Max-Utility Based Arm Selection Strategy For Sequential Query Recommendations

Shameem A. Puthiya Parambath sham.puthiya@glasgow.ac.uk
Christos Anagnostopoulos christos.anagnostopoulos@glasgow.ac.uk
Roderick Murray-Smith roderick.murray-smith@glasgow.ac.uk
Sean MacAvaney sean.macavaney@glasgow.ac.uk
University of Glasgow, Glasgow, UK

Evangelos Zervas ezervas@uniwa.gr
University of West Attica, Greece

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

We consider the query recommendation problem in closed loop interactive learning settings like online information gathering and exploratory analytics. The problem can be naturally modelled using the Multi-Armed Bandits (MAB) framework with countably many arms. The standard MAB algorithms for countably many arms begin with selecting a random set of candidate arms and then applying standard MAB algorithms, e.g., UCB, on this candidate set downstream. We show that such a selection strategy often results in higher cumulative regret and to this end, we propose a selection strategy based on the maximum utility of the arms. We show that in tasks like online information gathering, where sequential query recommendations are employed, the sequences of queries are correlated and the number of potentially optimal queries can be reduced to a manageable size by selecting queries with maximum utility with respect to the currently executing query. Our experimental results using a recent real online literature discovery service log file demonstrate that the proposed arm selection strategy improves the cumulative regret substantially with respect to the state-of-the-art baseline algorithms. Our data model and source code are available at `https://anonymous.4open.science/r/0e5ad6b7-ac02-4577-9212-c9d505d3dbdb/`.

Keywords: query recommendation, multi-armed bandits, arm selection, maximum utility

1. Introduction

We consider the problem of query recommendation in closed loop interactive computing environments, like exploratory data analysis, and online information discovery/gathering. In such applications, a user (data analyst) starts off a session by issuing an initial query related to a *topic of investigation/exploration* to the data system and, then, exploring the topic in-depth by executing further queries. Query recommendation algorithms are employed to recommend these 'future' further queries based on previously issued queries to improve user experience (Baeza-Yates et al., 2004). Furthermore, in this context, query recommendation algorithms can be envisaged as a pillar component in resource-efficient decision making methods in data management systems. In addition to providing recommendations for data exploration tasks, effective query recommendation algorithms are expected to greatly improve the way current data systems work (including query processing, pre-fetching data, pre-analysing data, managing cached data). Forecasting the correct next-query helps the system to proactively reserve resources and be prepared for any immediate downstream tasks. For example, if the forecast next query requires loading (or transferring through the network) a considerably huge amount of data to the main memory or running a CPU intensive task, the system can prefetch the data, free up caches and memory/processes to accommodate the tasks to carry out in the immediate short-term future. These advantages of timely next-query forecasting will anticipate impact on the way data centers work and schedule tasks in terms of throughput and scalability in task scheduling. Hence, an efficient query recommendation (forecasting) engine not only improves user engagement but also improves system performance and application-driven Quality of Experience.

Typical use case scenarios include: *(i)* data scientists analyzing a large volume of data to obtain in-depth knowledge about the data for follow-up tasks like, data trend explanation (Savva et al., 2020), report summarization (Marcel and Negre, 2011) and *(ii)* users gathering information by discovering scholarly articles to conduct literature review using online services (Krause and Guestrin, 2011). These tasks can be conceptualized as the system recommending queries and the user either accepting or rejecting these recommendations, thus forming a closed loop interactive environment. This learning environment can be viewed as a repeated game between an online algorithm and the user. At each trial $t = 1, \ldots, T$, the algorithm chooses a query for recommendation from a very large query set $\mathcal{A}$. Based on the utility of the recommendation, the user chooses either to execute the query or ignore it, thus resulting in a reward. Like in the standard recommendation settings, the rewards are in the form of 'clicks' for the correctly recommended queries.

The MAB framework is popular in personalized recommender systems to model the trade-off between *exploration* and *exploitation* over a set of items (queries in our context) with unknown reward distribution. We model query recommendations using the MAB framework. The standard MAB algorithms assume that the number of arms is fixed and relatively smaller than the number of trials. In query recommendation, the total number of queries (arms) far exceeds the number of times users will use the recommendation platform. Hence, the number of queries for recommendation is inherently huge and assumed to be countably many. We do not consider the queries to be infinite as there can be many implausible queries and query sequences. In addition, unlike in personalized recommendation settings, the queries can be about very general topics like COVID, MAB or SQL statements, and need not have a personalization component. Our goal is to recommend the next query to be executed based on past executed queries and the *topic of investigation*, i.e., currently executing query.

As opposed to the standard MAB settings, countably many armed bandit algorithms have to choose from an extremely large number of arms, often much larger than the number of experimental trials or time horizon. The sheer volume of possible arms make it computationally impossible to try each of the arms even once. The standard way to deal with countably many arms is to either randomly select a candidate set containing a fixed (k) but reduced number of arms ($k \ll T \ll |\mathcal{A}|$) from the pool of arms, and run standard MAB algorithms on this reduced arm set (Wang et al., 2008; Kalvit and Zeevi, 2020; Zhu and Nowak, 2020; Bayati et al., 2020), or exploit the benign reward structure of the arms (Kleinberg et al., 2019; Magureanu et al., 2014). As we demonstrate in our experimental evaluation, the random selection strategy often results in the optimal or near-optimal arms to be ignored in the selection process and, thus, hindering the performance of the bandit algorithm. The algorithms that exploit the reward structure of the arm contexts also fall short of expectations. As demonstrated in the experiments, the zooming based algorithm (Kleinberg et al., 2019) performs sub-optimally and the OSLB (Magureanu et al., 2014) is infeasible in our case as it requires one to solve an LP in each round. Hence, the key component in the algorithm design for countably many armed bandit problems is the arm selection strategy. To this end, we propose a strategy to select the candidate set of the most promising arms based on maximizing the utility of an arm with respect to the currently playing arm.

