Data Mining and Knowledge Discovery (2024) 38:3758–3782
https://doi.org/10.1007/s10618-024-01057-4

Sequential query prediction based on multi-armed bandits with ensemble of transformer experts and immediate feedback

Shameem A. Puthiya Parambath[1] · **Christos Anagnostopoulos**[1] · **Roderick Murray-Smith**[1]

Received: 7 February 2023 / Accepted: 30 June 2024 / Published online: 2 August 2024

Abstract

We study the problem of predicting the next query to be recommended in interactive data exploratory analysis to guide users to correct content. Current query prediction approaches are based on sequence-to-sequence learning, exploiting past interaction data. However, due to the resource-hungry training process, such approaches fail to adapt to immediate user feedback. Immediate feedback is essential and considered as a signal of the user's intent. We contribute with a novel query prediction ensemble mechanism, which adapts to immediate feedback relying on multi-armed bandits framework. Our mechanism, an extension to the popular Exp3 algorithm, augments Transformer-based language models for query predictions by combining predictions from experts, thus dynamically building a candidate set during exploration. Immediate feedback is leveraged to choose the appropriate prediction in a probabilistic fashion. We provide comprehensive large-scale experimental and comparative assessment using a popular online literature discovery service, which showcases that our mechanism (i) improves the per-round regret substantially against state-of-the-art Transformer-based models and (ii) shows the superiority of causal language modelling over masked language modelling for query recommendations.

Keywords Multi-armed bandits · Query recommendation · Immediate user feedback · Large language models (LLMs) · Transformers

Responsible editor: Matteo Riondato.

✉ 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

[1] School of Computing Science, University of Glasgow, Glasgow, UK

1 Introduction

The data exploration process consists of incremental knowledge extraction and gathering where users (like data scientists and analysts) sequentially explore the data space through sequences of queries predicted and suggested by online algorithms. It is evidenced that query recommendation speedups the data exploration process by addressing the need for disambiguation of queries and directing analysts to acquire the intended knowledge (Dehghani et al. 2017). Moreover, query prediction supports the management of distributed data systems in resource-efficient decision-making (Parambath et al. 2021). As pointed out in Parambath et al. (2021), an online interactive data exploration session starts with a user issuing an initial query pertaining to a specific topic. Then, the system responds with relevant results along with a *recommendation of a suitable query* to be executed in the next round of the exploration process. In such online interaction, the system can capture the immediate user feedback to the recommended query, e.g., the user accepts the recommended query in the form of a 'click'. This feedback serves as a signal of the user's intent; positive feedback (acceptance by a click) signals that the recommended query aligns with the user's intent, while negative feedback (no click) signals the opposite. Typical application scenarios involve literature discovery on the web, among others (Parambath et al. 2021).

In the online learning paradigm, the user feedback to the recommended action is considered as the *reward* received by the online algorithm (Lattimore and Szepesvári 2020; Cesa-Bianchi and Lugosi 2006). At each step, the algorithm recommends an action, i.e., a query in our case, and a click on the recommended query indicates a reward for a correct prediction, thus, correct inference of the user's intent. The online algorithm can make use of each reward received at each step to improve the query recommendation by gradually adapting its recommendation strategy. The overarching aim of this process is to maximize the cumulative reward captured by predictions (or, equivalently, minimize the cumulative loss) over a finite period of time.

The above-mentioned online user-system interactive learning process can be modelled through the Multi-Armed Bandits (MAB) framework with countably many arms (bandits) (Kalvit and Zeevi 2020; Bayati et al. 2020). A MAB framework models the trade-off between exploration and exploitation over a sequence of actions (queries in our case) (Lattimore and Szepesvári 2020; Cesa-Bianchi and Lugosi 2006). The MAB framework provides the principles of handling the inherent uncertainty about the suitable query to be predicted at each step of the exploratory data analysis and knowledge extraction process.

1.1 The query prediction problem in multi-arm bandits context

Let us first discuss the application of a standard stochastic MAB framework to the online query prediction problem. At each discrete step $t = 1, \ldots, T$, depending on the currently executing query, the (MAB-based) system chooses a query q_t to be

executed from a very large set of historical queries $\mathcal{Q}$. All the past executed queries until the current query in a session is referred to as *query context*, i.e., the sequence $(q_1, q_2, \ldots, q_t)$. The query context indicates the user's intent. Specifically, each predicted query q_t is associated with an intrinsic utility, which depends on the user's intent captured by the current query context. Depending on the user's intent, the user can either accept or ignore (reject) the recommendation of the predicted query, thus, revealing a reward r_t to the system at the t-th step. In this fashion, the system repeatedly predicts the next query to be recommended, observes the collected rewards by the user so far, and tries to maximize the expected cumulative rewards over time. In the standard stochastic MAB, there are $m > 0$ fixed unknown reward probability distributions associated with m queries available in the query set, i.e., $m = |\mathcal{Q}|$. The distributions are assumed to be independent and identical. The objective is to identify the *optimal* query with the highest expected reward distribution. This indicates that the MAB algorithm should explore the whole *query space* to find the optimal query with the highest reward value.

However, in practice, it is unrealistic to assume that a user progresses with a fixed number of queries during the data exploration process regardless of the past queries issued in the session. Moreover, the predicted query to be recommended should depend on the sequence of the past issued queries so far, reflecting a continuation and progression context of the exploratory data analysis, which this context is bound to change in every round. In fact, depending on the knowledge extraction and gathering process needs, a user might be interested in queries pertaining to different aspects of a topic of interest. For instance, consider a user who is exploring the research topic of 'multi-armed bandits'. The user could start the session with a simple query and dive into diverse subtopics of MAB, like applications, different MAB algorithms, reward distributions, action structures, etc., as the knowledge-gathering process progresses. Hence, by repeatedly recommending the next query pertaining to a single subtopic, disregarding the sequence(s) of past issued queries, is not expected to achieve positive responses (acceptances) from the user. Evidently, this hinders user satisfaction and system usability. Therefore, the stochastic reward assumption used in the standard MAB mechanisms does not successfully operate in online knowledge-gathering processes.

In this paper, we depart from these limitations of standard stochastic MAB mechanisms and tackle the query prediction problem in a non-stochastic fashion. Our principle is based on the autoregressive nature of the knowledge-gathering process, where we make no probabilistic assumptions on how the rewards, corresponding to different query predictions, are generated. In particular, the rewards may depend on previously predicted and executed queries, which can also be adversarial. Earlier work on query prediction using MAB assumes a fixed query set, which does not take into account the dynamics of the data exploration process (Parambath et al. 2021). Moreover, the aforementioned approach is based on the similarities between queries in the query context and the stochastic reward structure for the queries. Both of these aspects fail to model the autoregressive nature of the knowledge-gathering process. It should also be noted that standard adversarial MAB algorithms like Exp3 (Auer et al. 2002; Cesa-Bianchi and Lugosi 2006) cannot be directly used in query prediction tasks. This is due to the fact that such algorithms are designed to work with a

fixed and known set of queries only. Nonetheless, query predictions in knowledge-gathering require the next/predicted query to depend on the query context, which cannot be assumed to be known beforehand.

Due to the sequential nature of knowledge-gathering, autoregressive time series models for text generation have demonstrated promising results in many practical knowledge extraction tasks as shown in Dehghani et al. (2017), Ren et al. (2018), Ahmad et al. (2019), Jiang and Wang (2018), Wu et al. (2018), Mustar et al. (2020). Autoregressive query prediction models take into account the previously issued queries (in the same session) when proceeding with the query prediction to filter out irrelevant queries. This helps the user with the proper exploration of the data space. The state-of-the-art query prediction algorithms make use of the deep Recurrent Neural Networks (RNN) based autoregressive models to extract certain linguistic patterns from query logs and combine them with historical interaction data to predict the next query. Unfortunately, standard RNN models like Long-Short Term Memory (LSTM) are designed for sequential data processing and are limited by the relatively short span of the query dependency that can be captured by such networks (Mustar et al. 2020, 2021). On the other hand, the recently proposed Transformer networks (Vaswani et al. 2017) can process data in parallel. In addition, the standard deep autoregressive models are not designed for real-time interactive query predictions and recommendations as the learning is achieved over historical data. Hence, immediate user feedback is not considered in the decision-making. All in all, a query prediction mechanism in sequential knowledge gathering should be able to *adapt* to the immediate user feedback and query context to continuously improve its performance, as it will be demonstrated by our mechanism and our experiments.

