a benchmark.
- The system should be capable of generating relevant visualization to convey the results clearly.
- The system should have a provision to compare various bandit algorithms.

3.2.2 Performance Requirements

- The system must run efficiently on any platform that has Python.
- The tests must complete in feasible time
- Memory should be efficiently used such that the code runs even on standard systems.

3.2.3 Supportability

The system is Python based, and works on all devices that have Python installed. Since the system involves graphs for visualization, basic display facilities should be available.

3.2.4 Software Requirements

The various software requirements for the system are listed as follows:

- Python 3 for running experiments and visualizing results.
- Front end-HTML, CSS
- Operating system-Windows XP, Windows 7,8,10
- Web Browser- Google Chrome, Mozilla Firefox, Internet Explorer, Safari

3.2.5 Hardware Requirements

The minimum hardware requirements of the system are as follows:

- Quad Core Processor or better
- 8+ GB RAM
- 2 GB Hard Disk Space

3.2.6 Design Constraints

As the project is research oriented, less focus is given to the UI. Thus, instead of a ad recommender system which allows for a user to enter queries and view ads, a decision was made to create a simulation and analysis framework for visualization of the results.

3.2.7 Interfaces

This section talks about the user and software interfaces present in the system.

3.2.7.1 User Interfaces of the system

- Provision should be made for the user to upload a dataset.
- The user must be presented with options for the bandit algorithm and the associated parameters.
- Visuals such as graphs and relevant evaluation metrics should be displayed to the user.

3.2.7.2 Software Interfaces of the system

- An interface exists between the front end where the user interacts with the system, and the backend Python script that runs the bandit algorithm according to the user's specifications.
- Within the backend code, interfaces exist between the data collection and ad selection modules, as well as between the ad selection and reward mapping modules.

3.2.8 Non-Functional Requirements

These are requirements that aren't specific to the system's functionality, but they enhance the user experience and system quality.

- Availability : The system must be accessible to users without any downtime.
- Efficiency : The system should efficiently manage its resources.
- Uniformity : The application must be able to run on any standard platform with an accompanying browser.
- Speed : The system must be able to run a large number of trials in a short span of time.
- Maintainability : As the research progresses, the system must be modified to include the new findings such as new algorithms.

3.3 Summary

This section describes the system in terms of the functionality, the characteristics it possesses as well as the limitations that confine the scope of design. It also highlights the system requirements and interfaces involved.

Chapter 4

High Level Design of ARS

4.1 Design Considerations

This section discusses the general constraints that need to be kept in consideration when developing the system in question. Further, the approaches to development, methods and strategies used are outlined in section 4.1.2.

4.1.1 General Constraints

The display of ads to users is directly factored by the budget of the advertiser, as well as fairness to all advertisers with respect to their budgets. The performance of the system depends upon the nature of the data being used even though the algorithm developed is independent of the specific application and can be adapted as required.

4.1.2 Development methods

The functioning of this system depends on the base algorithm being used, and thus an incremental approach to software development was employed. At each stage, the software was designed and tested incrementally, until the development process was completed. This approach focuses on both development and maintenance.

4.2 Architectural Strategies

This section brings out the various strategies that were employed in the carrying out of this project to its completion.

4.2.1 Programming Language

The development of this system was carried out in Python, for a number of reasons. Firstly, Python is a very simple scripting language that does not involve complicated constructs,

which increases the readability of the code. Further, a number of python libraries provide functionality that is extremely useful to our work.

- **NumPy** : Numerical Python facilitates storing & manipulating large matrices efficiently, and also supports vectorized operations.
- **SciPy** : Scientific Python provides modules that generate data based on specific distributions, that are essential to our research and testing.
- **TensorFlow** : This package is a math and machine learning library that we used to develop a large-scale testing framework that runs efficiently on large amounts of test data.
- **pandas** : pandas is a popular python package that supports working with large datasets, including curating and manipulating the datasets. Given the large amounts of data we worked with, this package proved invaluable to efficient testing of our algorithm.
- **matplotlib** : This package aided us in visualization of our results in an easy-to-understand manner.

4.2.2 User Interface Paradigm

The user interface for this system provides functionality for selecting a dataset and modifying parameters of the run as desired. They can also choose the base algorithm they wish to run on the data. Fields will be provided for the essential algorithmic parameters. Once a simulation has completed, the results will be displayed to the user primarily in the form of graphs, along with other performance and accuracy metrics.

4.2.3 Error Detection and Recovery

The ads being submitted need to be checked for legality of content, as well as relevance of the advertisement to the title and truthfulness of the ad. Data needs to be cleaned before processing in order to ensure that no missing values cause the algorithm to malfunction.

4.2.4 Data Storage Management

The data to be used is stored locally on the system, which includes the algorithm state that makes use of user as well as ad context. This data is used to improve the accuracy of recommendations for internet advertisements. Further, the virtual fairness queue is also stored that helps to ensure that all advertisers are represented in proportion to their budget.

4.3 System Architecture

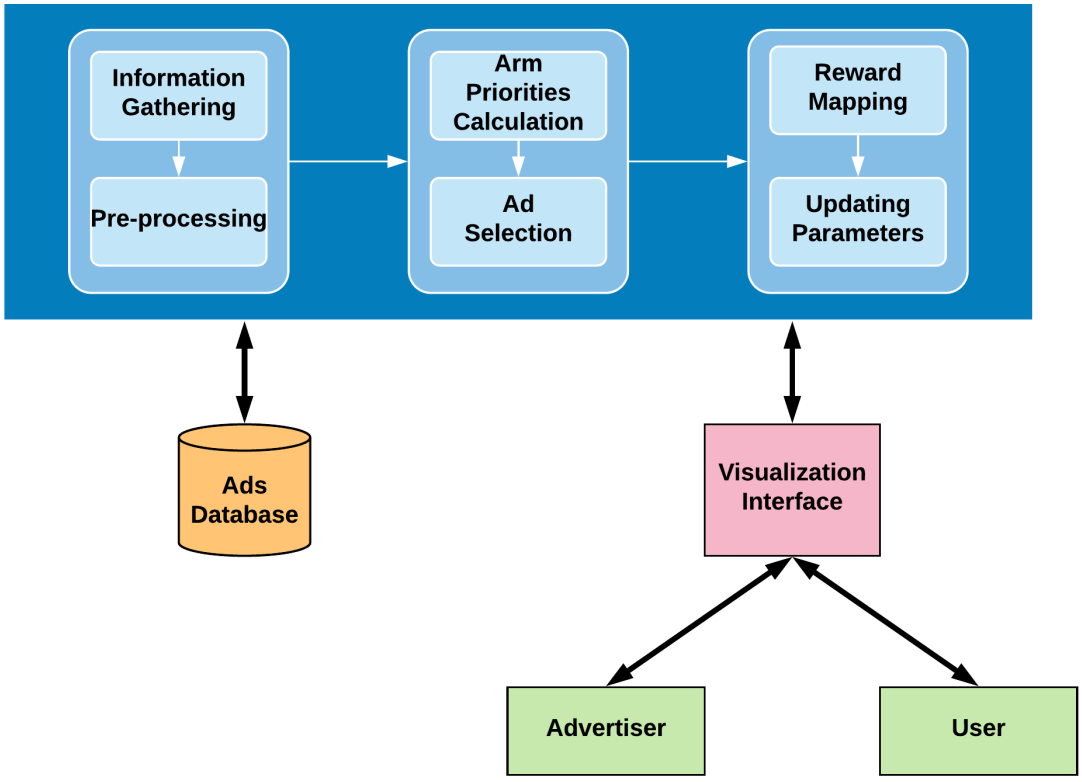

Figure 4.1: System Architecture

The high level organization of this system is described in this section, as represented by Fig. 4.1. The users of the system, who may be advertisers or researchers running experiments, interact with the visualization interface that allows them to select the algorithm and parameters as well as upload their dataset in a compatible format, and displays the results of the

simulation in an easy-to-understand format. The data is stored in the Ads Database and this is used by the processing modules as and when required. There are three main modules involved in running the system - the data collection module, the ad selection module, and the reward mapping and parameter updating module.