Given the currently executing query (currently playing arm)[1], we define the 'goodness of arm for selection' with respect to the currently playing arm based on the notion of utility. Our assumption is similar to the assumption that the rewards are Lipschitz function of the arm contexts (Magureanu et al., 2014; Kleinberg et al., 2019). Furthermore, we model the likelihood, i.e., probability of a query to be preferred by the current arm as a function of pairwise similarity between them. Specifically, the currently executing query provides us with the context information of the topic the user is interested in, and we make use of the shared context information between different queries, represented as real valued vectors, to choose the best next query to run. Then, we propose an effective selection strategy based on utility maximization. For theoretically plausible utility functions, our selection strategy reduces to monotonic submodular maximization problem with cardinality constraint. For a very large set of arms, the submodular maximization problem with cardinality constraint can be solved in nearly linear time using a distributed greedy algorithm (Mirzasoleiman et al., 2016). Our candidate set selection strategy allows us to dynamically set the number of arms in the candidate set, following the design principles (Berry et al., 1997): *(i)* when the number of trials is large, the algorithm is allowed to sacrifice short term gain by eschewing arms with larger reward, if necessary, while exploring for arms with even larger reward that will expect to yield a long-term benefit, *(ii)* when the number of trials is small, the algorithm is allowed to eschew new arm exploration in favor of selecting an arm that has a large immediate reward. We also elaborate on the case of setting the value of k in 'anytime' bandit settings.

The remainder of this paper is organized as follows: after a brief overview of the algorithms for countably many armed bandit problems and query recommendations in Section 2, we describe our framework and elaborate on the candidate set selection algorithm in Section 3. In Section 4, we report the results of our experimental study on real world sequential query recommendation for online literature discovery settings and compare against strong baselines like zooming algorithm (Kleinberg et al., 2019). Section 5 concludes the paper.

2. Related Work

We present studies related to our problem in different areas.

Multi-Armed Bandits: We limit our discussion to Countably Many Armed Bandit (CMAB) algorithms. For standard MAB algorithms, we recommend the readers to refer to

1. At the start of the session, the initial query will be the currently playing arm and as user interacts with the system the currently playing arm will be the recommended query, if the user accepts the recommendation, a newly issued query, otherwise.

Cesa-Bianchi and Lugosi (2006); Lattimore and Szepesvári (2020). As opposed to standard MAB settings, in CMAB settings, bandit algorithms have to choose from an extremely large number of arms, often much larger than the possible number of experimental trials (T). The sheer volume of possible arms makes it impossible to try each of the arms even once. In stochastic bandit problems with countably many arms, the learner is restricted to ignore many arms without even trying them once, and dedicate the valuable exploration scheme only to a certain number of arms. That is, in addition to the exploration-exploitation trade-off, which is typical to sequential learning algorithms, we need also to deal with the arm discovery-exploitation trade-off within the exploration phase. That is, while exploring, the algorithm has to decide whether it should try a new arm or revisit an already played arm to get a better estimate of the expected reward. Recently, many algorithms have been proposed in the CMAB settings. These algorithms can be broadly classified into: *(i)* failure-based approaches and *(ii)* pre-selection based approaches.

In failure-based approaches, there is no hard limit on the number of arms to explore. Such algorithms try different arms until the number of trials is exhausted or reach a pre-defined failure rate for the arms. Hence, the exploration phase lasts until the algorithms hit the end of time horizon. In Berry et al. (1997), the authors proposed *k-failure*, where an arm is played until it incurs k failures, and $m - run$, where *1-failure* is used until m arms are played or m success is obtained, algorithms for Bernoulli arms. Asymptotically, failure-based algorithms yield lower cumulative regret; but for finite $T < \infty$, they tend to perform poorly. Additionally, failure-based algorithms require prior knowledge of T. Similarly, Herschkorn et al. (1996) proposed a *non-recalling*, bounded memory, failure-based algorithm for Bernoulli arms. Kalvit and Zeevi (2020) proposed an algorithm in a setting where the arms are partitioned into different types, and the goal is to find the arm from the superior type. Their algorithm can be considered as failure-based, as it terminates only when the superior type is identified with large confidence. The failure-based approaches are better suited for *pure-exploration* settings.

In the pre-selection based approach, only a specific number of arms are explored. This number can be either fixed in advance (before the start of the experiment) or adapted as the trial progresses. Such approaches randomly choose k arms from the pool of arms and use standard MAB algorithms on this subset of k arms downstream. Indicatively, the pre-selection based algorithm in Wang et al. (2008) deals with selecting k randomly chosen arms for exploration and exploitation. The exact value of k in Wang et al. (2008) is defined as a function of the current trial count. They also adopt the 'goodness of arm' assumption which states that: the probability of the mean reward of a newly explored arm differing from the optimal arm is close to zero. Zhu and Nowak (2020) proposed an algorithm in the CMAB settings with multiple best arms. They also used k arms selected uniformly at random from the arms pool. In Bayati et al. (2020), authors proposed a subsampling (uniformly at random without replacement) based greedy algorithm in CMAB settings. The arm rewards of Lipschitz bandits Magureanu et al. (2014) are assumed to be *smooth*, but arms are sampled sequentially till every arm is tried at least a fixed number of times. The zooming algorithm proposed by Kleinberg et al. (2019) refines the region for sampling the arms based on the arms similarity. In recent years, there is a surge of literature dealing with bandit algorithms in countably many, and, the closely related, infinite arm settings. The reader is advised to refer to Kalvit and Zeevi (2020); Zhu and Nowak (2020); Bayati et al.

(2020); Kleinberg et al. (2019); Lattimore and Szepesvári (2020) and references therein for a detailed coverage of different algorithms.

In all the above discussed work, candidate arm selection is not carried out in a structured fashion and does not make use of the arms context effectively. To the best of our knowledge, our proposed method is the first non-random candidate arm selection strategy that makes use of the context information, which is evaluated in the context of sequential query recommendation systems.