1.2 Contribution

We contribute with a MAB mechanism dealing with the sequential query prediction problem. Our mechanism combines the effectiveness of Transformer-based auto-regressive models and the adaptivity of the MAB expert algorithms (Cesa-Bianchi et al. 1997; Cesa-Bianchi and Lugosi 2006) under non-stochastic rewards. We adopt an ensemble of state-of-the-art Transformer-based auto-regressive models for causal language modelling, hereinafter referred to as the *experts*, and propose a next-query prediction algorithm that adapts to immediate user feedback. In standard non-stochastic MAB approaches (Auer et al. 2002; Lattimore and Szepesvári 2020), the number of possible queries is fixed and known to the prediction mechanism in advance. On the other hand, in query prediction with Transformers, given the current query context, the number of possible candidate queries, even from a single expert, can be countably many, with varying relevance probabilities. Most importantly, possible candidate queries *change* in every step depending on the query context.

We first show that predicting and recommending the top query from a single expert (e.g., an autoregressive model) is not always the best strategy (Table 1). Then, departing from this fundamental outcome, our contribution is the introduction of a non-stochastic MAB algorithm, an extension to the popular Exp3 algorithm,

Table 1 Top-2 Recommendations (Representative Example)

Model 1's prediction	Model 2's prediction
Top-1: *factors that influence the profitability of insurance*	Top-1: *factors influencing the amount of insurance*
Top-2: *factors that influence purchasing decisions of insurance*	Top-2: *factors that affects getting disability insurance*

that *dynamically* chooses a *variable size* candidate set via an ensemble of different experts' recommendations/predictions extending the capacity of Exp3 by Auer et al. (2002) toward non-fixed and unknown sizes of sets of dynamically changing queries. In Sect. 3, we provide a theoretical analysis and prove the *regret bound* of our MAB algorithm showing that it achieves better regret than the standard baseline Exp3 algorithm, and matches the best attainable regret in adversarial settings (Audibert et al. 2009). In Sect. 4, we provide extensive experiments and comparative assessment against mechanisms and baseline models found in the literature namely: Exp3 (Auer et al. 2002), GPT2 (Radford et al. 2019), Transformer-XL (Dai et al. 2019), CTRL (Keskar et al. 2019), BERT (Mustar et al. 2020), BART (Lewis et al. 2020), and HRED (Sordoni et al. 2015), using query logs from online literature discovery services. Furthermore, our experiments show that fine-tuning publicly available pre-trained Transformer models trained on a huge corpus provide marginally better performance than learning a model from scratch with limited query log data. Finally, Sect. 5 concludes the article with future research directions. (Note: in the remainder, the terms recommended query and predicted query are used interchangeably, depending on the context.)

2 Related work

In this section, we review learning models adopted for query prediction providing a representative running example and elaborate on MAB models for on-line query prediction taking into account the user feedback.

2.1 Autoregressive and causal language models for query prediction

The standard unsupervised language Autoregressive (AR) models' objective is predicting the next word token (comprising a query), having processed all the previous ones. Such models have been popular in various Natural Language Processing (NLP) tasks, while recently they have been used for knowledge retrieval, query prediction, query reformulation, and document ranking (Dehghani et al. 2017; Ren et al. 2018). Specifically, Recurrent Neural Networks (RNN) are AR models adopted for query prediction due to their ability to process variable length input sequences (Ahmad et al. 2019; Jiang and Wang 2018; Wu et al. 2018; Sordoni et al. 2015). Dehghani et al. (2017) augmented the RNN-based sequence-to-sequence (seq2seq)

models for query prediction with a query-aware attention mechanism (Bahdanau et al. 2015) and a pointer network (Vinyals et al. 2015) to deal with the Out-of-Vocabulary words. Wu et al. (2018) proposed a method to integrate historical user feedback in RNN-based query prediction approaches. The proposed method uses historical interaction data to create embeddings and attention scores for queries by combining query-tokens associated with search result contents (e.g., hyperlinks), and corresponding 'click' streams. Depending on whether a hyperlink is 'clicked' or not, both positive and negative feedback is learned for different queries. Although such an approach makes use of historical user interaction data, the prediction strategy is not online and does not adapt to immediate feedback.

The authors in Ahmad et al. (2019) introduced a context attentive document-ranking and query prediction algorithm. Such algorithm encodes the query and search session activities into search context representation via a two-level RNN. Query and document embeddings are constructed from pre-trained word embeddings using bi-directional RNN with an inner attention mechanism. The search context embeddings are obtained by passing the query and relevant documents through a LSTM model. The query prediction consists of a series of word-level predictions that forecast the next word, given the current query and search context.

Query prediction is also studied under query reformulation. Ren et al. (2018) studied the query reformulation problem within the context of conversational information retrieval using a LSTM with self-attention for seq2seq modelling. Similarly, Jiang and Wang (2018) framed the query prediction task as a query reformulation one. At each step t, the new query to be predicted is a reformulation of the past issued queries until step $(t-1)$. The query generator in Jiang and Wang (2018) is an RNN-based decoder that generates the next query based on the context vector formed by query reformulation embeddings. Recently, with the advent of deep neural networks, many query prediction and recommendation algorithms have been proposed. The interested reader could refer to overview papers for query recommendation in Dehghani et al. (2017), Ren et al. (2018), Ahmad et al. (2019), Jiang and Wang (2018), Wu et al. (2018) and the references therein.

The parallel processing capabilities and the ability to capture complex query dependencies yield the Transformer networks (Vaswani et al. 2017) popular for Causal Language Modelling (CLM) tasks. Fundamentally, the CLM objective is predicting the next token following a sequence of tokens such that the prediction for the current position depends *solely on the known tokens* in the positions before the current position. The CLM objective is well suited for sequential query recommendations, as a Transformer network can access all the queries until the current round. Recently, Mustar et al. (2020, 2021) proposed bidirectional Transformer networks, like BERT (Devlin et al. 2019) and BART (Lewis et al. 2020) for query predictions. The empirical study in the BERT and BART papers showed a significant improvement in performance over the standard RNN-based models when pre-trained Transformer networks are fine-tuned for query prediction. In particular, Mustar et al. (2021) showed that even training hierarchical Transformer models from scratch on limited data under-performs compared to pre-trained models. However, BERT and BART are trained using masked language modelling objectives, and thus, the prediction for the current position makes use of random tokens in preceding and

following positions. In the context of query prediction, we show that CLM is better suited as the Transformer network cannot 'look into the future'. Nevertheless, Mustar et al. (2021) does not make use of immediate feedback when predicting the next query, thus, being unaware of the current user interactions and intention for knowledge extraction in exploratory tasks.

The current state-of-the-art AR models for query prediction recommend the top queries given a query context. Nonetheless, our preliminary analysis showed that combining different AR models, like Transformer-based models, query prediction is drastically enhanced by the query context awareness compared to employing single models.

Representative Example: Consider the following representative example. We fed two consecutive queries: q_1 = *'factors that affect getting life insurance'* and q_2 = *'purchasing factors in getting life insurance'* from a randomly selected user session from our query logs to two different Transformer-based networks (Model 1 and Model 2) for next query prediction. The results obtained based on the top-2 query recommendations are provided in Table 1. From the query context (q_1, q_2), it is evident that the user is interested in gathering knowledge regarding different aspects of getting life insurance. In that regard, both models suggest very relevant yet diverse terms as the predicted query keywords. But, as the top queries from the two models differ in the information content and direct the user to different search results, the simple strategy of recommending the top prediction result(s) from one model might not lead the user to reach the intended result. A better strategy could be to recommend from a mixture of the top results and then adapt the prediction based on the immediate user feedback (acceptance or rejection of the current forecast). In the example, the *actual* queries following the first two queries are q_3 = *'factors influencing the purchase of life insurance'* and q_4 = *'factors influencing the insurance amount'*. The second query (Top-2) from the Model 1 and the first one (Top-1) from Model 2 convey the user's correct intention.

In conclusion, the principle differences between our model and the state-of-the-art AR models in the context of query prediction and recommendation are: *(i)* instead of simply recommending the top predicted query from a single AR model, we dynamically create a candidate set of predicted queries by combining top queries from different models. Then, we recommend a query probabilistically by on-line estimating a probability distribution over this set of candidate queries; *(ii)* state-of-the-art AR models do not make use of immediate feedback, thus, being unaware of the current user intentions in the knowledge discovery process, whereas we exploit immediate user feedback to estimate and gradually adapt the probability distribution over the set of candidate queries w.r.t. feedback.