The data collection module carries out information gathering from various sources, followed by vectorization and normalization as part of pre-processing which brings the data into the required format. Following this, arm priorities are calculated for all the ads in the database, and the ads are chosen based on the algorithm's estimation of rewards. The final step involves mapping rewards to the chosen arms based on knowledge of the distributions for each arm, and updating all the algorithm parameters and the virtual queue based on the chosen arms and observed rewards.

4.4 Dataflow Diagram

4.4.1 DFD Level 0

The system (Fig. 4.2) interacts with two external entities for the purpose of advertisement recommendation. End-users view ads that are recommended by the system, and decide whether or not to click on the ad. User context is generally derived from the search query as well as user history and location. Advertisers provide ads along with context information to the system, and define a budget based on which the ads are chosen.

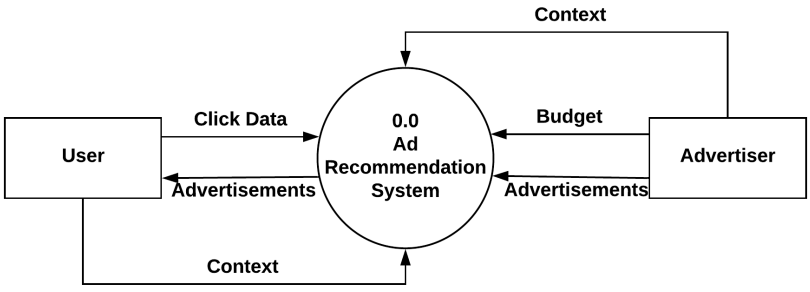

Figure 4.2: Data Flow Diagram – Level 0

4.4.2 DFD Level 1

The ad recommendation system (Fig. 4.3) comprises of three modules - Information Gathering & Pre-processing, Ad Selection, and Reward Mapping & Updating Parameters.

The information gathering module carries out data collection from various sources, followed by vectorization and normalization as part of pre-processing which brings the data into the required format. The Ad Selection module retrieves data history from the previous run, following which arm priorities are calculated for all the ads in the database. The ads to be displayed are chosen based on the algorithm's estimation of optimal rewards based on arm priorities. The final module involves mapping rewards to the chosen arms based on knowledge of the distributions for each arm, and updating all the algorithm parameters and the virtual queue based on the chosen arms and observed rewards.

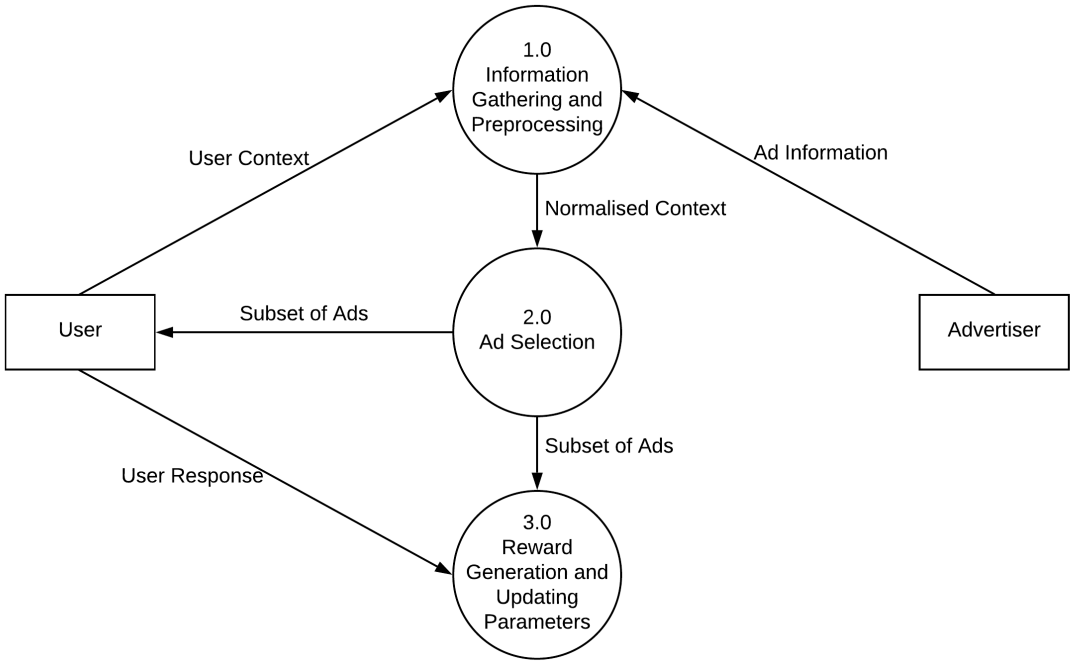

Figure 4.3: Data Flow Diagram – Level 1

4.4.3 DFD Level 2

The data collection & storage submodule represented in Fig. 4.4 receives the user context and ad information and stores them in the Ads database, with data including user history, search query, location, etc. The context is vectorized for ease of manipulation as the underlying algorithm requires all data in numeric format. The normalization submodule generates a normalized context vector to be fed to the next module.

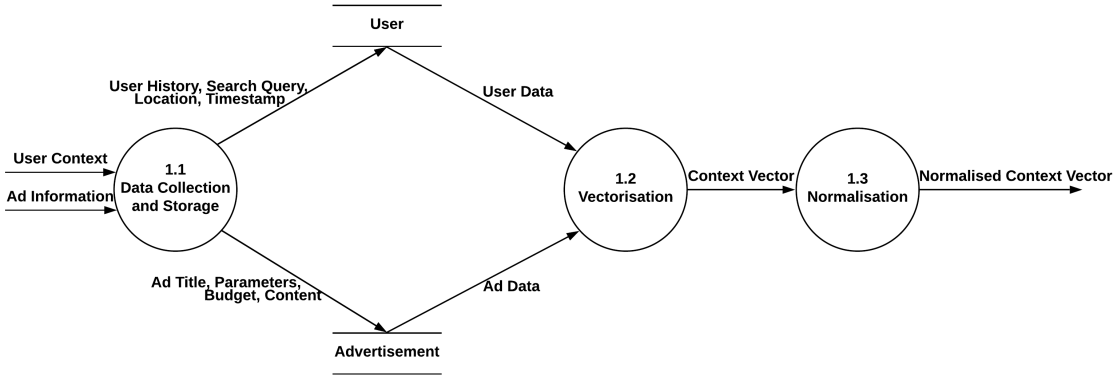

Figure 4.4: Data Flow Diagram – Level 2 (Information Gathering and Processing)

The Ad Selection module (Fig. 4.5) is central to the entire system, and implements the base algorithm on the chosen data. Stored parameters including the virtual fairness queue and the algorithm state are retrieved. The estimate, context and queue vectors are filtered based on availability of ads in each time step, and consolidated to form the knowledge base. Arm priorities are computed based on the available information retrieved, maintaining the exploration-exploitation trade-off. The ads to be displayed in the current time step are then chosen based on the arms with optimal expected rewards based on priorities computed.

The Reward Mapping and Updating Parameters module (Fig. 4.6) plays the selected subset of arms and the corresponding rewards are observed. Based on the observed rewards, the prior beliefs about the selected arms are updated to reflect the true distributions incrementally better at each time step, and regret is computed. Further, the virtual queueing system is also updated based on the chosen subset of ads.