Query Recommendations: Query recommendation based on historical query logs is extensively studied in the Information Retrieval (IR) and Database communities (Baeza-Yates et al., 2004; He et al., 2009; Dimitriadou et al., 2014; He et al., 2009; Li et al., 2019; Dehghani et al., 2017; Rosset et al., 2020). Most of the earlier work on query recommendation made use of simple IR techniques like query similarity and query support (Baeza-Yates et al., 2004). Dimitriadou et al. (2014) proposed a query recommendation algorithm for interactive data exploration applications. The proposed algorithm works by building on-the-fly decision tree classifiers from the user feedback obtained before the exploration starts. Other prominent approaches include building offline probabilistic models using historical session data. In He et al. (2009), supervised *N-gram* based Markov model is trained on historical session data to predict the sequence of next queries. The Click Feedback-Aware Network algorithm proposed in Li et al. (2019), models the sequential queries using deep neural networks. Given a query, the proposed method predicts a ranked list of queries for recommendations. The model is trained on positive and negative query instances, constructed from the historical logs. Similar to Li et al. (2019), Dehghani et al. (2017) proposed a sequence-to-sequence Recurrent Neural Network model with query-aware attention mechanism. Jiang and Wang (2018) also used sequence-to-sequence recurrent neural networks with attention mechanisms to recommend next query. The main difference between the above two approaches is: Jiang and Wang (2018) used a 'query reformulation inferencer' to obtain homomorphic embedding of the queries. As in the case of CMAB, recent years evidenced a surge in the query recommendation literature attributed mainly to the success of sequence-to-sequence deep learning models. The reader is advised to refer to Wu et al. (2018); Li et al. (2019); Jiang and Wang (2018); Dehghani et al. (2017) and references therein for a detailed coverage of different query recommendation algorithms.

All the above work considers query recommendation from a pure supervised / weakly supervised learning setting. The user feedback is not dealt with in an online fashion. Instead, it is used to create supervised training examples. In our limited knowledge, our work is the first attempt to formulate query recommendation as a pure online learning task using the MAB framework.

3. Countably Many Armed Contextual Bandits for Query Recommendations

3.1. Problem Fundamentals

Many information filtering tasks, e.g., information discovery in the Web (Krause and Guestrin, 2011) or exploratory data analysis (Marcel and Negre, 2011), involve interactive data exploration through user issued queries, often chosen from the recommendation provided by a query recommendation engine. In such environments, conceptually, a (user) session starts with a user issuing a task or topic specific initial query to the data system. Then, the system responds with a ranked list of relevant results and a query recommendation to be potentially executed at the next step of the process. Depending on the usefulness (or utility) of the recommended query, a user may wish to either execute it or ignore it. For meaningful exploration, the recommended queries are anticipated to be related to the topics corresponding to the currently executing query and, thus, expected to be semantically similar to the current one. Normally, the recommended queries are taken or derived from the historical query logs containing all queries executed in the past (Baeza-Yates et al., 2004). Hence, the potential queries for recommendation are considered as countably many. We envisage this problem as an optimal arm selection in stochastic multi-armed bandit settings with countably many arms (Kalvit and Zeevi, 2020) where queries correspond to arms.

Let $\mathcal{A}$ be the set of countably many arms and $a_i \in \mathcal{A}$ be the currently played arm. Our goal is to select an arm from the set $\mathcal{A}' = \mathcal{A} \backslash \{a_i\}$, which results in the maximum cumulative reward. The reward in our setting is the same as the reward in a typical recommendation problem using MAB. That is, rewards are in the form of clicks for the correctly recommended queries. If the cardinality of $\mathcal{A}'$ is significantly less than the time horizon, one could use standard stochastic multi-armed bandit algorithms like Upper Confidence Bound (UCB) (Auer et al., 2002) or ϵ-greedy (Lattimore and Szepesvári, 2020). However, as pointed out earlier, in query recommendation settings, the number of arms is countably many and any standard multi-armed bandit algorithm has to choose from an extremely large number of arms. The sheer volume of possible arms makes it impossible to try each of the arms (even once), and consequently the algorithm is enforced to ignore many arms and dedicate the valuable query exploration only to a certain number of arms.

Standard CMAB algorithms randomly select a fixed number of arms, called *candidate set*, from the pool of arms, and adopt standard MAB algorithms over this reduced arms set (Wang et al., 2008; Kalvit and Zeevi, 2020; Zhu and Nowak, 2020; Bayati et al., 2020). The random selection might result in the optimal arms to be ignored in the selection process, thus, hindering the performance of the recommendation engine. In our method, we make use of the shared context information between queries to select the candidate set. We assume that the queries are represented as d-dimensional real-valued vectors encoding the semantic content of the queries (Le and Mikolov, 2014; Mikolov et al., 2013). These vectors can be regarded as the contexts associated with the arms. Without ambiguity, we use the same notation a_i to represent the context vector of arm a_i. Conventional CMAB algorithms do not make use of context information when generating the candidate set.

We base our reasoning on using the context associated with an arm by analyzing a real query log files of an online literature discovery service. We plot the sequences of queries executed in five random user-sessions in Figure 1. Specifically, we plot the 2-dimensional query context vectors (we used PCA to reduce the original dimension $d = 128$) for five random user-sessions in five different colours (further details about the queries and sessions is given in Section 4 and the supplementary file). From the plot, one can observe that the queries issued within each session are contextually similar with respect to standard similarity measures like cos, and form relatively small loose clusters. Based on this observation on query similarity, we propose to make use of the shared context information between the currently executing query and queries in $\mathcal{A}'$ for candidate selection. The crux of our approach is the assumption that arms with similar context, with respect to a given arm, have similar *utility*. We formalize this assumption using the following definition of utility adopted from the information-theoretic interpretation of utility function in multi-agent systems. Specifically, one can conceive utility in terms of the preference probabilities for being at different states as given in (Ortega and Braun, 2010).