2.2 Multi-armed bandit systems for query prediction

MAB is a popular framework to model interactive learning applied, mostly, to web-based services like ad recommendation, dynamic pricing, digital health, etc (Bouneffouf et al. 2020; Lattimore and Szepesvári 2020). In query prediction, MAB provides a framework to develop on-line learning strategies that adapt to immediate

user feedback. The user feedback is assumed to be the *reward* for the recommendation of the predicted query. The MAB setting for query prediction can be envisaged as a two-player interactive game between the learner (algorithm) and the environment (e.g., user), where the following steps are repeated in a sequence of rounds $t = 1, \ldots, T$:

- For each query $q \in \mathcal{C}$, the environment chooses a loss vector;
- The learner chooses a query $q_t \in \mathcal{C}$ and recommends that to the user;
- The learner observes the reward r_t for the chosen query q_t and adapts its learning method.

As mentioned, the reward is observed as click/acceptance. For the correct prediction, the user clicks on the recommended query (accept) and ignores it otherwise (reject). The learner's goal is to predict queries so that their total reward is as high as possible.

The standard MAB algorithms assume that the number of queries is fixed and relatively small compared to the number of steps. However, models like Transformers can generate a relatively large number of predicted queries that can be used as possible candidates for recommendation. Hence, in this context, the candidate queries in a MAB algorithm can be countably many for query prediction and recommendation. In countably many candidate queries (corresponding to armed bandits in MAB terminology), the MAB algorithm has to deal with the discovery-exploitation trade-off during the exploration phase (Carpentier and Valko 2015), apart from the conventional exploration-exploitation trade-off. Several MAB algorithms for countably many or infinite armed bandits have been proposed. Fundamentally, Berry et al. (1997) proposed *k-failure* and *m-run* strategies. In a *k-failure* strategy, an arm is played until it incurs k failures. In *m-run* strategy, 1-*failure* is used until m arms are played or m successes are obtained. Another strategy to deal with infinite arms is *pre-selection*, where a subset of k arms is selected, and then standard MAB algorithms can be applied. Wang et al. (2008) proposed to select k randomly chosen arms for exploration and exploitation as a function of the current step, while Zhu and Nowak (2020) proposed to choose k arms uniformly at random. The authors in Bayati et al. (2020) proposed a greedy algorithm that selects arms uniformly at random without replacement. Recently, Parambath et al. (2021) proposed a stochastic MAB algorithm for query prediction and recommendation by selecting a candidate set using the utility of the queries. The work in Parambath et al. (2021) is fundamentally different from ours as their approach deals with a pre-selection strategy to choose the queries to run standard stochastic MAB algorithms. Nonetheless, the approach in Parambath et al. (2021) does not consider the AR nature of the query process, as elaborated earlier. The reader could refer to Kalvit and Zeevi (2020), Zhu and Nowak (2020), Bayati et al. (2020), Kleinberg et al. (2019), Lattimore and Szepesvári (2020), Parambath et al. (2021) and references therein for a comprehensive review on MAB algorithms.

Finally, we report on the *sleeping bandits* approach to provide a spherical review of the literature. Such an approach is closely related to the query prediction problem, albeit with significant differences. The standard adversarial sleeping bandit is

studied from the sleeping expertsâ€™ point of view and not from the sleeping action point of view (Kleinberg et al. 2010; Kanade et al. 2009; Saha et al. 2020). Specifically, in our context, experts are Transformer-based oracles, where the actions are the recommended queries from the oracles at any step. The oracles themselves will not be unavailable (*sleeping*) at any point in time. Moreover, the available queries for different oracles at any point in time can be different or the same. Once the queries become available, they will be added to the candidate set and will never become unavailable again. The fundamental difference is, then, that none of the sleeping bandits' works considers countably many action sets of arms; thus, they cannot be adopted in our context.

Overall, the main difference between our approach and the standard adversarial MAB approaches is that, in the standard adversarial settings, the number of queries is assumed to be finite and known beforehand. In our problem, the number of queries is *dynamic*, it can *change* in every round, and it is *not known* beforehand. To the best of our knowledge, our work in this paper is the first one that combines the non-stochastic MAB framework with an ensemble of Transformer networks for query prediction with real-time user feedback adaptation.

3 Ensemble-based sequential query prediction with immediate feedback

In this section, we introduce our MAB mechanism based on an ensemble of language model experts, which leverages immediate user feedback for the next query prediction tasks. The corresponding algorithm and regret analysis of our mechanism are reported to provide insights into the capacity of our mechanism to build incrementally candidate query sets for prediction.

3.1 Non-stochastic MAB with transformer-based experts and feedback

Our mechanism, coined *TEF* for non-stochastic MAB with Transformer-based Experts and Feedback, is provided in Algorithm 1. The fundamental feature of TEF is that the set of queries is not fixed and not given beforehand to the algorithm. Specifically, for each user session, TEF builds a candidate query set incrementally, starting with an empty set $\mathcal{C}_0$. Then, TEF starts off the learning process with an ensemble of Transformer-based experts $\mathcal{E}$, a recommendation threshold k, and a learning rate $\eta \in (0, 0.5)$.

At each time step $t > 0$ of the learning process, TEF observes the currently executing query q_t selected by the user. Then, using the query context $q_1, \ldots, q_t$, TEF probes each of the Transformer-based experts for the next queries to be predicted and then decides on the corresponding recommendation. These predicted queries are added to the candidate set. The algorithm then estimates the current distribution over the set of these queries in the candidate set, which is the basis for recommending a query by sampling from this distribution.

Algorithm 1 TEF -MAB Query Prediction w.r.t. Transformer-based Experts & Feedback

Parameters : Set of experts $\mathcal{E}$, learning rate $\eta \in (0, 0.5)$, recommendation threshold k

Initialization: query candidate set $\mathcal{C}_0 = \emptyset$

1 **for** *step* $t = 1, 2, \ldots T$ **do**

2 Get current executing query q_t and form the query context v_t;

3 $\mathcal{Q}_t = \bigcup_{e \in \mathcal{E}} f(e, v_t, k)$; `// API call to Transformer-based experts`

4 $\mathcal{C}_t = \mathcal{Q}_t \cup \mathcal{C}_{t-1}$;

5 /*Assign and update query weights*/;

6 **if** $(t == 1)$ **then**

7 $w_{i,t} = \frac{\eta}{(1-\eta)|\mathcal{C}_t|}, \quad \forall i \in \mathcal{C}_t$;

8 **else**

9 $w_{i,t} = \frac{\eta}{1-\eta} \frac{\mathbb{I}\{i \in \mathcal{C}_t \setminus \mathcal{C}_{t-1}\}}{|\mathcal{C}_t \setminus \mathcal{C}_{t-1}|} \sum_{i \in \mathcal{C}_{t-1}} \hat{w}_{i,t} + \hat{w}_{i,t} \mathbb{I}\{i \in \mathcal{C}_{t-1}\}$;

10 Estimation of query probability distribution

$p_{i,t} = \frac{(1-\eta)w_{i,t}}{\sum_i w_{i,t}} + \frac{\eta}{|C_t|} \quad \forall i \in \mathcal{C}_t$;

11 Sample I_t according to distribution p_t ;

12 Recommend the query $q'_{I_t} \in \mathcal{C}_t$ and capture user reward $r_{I_t,t}$;

13 Calculate the pseudo-rewards: $\hat{r}_{j,t} = \begin{cases} \frac{r_{j,t}}{p_{j,t}}, & \text{if } j = I_t \\ 0, & \text{else} \end{cases}$;

14 Update the query weights based on pseudo-rewards:

$\hat{w}_{j,t+1} = \begin{cases} w_{j,t} \cdot \exp(\eta \hat{r}_{j,t}), & \text{if } j = I_t \\ w_{j,t}, & \text{else} \end{cases}$;

In detail, at the time t, the algorithm observes the currently executing query q_t (line:2). The q_t can be either the free text entered by the user or one of the recommended queries in the previous step $t-1$. The sequence of queries issued by the user, i.e., the combination of currently executing query q_t and the queries issued till the time step $t-1$, form the current query context, i.e., the sequence $v_t = (q_1, q_2, \ldots q_t)$. In line:3, we call the function $f(e, v_t, k)$, which is a standard API function call for a finite set $\mathcal{E}$ of Transformer-based experts $e \in \mathcal{E}$. The function takes the query context v_t, expert instance e, and a threshold k as the arguments and returns the predicted queries from the expert e. For example, if e is an instance of a GPT model in Radford et al. (2019), the function returns the predicted queries obtained by GPT with current query context v_t as the input. The k argument controls the number of the predicted queries for each expert. It can be either a probability threshold or a count. In the case of probability threshold, the expert returns the predicted queries whose probability of relevance to the query context is at least $k > 0$. In the case of the count, this corresponds to a fixed number of predicted queries returned by the

expert, e.g., the top-5 predicted queries. In our experiments, we set k as a count threshold. The rationale of a count threshold is that, instead of simply obtaining only the top recommendation, which is not an efficient approach as described above, TEF considers *all* the available query predictions that satisfy the threshold k. We carry out this step for all the experts, and the set of the predicted queries $\mathcal{Q}_t$ is obtained by taking the union of the individual predictions per expert.

