A Contextual Bandit Approach to Dynamic Search

Angela Yang
Department of Computer Science
Georgetown University
Washington, D.C. 20057
asy24@georgetown.edu

Grace Hui Yang
Department of Computer Science
Georgetown University
Washington, D.C. 20057
huiyang@cs.georgetown.edu

ABSTRACT

When users engage in complex search tasks, they encounter two main questions at each point in the search: how to re-formulate the query and whether to continue the search or stop. In this paper, we propose a contextual bandit algorithm to model the dynamic search process. The proposed algorithm uses the context surrounding the current state of the search to select how the search will continue through different query re-formulation tactics. Furthermore, the algorithm automatically decides stopping condition for a search process. Using data from the Text REtrieval Conference (TREC) 2016 Dynamic Domain Track, we evaluate our system's search effectiveness over time and compare to the official runs. Our results show that the use of context as well as an automated stopping condition is effective in a dynamic search system.

1 INTRODUCTION

Dynamic search is a complex search process which involves multiple iterations of searches to accomplish an overall goal. A search topic can be broken down into subtopics that the user navigates through and discovers through their search. For example, a user looking for places to eat in New York City might choose to search the topic "NYC restaurants." In her initial search, she might find a wide variety of relevant information. Thus, she will choose one subtopic, such as "New York style pizza," and continue the search with that subtopic. Afterwards, she may choose another subtopic that she discovered, such as "Michelin star restaurants," and search on this new subtopic. Her complex search task would not be finished until she has searched through the different subtopics in her search task and her information need is fulfilled.

As we can see, in the current setting of a complex search task, a user with a broad information need goes through multiple iterations of query re-formulation to find relevant documents. This process is repeated until their information need has been satisfied. However, this current search process leaves a lot of responsibility with the user, especially placing the burden of query formulation and decisions of navigation on their shoulders. A few efforts [1, 4, 6] have thought about an alternative setting that instead shifts the responsibility to the systems.

ICTIR '17, October 1–4, 2017, Amsterdam, The Netherlands
 ISBN 978-1-4503-4490-6/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3121050.3121101

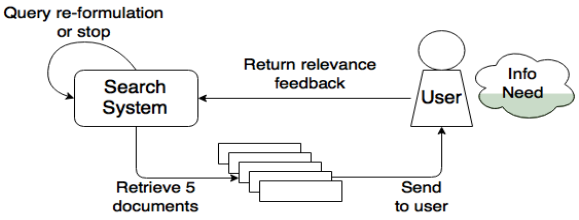

Figure 1: The TREC Dynamic Domain search process

The Text REtrieval Conference (TREC) Dynamic Domain (DD) Track [6], introduced in 2015, researches dynamic search using a simulated user called the "jig," that provides feedback for the documents returned by a search system. The task can be visualized in figure 1. In the TREC-DD task, the system uses a seed query to begin the search, providing 5 documents to the simulated user for feedback at each iteration. This feedback, instead of being click data, comes in the form of whether or not that document was relevant, and if it was, which passages within it were relevant and how relevant they were. Using that feedback, the system should then make a decision of whether to stop the search or continue it by retrieving 5 more documents in the next iteration.

Previous submissions to the Dynamic Domain Track use methods such as MDP [6] and passage language models [1]. While most algorithms focus on the document retrieval portion of the search process and manually specify the search's stopping condition, this paper proposes the addition of search completion automation.

In this paper, we propose using the context surrounding the current search to aid the system through the use of a contextual bandit algorithm. We ran our system on the TREC-DD 2016 Ebola dataset, and compared our system's runs with baselines extracted from the TREC-DD 2016 submitted runs. We also analyzed the search policy created by our system to better understand how the system navigated through the dynamic search. Our proposed system shifts more of the search responsibility from the user to the system, leading to a smarter and more agile search system.