Definition 1 ***Goodness of Arm for Selection*** *(Ortega and Braun, 2010, Definition 1) Given the currently playing arm* a_i*, let* $g\big(\{a_j\},\{a_i\}\big) = p(a_j|a_i)$ *be the conditional probability of the arm* a_j *to be included in the candidate set by a selection strategy. Then, there exists a real-valued utility function* util *which is: (i) sub-additive*[2]*, i.e.,* $\mathrm{util}\big(g\big(\{a_j,a_k\},\{a_i\}\big)\big) \leq \mathrm{util}\big(g\big(\{a_j\},\{a_i\}\big)\big)+\mathrm{util}\big(g\big(\{a_k\},\{a_i\}\big)\big)$*; and (ii) consistent, i.e.,* $g\big(\{a_j\},\{a_i\}\big) > g\big(\{a_k\},\{a_i\}\big) \Leftrightarrow \mathrm{util}\big(g\big(\{a_j\},\{a_i\}\big)\big) > \mathrm{util}\big(g\big(\{a_k\},\{a_i\}\big)\big)$*.*

The conditional probability, $p(a_j|a_i)$ can be considered as the normalized preference score for the arm a_j to be played next conditioned on the event that a_i is the currently playing arm. The utility function util defined above assigns a scalar value to each possible arm such that arms with higher utility correspond to arms that are more preferred. However, the challenge in our setting is that the preference scores $p(a_j|a_i)$ are not known *a priori.*

3.2. Arm Preference Probability

Typical multi-armed bandit algorithms work by adaptive hypothesis testing. From an abstract point, such algorithms randomly pick two arms (assuming only two types of arms: optimal and non-optimal) and run on-the-fly hypothesis testing for a predefined duration to estimate statistical properties of the arms. As noted earlier, contextually similar arms are preferred and, thus, query similarity is correlated to its utility. Hence, instead of randomly selecting the candidate set, we argue that the pair of arms is chosen using a joint probability distribution defined over the similarity of the arms.

Given two arms a_i, a_j and a similarity function evaluation oracle $s_{j,i} = \mathrm{sim}(a_j, a_i)$, let $S^i_{\geq}(\varepsilon) = \{a_j : \mathrm{sim}(a_j,a_i) \geq \varepsilon\}$ and $S^i_{<}(\varepsilon) = \{a_j : \mathrm{sim}(a_j,a_i) < \varepsilon\}$ be the partition of the arms for a given similarity threshold value $\varepsilon > 0$. Based on this similarity indices, a distribution is induced on the elements of $S^i_{\geq}(\varepsilon)$ $(S^i_{<}(\varepsilon))$ as:

$$s_{j,i}(\varepsilon) = p(a_j|a_j \in S^i_{\geq}(\varepsilon)) = \frac{s_{j,i}}{\sum_k s_{k,i}[\![a_k \in S^i_{\geq}(\varepsilon)]\!]}$$
$$\bar{s}_{j,i}(\varepsilon) = p(a_j|a_j \in S^i_{<}(\varepsilon)) = \frac{s_{j,i}}{\sum_k s_{k,i}[\![a_k \in S^i_{<}(\varepsilon)]\!]},$$

where $[\![\cdot]\!]$ is the indicator function. Extending over the whole population of arms, we define

$$\pi_{\star,i}(\varepsilon) = p(S^i_{\geq}(\varepsilon)) = \frac{\sum_j s_{j,i}[\![a_j \in S^i_{\geq}(\varepsilon)]\!]}{\sum_j s_{j,i}} \quad \text{and} \quad \bar{\pi}_{\star,i}(\varepsilon) = 1 - \pi_{\star,i}(\varepsilon).$$

Given the currently playing arm a_i, we posit that the candidate set of arms for adaptive hypothesis testing is sampled according to the following joint distribution; note, with $(s_{j,i}, s_{k,i}) \geq \varepsilon$i, we denote the pairwise comparison: $s_{j,i} \geq \varepsilon \wedge s_{k,i} \geq \varepsilon$.

2. Theorem:3 in Ortega and Braun (2010) holds for both additive and sub-additive utilities

$$
\begin{aligned}
p(a_j, a_k | a_i, \varepsilon) =& p(a_j, a_k | (s_{j,i}, s_{k,i}) \geq \varepsilon \vee (s_{j,i}, s_{k,i}) < \varepsilon) \\
=& \frac{\pi^2_{\star,i}(\varepsilon) s_{j,i}(\varepsilon) s_{k,i}(\varepsilon) + \bar{\pi}^2_{\star,i}(\varepsilon) \bar{s}_{j,i}(\varepsilon) \bar{s}_{k,i}(\varepsilon)}{\pi^2_{\star,i}(\varepsilon) + \bar{\pi}^2_{\star,i}(\varepsilon)}.
\end{aligned}
\tag{1}
$$

The detailed derivation of the above quantity is given in the supplementary file. The rationale behind the above sampling scheme is that, given the currently playing arm a_i, we independently draw two arms a_j and a_k and accept/reject them only if pairs are similar/dissimilar to a_i with confidence ε. To understand the concept, a detailed worked-example is given in the supplementary file.

Estimating the joint probability for every pair of arms is time consuming and computationally inefficient. In Lemma 2, we show that arms similar to the currently playing arm are selected with the probability in (2). This will help us to select a candidate set based on marginal probabilities only.