In turn, in line:4, the candidate query set is updated so that the candidate set at step t is obtained by combining different experts' predictions at step t along with the candidate query set at step $t-1$. In line:7 and 9, we define certain weights for the queries in the candidate set. We use a fixed weight for the newly predicted queries (at time t), whereas, for the older ones (at times $< t$), we adopt the same weight as in the previous rounds. Specifically, at step t, queries that are part of the candidate queries in step $t-1$, we use the same weights as those at the end of step $t-1$. While, for the newly added queries, i.e., those which are not part of the candidate query set at step $t-1$ but added at step t, a weight value is assigned factored by $\frac{\eta}{1-\eta}$ and divided by the number of the new queries in the candidate set. The weights will be used for sampling a query from the candidate set for the query prediction.

Given the updated weights, the algorithm estimates the current probability distribution p_t over the predicted queries in the candidate set. At each step, t, the algorithm samples an index I_t from the candidate set according to the distribution p_t that is proportional to the assigned weights at time t and *recommends* the query q'_{I_t} to the user (line:10). The algorithm then observes the reward for the recommended query and calculates the *pseudo* reward (line:13) as in the standard expert-based algorithms. The pseudo-reward is an unbiased estimator of the reward. Then, depending on the user feedback, the weights are updated at the end of each round to *reflect* that feedback (line:14). To update the weights, one can use any commonly employed *potential function* (Cesa-Bianchi and Lugosi 2006) like polynomial potential function or exponential potential function. Our algorithm's weights are updated using the exponential weighting scheme due to efficient regret performance (Cesa-Bianchi and Lugosi 2006).

3.2 TEF regret analysis

The performance of online algorithms is evaluated using regret. Formally, regret is defined as the loss suffered by the algorithm relative to the optimal strategy. After T rounds, the cumulative regret of an online algorithm is defined as:

$$R(T) = T - \sum_{t=1}^{T} r_t. \tag{1}$$

We prove that the cumulative regret in (1) of the proposed TEF matches with the regret order of the INF algorithm proposed in Audibert et al. (2009). The INF algorithm has the best obtainable regret in the adversarial MAB setting (disregarding the constant factors).

Theorem 1 *For any* $T > 0$, $|\mathcal{C}_T| > 0$, *and learning rate* $\eta = \min(0.5, \sqrt{\frac{T+1}{T(2|\mathcal{C}_T|+1)}})$, *the regret of the TEF algorithm is* $\mathcal{O}\left(\sqrt{\frac{|\mathcal{C}_T|}{T}}\right)$.

Proof Our proof is based on the assumption that at any time t, the candidate set contains the optimal query. The proof extends the proof of Exp3 to handle the dynamic nature of the candidate set. At each step t, the number of queries to be recommended may change. We denote the set of candidate queries available at time step t as $\mathcal{C}_t$. For convenience in analysis, we assume that $|\emptyset| = 1$.

Let us define the variable W_t as the sum of the weights of the queries in the candidate set $\mathcal{C}_t$ at the end of the t-th round, i.e., $W_t = \sum_{i \in \mathcal{C}_t} w_{i,t}$. Firstly, we derive a lower bound such that:

$$\ln \frac{W_{T+1}}{W_1} = \ln \sum_{i \in C_{T+1}} w_{i,T+1} - \ln W_1. \tag{2}$$

Then,

$$\ln \frac{W_{T+1}}{W_1} \geq \ln w_{j,T+1} - \ln \frac{\eta}{1-\eta} \qquad \text{for any } j \in \mathcal{C}_{T+1} \tag{3}$$

Thus,

$$= \eta \sum_{t=1}^{T} \hat{r}_{j,t} - \ln \frac{\eta}{1-\eta}. \tag{4}$$

$$\ln \frac{W_{T+1}}{W_1} \geq \eta \sum_{t=1}^{T} \hat{r}_{j,t} - \ln \frac{\eta}{1-\eta}. \tag{5}$$

It should be noted that (3) holds for any query in the candidate set (and not just for the query that is recommended). To emphasize this, we use the index j. Secondly, we bound the weights to derive a suitable upper bound for $t < T$, i.e.,

$$\ln \frac{W_{t+1}}{W_t} = \ln \left(\sum_{i \in \mathcal{C}_{t+1}} \frac{w_{i,t+1}}{W_t} \right) \tag{6}$$

$$= \ln \left(\sum_{i \in \mathcal{C}_t} \frac{\hat{w}_{i,t+1}}{W_t} + \sum_{i \in \mathcal{C}_{t+1} \setminus \mathcal{C}_t} \frac{w_{i,t+1}}{W_t} \right) \tag{7}$$

$$= \ln \left(\sum_{i \in C_t} \frac{\hat{w}_{i,t+1}}{W_t} + \frac{\eta}{1-\eta} \right) \tag{8}$$

$$= \ln\left(\sum_{i\in\mathcal{C}_t}\frac{w_{i,t}\exp\left(\eta\hat{r}_{I_t,t}\right)}{W_t} + \frac{\eta}{1-\eta}\right) \tag{9}$$

By using $|\mathcal{C}_t| \geq 1$ and $w_{i,t} = \frac{W_t}{1-\eta}\left(p_{i,t} - \frac{\eta}{|\mathcal{C}_t|}\right)$, and approximating $e^x \leq 1 + x + x^2$ for $x < 1.79$, we obtain that:

$$\ln\frac{W_{t+1}}{W_t} \leq \ln\left(\frac{\eta}{1-\eta} + \frac{1}{1-\eta}\sum_{i\in\mathcal{C}_t}\left(p_{i,t} - \frac{\eta}{|\mathcal{C}_t|}\right)\left(1 + \eta\hat{r}_{I_t,t} + \eta^2\hat{r}^2_{I_t,t}\right)\right) \tag{10}$$

$$\leq \ln\left(\frac{\eta}{1-\eta} + \frac{1}{1-\eta}\sum_{i\in\mathcal{C}_t}\left(p_{i,t} + \eta p_{i,t}\hat{r}_{I_t,t} + \eta^2 p_{i,t}\hat{r}^2_{I_t,t} - \frac{\eta}{|\mathcal{C}_t|}\right)\right) \tag{11}$$

$$= \ln\left(\frac{\eta}{1-\eta} - \frac{1}{1-\eta}\sum_{i\in\mathcal{C}_t}\frac{\eta}{|\mathcal{C}_t|} + \frac{1}{1-\eta}\left(\sum_{i\in\mathcal{C}_t}\left(p_{i,t} + \eta p_{i,t}\hat{r}_{I_t,t} + \eta^2 p_{i,t}\hat{r}^2_{I_t,t}\right)\right)\right) \tag{12}$$

$$= \ln\left(\frac{1}{1-\eta} + \frac{1}{1-\eta}\sum_{i\in\mathcal{C}_t}\left(\eta p_{i,t}\hat{r}_{I_t,t} + \eta^2 p_{i,t}\hat{r}^2_{I_t,t}\right)\right) \tag{13}$$

$$\leq \ln\left(2 + \frac{1}{1-\eta}\sum_{i\in\mathcal{C}_t}\left(\eta p_{i,t}\hat{r}_{I_t,t} + \eta^2 p_{i,t}\hat{r}^2_{I_t,t}\right)\right) \tag{14}$$

$$\leq 1 + \frac{\eta}{1-\eta}\sum_{i\in C_t} p_{i,t}\hat{r}_{I_t,t} + \frac{\eta^2}{1-\eta}\sum_{i\in\mathcal{C}_t} p_{i,t}\hat{r}^2_{I_t,t} \tag{15}$$

$$= 1 + \frac{\eta}{1-\eta}r_{I_t,t} + \frac{\eta^2}{1-\eta}\sum_{i\in\mathcal{C}_t}\hat{r}_{I_t,t}\mathbb{1}\{i == I_t\} \tag{16}$$