2 PRELIMINARIES: CONTEXTUAL BANDITS

Contextual bandit algorithms [7] are a type of reinforcement learning algorithm that uses the context given at each iteration to select an action that will maximize some sort of reward. Contextual bandits are similar to the multi-armed bandit with the augmentation of a historical context that is utilized alongside the reward from previous iterations to select an action. The context can be historical data that was collected since the start of the process, such as in Shivaswamy and Joachims [5], or use the contextual features of the

current action to influence the bandit selection policy, such as in LinUCB by Lihong et al. [2].

The LinUCB algorithm was originally designed for personal news recommendation. LinUCB uses contextual information gathered from the user and news articles to estimate a reward for each action. At each iteration, they pick the action that maximizes their estimated reward, using

$$a_t \stackrel{\text{def}}{=} \underset{a \in A_t}{\text{argmax}} \left(\mathbf{x}_{t,a}^{\top} \hat{\theta}_a + \alpha \sqrt{\mathbf{x}_{t,a}^{\top} \mathbf{A}_a{}^{-1} \mathbf{x}_{t,a}} \right) \quad (1)$$

where a_t signifies the selected action, $\mathrm{x}_{t,a}$ is the feature vector with an unknown but estimated coefficient vector $\hat{\theta}_a$ and α is a constant. $\sqrt{\mathbf{x}_{t,a}^{\top} \mathbf{A}_a{}^{-1} \mathbf{x}_{t,a}}$ is the standard deviation of the expected reward, where $\mathbf{A}_a{}^{-1}$ is the inverse of a matrix made by adding a design matrix corresponding to the previous contexts to an identity matrix the size of the context. In [2], they use click data as reward.

We adapt the LinUCB algorithm as its use of context at each iteration to select an action fits well with the multi-iterative approach of a dynamic search process. Our adaptation of LinUCB emphasizes the use of the context surrounding the search iteration, such as what the user found relevant and where previous actions have explored in the information space, to select what the appropriate next action should be. Furthermore, while [2] defines its action as the actual news article that is selected, we define actions as the tactics used to edit the query to retrieve documents, with the feedback information returned from the simulated user as the reward.

3 IMPLEMENTATION

Our framework uses a LinUCB contextual bandit algorithm[1] built on top of a search engine[2] to interact with the simulated user in order to navigate through the dynamic search process. At each iteration *t*, the bandit is provided with contextual information about the current state of the search. This feature vector is used by the bandit along with the reward information about each action to select one of four actions to play for that iteration. The relevance feedback provided in response to the documents retrieved by that action is then used to calculate a reward for the action selected. This process is repeated until the search is finished.

3.1 Bandit Actions

We design four actions in the bandit: add, remove, weight and stop. These actions use query re-formulation tactics in order to explore the search topic through its different subtopics. The actions "add," "remove," and "weight," explore the topic through a widening or narrowing focus of the search, while the action "stop" is akin to ending the search in that topic, and (if applicable) starting a new search in a different topic. The next topic can be randomly chosen.

3.1.1 Add. The action "add" provides more detail to the search by adding another word to the query in order to explore a new, more specific subtopic in the search. We use a form of tf-idf to select the word:

$$s_w = \underset{w \in D_{rel}}{\text{argmax}} \left(\frac{freq_{w,d}}{len(d)} * \ln\left(\frac{size(D_{rel})}{docfreq_w}\right) \right) \quad (2)$$

The score s_w of any given word is the maximum tf-idf score received for that word w in any document d in the set of relevant documents D_{rel} retrieved that iteration. In equation 2, the term frequency is calculated using $freq_{w,d}$, the frequency of a word in a document, divided by the document length, and inverse document frequency is calculated using the natural log of the total number of relevant documents that iteration divided by $docfreq_w$, the number of documents containing that word. The word with the highest score is considered to be the most representative word found in the documents with positive feedback, and is therefore selected to be added to the query.

In table 1, the example for "add" shows how the search is narrowed from a general search on the woman Kaci Hickox to a more specific search on her actions in particular.

