Search Agent Model: A Conceptual Framework for Search by Algorithms and Agent Systems

Jeffrey Dalton
University of Glasgow
jeff.dalton@glasgow.ac.uk

John Foley*
Smith College
jjfoley@smith.edu

ABSTRACT

We describe challenges using search systems designed for algorithms and agent systems rather than humans. As information access systems become more complex the users of retrieval systems are increasingly shifting from humans to agents that use search as a sensor for acquiring and interpreting knowledge of the world. First, we discuss work on prior search applications that fit into this agent model of search. We identify key challenges for current and future search systems including: confidence estimation, task state, and expressing complex long-term retrieval models. We propose a conceptual framework for understanding and creating search-focused agents that addresses these challenges, the Search Agent Model (SAM). SAM provides a shared model for complex search tasks requiring a variety of information processing and orchestration. Its main components include formalizing task state, an action policy, and a query language for agent interaction. We describe how this proposed agent model provides a roadmap for future research and system design in search.

1 INTRODUCTION

Traditionally, information retrieval systems are user-facing: users enter queries into the system and interact with a ranked list of documents. User-facing search systems are used by billions of people worldwide and are critical to how humans interact with information. However, humans are no longer the only users of such systems, and algorithmic and agent system users (AAS) are increasing in number. As more advanced and intelligent agent systems are designed, they will play a larger role in shaping the use and structure of information retrieval systems.

Already, many users choose to interact with personal virtual assistants (PVAs) such as: the Google Assistant, Amazon Alexa, Microsoft Cortana, Apple Siri, Samsung Bixby, and many others. These systems remove the traditional search interface and present a new voice interface, but more importantly, they focus on agent (algorithmic) processing of search result information in order to help the user perform actions like making a hotel or restaurant reservation. Because voice is a poor medium for transmitting document results, these systems must perform more algorithmic processing of the results. As agents like PVAs become more intelligent and helpful in their use of search systems, users may increasingly prefer interacting with such AAS users instead of traditional stand-alone search systems.

One interesting property of emerging PVAs is that they often consist of a collection of many agents for a diverse range of information tasks. The internal meta-assistant framework delegates handling of utterances to hundreds or thousands of possible bots (agents) to process the interaction. Current PVA platforms are experiencing an explosion of agents for every task imaginable: Alexa has over 30,000 skills in the United States. These agent systems are unlike traditional search engines in that while the agents may perform searches on behalf of users, their main function is to perform actions, such as making travel reservations and purchasing items. Internally, a single utterance may result in many searches inside of a variety of agents to diverse backend search systems - web search, personal document search, knowledge graphs, structured databases (products, movies, music, etc.), booking services, etc. The result is that algorithms orchestrate information processing and perform reasoning over heterogeneous information object results in increasingly complex ways that were previously done by humans. We propose that these algorithmic users will be important consumers of search in the future given the growth and increasing sophistication of PVAs and other sophisticated information processing systems.

One of the most successful aspects of the IR community is the availability of mature open-source search engines that implement state-of-the-art search algorithms and models as basic building blocks. However, the number of open-source systems for complex information tasks is quite limited. Beyond ad-hoc search, the IR community is actively engaged in research on increasingly complex information tasks, e.g., factoid question answering, synthesizing a Wikipedia article (TREC Complex Answer Retrieval track [9]), entity-centric information extraction and retrieval, and evaluating the trustworthiness and bias of results, etc. These advanced applications use search (possibly multiple times) throughout a complex algorithmic information process. While existing techniques for addressing complex information needs exist and models and demonstration systems are available for some tasks, the systems that perform these tasks are often large, complex, challenging to build, and lack a shared architecture. Importantly, there is little support from search frameworks to assist building these systems, resulting in unnecessary complexity and inefficiencies in the design and reproducibility of models and systems.