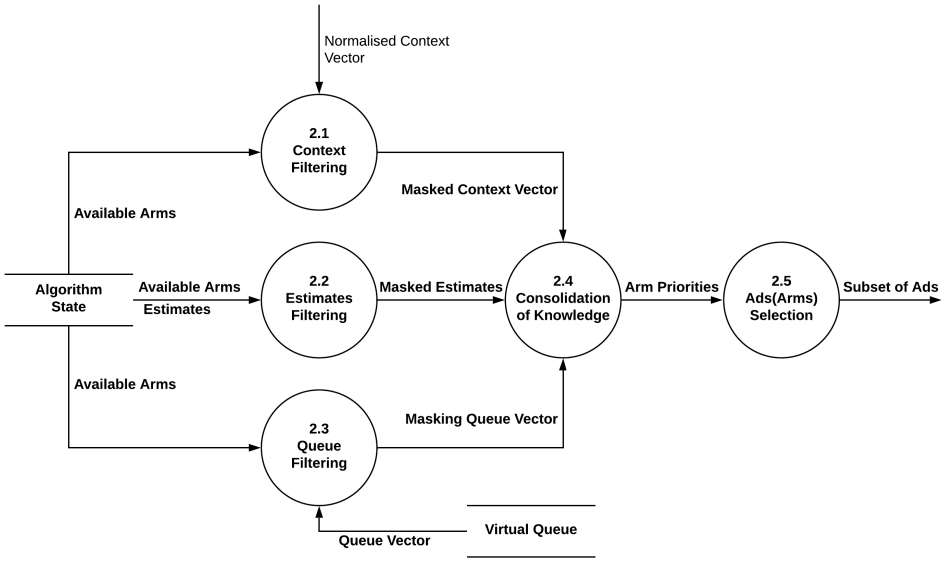

Figure 4.5: Data Flow Diagram – Level 2 (Ad Selection)

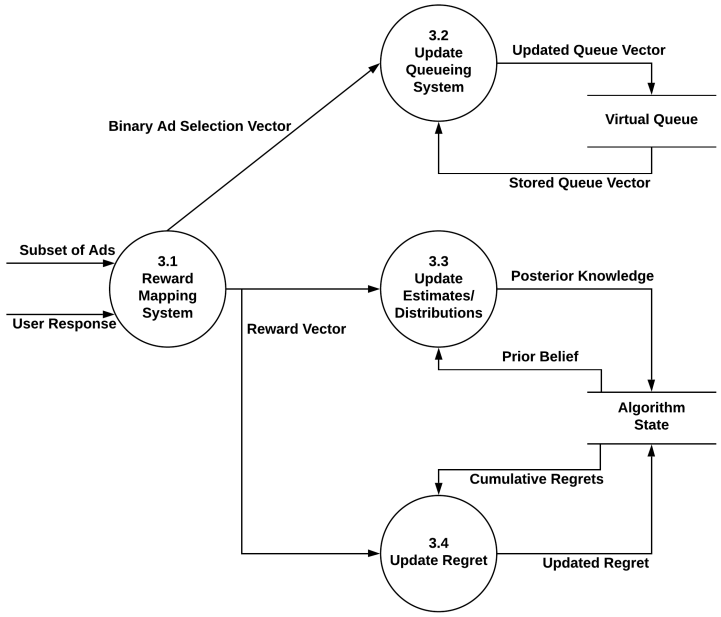

Figure 4.6: Data Flow Diagram – Level 2 (Reward Mapping and Updating Parameters)

4.5 Summary

This chapter gives a high level representation of the system-the entities present, the processes involved and their inter-relation. The system architecture portrays the high level organisation of the system, and the Data Flow Diagrams describes the high-level processes and how the data flows between processes.

Chapter 5

Detailed Design of ARS

This chapter discusses in detail the implementation of every module of the system. It talks about the input, the output and the processes involved in each module. This aids in understanding better the functionality of each individual component. This also helps explain the interrelation between various modules.

5.1 Structure Chart of Ad Recommendation System

A structure chart depicts the data & control flow among the various modules and submodules of a system. It also maps out the dependencies and interactions among the modules at various levels of the hierarchy, specifically the inputs and outputs to each module.

The Information Gathering module makes use of the incoming user and ad context information to generate a normalised context vector for use by the algorithm by passing the user and ad data among the sub-modules at the next lower level of hierarchy generating an intermediate context vector before normalisation. The second module in the Ad Recommendation System is Ad Selection. It uses the normalised vector generated from the previous module to perform the core operations of the system. This is achieved using data inputs from the User, Advertiser, Algorithm State, Queueing System which includes relevant context, queue vector, estimates and available arms. The Filtering sub-module uses the available arms to mask the vectors which are then consolidated to obtain a basis for deciding the subset of arms to be selected in the form of arm priorities. The Arm Selection sub-module then returns the selected subset of arms to be displayed for which the user response is recorded.

The Reward Generation and Updating Parameters module utilises the user response and accompanying information from the previous module to update the instantaneous average/cumulative regret in addition to modifying the estimates of distributions and the fairness queue.

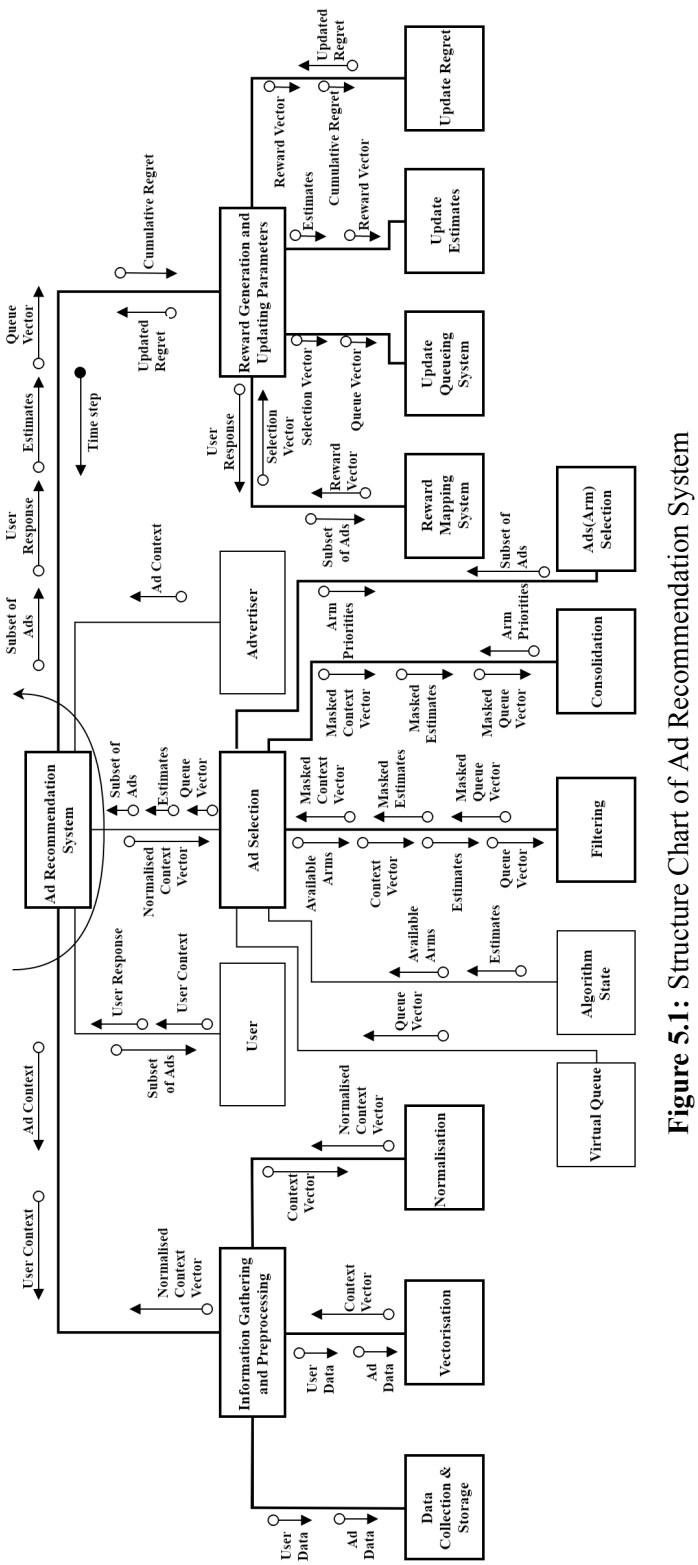

Figure 5.1: Structure Chart of Ad Recommendation System

The control information in the system is mainly in the form of time step records. As depicted in Fig. 5.1, the system iterates over the time horizon in a loop until the end of the experiment which is determined by the total number of time steps provided by the user during initialisation of experimental variables.