Table 1: Actions and Their Associated Directions

Action	Search Direction
Add	Explore a more detailed subtopic "Kaci Hickox"→"Kaci Hickox actions"
Remove	Broaden search from subtopic "existing robots"→"robots"
Weight	Search for more documents within the same subtopic "1.0 US 1.0 Military 1.0 crisis 1.0 response" →"0.8 US 0.85 Military 1.0 crisis 0.9 response"
Stop	End search for one topic "alleged alternative Ebola treatments"

3.1.2 Remove. The action "remove" also applies tf-idf like "add" to select which word to remove. This action widens the search from the previous subtopic. Notably, while the "add" action uses the set of relevant documents, "remove" uses the set of irrelevant documents in the feedback, or D_{nonrel} instead of D_{rel}. This generates a set of least relevant words retrieved in the last iteration. We then select the word in the current query that has the highest score in the set to be removed.

The example for "remove" shown in table 1 demonstrates this as the removal of the term "existing," which widens the search from searching about the currently existing robots being used to aid health care workers to all robots both existing and merely proposed to aid in the Ebola outbreak.

3.1.3 Weight. The action "weight" modifies the weights to each query term. This can be considered a continued search on the current subtopic with a shift in its focus. We calculate the adjustment as $weight_{t-1} \pm \alpha(rel(w))$, such that it is addition if the term is relevant and subtraction if irrelevant, and where $\alpha < 1$ is a constant and $rel(w)$ is the degree to which the term w is relevant or irrelevant. We decide the degree of a term's relevance or irrelevance by creating two lists—the top 20 most frequent words in the relevant documents and that in the irrelevant documents. A term's relevance or irrelevance is proportional to its rank on either list. Using mathematical representation, a term's weight is adjusted according

[1]We used the open source python package Striatum bandit at https://github.com/ntucllab/striatum.
[2]We used the Indri 5.8 search enginehttps://www.lemurproject.org/indri/.

Table 2: The features of the search

#	context feature	explanation
1	# of documents seen	how far search is
2	# of relevant documents last iteration	current effectiveness
3	# of relevant documents seen	depth of search
4	# of times action weight called	search path direction
5	# of times action add called	search path direction
6	# of times action remove called	search path direction

to eq. 3, with α set as 0.2, meaning a maximum of 0.2 is added (if relevant) or subtracted (if irrelevant) from the weight at any one time.

$$w_t = \begin{cases} w_{t-1} + .2(\frac{20-rel(w)}{20}) & \text{if word } w \text{ is relevant} \\ w_{t-1} - .2(\frac{20-rel(w)}{20}) & \text{if word } w \text{ is irrelevant} \end{cases} \quad (3)$$

3.1.4 Stop. While most dynamic search systems being proposed set a stopping condition for the search manually, either through the number of iterations or number of relevant documents seen, we propose giving the bandit an action "stop" to decide the search's stopping condition automatically.

The action "stop" ends the search of one topic and, if applicable, starts the next one. For example, once the bandit has searched the topic "alleged alternative Ebola treatments" adequately and found enough information, the bandit will select the action "stop" and end the search on that topic. The bandit will then start the search on the next topic in the list provided by TREC, "urbanization/urbanisation."

3.2 Bandit Reward and Context Features

3.2.1 Reward. A reward is given to denote the effectiveness of an action. In a search process, the reward for an iteration can be considered the amount of information gained at that iteration. Therefore, we define the reward r_t for a selected action at iteration t as the sum of the relevance score RS for each relevant passage p in the set of all documents D retrieved at that iteration.

$$r_t = \sum_{p \in D} RS_p \quad (4)$$

The relevance score for a relevant passage is a score from 1 to 5, with 5 meaning extremely relevant and 1 being only marginally relevant. Note that the reward is never negative but it can be 0 if there were no relevant passages, which would show that the action was not effective in retrieving relevant documents for that iteration. The reward for "stop" is calculated using the set of documents retrieved by the next topic's initial search.

3.2.2 Context. The context given to the LinUCB model consists of six features, listed in table 2. These features span different aspects of the previous iteration(s) in order to provide the bandit with more information about the current state of the search and search path. Using the reward and the context, the bandit selects the action to play at each round that will maximize the reward.