In this work we propose a departure from current search engines and propose a new model to enable more complex information tasks. The Search Agent Model (SAM) is a conceptual model for understanding and creating a vision for the future of search-focused agents. SAM can encode straightforward ad-hoc retrieval and relevance feedback tasks, but more importantly provides a shared

*Work done while at the University of Massachusetts Amherst

DESIRES 2018, August 28-31, 2018, Bertinoro, Italy. Copyright held by the author(s).

DESIRES 2018, August 2018, Bertinoro, Italy

https://www.wired.com/story/inside-amazon-alexa-prize/
https://www.voicebot.ai/2018/03/22/amazon-alexa-skill-count-surpasses-30000-u-s/

model for more complex tasks that require a variety of information processing and orchestration. We see SAM supporting tasks such as intent classification, document summarization, learning to rank, managing task state, inference and reasoning across information objects, structured planning and constraint modeling, and others. Today, rudimentary 'information agents' are created manually - with researchers hard-coding architectures and policies that address very specific tasks in limited domains. However, we need conceptual frameworks like SAM to help us develop more general strategies for implementing intelligent AAS-based systems.

The rest of this paper is structured as follows: we first identify the shared challenges that emerge from our AAS applications, as well as related work for each of these challenge areas. Next, we discuss in more detail our motivations for proposing the Search Agent Model through a number of case studies. We formally introduce SAM and describe how its different main components - task state, policy, and query language - address the challenges identified. SAM is our vision for future work: and our challenge areas will require innovations and new models as we move into an era of agent-users of search systems.

2 KEY CHALLENGES FOR AGENT SYSTEMS

AAS users of search mean that architectures need to change in fundamental ways. The challenges we highlight here are inspired by the broader set of key challenges identified at the Strategic Workshop on IR (SWIRL) 2018 [4]. In particular: stateful search in conversational search systems, system confidence for knowledge driven decision making, and flexible query and response APIs using Generated Information Objects. We now describe each of the challenges and how they relate to AAS users and our SAM vision in more detail.

2.1 Confidence in results

Algorithms and agent systems need more effective models for the confidence of the results - a meaningful confidence (or probability) of relevance that is stable within a ranking, across queries, and across heterogeneous collections of information objects. Most user-facing search systems do not present scores to users. This is not sufficient for an AAS. An AAS must have a meaningful confidence of the relevance of the search result. Ideally, instead of returning a top-k fixed list, search systems should return arbitrary-sized lists of documents up to a meaningful confidence level, depending on the requirements of the AAS.

Further, beyond just a meaningful confidence, the scores or probabilities for the objects returned need to be useful as part of larger (agent) systems. Today, it is often difficult to incorporate scores or rankings into further machine learning steps because scores on documents often do not strongly relate to confidence of relevancy. As an example, in most implementations of the query-likelihood language modeling framework, we ignore the fact that the $P(Q)$, and the query length prevent comparisons of scores between queries, and that collection statistics prevent comparisons across retrieval collections.

There are a number of previous approaches to acquiring confidence estimates for search results, ranging from the simple, e.g., max-min score normalization of scores or using a function of the rank instead of a score to more sophisticated query performance predictors [3]. Another strategy for estimating confidence or performance relies on user signals, which are obviously not usable when the direct users of our search systems are algorithmic in nature. The IR community developed rich user interaction models for humans interacting with search results. There are sophisticated models of user interactions with search engines, that study many diverse properties including attention, page dwell time, and click behavior [19]. It also developed simulated models for search and when users stop searching [17]. These can be used to tune the relevance of the search system and more effectively estimate relevance.

We need new mechanisms for studying and developing AAS user models that support explicit agent feedback on the utility of results. We describe some ideas for these in §4. Early work on comparing (simulated agents) with humans is just beginning [15]. We do not yet have concrete ways of defining query strategies, stopping strategies, or decision making strategies for agents. And for metrics, we should look beyond tradition measures of document relevance (MAP, ERR, etc..) to more fine-grained measures (larger scales) that measure the utility of the information provided in algorithm task context (e.g. used in a summary that resulted in a purchase with monetary value). This is required for more fine-grained tuning of model parameters and learning effective search agent policies.