In (10), we expanded the equation by multiplying respective terms. In (11), we used the fact that η, and $\hat{r}_{I_t,t}$ are non-negative values. In (13), we used the fact that $\sum_{i\in\mathcal{C}_t} p_{i,t} = 1$ and $\sum_{i\in\mathcal{C}_t}\frac{\eta}{|\mathcal{C}_t|} = \eta$. In (15), we used the inequality $\ln(1+x) \leq x$ for $x > 0$ and in (15) we assume that rewards are bounded in (0, 1). $\mathbb{1}\{i == I_t\}$ in (16) denotes the indicator variable. By summing over t, we obtain:

$$\ln\frac{W_{T+1}}{W_1} \leq T + \frac{\eta}{1-\eta}G_T + \frac{\eta^2}{1-\eta}\sum_{t=1}^{T}\sum_{i\in\mathcal{C}_t}\hat{r}_{I_t,t}\mathbb{1}\{i == I_t\} \tag{17}$$

$$\leq T + \frac{\eta}{1-\eta} G_T + \frac{\eta^2}{1-\eta} |\mathcal{C}_T| \sum_{t=1}^{T} \hat{r}_{I_t,t}. \tag{18}$$

In (17), G_T refers to the sum of the rewards, i.e., $G_T = \sum_{t=1}^{T} r_{I_t,t}$. We then combine the lower bound in (5) and upper bound in (18), thus, obtaining,

$$\eta \sum_{t=1}^{T} \hat{r}_{j,t} - \ln \frac{\eta}{1-\eta} \leq \frac{\eta}{1-\eta} G_T + \frac{\eta^2}{1-\eta} |\mathcal{C}_T| \sum_{t=1}^{T} \hat{r}_{I_t,t} + T \qquad \text{or,} \tag{19}$$

$$\frac{\eta}{1-\eta} G_T + \ln \frac{\eta}{1-\eta} \geq \eta \sum_{t=1}^{T} \hat{r}_{j,t} - \frac{\eta^2}{1-\eta} |\mathcal{C}_T| \sum_{t=1}^{T} \hat{r}_{I_t,t} - T \tag{20}$$

$$\frac{\eta}{1-\eta} G_T + 1 \geq \eta \sum_{t=1}^{T} \hat{r}_{j,t} - \frac{\eta^2}{1-\eta} |\mathcal{C}_T| \sum_{t=1}^{T} \hat{r}_{I_t,t} - T \tag{21}$$

In (21), we used the fact that,for any plausible values of η, $\ln \frac{\eta}{1-\eta} \leq 1$. Taking the expectation in both sides, we get

$$\frac{\eta}{1-\eta} \mathbb{E}[G_T] + 1 \geq \eta \sum_{t=1}^{T} r_{j,t} - \frac{\eta^2}{1-\eta} |\mathcal{C}_T| \sum_{t=1}^{T} r_{I_t,t} - T. \tag{22}$$

Based on the fact that the inequality holds for any $j \in \mathcal{C}_t$ and the assumption that the optimal query always exists in the candidate set to obtain, we have that,

$$\frac{\eta}{1-\eta} \mathbb{E}[G_T] + 1 \geq \eta G^* - \frac{\eta^2}{1-\eta} |\mathcal{C}_T| G^* - T, \tag{23}$$

where in (23), $G^* = \max_j \sum_{t=1}^{T} r_{j,t}$ is the cumulative reward from recommending the single globally best query.

As the rewards are bounded in (0,1), the maximum cumulative reward is therefore bounded by T, i.e., $G^* \leq T$. We can then re-write (17) and obtain,

$$\eta G^* \leq 1 + \frac{\eta}{1-\eta} \mathbb{E}[G_T] + \frac{\eta^2}{1-\eta} |\mathcal{C}_T| T + T. \tag{24}$$

By converting the *gain* into *loss*, i.e., $L_T = T - G_T$ and $L^\star = T - G^\star$, and using the facts that $\frac{1}{1-\eta} \leq 2$ and $\mathbb{E}[G_T] \leq T$, we obtain:

$$\mathbb{E}[L_T] - L^\star \leq \frac{T+1}{\eta} + 2\eta |\mathcal{C}_T| T. \tag{25}$$

By taking a learning rate $\eta = \sqrt{\frac{T+1}{2T|\mathcal{C}_T|}}$, we obtain that the per-round regret:

$$\frac{L_T - L^\star}{T} \leq \mathcal{O}\left(\sqrt{\frac{|\mathcal{C}_T|}{T}}\right). \tag{26}$$

□

We can observe that the regret of TEF matches the best obtainable regret in the adversarial bandit setting (disregarding the constant factors) in Audibert et al. (2009), and it is better than the regret obtained using the baseline Exp3 algorithm (Auer et al. 2002), which the later assumes that the candidate set is known in advance.

Remark 1 The Exp3 (Auer et al. 2002) algorithm with the known candidate set and known cardinality m achieves a regret of the order $O\left(\sqrt{\frac{m \ln m}{T}}\right)$, which is $\ln m$ higher than the regret obtained by our TEF algorithm with $m = |\mathcal{C}_T|$.

4 Experimental evaluation

In this section, we report on the dataset, baseline approaches and models under comparison, and the performance metrics adopted in our experimental study. Furthermore, we provide an in-depth analysis of the results of different query prediction and recommendation mechanisms using the Transformer-based models. For reproducibility, our source code is publicly available at: https://github.com/shampp/tef.

The purpose of this experimental study is threefold: We empirically show that (i) Transformer-based models trained with CLM objective perform better than Transformer-based models trained using Masked Language Modeling (MLM) objective; (ii) adversarial MAB strategies by combining the query predictions from an ensemble of experts outperform strategies based on recommendations using a single CLM-based Transformer model as an expert, and (iii) Transformer-based models trained using limited historical query logs slightly underperform publicly available pre-trained Transformer models trained on huge corpus and fine-tuned for query recommendations.

4.1 Datasets and protocol

Our experiments are conducted on a real large-scale query log from an online literature discovery service. The dataset is used in the past (Parambath et al. 2021) to test stochastic MAB algorithms for query recommendation. The online literature discovery service lets the user/analyst start the knowledge-gathering session by entering a free-text query. In each subsequent step, the user either chooses from the recommended queries or inputs another free-text new query. The logs contained 4.5 million queries grouped into 547,740 user sessions. We followed the same data preparation techniques as in the paper (Parambath et al. 2021). As the pre-processing step, we removed user sessions with less than four queries. We also removed sessions

Table 2 Experimental Dataset Statistics

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

containing non-English queries. The statistics of the final data used in the experiment are given in Table 2. We refer the readers to Parambath et al. (2021) for further details about the data.

In our experiments, at each round, we select user sessions one by one and recommend the predicted queries in each session. The process is carried out for $T = 500$ rounds, where T is the horizon. In each user session, we treat the first query as the initial query. We then recommend predicted queries sequentially. In each round, we treat all the executed queries in the selected user session as the query context v_t. We then feed the query context to get recommendations from the ensemble of experts. For user session with less than 500 queries, we reset the initial query and the query context to restart the on-line learning process.

4.2 Performance metrics

The performance of online algorithms is evaluated using the regret defined in (1). In the predicted query recommendation, we adopt binary rewards, such that the query recommendation algorithm receives a reward $r_t = 1$ if the recommended query is clicked (accepted) by the analyst at step t; $r_t = 0$ otherwise. In our offline experiments, we mimic the click/accept behaviour using the metric BLEU. BLEU is a well-established metric widely used in evaluating the performance of query recommendation tasks (Mustar et al. 2021; Ahmad et al. 2019; Ren et al. 2018; Mustar et al. 2020). BLEU was originally proposed to measure the correctness of machine translation tasks (Papineni et al. 2002) and later adopted for tasks like query recommendation and prediction. Specifically, we adopt the BLUE metric to determine the reward value r_t. BLEU is defined as the ratio of the generated words in the queries that are present in the predicted query *and* the observed (actual query issued by the user) query. In our experiments, to calculate the regret, we assign rewards based on the value of the BLEU metric. We assign $r_t = 1$, if the predicted query and observed query share more than half of the words, and a reward $r_t = 0$, otherwise.