Lemma 2 *Given the set of arms $\mathcal{A}'$ and the currently playing arm a_i, candidate arms are independently drawn with probability*

$$
P(a_j | a_i, \varepsilon) = \frac{\pi^2_{\star,i}(\varepsilon) s_{j,i}(\varepsilon) + \bar{\pi}^2_{\star,i} s_{j,i}(\varepsilon)}{\pi^2_{\star,i}(\varepsilon) + \bar{\pi}^2_{\star,i}(\varepsilon)}.
\tag{2}
$$

The proof of Lemma 2 is given in the supplementary file. A naive approach for generating candidate arms set will be to fix a probability threshold and take all the arms with highest marginal probability according to (2), or randomly sample k arms with the estimated marginal probability. However, such an approach is not effective in practice, i.e., users do not prefer to issue the most similar queries in data exploration tasks, as demonstrated in our experimental study in Section 4. Moreover, in the CMAB settings, there can be a huge number of similar arms for a fixed threshold. So randomly picking k number of arms as per the marginal probability might lead the optimal or near-optimal arms to be ignored as discussed in our problem statement. The number of arms for different threshold values of marginal probability is given in Table 2. To this end, we propose to select a candidate arms set that maximizes the *utility* of the arms as given in Definition 1. Furthermore, we elaborate on an easy-to-implement distributed framework for finding a candidate set with varying numbers of candidate arms.

3.3. Candidate Arm Selection Using Maximum Utility

To select arms based on *utility*, we need a function that realizes real valued utilities from the probabilistic preference scores obtained using the similarity between shared contextual information content. According to Theorem 3 (Ortega and Braun, 2010), the *logarithm* function is the only function that can express such a relationship between the preference probability scores and utility function. For completeness, we restate Theorem 3 by Ortega and Braun (2010).

Theorem 3 ((Ortega and Braun, 2010, Theorem 1)) *Given the arm set $\mathcal{A}$ with the probability space defined as in (2), a function* util *is a utility function on the probability*

Algorithm 1: Max-utility based Countably Many-armed Contextual Bandits

```
Input   : $\mathcal{A}', a_i$
1 Initialize $\mathcal{C}$ by executing Algorithm 3 for $t = 1, \dots, T$ do
2     run Algorithm 2 to update $\mathcal{C}$;
3     run standard stochastic contextual
        bandit algorithm on arm set $\mathcal{C}$ using
        $a_i$ as context;
4 end
```

Algorithm 2: Candidate Selection

```
Input   : $\mathcal{A}', \mathcal{C}, t, \alpha \geq 1$
1 $k_t = t^{\alpha} \mathrm{e}^{-t}$ ;
2 $\mathcal{S} = \mathcal{A}' \setminus \mathcal{C}$ ;
3 while $|\mathcal{C}| < k_t$ do
4     $e^{\star} = \mathrm{argmax}_{e \in \mathcal{S}} \log \big(g\big(\mathcal{C} \cup \{e\}, \{a_i\}\big)\big)$;
5     $\mathcal{C} = \mathcal{C} \cup \{e^{\star}\}$;
6     $\mathcal{S} = \mathcal{S} \setminus \{e^{\star}\}$;
7 end
Output: $\mathcal{C}$
```

space, if and only if for all $a_i, a_j \in \mathcal{A}$*,* $\mathrm{util}\big(g\big(\{a_j\}, \{a_i\}\big)\big) = c \cdot \log\big(g\big(\{a_j\}, \{a_i\}\big)\big)$*, where* $c > 0$ *is an arbitrary constant.*

It can be easily verified that the log function satisfies all the properties given in the Definition 1 with our preference probability space defined in (2). Given log as the utility function, $c = 1$ and currently executing arm a_i, a candidate set $\mathcal{C}$ of cardinality k realizes the optimal utility if it solves[3]:

$$\max_{\substack{\mathcal{C} \subseteq \mathcal{A}' \\ |\mathcal{C}| \leq k}} \log\big(g\big(\mathcal{C}, \{a_i\}\big)\big) \tag{3}$$

Since log is a concave function, the objective function in (3) is a submodular maximization problem with cardinality matroid constraint. Though submodular maximization is NP-hard in general, a simple greedy heuristic due to Nemhauser et al. (1978) guarantees a solution with constant approximation factor equal to $1 - \frac{1}{e}$. Our CMAB algorithm with max-utility based candidate arm selection procedure is given in Algorithm 1. The input to the algorithm is $\mathcal{A}'$ and the currently running query (arm) a_i. The algorithm starts with an initial set of candidate arms with fixed cardinality. This set is constructed following the greedy heuristic. At each trial, the algorithm dynamically updates the candidate set based on the current trial count using Algorithm 2. At the initial stages of trials, we allow the candidate set to grow as the algorithm is allowed to sacrifice short term gain by exploring for arms with even larger reward that will expect to yield a long-term benefit. As the trial progresses we keep the candidate set fixed as the algorithm is allowed to eschew new arm exploration, by exploiting the unimodality of $t^{\alpha}\mathrm{e}^{-t}$. Finally, we run the standard stochastic contextual MAB algorithm using the candidate set and currently executing arm.

3.4. Fast Submodular Function Evaluation

The standard greedy algorithms for submodular maximization guarantee a near-optimal candidate selection, without room for further improvement using current computing environments. However, greedy algorithms do not scale well when applied to massive data and, thus, initialization of $\mathcal{C}$ will incur considerable computation time. The greedy algorithms

3. For the set $\mathcal{C}$, $g(\mathcal{C}, a_i)$ is defined as the joint probability of the arms in $\mathcal{C}$ conditioned on a_i

```
Algorithm 3: Distributed Submodular Maximization
Input   : $\mathcal{A}', k, m, a_i$
1 Partition $\mathcal{A}'$ into $m$ sets $\mathcal{A}_1, \mathcal{A}_2, \cdots \mathcal{A}_m$ (arbitrarily/at random);
2 Run standard (lazy) greedy algorithm on each $\mathcal{A}_i$ to obtain solution with cardinality $k$
    that maximizes $\log\big(g(\mathcal{A}_i, \{a_i\})\big)$ to get set $\mathcal{P}_i$ ;
3 Find $\mathcal{P}_{max} = \text{argmax}_{\mathcal{P}_{i|1\ldots m}} \log\big(g(\mathcal{P}_i, \{a_i\})\big)$ ;
4 Merge the sets into $\mathcal{B} = \bigcup_{i=1}^{m} \mathcal{P}_i$;
5 Run lazy greedy algorithm on $\mathcal{B}$ to obtain solution that maximizes util to get $\mathcal{P}_{\mathcal{B}}$;
Output: $\text{argmax}_{e \in \{\mathcal{P}_{max}, \mathcal{P}_{\mathcal{B}}\}} \log\big(g(e, \{a_i\})\big)$
```