2.2 Task state and information reasoning

Current search systems, including the main open source research engines (Galago, Lucene, Terrier) are largely stateless. Queries are usually treated independently with little or no shared context across them. Although there exists plenty of study of human search sessions, e.g., work done in the TREC Session Track [14], AAS users will have different and more explicit sessions in comparison to human users. In contrast, one of the fundamental differences for search with AAS users is that given the right APIs they can provide fine-grained updates on processing ('reading') of information as well as having an explicit machine-readable representation of current world knowledge and ongoing "task state". This state is needed to provide a working memory for the information agent as events happen and the task evolves.

Beyond simply storing stateful interaction history and feedback information, AAS users perform complex information summarization and reasoning task that evolve the state. The resulting state will be large and complex. Search systems need to be aware of past queries and candidate pools of results. Further the AAS user will take actions to update the state to summarize information, perform proactive searches, rerank results, and update possible structured plans. Current systems lack the richness to describe and utilize this kind of state without significant and highly specialized engineering effort.

Reasoning about the task at large is going to be the chief responsibility of the AAS. One of the key challenges here will be the "joining" of information from separate verticals and separate queries. While maintaining a large task state and reasoning about it is motivating for efficiency research, it also suggests that we need more sophisticated search APIs and models that can use the state context to affect results.

2.3 First-class Learning in Agents

With current approaches to agent design, integration of machine learning requires a lot of manual work to extract training data, train models, load and execute them at appropriate times. While we have research approaches to online learning and annotation, actually integrating these ideas into runtime systems is quite difficult because our systems are optimized for the Cranfield paradigm: batch submission of TREC-style queries.

Recent advances in deep learning libraries show one possible path forward: by using dynamic computation graphs when possible, a system can reason about the "learnable" pieces of the graph and how they interact with the full system. We partially address this vision with an evolution of the INQUERY/Indri and Galago query language we call AgentQL, but more research and exploration is needed to evolve search APIs for current and future AAS users.

3 CASE STUDIES: ALGORITHM AND AGENT SYSTEM USERS

We now highlight several algorithmic information tasks that illustrate the challenges using existing search systems for complex information tasks. In particular, given the authors' focus on research applications, we provide "war stories" of using the Galago search system for building this style of systems. For each application we provide an overview of the application, the challenges involved in using current search systems, and how SAM would simplify it.

3.1 Travel Agent System

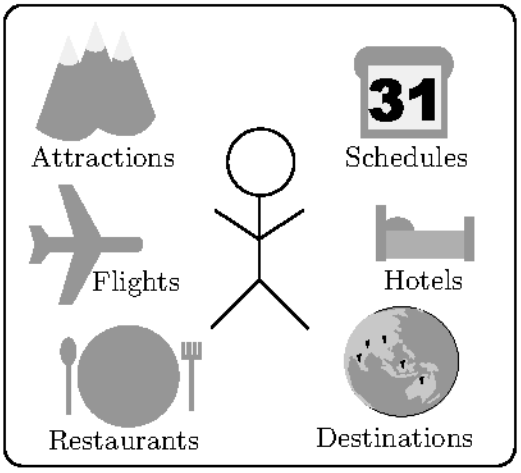

Figure 1: Travel Planning Example: To successfully plan travel without an agent, a user must coordinate search results and constraints across many dimensions.

As a motivating example, consider a virtual travel assistant agent system with the task of planning a trip. The information handled by this system is an evolving plan of destinations and travel plans with transportation, accommodation, restaurants, and attractions. This agent will take actions on behalf of the traveler, e.g., to recommend restaurants and reserve tables, reserve museum tickets, and other tasks. The agent will provide access to trip information (QA) as well as proactively assisting the throughout the trip - before the trip with with planning and bookings, with real-time status updates (and prompts) during the trip, and finally follow-up afterwards (sharing pictures, reviews, expenses, etc.).