4 RESULTS

Experiments were run on the Ebola dataset from TREC 2016 DD Track dataset, using the ground truth and topics provided by TREC to retrieve results and generate relevance feedback. Since a relevant document is no longer relevant the second time it is seen, we prevent duplicate documents being retrieved for any one topic by assigning a prior value of −1e−24 to a document that was retrieved once to reduce the chance that it is retrieved again.

Results were evaluated using the official TREC-DD metrics, the Cube Test (CT) and Average Cube Test (ACT) [3]. CT and ACT metrics show gain over time, with gain calculated using

$$\text{Gain}(q, d_j) = \sum_i \Gamma\theta_i rel(d_j, c_i)\mathbb{1}(\sum_{k=1}^{j-1} rel(d_k, c_i < \text{MaxHeight}) \quad (5)$$

where $rel()$ is the relevance of document d_j to subtopic c_i, θ_i denotes the importance of the subtopic, Γ is the discount factor for previous gains to that subtopic, and $\mathbb{1}$ is the indicator function. MaxHeight constrains the gain such that once an information need has been fulfilled (has reached 1), no further gain can be added. CT at each iteration representing the change in information gain that occurred through that iteration.

Table 3: Search effectiveness from Iteration 1 to Iteration 10 on TREC DD 2016 Ebola Dataset (* indicates stat. significant improvement over rmit-lm-psg-max, (p<0.05, t-test, one-sided))

Algorithm	CT@1	ACT@1	CT@10	ACT@10
LinUCB	0.278	0.208	0.098*	0.135*
rmit-lm-psg-max	0.246	0.268	0.049	0.089
LDA_Indri73	0.260	0.187	0.069	0.115
TREC Median	0.260	0.187	0.076	0.119
TREC Best	0.319	0.236	0.176	0.150

Table 4: A comparison of the scores of system with manual search completion and automated search completion (* indicates stat. significant improvement of automated system over manual system, (p<0.05, t-test, one-sided)).

Algorithm	1	3	5	7	10
Manual CT	0.277	0.139	0.090	0.068	0.049
Automated CT	0.277	0.149	0.116	0.104*	0.098*
Manual ACT	0.208	0.170	0.141	0.122	0.101
Automated ACT	0.208	0.172	0.153*	0.143*	0.135*

4.1 Search Effectiveness

We compare our results to other TREC-DD 2016 submitted runs. The TREC median and best scores are aggregated from the median and best score of all the runs at each iteration. We also compare our system with a submitted run called rmit-lm-psg-max[1] that uses a passage language model for document ranking, which represents a vastly different approach to the one presented in this paper, and a submitted run, LDA_Indri73 [6] that uses MDP, a more similar approach to our system. The results in table 3 show that our bandit achieves significantly better CT and ACT scores compared to rmit-lm-psg-max by the tenth iteration. Our system also shows scores similar to the TREC median scores and LDA_Indri73 from iteration one to iteration ten.

Table 5: Example policy created by the bandit algorithm.

Action	Results
Add "Kaci Hickox actions"→"Kaci Hickox actions CDC"	3 relevant
Add "Kaci Hickox actions CDC"→"Kaci Hickox actions CDC obtain"	5 relevant
Weight "1.0 Kaci 1.0 Hickox 1.0 actions 1.0 CDC 1.0 obtain"→"1.0 Kaci 1.0 Hickox 1.0 actions 0.91 CDC 1.0 obtain"	4 relevant
Weight "1.0 Kaci 1.0 Hickox 1.0 actions 0.91 CDC 1.0 obtain"→"0.93 Kaci 0.96 Hickox 1.0 actions 0.91 CDC 1.0 obtain"	2 relevant
Stop	
Weight "1.0 natural 1.0 immunity"→".87 natural .9 immunity"	1 relevant
Weight ".87 natural .9 immunity"→".7 natural 1.0 immunity"	no relevant
Stop	

Our comparison with TREC best, however, shows that there is still much potential for our system to improve. The basic algorithms used for our actions and our use of the contextual bandit algorithm overall provides an effective method for retrieving relevant documents and selecting when to stop but leave room for continued improvement.