5.2 Functional Description of the Modules

5.2.1 Information Gathering and Pre-Processing

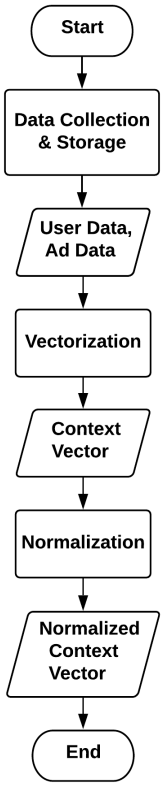

Figure 5.2: Flowchart of Information Gathering and Pre-Processing

Purpose : Consolidation of data and pre-processing for storage in required format.

Functionality : This module receives the user context and ad information and stores them in

the User and Advertisement tables of the Ads database. This data includes but is not limited to user history, search query, location, ad titles and search filters.Non numeric data, which comprises a substantial portion of the context, needs to be converted in to a numeric form as the underlying algorithm processes only numeric data. The context is thus vectorized for ease of manipulation.
Also, as different parameters may be bounded by different values, all the context information needs to be normalised to ensure that no parameter is given undue priority. This is carried out by the normalization submodule, which generates a normalized context vector to be fed to the next module.

Input : User context and ad information

Output : Normalised context vector

Flowchart : The flowchart for this module is shown in Fig. 5.2.

5.2.2 Ad Selection

Purpose : Use available context as well as priority estimates and fairness queue to select a subset of available ads to display.

Functionality : The first step in this module is retrieval of stored parameters including the virtual fairness queue and the algorithm state (estimates, available arms and current time step). The estimate, context and queue vectors are masked based on availability of ads in each time step. After this, the filtered vectors are consolidated to form the knowledge base for the current time step.
Arm priorities are computed based on the context available and the algorithm estimates & fairness queue, with a trade-off between exploration and exploitation decided by a constant runtime parameter. The ads to be displayed in the current time step are then chosen based on the arms with optimal expected rewards based on priorities computed.

Input : Normalised context vector

Output : Subset of ads to be displayed

Flowchart : The flowchart for this module is displayed in Fig. 5.3.

5.2.3 Reward Mapping & Updating Parameters

Purpose : Compute rewards for each chosen ad based on inherent distribution, and update algorithm parameters based on rewards.

Functionality : Initially, the selected subset of arms are played and the corresponding rewards are observed. Based on the observed rewards, the prior beliefs about the selected arms are updated to reflect the true distributions incrementally better at each time step. The regret, which is defined as the difference between the observed rewards and the optimal rewards that could have been obtained if complete information was available, is computed for purposes of analysis.
Further, the virtual queueing system which is used to maintain the notion of fairness is also updated based on the chosen subset of ads, irrespective of the rewards observed. In particular, it stores the debt owed to each arm which increases if the arm is not played, and reduces on each play of the arm.

Input : Subset of ads to be displayed, user context

Output : None

Flowchart : The flowchart for this module is displayed in Fig. 5.4.