The information travel planning task involves processing both structured and unstructured data - performing reasoning, summarizing information, resolving constraints, and finally creating a structured output. As shown in Figure 1, it requires contextual information and search in a dynamic information space across multiple information domains and object types (e.g. restaurants, museums, hotels, flights, etc.). For example, queries to backend search systems would use shared geographical and personalization context for the planning task. The context would also incorporate diverse sets of preferences and constraints: budget constraints, previous preferences for types of attractions, food allergies (e.g. vegetarian or gluten-free), the purpose of the trip (business or holiday), group constraints (e.g. family or friends), and others. Further, as demonstrated in previous work on task modeling [1, 10, 13] - the task and subtasks are long-running and may be active for days, weeks, or months as details of a users' primary destination evolve and become fixed. The plan may also need to evolve rapidly due to changes in the environment, such as weather or delays in travel.

Today, Galago and other existing search systems do not explicitly support these tasks. As a result, systems that tackle these kinds of problems are created outside of search with significant effort. Increasingly the key building blocks and subsystems exist and can be combined algorithmically. For example there are APIs for booking flights or hotels (the agent system needs to be grounded to use them), subjective review information on review websites via Yelp or TripAdvisor (the system needs to process semi-structured and unstructured text and summarize them). Other challenges include the collating and joining the information across sources (coreference resolution and information integration [12]), and finally creating actionable structured plans (constraint satisfaction and structured prediction).

The proposed SAM will enable this new application along multiple dimensions. First, it will have a fundamental mechanism for creating and storing long-term information state. The proposed state model will support the diverse array of information objects that mixes structured data and unstructured data – with history and provenance for how they have been summarized, extracted, and joined. More importantly, it provides an agent layer that performs actions on information to transform it - object extraction, cross-object information integration and coreference resolution, information summarization, constraint and preference modeling, and structured result composition in ways that can be more flexibly performed and more importantly be optimized in conjunction with the search system.

3.2 Entity-Aware Retrieval Models

Modern retrieval models could be considered agent users of search, as they tend to combine evidence from multiple sources and queries in order to improve results.

http://lemurproject.org/galago.php

In prior work, we presented an entity-based query feature expansion (EQFE) model to improve ad-hoc retrieval effectiveness [7]. Today, many of the challenges remain unsolved and an area of active research, with the recent TREC Complex Answer Retrieval (CAR) track generating synthetic Wikipedia-like summary pages [9, 18]. In all of these cases, effective models require multiple stages of retrieval across heterogeneous collections and joint reasoning about the relevance of entities and documents in relation to an information need. Currently, systems are difficult to share and reuse because they consist of many layers of handcrafted code for coordinating these steps, and learning (or using learned models) must be carefully interwoven by hand.

Building EQFE-like models was (and remains) challenging with existing open source search engines. Our current tools assume a homogeneous set of objects in the index for the purposes of normalization, calculation of statistics, and presentation of results. In the original implementation of EQFE, there were separate document and KB indices that needed to be searched separately and joined to preserve statistics. Further, after top documents are retrieved the next step is to extract passages and entities and to create new entity information representations from retrieved results.

However, leveraging dynamic objects that are never indexed means that the search engine almost immediately stops being the correct code tool. To the best of our knowledge, no open-source search engines are capable of scoring documents that are not indexed. This means that collection statistics need to be computed on the fly and we must re-implement basic scoring models on our custom datasets because (typically) ranking models are intertwined with search index implementations – one cannot score BM25 without having a posting list. The result is brittle and "hacky" code, despite this re-ranking approach being at the forefront of IR research. Any student of IR knows how easy it is to implement a nearly correct ranking model that misses edge cases or subtly affects ranking performance.