4.3 Baseline approaches and models under comparison

We conducted a comprehensive experimental evaluation of different models and algorithms with different Transformer-based models. In Mustar et al. (2021), authors compared different state-of-the-art query recommendation models, including RNN models, and established that the Transformer model outperforms other models. In our study, we employed three CLM-based models and two MLM-based Transformer

models. We used GPT2, CLRT, and Transformer-XL as the CLM models. Moreover, following Mustar et al. (2020, 2021), we used BERT and BART as the MLM models. For our proposed algorithm TEF , we used the above-mentioned five Transformer models as experts, i.e., TEF makes use of predictions from an ensemble of five experts: GPT2, CLRT, Transformer-XL, BERT and BART.

We experimented with two different setups for the above-discussed Transformer models. In the first setup, we used the publicly available pre-trained models from HuggingFace.[1] We fine-tuned the pre-trained models using our training data and used them as the experts for TEF . In the second setup, we trained each of the Transformer models discussed above from scratch using the query logs as the training data. We used the default hyperparameter values for the deep neural networks in both setups. The most important sets of hyperparameter values are given in Table 3.

The comparison models against TEF in our experiments are as follows:

- *GPT2* is a popular Transformer-based language model trained with the CLM objective, i.e., predict the next word given all of the previous words within some text (Radford et al. 2019). It is, therefore, powerful at predicting the next token in a sequence. The original GPT2 model has 1.5 billion parameters and is trained on a dataset of 8 million diverse web pages containing English text. The GPT model used in our experiments is available from the HuggingFace interface as: *gpt2-medium*.
- The *CTRL* model was proposed in Keskar et al. (2019). Similar to GPT2, it is a unidirectional Transformer network. CTRL makes use of control codes derived from naturally co-occurring structures in the raw text to generate coherent text. CTRL is trained over a huge amount of data from different online services like Amazon reviews, Wikipedia, and Reddit. Our CTRL model is available from HuggingFace interface as: *ctrl*.
- The *Transformer-XL* model was introduced in Dai et al. (2019). It is a model with relative positioning (sinusoidal) embeddings that can reuse previously computed hidden states to attend to a longer context. Hence, it has the potential to learn longer-term dependency. The Transformer-XL model is trained on Wiki-Text-103 dataset and is available as: *transfo-xl-wt103* in the HuggingFace interface.
- *BERT* is a bidirectional (MLM) Transformer model, which is pre-trained using a combination of masked language modelling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia Devlin et al. (2019). We used the same pre-trained model as in Mustar et al. (2020), which is available as: *bert-base-uncased* in the HuggingFace interface.
- *BART* uses a standard seq2seq/machine translation architecture with a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT) (Lewis et al. 2020). The pre-training task involves randomly shuffling the order of the original sentences and a novel in-filling scheme, where spans of text are replaced with a single mask token. The BART model used in our experiments is trained over

[1] https://huggingface.co/models.

the CNN/DM dataset and is available as *facebook/bart-base* in the HuggingFace interface.

- The *HRED* is a hierarchical recurrent encoder-decoder model introduced in Sordoni et al. (2015). Given a user session and any query in that session, HRED encodes the information seen up to that query, i.e., all the previous queries in the session, and then tries to predict the next query, just like in typical language models. But HRED makes use of hierarchical encoder-decoder architecture. The process is iterated throughout all the queries in the session. We used the original implementation provided by the authors.[2]
- *Exp3* Auer et al. (2002) is the baseline MAB model for TEF . Like in our case, Exp3 makes use of the immediate user feedback to estimate the sampling distribution to choose the next query to be recommended. The original Exp3 algorithm works only with a fixed set of arms. Hence, it cannot be used directly in sequential query prediction using an ensemble of Transformer experts. To circumvent this issue, we take at most fifty recommendations from the experts for every initial query and fix this as a static candidate arm set. We chose fifty recommendations, as they can include all the relevant and yet diverse queries pertaining to the initial query.

Remark 2 We excluded the following two baseline models from our experimental evaluation for the following reasons. We excluded the continuously re-trained Transformer model. As a baseline, we could try to retrain or fine-tune any Transformers after accumulating a fixed number of user responses. However, the Transformer models have a large number of parameters which makes constant retraining and fine-tuning significantly inefficient in space and time and contributing to a higher carbon footprint. Furthermore, it is impractical in knowledge-gathering tasks involving query recommendation applications, as predictions should be served in seconds.

In addition, we excluded the EXP4 algorithm proposed in Auer et al. (2002). At each time step of the EXP4 algorithm, it gets a single recommendation from the experts with the goal of finding the best expert (regarding the recommendation) in hindsight. As we pointed out in the revised rationale & motivation, the top-1 recommendation is not always the best one. That is the motivation for taking top-k recommendations from all experts to build the candidate set sequentially. To use EXP4 in our settings, we may have to consider the top-k recommendations from each expert. However, the current literature lacks how one can calculate the rewards for the $k-1$ non-chosen queries from the chosen expert. It is something we are considering as a future work in our agenda. Additionally, in the rationale and motivation section, we exemplified the suboptimal performance of the strategy of recommending from a single expert.

Additionally, EXP4 algorithm is similar to the Hedge algorithm (Freund and Schapire 1997) and aims to find the best-performing expert from the given set of experts by adapting to the immediate feedback. Consequently, EXP4 cannot perform

[2] https://github.com/sordonia/hred-qs.

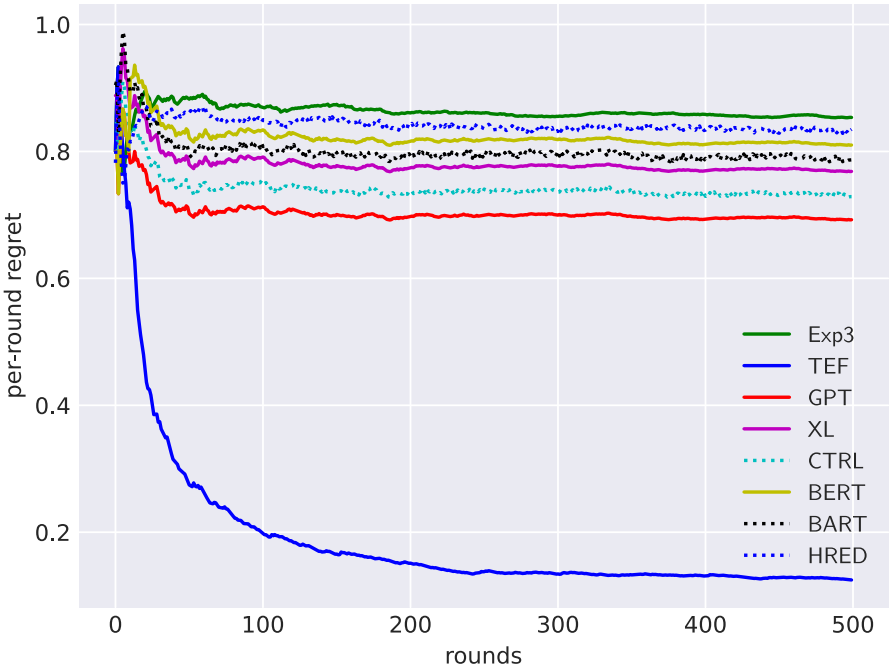

Fig. 1 Per-round regret of different comparison algorithms and baselines using *pre-trained* models

better than the best-performing expert. Since all the individual experts perform poorer than our proposed algorithm, as evidenced in our experimental study (see Figs. 1 and 2), we exclude EXP4 algorithm as a baseline.