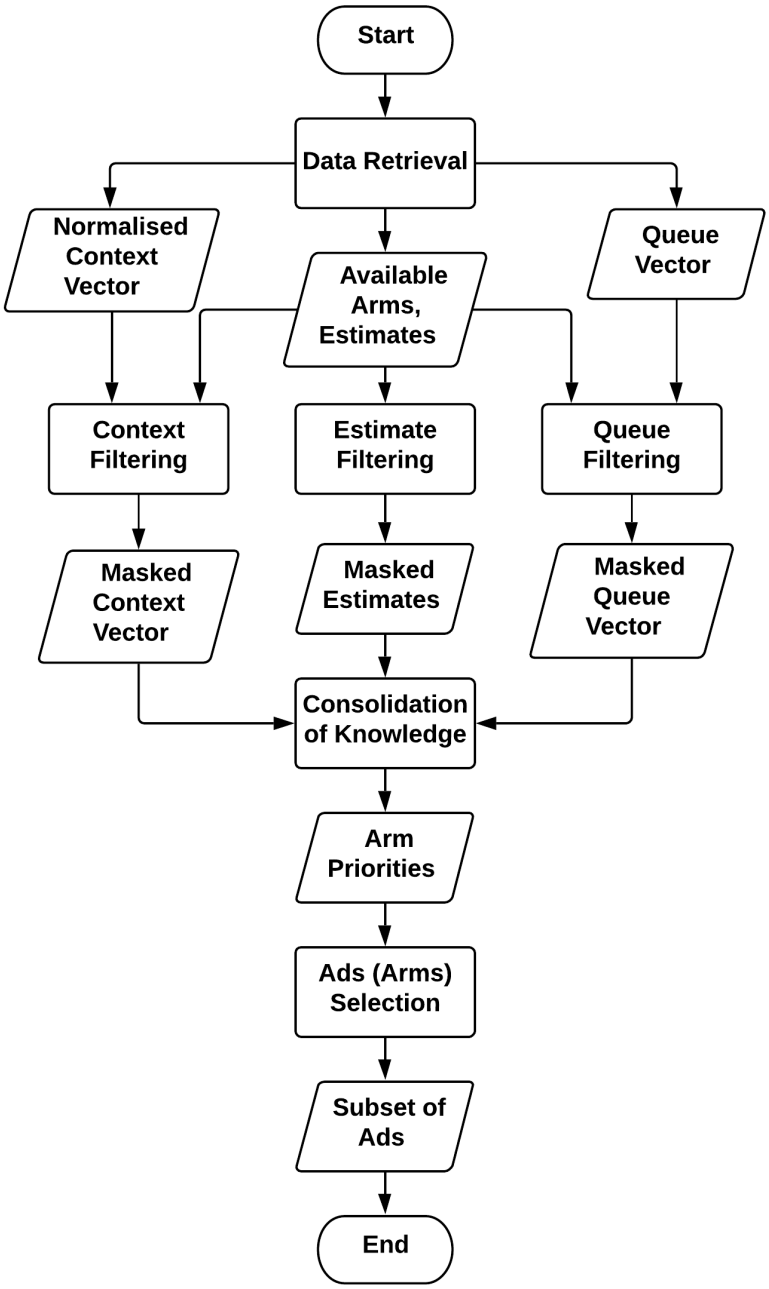

Figure 5.3: Flowchart of Ad Selection

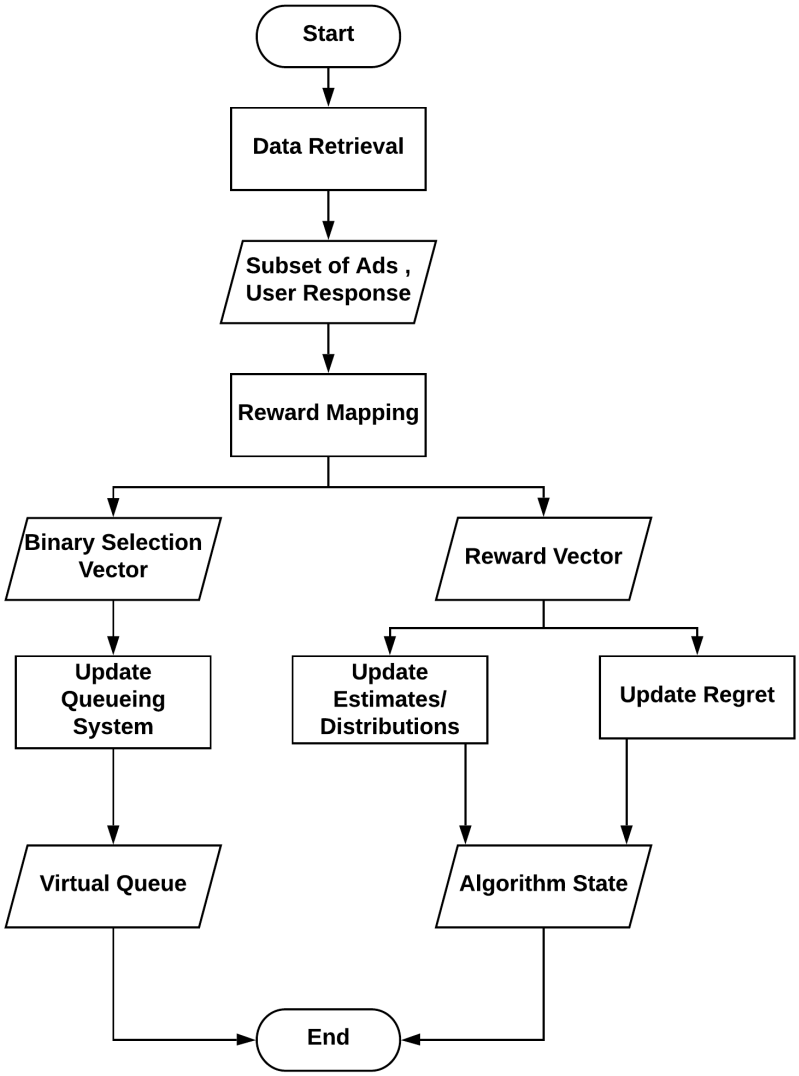

Figure 5.4: Flowchart of Reward Mapping and Updating Parameters

5.3 Summary

The detailed design of the simulation software has been depicted in this chapter. The structure chart shows how data and control flow information flows across the system. The functional description of the modules shows the various steps involved in each module and the relationship that exists between the modules.

Chapter 6

Implementation of Ad Recommendation System

6.1 Programming Language Selection

Python was the programming language of choice, as it has a variety of features that make it ideal for the construction of a framework that allows for development and testing of the proposed algorithm. Some of these are outlined below :

- Python has a number of libraries it supports that proved essential to our work, such as pandas, which helped handle the large amounts of data we worked with efficiently.
- Python has a number of scientific libraries such as NumPy and SciPy that provided us with distributions and support for matrix manipulations that we used very often.
- Matplotlib, another library offered by Python, helped us visualize our results in a clean and straightforward manner.
- Python being a scripting language, has been designed for ease of use by the programmer and this made working with it quite uncomplicated.

6.2 Platform Selection

The system can be deployed on any platform that supports Python and has 16 GB RAM to facilitate acceptable rates for the arm selection and parameter updation processes. The presence of a GPU can accelerate the intensive computations involved and is desirable, especially for the purpose of training complex models.These needs show that instead of every user acquiring this hardware, the application can be deployed on a cloud service and a web-interface can be provided for the user. A cloud storage service can be used for storing the parameter data as well.This is the most suitable platform for the application, but provided the user has the required hardware, the application can be deployed locally as well. The entire application has been developed on Linux and the end result can be run on Linux, Windows

and MacOS.

6.3 Proposed algorithm

The proposed CSLogUCB-F algorithm makes use of context in the Combinatorial Sleeping MAB algorithm with fairness guarantees, through the use of logistic regression to learn the coefficients for the context.

Algorithm 1: CSLogUCB

Input: $\mathbb{A}$, f, x, $m > 0$, $\eta > 0$, $\alpha > 0$

Result: Best m arms

for *all* $a \in \mathbb{A}$ **do**

$A_a \leftarrow I_d$

$q_a \leftarrow 0$

end

for *t=1, 2, 3,...,T* **do**

observe features of all arms a $\in A_t$: $x_{t,a} \in \mathbb{R}^d$

for *all* $a \in \mathbb{A}_t$ **do**

$\hat{\theta}_a, b \leftarrow$ learned from logistic regression

$U_{t,a} \leftarrow (\hat{\theta}_a^T x_{t,a} + b) + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$

$p_{t,a} \leftarrow q_a + \eta U_{t,a}$

end

$a_t \leftarrow$ top m arms with highest $p_{t,a}$

reward $r_{t,a}$ used to update logistic regression model

$A_a \leftarrow A_{a_t} + x_{t,a} x_{t,a}^T$

for $a \in a_t$ **do**

$q_a \leftarrow \tanh((q_a + f_a - 1)/T)$

end

for $a \notin a_t$ **do**

$q_a \leftarrow \tanh((q_a + f_a)/T)$

end

end

6.4 Code Conventions

Code conventions are a set of guidelines provided for a particular programming language or a framework. They give recommendations about the best practices to be followed while writing code. These conventions usually include naming conventions, file organization and comments. Python has a number of standard code conventions, that we followed in the development of our program.

6.4.1 Naming Conventions

Standard Python naming conventions have been followed, with CAPITALIZED_WITH_UNDERSCORES for constants, and lowercase_separated_by_underscores for other names.

6.4.2 File Organization

There is a folder dedicated to store the dataset across files, containing advertisement data, categorical data, user information, click data and user search data. All the python scripts were developed in notebooks and integrated into a final script. The notebooks include standard MAB algorithms such as UCB and Thompson Sampling, the context based algorithms such as LinUCB, the proposed algorithm, etc.

The results of the tests that were run were sorted by the number of bandits in the test run as well as the number of time steps the test was run for.

6.4.3 Comments

Standard Python comment conventions have been followed, with
” ” ”

Enclosed in double quotes
” ” ”

for docstrings, and
#comment
for inline comments.

6.5 Difficulties Encountered and Strategies used to Tackle

Due to the developmental nature of the project and the focus on research, there were a number of difficulties we came across, that we needed to overcome. Some of these are outlined in the following sections.

6.5.1 Dataset Pre-processing

After an extensive search for suitable public dataset to use, we were only able to obtain the Avito Context Ad Clicks dataset [30], which is entirely in Russian. Thus, we were tasked with translating the entire dataset into English using the Google Translate API, which was fairly expensive and also time consuming.

This was necessary in order to form word vectors for use in subsequent modules, and led to us filtering out only the advertisements that had most data points present.

6.5.2 Removing the effect of randomness

Due to the large number of parameters used in the algorithm - many of which are randomly sampled from a distribution - none of the empirical results could be taken at face value, because of the amount of inherent randomness in the tests.

Thus, it was necessary to run the tests over a large number of samples for each parameter, and take an average over these to estimate an accurate result. This was an extremely time-consuming task and also required extensive computational power at our disposal, something we did not have. We thus had to segment the tests and combine the results correctly at the end.

Chapter 7

Software Testing of Ad Recommendation System

7.1 Test Environment

Testing of the system was carried out in Python, with unit test cases run in the development environment itself. The pytest module was used to run the test cases on the unit modules.

7.2 Unit Testing

Unit testing involves individually testing all the components or functional units of the system. This is done to discover errors and bugs in the functional units. The following test cases were carried during the implementation phase of the project.

The unit modules in our project include the information gathering and preprocessing module, the ad selection module, and the reward mapping and parameter updation module.

7.2.1 Unit Testing of Information Gathering & Pre-Processing module

This module takes the dataset as an input, and constructs a context vector which is returned as output. In order for the data to be vectorized accurately, the embedding dimensionality needs to be sufficiently large. This will ensure that no information is lost in the vectorization process.

Table 7.1 displays a case of successful processing of data with an embedding dimensionality of 32, whereas the second testcase (Table 7.2) was run with a dimensionality of 2, which led to the data not being represented accurately in vector space.

