AgentBuddy: an IR System based on Bandit Algorithms to Reduce Cognitive Load for Customer Care Agents

Hrishikesh Ganu
Mithun Ghosh
Freddy Jose
Shashi Roshan
hrishikesh_ganu@intuit.com
mithun_ghosh@intuit.com
Intuit.com

ABSTRACT

We describe a human-in-the loop system - AgentBuddy, that is helping Intuit improve the quality of search it offers to its internal Customer Care Agents (CCAs). AgentBuddy aims to reduce the cognitive effort on part of the CCAs while at the same time boosting the quality of our legacy federated search system. Under the hood, it leverages bandit algorithms to improve federated search and other ML models like LDA, Siamese networks to help CCAs zero in on high quality search results. An intuitive UI designed ground up working with the users (CCAs) is another key feature of the system. AgentBuddy has been deployed internally and initial results from User Acceptance Trials indicate a 4x lift in quality of highlights compared to the incumbent system.

CCS CONCEPTS

• **Information systems → Expert systems**; **Environment-specific retrieval**; • **Human-centered computing → User interface toolkits**; • **Computing methodologies → Reinforcement learning**; *Machine learning*; • **Computer systems organization →** *Real-time system architecture*;

KEYWORDS

Bandit algorithms, LDA, user experience, human-in-the-loop

ACM Reference Format:
Hrishikesh Ganu, Mithun Ghosh, Freddy Jose, and Shashi Roshan. 2019. AgentBuddy: an IR System based on Bandit Algorithms to Reduce Cognitive Load for Customer Care Agents. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331408

1 INTRODUCTION

We are a Machine Learning team which works closely with Customer Care Agents (CCAs) from the Customer Support team. CCAs interact with end customers through web chat. In their effort to address customer issues they perform search over our internal Knowledge Base (KB) of curated articles through an in-house federated search engine. The current system has two issues which affect the speed and quality of search:

- The median length of curated articles is 4 paragraphs and hence CCAs spend a lot of time reading through the retrieved articles before they can figure out whether the article is relevant to the customer's issue. The CCA leader had the following ask "*Can you automatically highlight the key sentences in each article, looking at the question? This will save us a lot of time.*"
- Our in-house federated search is a legacy system backed by a large set of "Collections" of articles. Each collection has its own API and all APIs don't serve the same type of answers. These APIs have evolved over time and hence there isn't a sharp mapping of APIs to answer types. The legacy system however relies on hand-crafted rules which mostly focus on the CCA attributes rather than the attributes of the question to determine which collections to search over for federated search. Ignoring the question content leads to poorly tuned federated search resulting in low relevance or high retrieval latency. Further, the individual search APIs are owned and maintained by disparate teams so condensing them into a single API is not easy from an engineering perspective.

Intuit has built up a brand reputation for customer friendly products and hence customer experience is the highest priority for Intuit. Given the current abilities of ML systems it is not possible to completely revamp the process and replace agents with automated ML driven systems without taking a hit on the customer experience. Hence we have created a human-in-the-loop system which does not change the workflow of CCAs completely but assists them in performing their tasks with greater efficiency. We term it a human-in-the-loop system because the ML system only offers recommendations - the final call on whether to accept the recommendation lies with the agents. This system which is meant to be used internally by CCAs has been deployed and is under User Acceptance Trials currently.

2 RELATED WORK

The contextual bandit framework has become well known in recent years . In this short paper we cannot cover the entire body of literature related to this problem. However, Bietti et. al. [1] provide

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331408

a summary of the most popular algorithms used in this formulation. In terms of applying contextual bandits to search problems Jie et. al. [3] have done some early work. The present work is similar in principle to [3] but with the following key differences:

- Since we receive feedback from internal customer care agents, we are able to capture much richer implicit feedback and are able to track the follow through actions till the agent "closes" the case with the customer. See Section 3.1 for details. This situation is different from that in [3] since they learn from external customer interactions.
- The scope of our work is different. We have built an entire Decision Support System (DSS) for customer care agents which includes other ML components like the highlighting model etc. The bandit algorithm is just one of the several sub-modules we employ to build this system.

3 USER INTERACTION

For the AgentBuddy system, the user is a Customer Care Agent (CCA) who answers end customers' questions, and the end customers are the users of our accounting software. When a CCA submits a question to the AgentBuddy system, search results are shown to the CCA. Each search result has a relevance score associated with it, which is used to sort them. The user interface provides an option to zoom in on a search result, as well as to navigate through all the search results. Each search result consists of a title and a body text.