As most of our actions use quite straightforward tactics for query re-formulation, it is clear there is large room to improve the search effectiveness if we employ more sophisticated query re-formulation tactics. Nonetheless, these comparisons show that using the context of the current iteration to select the next action in the search provides a good method for navigating through the search and finding relevant documents. These results also indicate that our approach is effective with even a simple set of actions.

4.2 Automated Search Completion

To analyze the effects of automating the search completion, we ran our system with a manually set stopping condition instead of the "stop" action. Table 4 shows a comparison of that run with our automated system. Both systems begin with the same effectiveness, but the faster rate of decrease in the manual system demonstrates that our system is capable of detecting a good end to the search and provides a statistically significant improvement for our system's search effectiveness.

4.3 Bandit Search Policy

The policy created by the bandit to make decisions on how to navigate through the search focused on retrieving as many relevant documents as possible.

For example, table 5 shows an example from a search on Kaci Hickox, who resisted a quarantine as she insisted she was not infected. First, the bandit selected "add", and found that had good results. Specifically, the bandit system found that exploring a subtopic on her actions against the CDC led to many relevant documents. Then, when the bandit discovered a subtopic that seemed to be very relevant (with 5 relevant documents), it chose to continue the search that subtopic with a shift in focus through the action "weight," also to good results. It finally ended the search when the effectiveness started to decrease.

During another search task, the bandit system tried using action weight to good results, turning "1.0 natural 1.0 immunity" to ".87 natural .9 immunity," with 1 relevant document. It therefore continued to choose action weight again. However, the new query ".7 natural 1.0 immunity" retrieved no relevant documents. At this point, the bandit system chose to stop the search, deciding that there had been enough relevant documents retrieved and that the search was becoming ineffective.

These examples demonstrate that the bandit understands the search path being explored and the search's current effectiveness. It is able to discover the strategies that lead to relevant results, and chooses "stop" at points when the effectiveness of the search begins to drop off.

5 CONCLUSIONS

In this paper, we proposed a solution to the dynamic search task that uses the context of the current search state in order to determine the next action in the search. We designed actions, add, remove, weight and stop, to explore the information space in order to find relevant documents quickly and exhaustively before deciding to end the search. Based on our system's interaction with the TREC-DD Ebola domain, we found that taking into account the context of the current search is relevant towards improving the search's effectiveness over time. In the future, we can continue to improve these results by applying more advanced query expansion and weight adjustment techniques to increase the bandit's search effectiveness over time. Further research can also be done on the search completion and next topic selection portion of the search, in order to build an even more advanced search system.

6 ACKNOWLEDGEMENTS

This research was supported by NSF grant IIS-145374 and DARPA FA8750-14-2-0226 and the SIGIR student travel grant. Any opinions, findings, conclusions, or recommendations expressed in this paper are of the authors, and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Ameer Albahem, Lawrence Cavedon, Damiano Spina, and Falk Scholer. 2016. RMIT @ TREC 2016 Dynamic Domain Track: Exploiting Passage Representation for Retrieval and Relevance Feedback. In *TREC '16*.
[2] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation *(WWW '10)*. 10.
[3] Jiyun Luo, Christopher Wing, Hui Yang, and Marti Hearst. 2013. The Water Filling Model and the Cube Test: Multi-dimensional Evaluation for Professional Search *(CIKM '13)*. 6.
[4] Felipe Moraes, Rodrygo L. T. Santos, and Nivio Ziviani. UFMG at the TREC 2016 Dynamic Domain track.
[5] Pannagadatta K. Shivaswamy and Thorsten Joachims. 2012. Multi-armed Bandit Problems with History. In *AISTATS '12*.
[6] Hui Yang and Ian Soboroff. 2016. TREC 2016 Dynamic Domain Overview. In *TREC '16*.
[7] Li Zhou. 2015. A Survey on Contextual Multi-armed Bandits. *CoRR* (2015).