Table 7.1: Test case 1

Sl No. of Test Case	1
Name of Test Case	VALID_PROCESSING
Feature being Tested	Information gathering and pre-processing
Description	Retrieving data from the dataset, and converting it to the normalised vector format required
Sample Input	Avito Ads Click dataset, context
Expected Output	Normalised Context Vector
Actual Output	Expected normalised context vector
Remarks	Test case successful

Table 7.2: Test case 2

Sl No. of Test Case	2
Name of Test Case	INVALID_PROCESSING
Feature being Tested	Information gathering and pre-processing
Description	Retrieving data from the dataset, and converting it to the normalised vector format required
Sample Input	Avito Ads Click dataset, context
Expected Output	Normalised context vector
Actual Output	Incorrect normalised context vector
Remarks	Test case successful

7.2.2 Unit Testing of the Ad Selection Module

This module is responsible for using the input context vector to determine which ad is optimal in each time step, based on all available information at that time.

Table 7.3 shows the case where the best advertisement is not chosen. This is because the algorithm still hasn't gained enough information about the the click-through rates of the ads, and thus is in an exploration stage. Table 7.4 displays the case when the algorithm has gathered sufficient information, and thus displays the most optimal advertisement.

Table 7.3: Test case 3

Sl No. of Test Case	3
Name of Test Case	AD_SELECTION_1
Feature being Tested	Ad Selection
Description	Selecting ads based on given context vector, time step 3
Sample Input	$\begin{bmatrix} 0.07 & 0.5 & 0.08 \\ 0.4 & 0.03 & 0.7 \end{bmatrix}$
Expected Output	ad 2
Actual Output	ad 1
Remarks	Test case unsuccessful

Table 7.4: Test case 4

Sl No. of Test Case	4
Name of Test Case	AD_SELECTION_2
Feature being Tested	Ad selection
Description	Selecting ads based on given context vector, time step 1000
Sample Input	$\begin{bmatrix} 0.07 & 0.5 & 0.08 \\ 0.4 & 0.03 & 0.7 \end{bmatrix}$
Expected Output	ad 2
Actual Output	ad 2
Remarks	Test case successful

7.2.3 Unit Testing of the Reward Mapping & Updating Parameters module

This module computes rewards for each chosen ad based on inherent distribution, and updates algorithm parameters based on rewards.

The test case for the reward mapping sub-module is displayed inTable 7.5. Based on the quality of the ad and the context, a reward is generated based on the user's response-which are Bernoulli rewards in this case.

Table 7.6 shows the test case where the algorithm parameters are updated. The estimates, regret and fairness queue are updated based on the advertisements chosen by the algorithm at that time step, and also based on the user's response.

Table 7.5: Test Case 5

Sl No. of Test Case	5
Name of Test Case	REWARD_MAPPING
Feature being Tested	Reward mapping
Description	Getting rewards for chosen arms.
Sample Input	ad 2
Expected Output	1 with probability E(0.76)
Actual Output	1
Remarks	Test case successful

Table 7.6: Test Case 6

Sl No. of Test Case	6
Name of Test Case	UPDATING_PARAMETERS
Feature being Tested	Updating parameters
Description	Updating algorithm parameters based on reward.
Sample Input	ad 2, reward = 1
Expected Output	Estimates, regret and queue updated
Actual Output	Estimates, regret and queue updated
Remarks	Test case successful

7.3 Integration Testing

Integration testing is the process of testing the system as a whole after connecting all the modules. New bugs and errors might get introduced into the system when integration of the components are done. Integration testing is generally done in an incremental fashion.

7.3.1 Integration Testing of Information Processing and Ad Selection modules

The information processing module takes the dataset as an input, and constructs a context vector which is returned as output. In order for the data to be vectorized accurately, the embedding dimensionality needs to be sufficiently large. This will ensure that no information is lost in the vectorization process. The Ad Selection module is responsible for using the input context vector to determine which ad is optimal in each time step, based on all available information at that time.

Table 7.7 shows the test case where search queries from the Avito dataset are fed to the algorithm, and upon generation of the context vectors, the ad selection is correctly performed.

Table 7.7: Test Case 7

Sl No. of Test Case	7
Name of Test Case	PREPROCESSING_AD_SELECTION
Feature being Tested	Information gathering, pre-processing and ad selection
Description	Retrieving information from dataset and choosing ad.
Sample Input	Avito Ads Click dataset, context
Expected Output	Optimal ad
Actual Output	Optimal ad
Remarks	Test case successful

7.3.2 Integration testing of the Ad Selection & Reward Mapping modules

Table 7.8: Test Case 8

Sl No. of Test Case	8
Name of Test Case	AD_SELECTION_REWARD_PARAMETER
Feature being Tested	Ad selection, reward mapping and updating parameters
Description	Selecting ads based on context, observe rewards and updating parameters accordingly.
Sample Input	Context vectors for all arms
Expected Output	Update parameters for optimal arm
Actual Output	Parameters for optimal arm updated
Remarks	Test case successful

The Ad Selection module is responsible for using the input context vector to determine which ad is optimal in each time step, based on all available information at that time. The reward mapping module computes rewards for each chosen ad based on inherent distribution, and updates algorithm parameters based on rewards.

Table 7.8 displays the test case where the context vector for all the arms are input to the ad selection module, which selects the ads that are more likely to generate clicks. The parameters(regret, fairness queue and estimates) for the chosen ads are updated based on whether the ad was clicked or not.

7.4 System Testing

System testing is the process in which the entire system is tested in an end to end manner. In this section, basically all the modules obtained as a result of integration testing are combined together to form a single system All the aspects of the system are tested to prove that the requirements are met by implementation. Hence, the proper functionality of the system is concluded in system testing phase of software testing.

Table 7.9: Test Case 9

Sl No. of Test Case	9
Name of Test Case	CONSTANT_CONTEXT
Feature being Tested	Effect of context on algorithm
Description	Testing the effect of context on the algorithm
Sample Input	Same context across all timesteps
Expected Output	Non-contextual MAB
Actual Output	Non-contextual MAB
Remarks	Test case successful

Table 7.10: Test Case 10

Sl No. of Test Case	10
Name of Test Case	VARIABLE_CONTEXT
Feature being Tested	Effect of context on algorithm
Description	Testing the effect of context on the algorithm.
Sample Input	Variable context across timesteps
Expected Output	Contextual MAB
Actual Output	Contextual MAB
Remarks	Test case successful

Table 7.9 represents the case where the same context vectors are generated at each time step for each ad. This reduces to the trivial non-contextual multi-armed bandit problem-which

the proposed algorithm also solves correctly.

Table 7.10 shows the case where different context vectors are produced at each time step for each ad. The proposed algorithm is also able to solve this correctly.

Chapter 8

Experimental Analysis and Results

8.1 Evaluation Metric

The performance of the strategic agent implementing the bandit strategy is measured in comparison with that of an optimal strategy that consistently plays the arm that is best in the first n timesteps for a horizon of n time steps. This represents the *regret* of the agent for making choices that are not optimal. Formally, with $K \geq 2$ arms and given sequences $X_{i,1}, X_{i,2}, ...$ of unknown rewards associated with each arm $i = 1, ..., K$, agent selects a subset of arms $\mathcal{S}_t \in \mathcal{A}_t$, where $|\mathcal{S}_t| = m$ and $m \leq K$, at each time step $t = 1, 2, ...$, and receives the associated rewards $X_{I_t,t}, \forall I_t \in \mathcal{S}_t$. The regret after n plays $S_1, ..., S_n$ is given by

$$R_n = \sum_{t=1}^{n} r_t^* - \sum_{t=1}^{n} r_t \tag{15}$$

where r_t^* is the sum of the expected rewards obtained if the m most optimal arms are played at time t, and r_t is the sum of the expected rewards of the arms that are actually played at time t.

The expected reward for any arm a is given by

$$\mathbb{E}[X_{I_t}|x_{I_t,t}] = f(x_{I_t,t}^T \, \theta_{I_t}^*) \tag{16}$$

where $f(\cdot)$ is the sigmoid function.

8.2 Experimental Dataset

The algorithm was tested on the Avito Context Ad Clicks Dataset [30], which is a publicly available dataset. This dataset contains details of traffic to an e-commerce website over a span of 16 days.