For each search result, the AgentBuddy system highlights the part which is relevant to the search query. The highlights are created using Machine learning models as described in Section 4.1.2. If a result is too long to read, the CCA can choose to read its highlighted portion to get the important information. The CCA also has an option to copy and send the highlighted part of a search result directly to the end customer, if the highlighted part is an optimal answer. Else we proviide a option for the CCA to compose the answer in a rich text box using a combination of highlighted content, non-highlighted content and free form text. The screenshot of AgentBuddy interface and features is shown in Figure 3. If a CCA wants to save a manually created answer so that it can be used later for answering similar questions, this can be done by using the "Persist Answer" option.

3.1 User Feedback and Bandit Learning

We derive the user feedback implicitly from the actions performed by the CCA. The feedback received from the CCAs is used to train a contextual bandit. When a CCA sends a search result directly to the end customer without modifying it, this corresponds to the best feedback that the system can receive and we note a high reward. Alternatively, when a CCA copies only a part of a search result and sends it to the end customer (maybe after modifying it), it corresponds to a medium reward. Finally, if a CCA chooses not to use anything from the search results, this implies that the search results were not useful. This corresponds to a very low reward. This implicit feedback is used to train the bandit algorithm as explained in Section 4.1.1.

4 SYSTEM COMPONENTS AND TECHNICAL DETAILS

AgentBuddy is a system which has several components. It consists of a user interface (frontend) which takes the question as an input and passes it to the bandit algorithm. The bandit algorithm determines which search collection is to be used and calls the federated search system for articles. Once the articles are retrieved they are passed on to the highlighting module. This module decides which portions of the article should be highlighted in the context of the question. The query, articles and the highlights are finally sent to the frontend to be rendered. Each of the individual components : frontend, Bandit algorithm, Highlighter are separate web applications running with the popular flask-gunicorn combination. In AgentBuddy, we use a mix of AWS EC2 and AWS Lambda. We have used AWS EC2 for compute intensive tasks like highlighting relevant portions of search results, whereas we have used AWS Lambda for input/output bound tasks, like fetching search results from various sources. AWS Lambda also provides finer auto scaling controls as compared to EC2.

4.1 ML Frameworks

As explained in Section 1 , we had to solve for two major issues which was affecting the speed and quality of the search.

4.1.1 Choosing the right search collections: As discussed in Section 1, one of the key decisions is to determine which collections to search over during federated search. The problem can be formulated as a contextual bandit problem. Each combination of search collections becomes an arm of the contextual bandit and given a query, there is a reward distribution over combinations of search collections. This formulation allows AgentBuddy to treat the search collections as black box. We have used contextual bandit algorithms described in [1]. Associated with each arm of the bandit is a reward feedback from the CCAs. We have three levels of reward: high, medium and low as described in Section 3.1 which are mapped to scalar numbers in consultation with CCAs.

4.1.2 Highlight the most relevant parts of search results using Latent Dirichlet Allocation. Once a customer care agent is presented with the relevant search results for a search query, the next task is to extract information from the search results which is relevant to the query. This problem nicely fits into text highlighting problem or context aware text summarization problem. After trying out many different approaches, we have built Latent Dirichlet Allocation (LDA) based model [2] for highlighting the answer text based on given query. We have built a model which is trained on a large number of Customer Support articles and User Manuals (of our accounting software) to get all the different topics available in relevant discourses. This is a one time training of our LDA based Machine Learning model. In terms of best fit we had 25 topics which has given us best coherence score. Once we receive a customer query, we quickly map that query to the right intent or latent topic based on the ML model. We then accordingly pick up the right set of sentences in the search results which is relevant with the search query in terms of the topic or associated word distributions.