work well for centralized submodular maximization problems; but this requires $O(nk)$ value oracle calls to select k arms from n arms. The adaptive addition of new arms to the existing $\mathcal{C}$ (Algorithm 2) can be achieved in linear time, as the number of arms to be added will be relatively very small. For large data, submodular maximization with cardinality constraint can be solved in nearly linear time using a distributed greedy algorithm (Mirzasoleiman et al., 2016). We use a faster version of the submodular maximization algorithm (Mirzasoleiman et al., 2016), which can be parallelized. The proposed algorithm adopted to our setting is then given in Algorithm 3. In line 2 & line 3 of the Algorithm 3, as mentioned earlier, g is the joint probability of all the arms in the input set conditioned on the currently playing arm. Due to the use of a greedy algorithm, we do not explicitly calculate it, but incrementally construct the set.

3.5. Choosing k and Anytime Algorithm

If the total number of trials T is known beforehand, one can choose a value of k that minimizes the regret associated with the underlying bandit algorithm. For instance, in the UCB-∞ algorithm in Wang et al. (2008), for a distribution specific parameter β, one can choose k to be of the order of $T^{\frac{\beta}{2}}$ or $T^{\frac{\beta}{\beta+2}}$ depending on the range of β[4]. When the number of trials T is not known in advance, one can adaptively choose k depending on the current trial number. A MAB algorithm is *anytime*, if the regret bounds on the expected regret holds for all values of T (up to constant factors). We can make our algorithm *anytime* by employing *anytime* bandit algorithm like UCB-∞ in line 3 of Algorithm 1. For example, in case of using UCB-∞, we can use $k_{t-1} < t^{\frac{\beta}{2}}$ or $k_{t-1} < t^{\frac{\beta}{\beta+1}}$ (depending on the value of β) as shown in line 1 of Algorithm 2.

4. Experimental Evaluation & Comparative Assessment

We discuss the dataset, baselines, and provide in-depth analysis to verify the performance of the proposed arm selection scheme. Our source code is available at: `https://anonymous.4open.science/r/0e5ad6b7-ac02-4577-9212-c9d505d3dbdb/`.

4. Better approximation for k can be achieved by adopting the Lambert W function.

4.1. Dataset & Context Vector

Our experiments are conducted on recent large scale query logs from SemanticScholar an online literature discovery service. The application works by accepting a user query and returning a list of the most relevant research articles to the query. The logs contained around 4.5 million queries grouped into 547740 user sessions. For our experimental study, as a pre-processing step, we removed user sessions with less than 4 queries and more than 50 queries. We also removed sessions containing non-English queries. The statistics of the final data used in the experiment are given in Table 3. An example of the queries issued during literature discovery service, with session ids anonymized, is given in Table 1. Further details about the query log is given in the supplementary file.

Table 1: Example Queries

Session ID	Timestamp	Query Text
XXXX495	2020-X-X X:04:03	protocol state fuzzing of tls implementations
XXXX495	2020-X-X X:08:45	aflnet a greybox fuzzer for network protocols
XXXX495	2020-X-X X:33:04	protocol learning fuzzing
XXXX495	2020-X-X X:42:33	improving grey box fuzzing by modeling program behavior
XXXX495	2020-X-X X:03:22	poster fuzzing iot firmware via multi stage message generation
XXXX495	2020-X-X X:05:34	fuzzguard filtering out unreachable inputs in directed grey box fuzzing through deep learning
XXXX495	2020-X-X X:19:17	not all bytes are equal neural byte sieve for fuzzing

For the query context vector, we used a transformer-based deep neural network (Vaswani et al., 2017) to extract the contexts from the queries. We train a bidirectional BERT model (Devlin et al., 2019) on queries to get a word level vector embeddings. We feed these embeddings to a sentence embedding algorithm (Reimers and Gurevych, 2019) to get the final context vector representation for each query in the log. We set the dimension of the context vectors $d = 128$.

Table 2: # arms for different $p(a_j|a_i)$

$p(a_j\|a_i, 0.5)$	COUNT
0.25	1,116,801
0.40	949,365
0.50	523,750
0.60	161,529

Table 3: Dataset Statistics

Description	COUNT
# of queries (original)	4,687,947
# queries (pre-processed)	1,120,461
# user sessions (")	159,237
# avg queries/session (")	7

4.2. Models Under Comparison

We compare our max-utility selection strategy against a variant of the zooming algorithm (Kleinberg et al., 2019) and the random selection strategy that is used in standard CMAB algorithms (Wang et al., 2008; Kalvit and Zeevi, 2020; Zhu and Nowak, 2020; Bayati et al., 2020). In random selection, we randomly select k arms and pass to the downstream MAB algorithm whereas in max-utility we follow the Algorithm 1. In case of random and max-utility schemes, we experimented with four different contextual bandit algorithms (LinUCB, LinThompSamp, Random and Similar) as downstream MAB algorithms and for the zooming variant, we used LinUCB as the downstream MAB algorithm.

Linear UCB (LinUCB) was proposed in the context of news recommendation (Li et al., 2010) where the algorithm sequentially selects news articles based on contextual information about the users and articles, while simultaneously adapting to the user-click feedback. LinUCB models the reward as a linear function of the context vector.

Linear Thomson Sampling(LinThompSamp)was proposed as an extension to the Thomson sampling scheme to the stochastic contextual bandit settings (Agrawal and Goyal, 2013). The rewards are assumed to be a linear function of the context vectors. The algorithm itself is based on Bayesian ideas and assumes that the reward likelihood and mean parameter follow the Gaussian distribution.