AdsInfo.csv
AdID
CategoryID
Params
Price
Title
Latitude
Longitude
HistCTR

SearchInfo.csv
SearchID
SearchDate
UserID
SearchQuery
CategoryID
SearchParams
Latitude
Longitude
Budget

SearchStream.csv
SearchID
AdID
HistCTR

Figure 8.1: Data used

The data is organised into a number of data files, of which three were of relevance to our project, as displayed in Figure 8.1. We describe these further.

The AdsInfo file contains the details of every advertisement that was available to be displayed during the period that data was collected in. This includes the Advertisement ID, the title of the ad and the search filters relevant to the ad. Further, the target location of the advertiser (in terms of latitude and longitude) and the price of the product/service being advertised is also mentioned.

The SearchInfo file contains the details of every search query made on the website during the period the data was collected. This includes the query ID, the user's query and the search filters selected. Further, the location of the user submitting the query (in terms of latitude and longitude) and the user's chosen budget is also mentioned.

The SearchStream file contains the details of advertisements displayed for selected search queries from SearchInfo, and user feedback. This includes the search query ID, the corresponding advertisement ID, the estimated click through rates for that display of the ad, and the user

8.3 Performance Analysis

The performance of the algorithms have been analysed using Cumulative Regret. This parameter begins at a value of zero and in each time step, the regret incurred is added to it to represent the total regret so far. It is expected that for a good algorithm, once the algorithm has gained sufficient information and has started to pick optimal or near-optimal arms, the regret added at each time step is zero or negligible. This results in the cumulative regret curve across time steps eventually plateauing after an initial increase as the algorithm learns optimal behaviour. The following section shows cumulative regret plots for various benchmark algorithms and shows how the proposed algorithm performs in comparison.

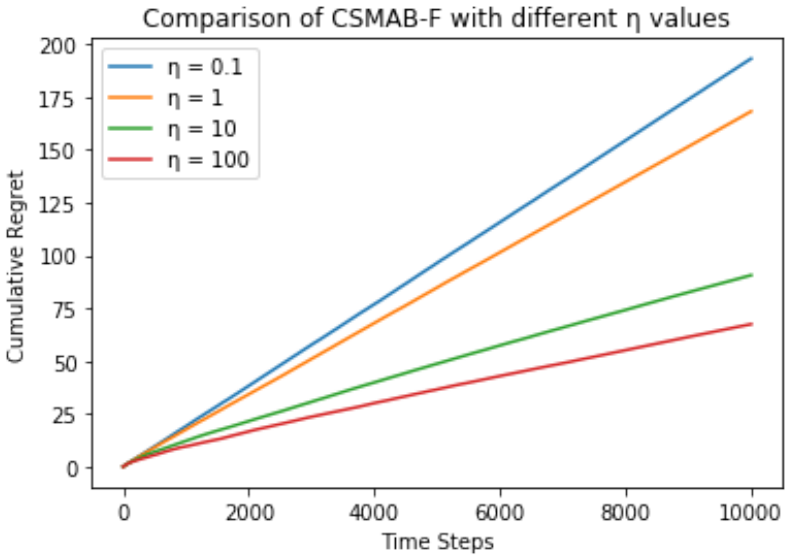

Figure 8.2: Comparison of CSMAB-F with different η values

Figure 8.2 shows the plot of cumulative regrets versus time for the algorithm proposed in [17] for various values of η. η is the parameter which balances the weightage of UCB estimates of the arm and the fairness parameter. Higher values of η imply more importance to the UCB estimate thus resulting in lower regret which is clearly shown.

Figure 8.3 shows the plot of cumulative regrets versus time for our proposed algorithm for various values of η. It also compares the performance of the proposed algorithm without the

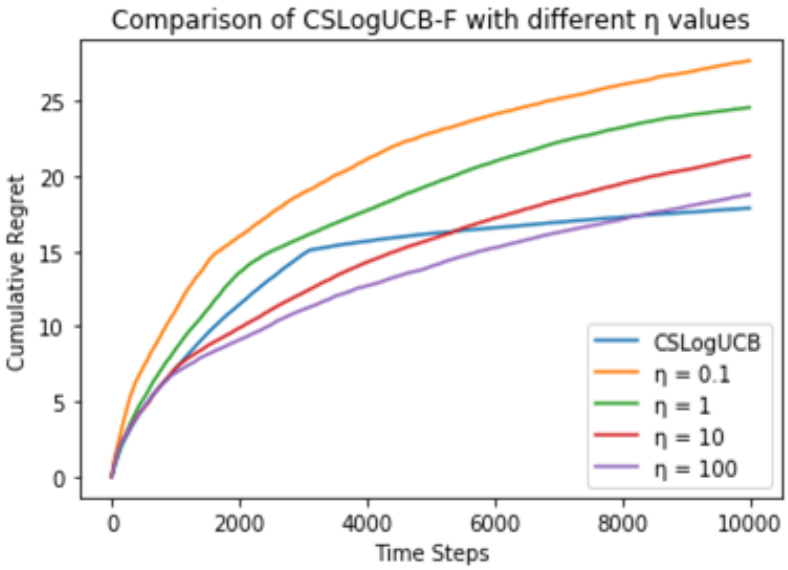

Figure 8.3: Comparison of CSLogUCB-F with different η values

fairness constraint and shows that when there are no fairness constraints to be considered, the algorithm is free to simply choose the best arms at each time step thereby acquiring lower regret.

Figure 8.4 compares the performances of the algorithm in [17] and our proposed algorithm. It can be seen that the latter significantly outperforms the former based on empirical results. This is because our algorithm considers the contextual information available at each time step which can better inform the algorithm in the comparison of arms. It can also be attributed to the fairness notion used in the CSMAB-F algorithm which causes the system to cease learning the best arms and instead focuses purely on satisfying the imposed fairness constraints. We develop an alternative notion of fairness that ensures that the balance between the estimates and fairness requirement is kept in check throughout the time horizon by considering the effects that could accumulate over time due to the fairness restriction.

Figure 8.5 shows the relative performances of the LinUCB algorithm in [26] and our proposed algorithm. LinUCB is a contextual algorithm but it is not suitable to our application of internet advertising since it has been developed for a system with continuous rewards