4.4 Experimental results and analysis

We carried out three sets of experiments. For all the sets of experiments, we used the top-3 predicted query prediction per expert for TEF , i.e., threshold $k = 3$. The first set of experiments involved comparing fine-tuned pre-trained CLM models against pre-trained MLM models. We used our query logs to fine-tune the publicly available models. These experiments are carried out to show the superiority of the Transformer models trained with CLM objective over the same trained with MLM objective. This set of experiments is inspired by the results of the recent work in Mustar et al. (2020), where the authors showed that fine-tuning pre-trained MLM-based Transformer models outperform standard RNN-based models. For this set of experiments, we compared the three CLM models against two MLM models used in Mustar et al. (2020). The results of this experiment are shown in Fig. 1. As it is evident from the plot, CLM-based models (GPT, CTRL, and Transformer-XL) resulted in lower regret than MLM-based models (BERT and BART). Hence, it is advisable to adopt CLM-based transformer models for query prediction as opposed to

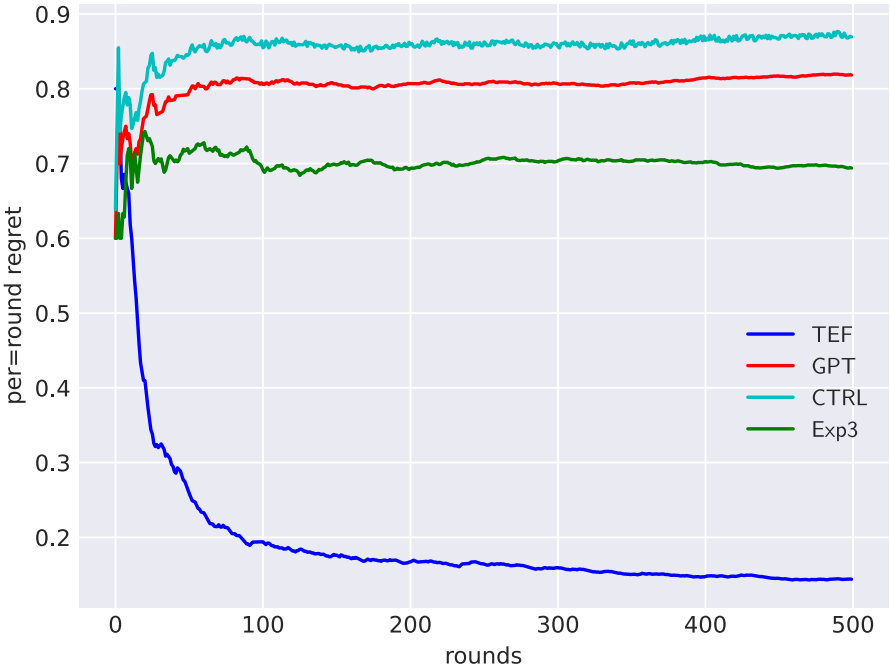

Fig. 2 Per-round regret of different comparison algorithms and baselines using models learned *from scratch*

previous research (Mustar et al. 2020). It is also worth noting that Exp3 with a fixed candidate set performs poorly compared to other baselines, even though it makes use of user feedback. This clearly demonstrates the efficacy of TEF .

In the second and third classes of experiments, we study the performance of different algorithms using experts learned from scratch. We compared different CLM-based transformer models trained only on our query logs (no fine-tuning of pre-trained public models). For this experiment, we used two top-performing CLM models from 1st set of the experiment: GPT and CTRL, and trained them from scratch using the query log as the training data. We used these models learned only from the raw data as the experts in the Exp3 and TEF algorithms. The results are given in Fig. 2. One notable point of this result is that Exp3 performed better than CLM-based baselines. This clearly indicates that fine-tuning models trained on vast amounts of data is very important to get good performance. In both experiments (Figs. 1 and 2) TEF performed significantly much better than recommending the top result from the experts, irrespective of the fact that the experts are fine-tuned or learned from scratch. TEF significantly reduces the regret substantially compared to individual expert recommendations by adopting the adversarial recommendation strategy. This is due to the fact that TEF algorithm is able to recommend the next queries by combining the top predictions from an ensemble of different experts and, thus, increasing the probability

Table 3 Training Parameters of the Models under Comparison

	GPT2	CTRL	Trans-XL	BERT	BART
Embedding dimensionality	1024	1280	1024	768	768
# layers	24	48	18	12	6
# attention heads	16	16	16	12	12
Dropout probability	0.1	0.1	0.1	0.1	0.1
Maximum sequence length	1024	50,000	-	512	1024

of selecting the best recommendation, depending on immediate user feedback and query context. Moreover, TEF is able to exploit not just the top recommendation but the top-*k* recommendations from different experts. As pointed out earlier, by comparing Figs. 1 and 2, one can observe that fine-tuning pre-trained models achieve remarkably better regret compared to models learned from limited data.

To directly compare the performance of TEF with and without fine-tuning, we plot the regret of TEF with fine-tuned pre-trained models against TEF with models learned from scratch in Fig. 3. It can be seen that fine-tuned models give slightly better performance and are preferred over models trained with limited data.

We also study how *fast* the candidate set is increasing as shown in Fig. 4. After 500 rounds, the average size of the candidate set (overall queries) is 23 (with min

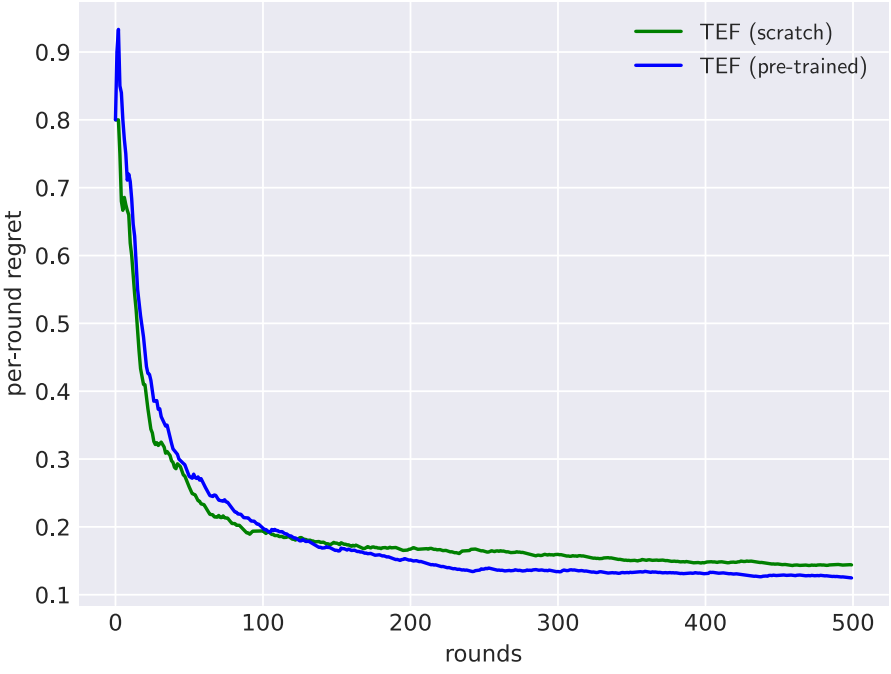

Fig. 3 Per-round regret of TEF using both fine-tuned and trained-from-scratch experts

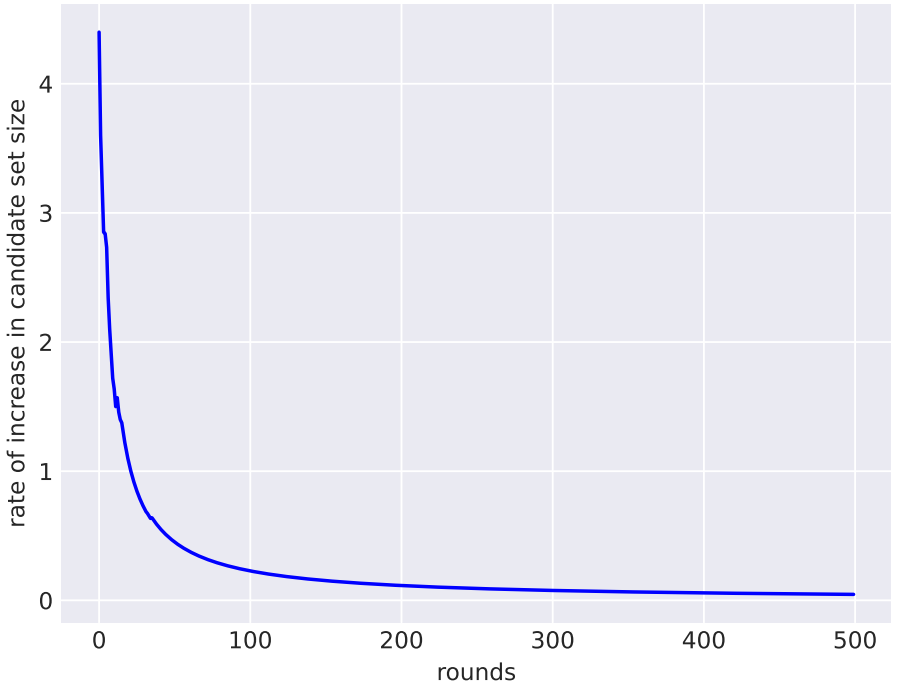

Fig. 4 Increase the rate of the candidate set cardinality vs. rounds (using $k = 2$ query predictions per expert)

and max size values of 8 and 43, respectively). The candidate set remains almost the same after 100 rounds indicating the scalability of our model.