Most Similar strategy recommends one query from the top-five queries with highest conditional marginal probability with respect to the currently executing query from the candidate set. In Figure 1, we plot the context vectors associated with queries issued in five different user sessions in five different colors. Each data point represents a query. The queries in each session are contextually similar and form relatively loose clusters. Looking at the figure, one might expect that recommending the most similar query might be a useful strategy. We used this baseline strategy to demonstrate that the naive strategy of picking the queries with high marginal probability or similarity with respect to the currently executing query underperform compared to other algorithms.

Random strategy randomly selects a query for recommendation from the candidate set.

Zooming LinUCB is based on the zooming algorithm proposed by Kleinberg et al. (2019). It combines the upper confidence bound technique with an adaptive refinement step that selects a candidate set region. For each currently executing query, we select candidate arms that is equal or higher than the similarity threshold and run LinUCB on this arm set.

4.3. Results & Analysis

We used the cumulative regret to compare the performance of the algorithms. Formally, we compare the quantity

$$R(T) = T - \sum_{t=1}^{T} r(a^t),$$

where $r(a^t)$ is the reward obtained from the arm played at the t^{th} round. In the experiments, we used the first query in each user session as the initial query and $\mathcal{A}'$ is given as the input to the candidate set selection algorithm. We used cosine as the similarity function between arms. If the recommended query is the one of the next queries executed in the corresponding user session, the algorithm is rewarded with $r(a) = 1$, otherwise, $r(a) = 0$. The per-round regret of different MAB algorithms against the random and max-utility selection strategies is shown in Figure 2. At round t, the per-round regret is defined as $\frac{R(t)}{t}$. As it can be seen from the plot, random selection strategy results in optimal or near-optimal arms to

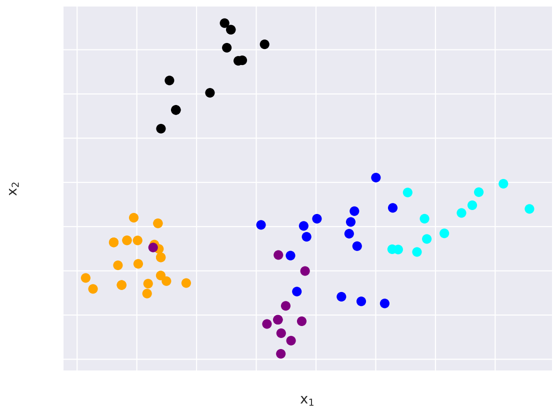

Figure 1: Similarity of the queries in five user sessions in five different colors.

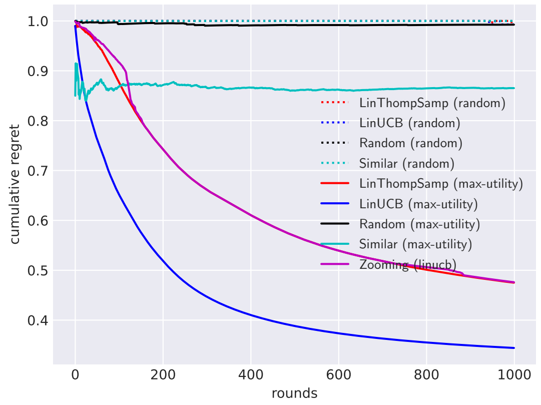

Figure 2: Per-round regret for the different algorithms.

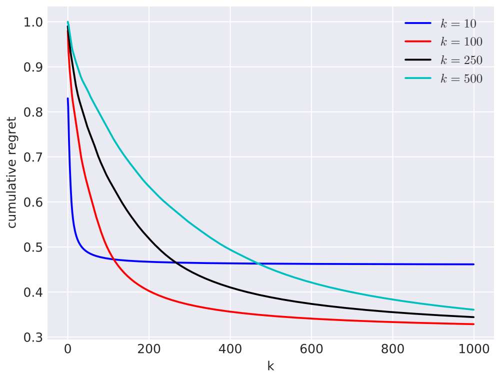

Figure 3: Per-round regret for different k.

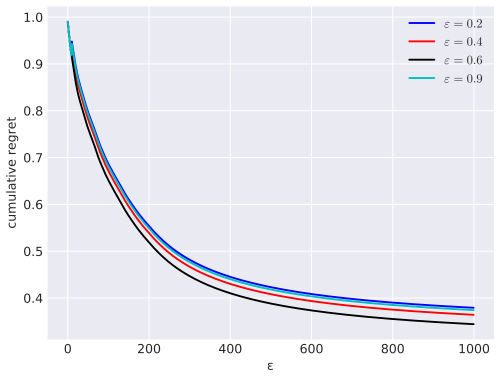

Figure 4: Per-round regret for different ε.

be ignored whereas max-utility based candidate selection always includes optimal or near-optimal arms in the candidate set. Though the zooming algorithm with LinUCB performed as well as LinThompSamp with max-utility, it resulted in higher regret compared to LinUCB with max-utility. Our results are also inline with the regret guarantees proved for LinUCB and LinThompSamp. In the case of LinUCB, the per-round regret is of the order of $\frac{d}{\sqrt{T}}$ whereas in the case of LinThompSamp it is of the order of $\frac{d^2}{\sqrt{T}}$. Thus LinUCB resulted in lower regret compared to LinThompSamp with max-utility. Even with superior regret guarantees of the LinUCB algorithm, the zooming algorithm failed to achieve the regret of LinUCB with max-utility. Looking at the plot in Figure 1, one might be tempted to use simple strategies like recommending similar queries (our Similar strategy), but from the performance comparison in Figure 2 it is very evident that even with the proper candidate selection strategy, it performs very badly in the longer runs.