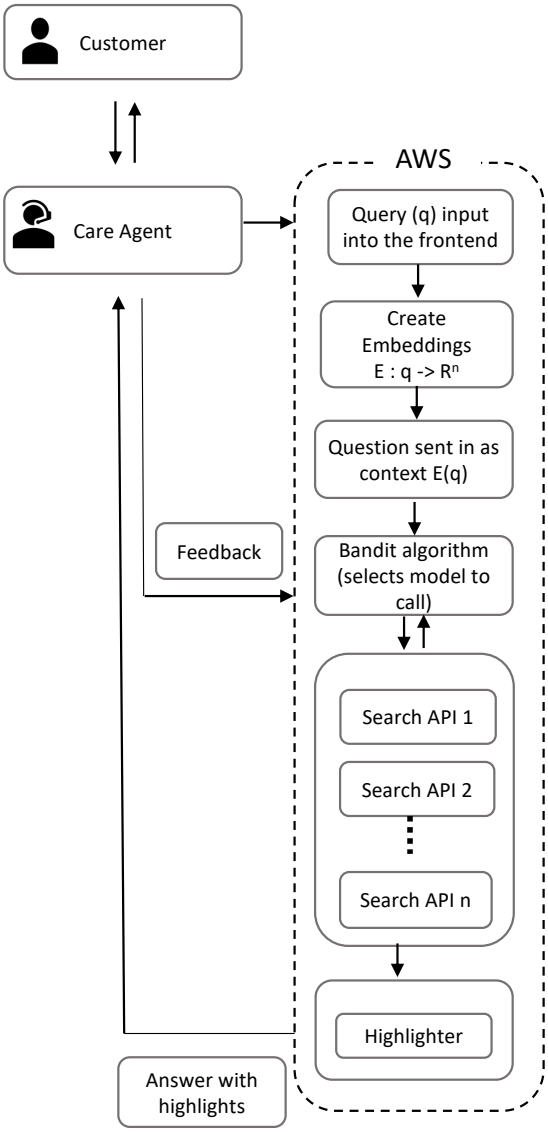

Figure 1: AgentBuddy Workflow

4.1.3 Highlight the most relevant parts of search results using a relevance predictor . An alternate approach was to use a binary classifier to determine whether a sentence within an answer is relevant to the query and then highlight it if it's relevant. We used an implementation of a Siamese network type architecture provided by Amazon SageMaker as a binary classifier.[1]. We used a proprietary internal dataset of <question, answer> pairs where the answers are especially short to create training data. We call this the "short answers" dataset. In this case because the answers are short and crisp, each sentence in the answer can be assumed to be relevant to the question. E.g. in our training dataset, the relevance label for

[1]https://aws.amazon.com/blogs/machine-learning

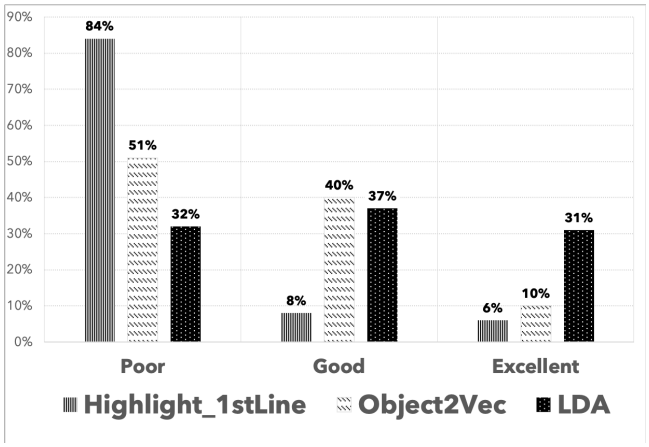

Figure 2: Comparison of the LDA and Object2Vec model with the incumbent highlighting logic on the "quality of highlights" metric. The quality was assessed by highly skilled CCAs over a dataset of 126 question-article-highlight tuples.

a matching <question,answer> pair like ("How to get GST input credit", "Reduce your GST payment by getting the right invoice from your vendor") would be "Yes". The training process is fairly standard and we minimize the cross-entropy loss. After several rounds of tuning we finalized a model with a test AUC of 0.82.

5 RESULTS & DISCUSSION

To evaluate the quality of the highlight, it is important to have a relevant search result for a given search query. This is because if the retrieved search result itself is irrelevant to the query, any highlighting inside the article is also irrelevant. To evaluate this in a controlled setup, we selected a dataset of 126 queries, each paired up with one article. The pairs were such that the article in each pair was "relevant" to the search query as assessed by CCAs. Then we created highlights inside the articles in each pair using ML models and asked the CCAs to tag each highlight into: *Excellent, Good or Poor*. Note that before deployment of our work, CCAs used a legacy system which always chose the first line of the article as a highlight. We call this the incumbent system. The system using LDA highlighting (Section 4.1.2) has 4X as many highlights in the "Good" or "Excellent" category as compared to the incumbent and this metric is 1.5X when comparing the LDA based system with that based on the binary classifier (Section 4.1.3). While this result might seem counter-intuitive at first, one of the reasons why the supervised binary classifier does not perform as well as the LDA model could be that the binary classifier was trained on the "short answers" dataset (See Section 4.1.3). While this dataset is also about accounting, it is from a geography other than India.