EQFE builds upon the foundation of strong entity recognition and entity linking systems. We found that while search can be helpful for improving these tasks, to do so requires an agent to make decisions and incorporate results, leading us to struggle with confidence and query-specific relevance estimations [5] for entity recognition across sentences. We also explored improving entity linking using PRF [6] and query-time entity linking [11] – but in these applications we struggled to normalize query scores across different collections and query formulations.

We must address this challenge of unifying query and re-ranking systems for the purposes of improved reproducibility and confidence in our results, but also for the ease of future research into novel retrieval models and the agent systems we focus on in this work. Future research will look at dynamically generated data objects that have relationships across multiple databases and corpora. Being able to confidently express baseline models and compose them into novel inferences will be critical to the success of research in the future.

4 SEARCH AGENT MODEL

We propose a new layer above the search engine, the Search Agent Model. It provides APIs for interacting with search for AAS users, which may themselves be agents. It provides capabilities for support complex information needs and addresses the challenges in the previous sections. SAM is a new proposed framework with prototypes under development for the future.

4.1 Agent Task State

As described in challenges, a fundamental change for AAS users in SAM is the importance of short-term and long-term state. For this purpose, we propose building on the recently proposed Generated Information Objects (GIOs) [4] as the representation of information state that can be explicitly modeled, persisted, and used across many queries and tasks.

GIOs are generalizations of the document objects typically returned from search systems. While usually the elements in a ranked list refer directly to documents that have been indexed by a search system, more complicated search tasks have always needed to represent portions of documents and to transform the representation shown to the user (e.g., entity, XML and expert search, document expansion, NLP tagging, multimedia-enrichment or multi-document summarization). These GIOs may be indexed directly or constructed on the fly and stored only as long as they are relevant to the user – anywhere from milliseconds (PRF summary documents) to years (related work for a thesis).

In a conversational system, tracking the utterances of the user becomes important. With natural language understanding processing including: intent identification, entity recognition and linking, and semantic parsing, the unstructured utterances are transformed to become true GIOs. Similarly, an agent may summarize a sequence of clarification questions and documents into a GIO that represents the evolution of the information need. This will be needed to represent any long running task to a user who may wish to resume, e.g. planning their trip or researching a particular topic. This summarization of multiple GIOs or a sequence of GIOs demonstrates a key challenge of such a knowledge-rich system: merging, de-duplication and evolution of GIOs.

Over time, the user and the agent will interact, whether it is through the collection of true relevance labels or the submission of queries or questions. Additionally, actions made by the agent on behalf of the user will accumulate: reservations, summaries, tickets, etc. These GIOs are also an important part of task state that the agent must understand and maintain in order to be successful.

Although the idea of GIOs are still abstract, this is because they are meant to be flexible, and the goal of future experimental search systems should be to provide for the flexibility needed in dynamic objects while still easing reproducibility and system design. This flexibility is the key to robust management of the task state needed for intelligent agent systems.

4.2 Agent User Policy

The core of an agent-based search system is the agent's user policy. While in some cases it will make sense to hard-code behavior in user-defined or developer-defined policies, we expect that the ultimate future of such engines will involve learned policies, i.e., using reinforcement learning: where the next action a_{t+1} is chosen based from a set of actions A based on maximizing the expected reward $r(a)$.