Another interesting observation is that with the max-utility arm selection strategy, even the trivial random recommendation strategy performs better than random recommendation with random arm selection strategy. Moreover, random recommendation with max-utility arm selection strategy results in lower regret than recommending similar queries from a randomly picked candidate set.

Hyperparameter Analysis. We further investigated the effect of the size of the candidate set and the similarity threshold (ε) on the algorithm performance. For this set of experiments, we used LinUCB as the downstream MAB algorithm. Our results are shown in Figure 3 & Figure 4. In Figure 3, we varied the value of k from 10 to 500 and compared the performance as a function of k. Here, we used $\varepsilon = 0.5$. Keeping the cardinality of the candidate set to small or high does not give any performance advantage. Though the average number of queries per session in our dataset is $\sim$7, setting $k = 10$ performed very poorly. So it is very important to choose the correct size for the candidate set. Precisely, k should be chosen large enough such that the candidate set contains all possible, relevant and diverse queries with respect to the currently running query. We also analyzed the performance of the max-utility candidate selection strategy for different values of ε. Probability preference score for an arm is determined by the value of ε, thus, it is an hyperparameter of the selection strategy. The performance of the LinUCB algorithm for different values of ε is plotted in Figure 4. Here, we used $k = 250$. For small and very large values of ε, the per-round regret is slightly higher than mid-range (0.4 - 0.6) values of ε. By keeping ε to small and large values, we make the preference probability score between the currently running query and the remaining queries to be high and low respectively. As a result, we notice the same trend as in Figure 3. When ε is small many irrelevant queries will have high preference probability scores. Similarly, when ε is large, many diverse but relevant queries will have low preference probability scores.

5. Conclusions

We modelled the query recommendation problem in closed loop interactive learning settings like online information gathering using a MAB framework with countably many arms. The standard way to solve MAB problems with countably many arms is to select a small set of candidate arms and then apply standard MAB algorithms on this candidate set downstream. We showed that such a selection strategy often results in higher cumulative regret and proposed a selection strategy based on the maximum utility of the arms. Our experimental results using a real online literature gathering service log file demonstrated that the proposed arm selection strategy significantly improves the cumulative regret compared to zooming algorithm and the commonly used random selection strategy for a variety of contextual multi-armed bandit algorithms.

Acknowledgments

This work is partially funded by the UK EPSRC Grant Number: EP/R018634/1. We thank Allen Institute for providing the SemanticScholar query logs.

References

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *ICML*, pages 127–135. PMLR, 2013.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, (2):235–256, 2002.

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *EDBT*, pages 588–596. Springer, 2004.

Mohsen Bayati, Nima Hamidi, Ramesh Johari, and Khashayar Khosravi. Unreasonable effectiveness of greedy algorithms in multi-armed bandit with many arms. In *NeurIPS*, pages 1713–1723, 2020.

Donald A Berry, Robert W Chen, Alan Zame, David C Heath, and Larry A Shepp. Bandit problems with infinitely many arms. *The Annals of Statistics*, pages 2103–2116, 1997.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games.* Cambridge university press, 2006.

Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. Learning to attend, copy, and generate for session-based query suggestion. In *CIKM*, pages 1747–1756, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NACL*, pages 4171–4186, 2019.

Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *SIGMOD*, pages 517–528, 2014.

Qi He, Daxin Jiang, Zhen Liao, Steven CH Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. Web query recommendation via sequential query prediction. In *IEEE ICDE*, pages 1443–1454, 2009.

Stephen J Herschkorn, Erol Peköz, and Sheldon M Ross. Policies without memory for the infinite-armed bernoulli bandit under the average-reward criterion. *Probability in the Engineering and Informational Sciences*, 10(1):21–28, 1996.

Jyun-Yu Jiang and Wei Wang. Rin: Reformulation inference network for context-aware query suggestion. In *CIKM*, pages 197–206, 2018.

Anand Kalvit and Assaf Zeevi. From finite to countable-armed bandits. In *NeurIPS*, pages 8259–8269, 2020.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Bandits and experts in metric spaces. *Journal of the ACM*, 66(4), 2019.

Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM Transactions on Intelligent Systems and Technology*, 2(4):1–20, 2011.

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

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196. PMLR, 2014.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010.

Ruirui Li, Liangda Li, Xian Wu, Yunhong Zhou, and Wei Wang. Click feedback-aware query recommendation using adversarial examples. In *WWW*, pages 2978–2984, 2019.

Stefan Magureanu, Richard Combes, and Alexandre Proutiere. Lipschitz bandits: Regret lower bound and optimal algorithms. In *COLT*, pages 975–999. PMLR, 2014.

Patrick Marcel and Elsa Negre. A survey of query recommendation techniques for data warehouse exploration. In *EDA*, pages 119–134, 2011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *JMLR*, (1):8330–8373, 2016.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, (1), 1978.

Pedro A. Ortega and Daniel A. Braun. A conversion between utility and information. In *Proceedings of the 3rd Conference on Artificial General Intelligence (2010)*, pages 47–52. Atlantis Press, 2010.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*, pages 3973–3983, 2019.

Corbin Rosset, Chenyan Xiong, Xia Song, Daniel Campos, Nick Craswell, Saurabh Tiwary, and Paul Bennett. Leading conversational search by suggesting useful questions. In *Proceedings of The Web Conference 2020*, pages 1160–1170, 2020.

Fotis Savva, Christos Anagnostopoulos, Peter Triantafillou, and Kostas Kolomvatsos. Large-scale data exploration using explanatory regression functions. *Transactions on Knowledge Discovery from Data*, 14(6), 2020. ISSN 1556-4681.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 6000–6010, 2017.

Yizao Wang, Jean-Yves Audibert, and Rémi Munos. Algorithms for infinitely many-armed bandits. *NIPS*, pages 1729–1736, 2008.

Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Query suggestion with feedback memory network. In *WWW*, pages 1563–1571, 2018.

Yinglun Zhu and Robert Nowak. On regret with multiple best arms. In *NeurIPS*, pages 9050–9060, 2020.