Finally, we tested the sensitivity of the number of expert predictions to be included in the candidate set, i.e., how the regret changes as we change the number of predictions used for the query recommendation. In Fig. 5, we plot the *per-round* regret as the function of the number of the predictions (k) used by TEF from the experts. It is evident that increasing the number of arms (predicted queries) does not significantly improve performance. It should be noted that there can be duplicate entries from the same expert, and often increasing the number of predicted queries and recommendations does not necessarily increase the size of the candidate set.

5 Conclusions

We introduced the TEF algorithm for query prediction and recommendation tasks in interactive data exploratory analysis processes, like knowledge discovery and information gathering. Our algorithm exploits the adversarial MAB setting with experts' advice, to take into consideration immediate user feedback. Our MAB

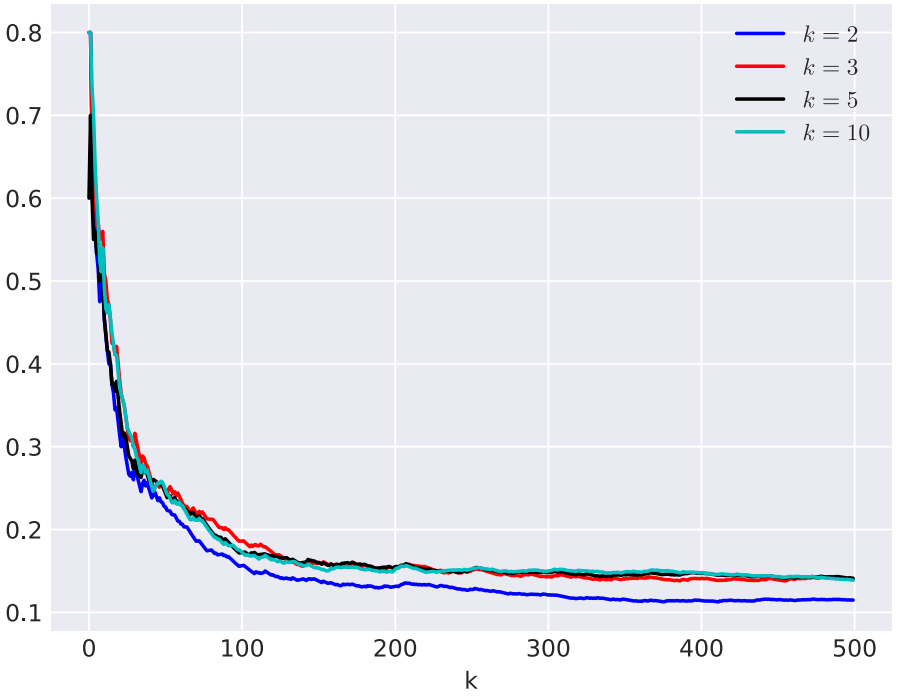

Fig. 5 The per-round regret for different values of k in TEF

mechanism augments the Transformer-based language AR models for query prediction and, in turn, recommendations by combining the predictions from different experts and dynamically building a candidate set in each step of the exploration. We make use of immediate user feedback to choose the appropriate recommended query from the candidate set probabilistically. Our experimental results along with a comprehensive comparative assessment using real data showcasing that TEF significantly improves the cumulative regret compared to Transformer-based models and relevant models found in the literature. In our future research agenda, we are investigating non-linear reward structures for query recommendation in the new framework of 'deep bandits'.

Acknowledgements This research is partially funded by the UK 'Closed-Loop Data Science for Complex, Computationally-and Data-Intensive Analytics' (#EP/R018634/1) and the EU Horizon Grant 'Integration and Harmonization of Logistics Operations' TRACE (#101104278).

Declarations

Competing interests The authors have no Conflict of interest to declare. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication.

References

Ahmad WU, Chang KW, Wang H (2019) Context attentive document ranking and query suggestion. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 385–394

Audibert JY, Bubeck S et al (2009) Minimax policies for adversarial and stochastic bandits. In: COLT, pp 1–122

Auer P, Cesa-Bianchi N, Freund Y et al (2002) The nonstochastic multiarmed bandit problem. SIAM J Comput 32(1):48–77

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: 3rd international conference on learning representations, ICLR 2015

Bayati M, Hamidi N, Johari R et al (2020) Unreasonable effectiveness of greedy algorithms in multi-armed bandit with many arms. In: Larochelle H, Ranzato M, Hadsell R et al (eds) NeurIPS, pp 1713–1723

Berry DA, Chen RW, Zame A et al (1997) Bandit problems with infinitely many arms. Ann Stat 25:2103–2116

Bouneffouf D, Rish I, Aggarwal C (2020) Survey on applications of multi-armed and contextual bandits. In: 2020 IEEE Congress on evolutionary computation (CEC). IEEE, pp 1–8

Carpentier A, Valko M (2015) Simple regret for infinitely many armed bandits. In: ICML, PMLR, pp 1133–1141

Cesa-Bianchi N, Lugosi G (2006) Prediction, learning, and games. Cambridge University Press, Cambridge

Cesa-Bianchi N, Freund Y, Haussler D et al (1997) How to use expert advice. J ACM 44(3):427–485

Dai Z, Yang Z, Yang Y et al (2019) Transformer-XL: attentive language models beyond a fixed-length context. In: ACL, pp 2978–2988

Dehghani M, Rothe S, Alfonseca E et al (2017) Learning to attend, copy, and generate for session-based query suggestion. In: Proceedings of the 2017 ACM on conference on information and knowledge management. ACM, pp 1747–1756

Devlin J, Chang MW, Lee K et al (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, pp 4171–4186

Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139

Jiang JY, Wang W (2018) RIN: reformulation inference network for context-aware query suggestion. In: Proceedings of the 27th ACM international conference on information and knowledge management, pp 197–206

Kalvit A, Zeevi A (2020) From finite to countable-armed bandits. In: Advances in neural information processing systems, pp 8259–8269

Kanade V, McMahan HB, Bryan B (2009) Sleeping experts and bandits with stochastic action availability and adversarial rewards. In: AISTATS, PMLR, pp 272–279

Keskar NS, McCann B, Varshney L et al (2019) CTRL—A conditional transformer language model for controllable generation. arXiv preprint

Kleinberg R, Niculescu-Mizil A, Sharma Y (2010) Regret bounds for sleeping experts and bandits. Mach Learn 80(2):245–272

Kleinberg R, Slivkins A, Upfal E (2019) Bandits and experts in metric spaces. J ACM 66(4):1–77

Lattimore T, Szepesvári C (2020) Bandit algorithms. Cambridge University Press, Cambridge

Lewis M, Liu Y, Goyal N et al (2020) BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th annual meeting of the Association for Computational Linguistics, pp 7871–7880

Mustar A, Lamprier S, Piwowarski B (2020) Using BERT and BART for query suggestion. In: Joint conference of the information retrieval communities in Europe. CEUR-WS.org

Mustar A, Lamprier S, Piwowarski B (2021) On the study of transformers for query suggestion. ACM Trans Inf Syst 40(1):1–27

Papineni K, Roukos S, Ward T et al (2002) BLEU: a method for automatic evaluation of machine translation. In: ACL, pp 311–318

Parambath SAP, Anagnostopoulos C, Murray-Smith R et al (2021) Max-utility based arm selection strategy for sequential query recommendations. In: ACML, PMLR

Radford A, Wu J, Child R et al (2019) Language models are unsupervised multitask learners. OpenAI Blog 1(8):9

Ren G, Ni X, Malik M et al (2018) Conversational query understanding using sequence to sequence modeling. In: Proceedings of the 2018 World Wide Web conference, pp 1715–1724

Saha A, Gaillard P, Valko M (2020) Improved sleeping bandits with stochastic action sets and adversarial rewards. In: ICML, PMLR, pp 8357–8366

Sordoni A, Bengio Y, Vahabi H et al (2015) A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In: CIKM, pp 553–562

Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. In: Guyon I, Luxburg UV, Bengio S et al (eds) Advances in neural information processing systems, vol 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vinyals O, Fortunato M, Jaitly N (2015) Pointer networks. In: Cortes C, Lawrence N, Lee D et al (eds) Advances in neural information processing systems, vol 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf

Wang Y, Audibert JY, Munos R (2008) Algorithms for infinitely many-armed bandits. NIPS 21:1729–1736

Wu B, Xiong C, Sun M et al (2018) Query suggestion with feedback memory network. In: Proceedings of the 2018 World Wide Web conference, pp 1563–1571

Zhu Y, Nowak R (2020) On regret with multiple best arms. In: NeurIPS, vol 33. Curran Associates, Inc., pp 9050–9060