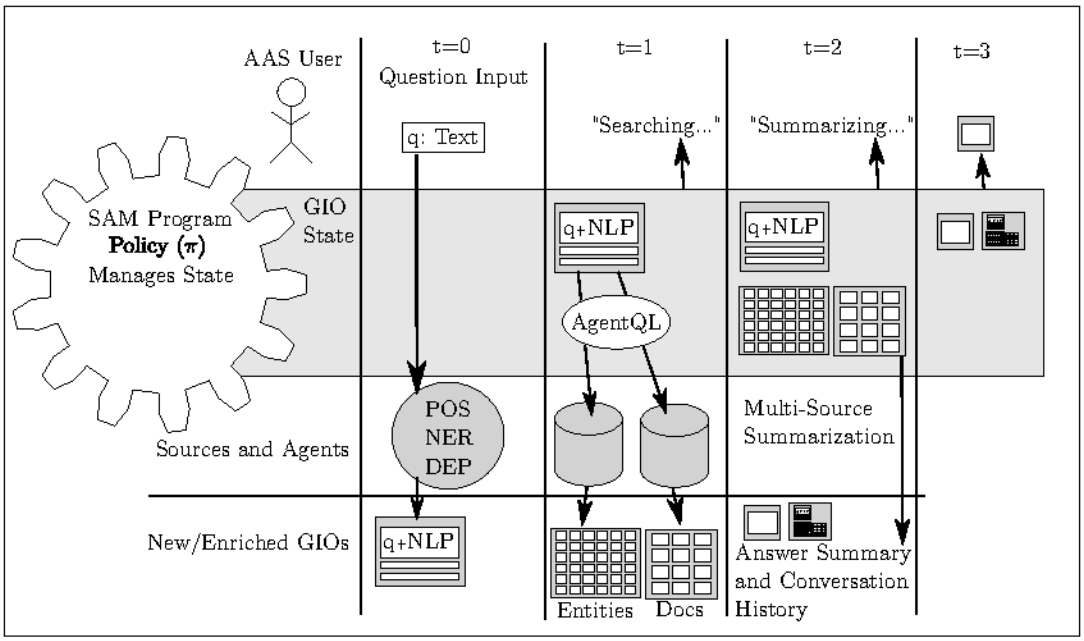

Figure 2: SAM Conversation Example: At each timestep, the SAM program takes input, manipulates GIO state, and optionally presents progress or data to the user.

$$a_{t+1} = \arg\max_{a \in A} E(r(a))$$

An agent will have to decide between many possible actions, which will be task-specific. An agent can choose between searches to perform, what analysis to compute, what questions to ask a user, whether to explore more results from a given query sub-task or to reformulate that query sub-task. While additional actions will need to be defined that are task specific (i.e., booking a hotel, calling a taxi, renting a movie), our analysis focuses on actions and decisions around the agent's information and knowledge, as these policies will be shareable and reusable across tasks. Even domain-specific actions can share a lot of logic and reasoning about how certain the agent is about its current knowledge (e.g., all purchase actions should have been confirmed before being acted upon). A key research challenge here will be how to share effort between different kinds of search agents, and how to learn policies over a large variety of actions.

Models of user simulation in IR have formalized the way that humans interact with rankings [16], but in this section we identify that although we expect some similarities to human usage of search systems, this will really be set by the Agent's user policy. Trying to learn importance or relevance of documents through, e.g., an agent's click model will simply result in slowly reverse-engineering something the agent already knows – it's policy. Therefore, we must have new solutions for providing feedback and tagging actions as user-inspired or speculative.

Agent Feedback & Integrated Learning: In addition to learning how to estimate the expected reward for potential actions, an agent potentially has many sub-tasks that involve machine learning models. An Agent should gather feedback on whether results are processed (viewed), whether the result appears to be relevant (click), and the final utility of the result (judgment). Because we have an agent and not a user, it is possible that these types of rewards are not directly related to direct user actions, but based on the resulting utility of the information. These could be indirectly propagated based on some non-traditional type of user feedback, e.g., hesitation, frustration or other more novel user-signals.

As a basic example, we expect that feedback can be used for updating search model parameters. An agent may make decisions based on the output of many models and will need to know which models feedback can improve. An agent may obtain these labels directly from an end user (through their interactions), but also from interactions with other algorithms and agent systems. Machine learning should be a first-class part of a future SAM system, where weights can be learned, updated and stored at any step, much like many libraries now provide for auto-differentiation of computations.

Query Generation for Confidence: An Agent program could generate multiple queries for the same task or sub-task, and aggregate confidence indicators across different formulations derived from the same information need. In this situation, results that were not semantically coherent with results found by alternates could be considered to have less confidence.