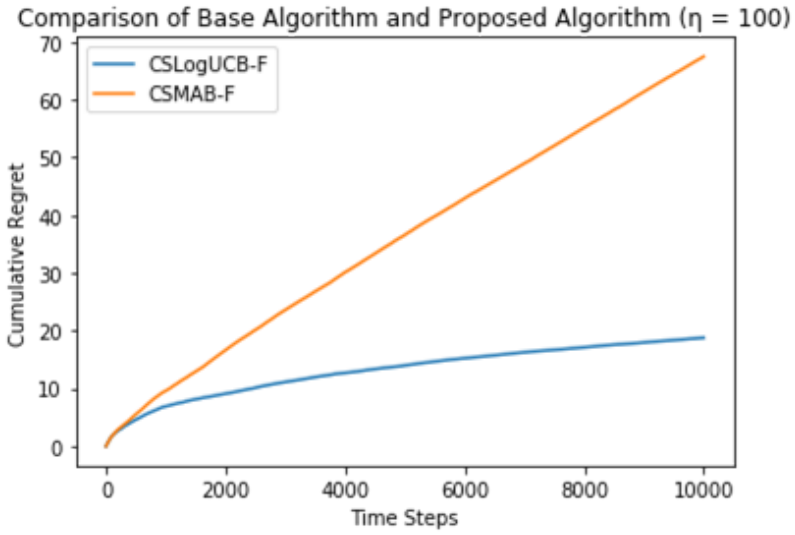

Figure 8.4: Comparison of CSMAB-F and CSLogUCB-F Algorithms

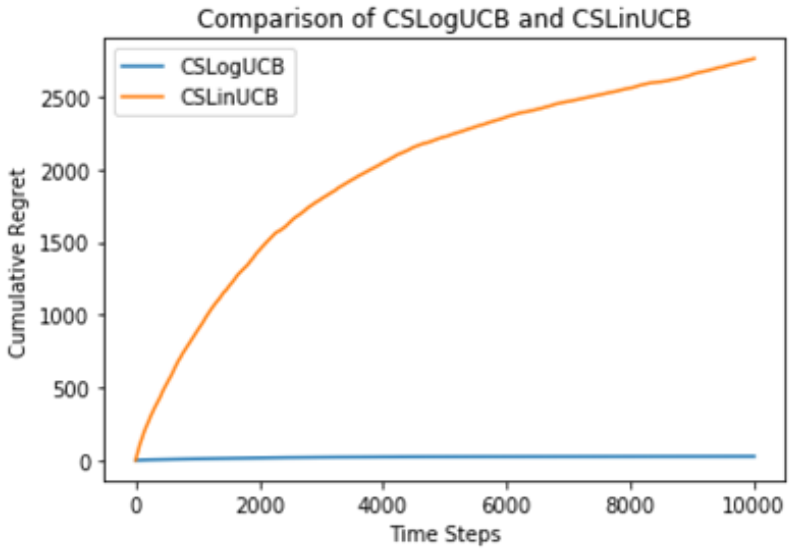

Figure 8.5: Comparison of LinUCB and CSLogUCB-F Algorithms

whereas our application is an inherently binary system that is best represented by Bernoulli rewards $\{0, 1\}$. This is taken care of in our proposed algorithm where we use Logistic Regression instead of Linear Regression.

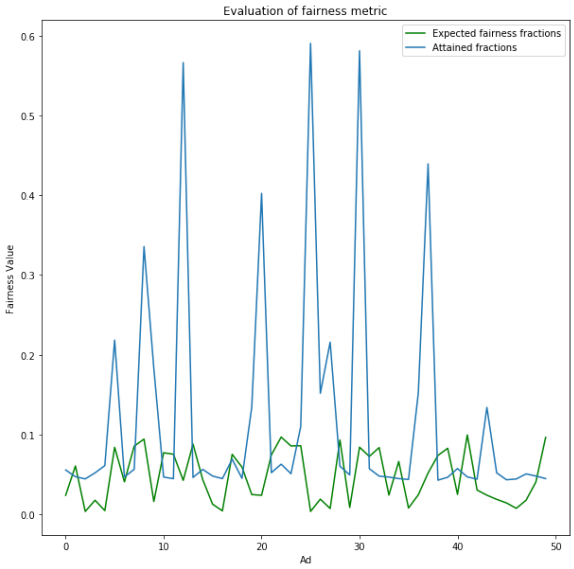

Figure 8.6: Evaluation of Fairness Metric

Figure 8.6 shows that our algorithm satisfies the fairness constraint. It can be seen that the fairness fractions imposed have been attained over the time horizon for most of the arms. For the best ads, it can be seen that their display fractions exceed the lower bound of the fractions imposed by the fairness constraint as expected while providing a fair chance for the rest of the ads as well.

Chapter 9

Conclusion

In the general MAB setting without the use of context to determine relevance, advertisement recommendations may not have reflected the true estimates of each arm specific to the current context. The introduction of context into CSMAB algorithm, resulting in the proposed algorithm being developed, helped reduce the regret incurred through the entire run across time steps empirically.

An improved fairness notion was proposed that models the real world scenarios more accurately was introduced, and was found to reduce the regret based on the experiments conducted. Logistic regression was used to learn weights for the context, which was also a factor in the reduced regret.

9.1 Limitations of the Project

Some of the limitations of the proposed algorithm include :

- The regret bounds for the proposed algorithm have been evaluated empirically, but are yet to be mathematically proven.
- The fairness notion that has been proposed requires a knowledge of the duration of the algorithm being run, which may not always be known in advance.
- There is no closed form solution for logistic regression as there is for linear regression, resulting in the scalability of the model not being as optimal as desired.

9.2 Future Enhancement

Some possible additions to the project include :

- Performing a thorough theoretical analysis of the regret bounds of the proposed algorithm.
- Exploring various other notions of fairness to determine if a more efficient one can be implemented.
- Incorporating additional parameters into the context vector in order to better represent context.

References

[1] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multi-armed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[3] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *European conference on machine learning*. Springer, 2005, pp. 437–448.

[4] K. Jamieson and R. Nowak, "Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting," in *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*. IEEE, 2014, pp. 1–6.

[5] E. V. Denardo, E. A. Feinberg, and U. G. Rothblum, "The multi-armed bandit, with constraints," *Annals of Operations Research*, vol. 208, no. 1, pp. 37–62, 2013.

[6] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, "Combinatorial multi-armed bandit with general reward functions," in *Advances in Neural Information Processing Systems*, 2016, pp. 1659–1667.

[7] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Advances in Neural Information Processing Systems*, 2011, pp. 2312–2320.

[8] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *International Conference on Machine Learning*, 2013, pp. 151–159.

[9] D. Cortes, "Adapting multi-armed bandits policies to contextual bandits scenarios," *arXiv preprint arXiv:1811.04383*, 2018.

[10] L. Chen, J. Xu, and Z. Lu, "Contextual combinatorial multi-armed bandits with volatile arms and submodular reward," in *Advances in Neural Information Processing Systems*, 2018, pp. 3251–3260.

[11] A. Slivkins, "Contextual bandits with similarity information," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2533–2568, 2014.

[12] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 207–216.

[13] M. Niimi and T. Ito, "Budget-limited multi-armed bandit problem with dynamic rewards and proposed algorithms," in *Advanced Applied Informatics (IIAI-AAI), 2015 IIAI 4th International Congress on*. IEEE, 2015, pp. 540–545.

[14] R. Combes, C. Jiang, and R. Srikant, "Bandits with budgets: Regret lower bounds and optimal algorithms," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 245–257, 2015.

[15] M. Joseph, M. Kearns, J. H. Morgenstern, and A. Roth, "Fairness in learning: Classic and contextual bandits," in *Advances in Neural Information Processing Systems*, 2016, pp. 325–333.

[16] M. S. Talebi and A. Proutiere, "Learning proportionally fair allocations with low regret," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 2, p. 36, 2018.

[17] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," *arXiv preprint arXiv:1901.04891*, 2019.

[18] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal, "Mortal multi-armed bandits," in *Advances in neural information processing systems*, 2009, pp. 273–280.

[19] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma, "Regret bounds for sleeping experts and bandits," *Machine learning*, vol. 80, no. 2-3, pp. 245–272, 2010.

[20] V. Kanade and T. Steinke, "Learning hurdles for sleeping experts," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 11, 2014.

[21] F. Basık, B. Gedik, H. Ferhatosmanoglu, and K.-L. Wu, "Fair task allocation in crowdsourced delivery," *IEEE Transactions on Services Computing*, 2018.

[22] E. M. Schwartz, E. T. Bradlow, and P. S. Fader, "Customer acquisition via display advertising using multi-armed bandit experiments," *Marketing Science*, vol. 36, no. 4, pp. 500–522, 2017.

[23] H. Yang and Q. Lu, "Dynamic contextual multi arm bandits in display advertisement," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 1305–1310.

[24] L. Zhou and E. Brunskill, "Latent contextual bandits and their application to personalized recommendations for new users," *arXiv preprint arXiv:1604.06743*, 2016.

[25] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra, "Explore, exploit, and explain: personalizing explainable recommendations with bandits," in *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 2018, pp. 31–39.

[26] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 661–670.

[27] T. Lu, D. Pál, and M. Pál, "Showing relevant ads via context multi-armed bandits," Tech. rep, Tech. Rep., 2009.

[28] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Efficient hyperparameter optimization and infinitely many armed bandits," *CoRR, abs/1603.06560*, 2016.

[29] D. Agarwal, B.-C. Chen, and P. Elango, "Explore/exploit schemes for web content optimization," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 1–10.

[30] "Avito context ad clicks dataset," https://www.kaggle.com/c/avito-context-ad-clicks/data, accessed: 2019-02-25.