6 CONCLUSION

We have described a human-in-the loop system to help Customer Care Agents become more efficient. The system provides adaptive (question-specific) highlights inside articles. We experimented with multiple ML formulations for highlighting to ensure that the business team gets a model that fits their use case. We have also helped

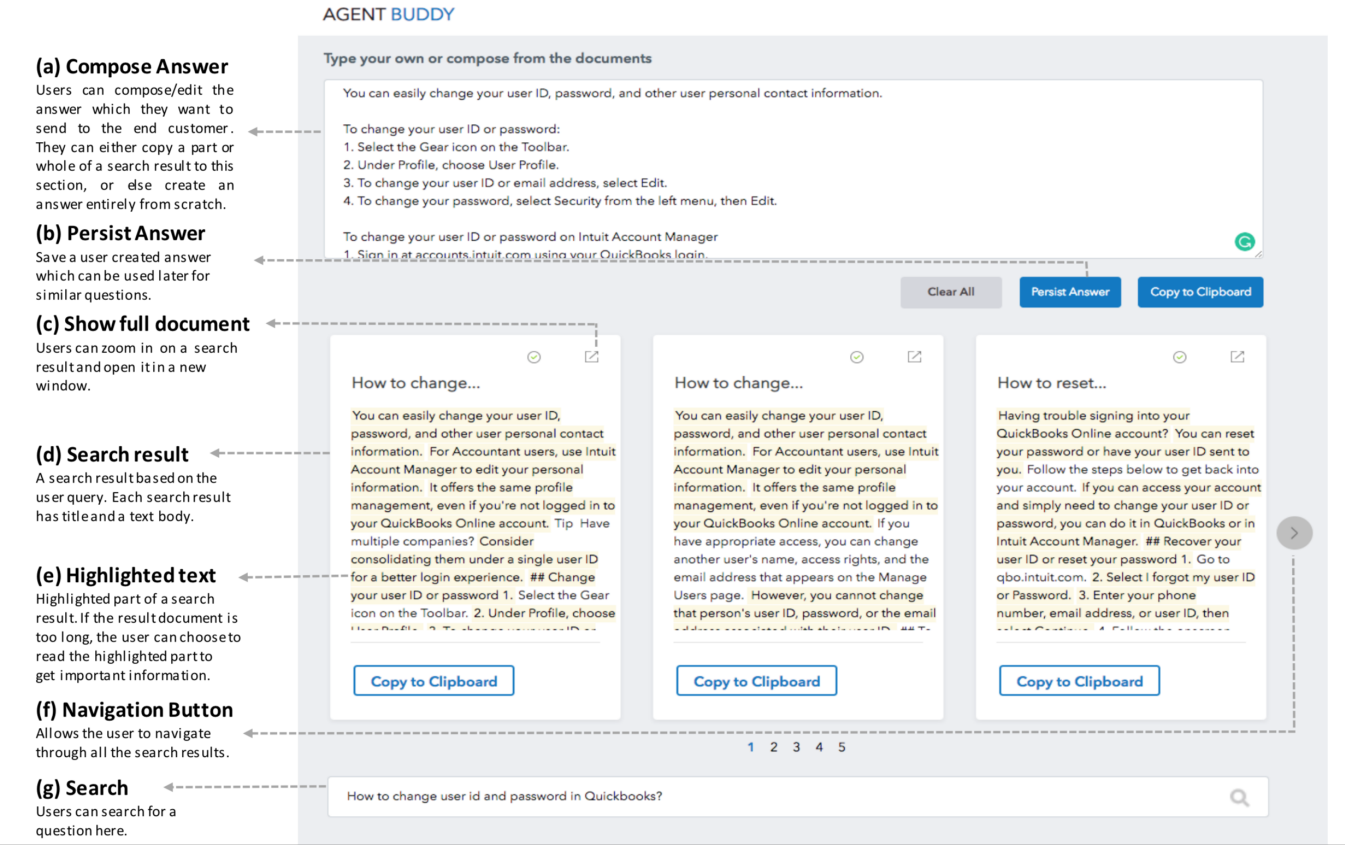

Figure 3: AgentBuddy User Interface

improve the quality of federated search by using a bandit model to determine sets of collections to search over, using feedback from CCAs as a reward signal. We are currently compiling data on performance of the bandit algorithm so that we can report it in the future.

ACKNOWLEDGMENTS

We thank Aminish Sharma and Padmakumar Nair for their sponsorship of this work. John Varghese provided all the human annotations for validation of the system. Thanks are also due to our partner team of engineers and UX experts.

REFERENCES

[1] Alberto Bietti, Alekh Agarwal, and John Langford. 2018. Practical Evaluation and Optimization of Contextual Bandit Algorithms. *CoRR* abs/1802.04064 (2018). arXiv:1802.04064 http://arxiv.org/abs/1802.04064

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022. http://www.jmlr.org/papers/v3/blei03a.html

[3] Luo Jie, Sudarshan Lamkhede, Rochit Sapra, Evans Hsu, Helen Song, and Yi Chang. 2013. A unified search federation system based on online user feedback. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy (Eds.). ACM, 1195–1203. https://doi.org/10.1145/2487575.2488198

A ONLINE RESOURCES

Since AgentBuddy is an internal application to be used by CCAs, we cannot expose a demo url to the public. Instead, we have uploaded a video to the following url to help readers understand the capabilities of the application: https://archive.org/details/Demo1_201902