Constraint Resolution: An Agent policy is likely to have more information about the objects being retrieved, i.e., while a search system would potentially have the GPS coordinates of museums while retrieving a ranking, and a users' desired destinations could be incorporated into a ranking function, a program might choose to resolve constraints, such as being close to affordable hotels in its own working space, which would allow for the Agent to have

sufficient information to propose that a user relax or clarify some of their task constraints.

By performing constraint resolution, an Agent may need to request more results from a particular query (e.g., the top-k relevant hotels are too far away from a museum the user just decided to visit and should be re-ranked lower, so we should search for more hotels in Florence). Constraint resolution therefore needs to be integrated into the policy and part of the decision-making process.

4.3 Agent Query Language (AgentQL)

We propose a high-level language (or library) AgentQL that supports the construction of graph-based representations of an agent's information needs. Because an Agent will submit many parallel queries, and will be more likely to exercise weighting options than a human user, we need a richer, more expressive query language for an agent's needs.

4.3.1 Complex and Nested Queries. Any agent query language must be able to express complex queries. Being able to specify weights on queries and subqueries to multiple fields and to tune importances requires the complex nesting of queries. Already, open source retrieval systems have the ability to accept such complicated, deeply nested queries and transform them into more efficient, flattened structures if possible [2]. Being able to compose queries effectively means that larger queries can be built, reformulated, and then put together into meaningful joint models. While this is mostly provided for in the Indri and Galago query languages, these languages are meant to be used by humans and make certain assumptions that do not make sense with an agent user. For example, in Indri and Galago the parser assumes typical English text with operators delimited by '#' symbols. Any system that aims to use unstemmed, binary data, or even non-traditional text (or just hashtags) will find the textual interface challenging.

4.3.2 Query Processing in Context. Existing models require significant amounts of pre-processing that are performed outside of the retrieval engine. AgentQL makes these explicit with support specifying components of processing. This includes query weighting options, query rewriting mechanisms (correction, expansion, etc...) and parameters, and different algorithms for transforming the query in the context of the current task state. For each of these, the language natively supports learned components. Unlike users, we expect AAS users to take closer algorithmic control over the results.

4.3.3 Static Analysis of Confidence. Modeling the whole system in a single language opens up opportunities for understanding the confidence of results, which is key to allowing our agent policies to make intelligent decisions in the face of uncertainty. By being able to analyze the structure of ranking models, we can acquire analytic bounds for the maxima and minima of all sub-expressions involved in scoring of documents (and empirical bounds can be discovered through exact or inexact sampling procedures). Confidence in general is an area that is ripe for future research and having a shared query language for future retrieval models can provide benefits to the research community.

4.3.4 Multiple-Round and Source Query Execution. AgentQL should support models that involve multiple rounds of retrieval (e.g., RM3) or the calculation of weights from statistics. While SQL in databases supports nested queries, we are unaware of any retrieval systems that support such queries at this time: Galago's implementation of RM3 issues a query as a specialized pre-processing step, which limits the ability to compose this model with others or to use different RM parameters for each field.

Running query expansion on external document collections can result in better expansion terms being selected, especially if a target collection is small [8]. In agent-based systems, the requirement for multiple sources of statistics and data is fundamental to many interesting applications, such as our example of a travel agent.

5 CHALLENGES FOR FUTURE WORK

We identified areas where our vision calls for future work throughout this paper, but we revisit some ideas here. SAM provides a framework for thinking about future agent-based systems as well as state-of-the-art retrieval models and tasks that use IR as a subroutine.

For our future agent-based users of search, we need:

(1) search systems that allow us to re-use retrieval models in novel settings, e.g., for re-ranking of ad-hoc documents or summaries.
(2) predictable, self-aware retrieval models that can give us meaningful scores across different collections and queries while still producing state-of-the-art rankings.
(3) tools for constructing "learnable" agent policies that can be trained either online or offline and can manipulate generic information objects (GIOs) in order to synthesize knowledge from multiple sources.

These are the three core challenges we identified as we developed SAM, but we expect there will be more as the world and our field moves toward intelligent autonomous agent users as consumers of our systems rather than interfacing directly with human users.

6 CONCLUSION

Algorithms and agents are important new users with very different needs and ways of consuming information than humans. Complex information tasks shift the burden of information processing to algorithms that need to reason over heterogeneous data and perform constraint resolution, reasoning, and summarization We introduce the Search Agent Model to address key challenges building complex systems. SAM addresses the key challenges of task-state modeling, policies for information agents, and efficient and effective communication with backend search systems and agents. We demonstrated how SAM could replace custom and inflexible hard-coded search 'agents' for tasks ranging from NLP, advanced retrieval applications, and learning to rank. The model is the first step towards enabling AAS users in developing new information agent applications. Next steps will focus on key system components, particularly on more detailed SAM architectures and prototype systems.

ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval.

REFERENCES

[1] Krisztian Balog. 2015. Task-completion Engines: A Vision with a Plan.. In *SCST@ ECIR*. Citeseer.

[2] Marc-Allen Cartright and James Allan. 2011. Efficiency optimizations for interpolating subqueries. In *CIKM*. 297–306.

[3] Stephen Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In *SIGIR*.

[4] J. Shane Culpepper, Fernando Diaz, and Mark Smucker. 2018. Research Frontier in Information Retrieval – Report from the Third Strategic Workshop on Information Retrieval in Lorne (SWIRL 2018). In *ACM SIGIR Forum*, Vol. 52. ACM, 34–90.

[5] Jeffrey Dalton, James Allan, and David A. Smith. 2011. Passage retrieval for incorporating global evidence in sequence labeling. In *CIKM*. 355–364.

[6] Jeffrey Dalton and Laura Dietz. 2013. A Neighborhood Relevance Model for Entity Linking. In *Proceedings of the 10th International Conference in the RIAO series (OAIR) (RIAO '13)*. ACM, New York, NY, USA. https://doi.org/10.1145/2063576.2063633

[7] Jeff Dalton, Laura Dietz, and James R Allan. 2014. Entity query feature expansion using knowledge base links. In *SIGIR*.

[8] Fernando Diaz and Donald Metzler. 2006. Improving the estimation of relevance models using large external corpora. In *SIGIR*. 154–161.

[9] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC Complex Answer Retrieval Overview. In *TREC*.

[10] Henry A Feild. 2013. *Exploring privacy and personalization in information retrieval applications*. Ph.D. Dissertation.

[11] John Foley, Brendan T. O'Connor, and James R Allan. 2016. Improving Entity Ranking for Keyword Queries. In *CIKM*.

[12] Behzad Golshan, Alon Y. Halevy, George A. Mihaila, and Wang Chiew Tan. 2017. Data Integration: After the Teenage Years. In *PODS*.

[13] Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM*. 699–708.

[14] Evangelos Kanoulas, Ben Carterette, Mark Hall, Paul Clough, and Mark Sanderson. 2011. Overview of the trec 2011 session track. (2011).

[15] David Maxwell and Leif Azzopardi. 2016. Agents, Simulated Users and Humans: An Analysis of Performance and Behaviour. In *CIKM*.

[16] David Maxwell and Leif Azzopardi. 2016. Agents, Simulated Users and Humans: An Analysis of Performance and Behaviour. In *CIKM*. 731–740.

[17] David Maxwell, Leif Azzopardi, Kalervo Järvelin, and Heikki Keskustalo. 2015. Searching and Stopping: An Analysis of Stopping Rules and Strategies. In *CIKM*.

[18] Federico Nanni, Bhaskar Mitra, Matt Magnusson, and Laura Dietz. 2017. Benchmark for complex answer retrieval. In *ICTIR*. 293–296.

[19] Ryen William White. 2016. 978 - 1 - 107 - 03422 - 8 - Interactions with Search Systems.