Crowdsourcing Relevance Judgments Using Reinforcement Learning

Tyler McDonnell
Department of Computer Science
University of Texas at Austin

1. INTRODUCTION

Evaluation of Information Retrieval Systems (IR) traditionally relies upon expert annotators to provide both search queries and sets of *relevance judgments* for documents indexed by the retrieval system, which indicate, for a given query whether a document is relevant or not relevant. Unfortunately, hiring expert judges to provide these judgments is increasingly expensive, as the need to scale evaluation to larger and more diverse bodies of queries and web pages grows as the number of users of search engines increases.

More recently, the rise of platforms such as Amazon's Mechanical Turk has prompted the use of crowdsourcing as a scalable alternative to hiring expert annotators for collecting relevance judgments. However, the opportunity to leverage a large and diverse body of workers comes with new concerns, namely quality control, since it can be extremely difficult or manually intensive to verify the work products of vast numbers of unreliable workers in practice.

In fact, quality control is a central challenge when deploying crowdsourcing to any domain, and as a result, a wide body of techniques have been developed to address this concern. The most popular by far relies on *repetition* and *aggregation*. Rather than having a single unreliable crowdworker provide a relevance judgment, one can simply collect several responses from different crowdworkers and choose the most common answer (i.e., majority vote). Under the assumption that *most* workers are reliable, we can rely on the law of large numbers to converge on correct answers through this repetition. Additionally, we can relax this assumption and arrive at quality answers more efficiently or for very difficult tasks by using more intelligent aggregation techniques. For instance, a more sophisticated aggregation technique developed by Dawid and Skene relies on using an Expectation Maximization (EM) algorithm to estimate the reliability of workers and weight the answers of reliable workers more heavily than others [4]. Despite varying levels of sophistication, all aggregation techniques try to form high-quality answers by consulting multiple noisy labels.

While repetition and aggregation have done much to assuage quality control concerns in the crowdsourcing domain, asking many people to complete a task is often wasteful and undermines the scalability and cost-effectiveness of crowdsourcing. When crowdsourcing relevance judgments, one would ideally like to collect the fewest number of judgments possible to arrive at an accurate conclusion of relevance for any *particular* web page. In fact, in practice there exists an asymmetry, in that some web pages or search queries are more difficult to provide accurate relevance labels for than others. In these cases, we may require the help of more crowdworkers on average to arrive at a correct label. To this end, this paper explores the use of reinforcement learning (RL) techniques for crowdsourcing relevance judgments.

Whereas most previous attempts to adopt RL for crowdsourcing have explicitly studied a trade-off between exploration and exploitation of the crowd worker space and their respective skill sets, this work takes a slightly different approach. The techniques in this paper will attempt to model a state space defined by the actions taken by the crowd requester (i.e., the decisions made by the person hiring crowdworkers, such as how many judgments they have collected for a particular document) and discriminating features of the answers that were collected through these actions. The goal is to minimize the number of judgments collected for each individual document subject to some accuracy goal.

Whereas many traditional RL applications in crowdsourcing assume the ability to instantly validate answers from crowdworkers, no such assumptions are made in this paper. The model will seek to learn from features derived from annotator judgments and task idiosyncrasies (e.g., characteristics about how long or what *type* of web page is being reviewed). Unfortunately, in the traditional formulation of the relevance judgment task, we receive nothing more than a binary answer from the worker, which is insufficient for supervision. To this end, we adopt the *annotator rationale* paradigm proposed by [15] and [11], in which annotators provide excerpts from documents to support their relevance judgment for that document. Discriminating features such as inter-annotator rationale overlap or the presence of search query keywords can then be derived from these rationales.

The simplest formulation of this task is as a small, finite episodic Markov Decision Process (MDP), in which states

are defined by the number of judgments that have been collected and state transitions occur when the agent decides to either (a) collect an additional relevance judgment from a random crowd worker or (b) commits a decision about the relevance of a document according to some aggregation scheme and the answers collected up to that point from the crowd. To encourage cost-efficiency, negative rewards are given for each state transition that involves collecting an additional judgment from the crowd, and episodes are terminated when the agent decides to commit a relevance decision. Positive rewards are given for correct relevance decisions.

While this model is useful, it does not address the cost-efficiency trade-offs on a document-by-document basis or learn from the signal received from the worker (e.g., rationales) or task itself (e.g., query topic). To integrate these features, this work will also explore techniques based on function approximation.

2. RELATED WORK

2.1 Relevance Judgments

For 25 years, the Text Retrieval Conference of the National Institute of Standards has organized and collected relevance judgment datasets to support IR evaluation [14]. Trusted relevance judgments are a cornerstone of TREC and IR evaluation, and they serve as a gold standard in this paper for evaluating reinforcement learning techniques.

Some prior literature has also sought to improve the scalability of relevance judgment-based evaluation methodologies. For example, [2] propose the use of relative, rather than absolute, relevance labels for documents. However, most work in this space has focused on leveraging crowdsourcing for collecting relevance judgments. [1] hire crowdworkers to complete micro-tasks consisting of individual relevance judgment tasks on a four-point relevance scale. They found that this technique provided fast turnaround for posted tasks at a low cost. Though the quality of individual workers was inconsistent in their study, they found that they were able to achieve high quality results through repetition and aggregation of answers. These results are consistent with how crowdsourcing has been applied to tasks in various domains, and has been widely adopted as the standard way to crowdsource relevance judgments for subsequent studies. This study will use data collected using this framework.

2.2 RL for Crowdsourcing

Reinforcement learning techniques have been applied to problems within the crowdsourcing domain. Crowdsourcing has been likened to the multi-armed bandit problem in the sense that there is a trade-off between exploration, to identify and characterize the skills of new workers, and exploitation (unfortunate word choice!), of known workers or skill sets. However, unlike the traditional bandit problem, crowdsourcing is subject to more prohibitive budget constraints which can make greedy selection for exploitation sub-optimal. As a result, more recent works have devised budgeted variants of the bandit problem [7], [6]. Each of these approaches is related to the goal of this paper, which involves a trade-off between exploration and exploitation under budget constraints. However, in this case we are not interested in explicitly exploring and modeling the worker pool, but rather modeling a state space based on characteristics of each individual task and features derived from the answers that workers provide for it.

[8] formalize the *online task assignment problem*, in which a requester has a fixed set of tasks and a budget for the maximum number of times he would like each task completed and must dynamically allocate incoming workers to tasks. They present a two-phase exploration/exploitation algorithm and show that it outperforms both random assignment and greedy approaches. Their model assumes that requesters are able to instantly evaluate the accuracy of answers provided by workers and that the quality of incoming workers is unknown and must be discovered through exploration. The problem formulation in this paper is similar, since we seek to minimize the number of relevance judgments collected on a document-by-document basis to maximize a reward function. However this approach does not explicitly model individual crowdworkers or assume that we are able to instantly evaluate the accuracy of answers collected.

[3] develop agents which use Bayesian network learning and inference in combination with Partially-Observable Markov Decision Processes (POMDPs) to control crowd voting for a binary-choice task and show that their approach outperforms simple majority vote, especially as the importance of accuracy increases and affects the reward function. This task formulation is analogous to the simplest formulation of the relevance judgment task, which simply asks if a document is relevant or not relevant without additional granularity.

2.3 Learning for Relevance Judgments

As previously discussed, trusted relevance judgments are a corner of the evaluation of IR systems. Nevertheless, prior work has shown that even expert annotators frequently disagree about the relevance of some documents in practice [13]. This indicates an asymmetry in the subjectivity of documents that complicates the crowdsourcing process: some documents require more extensive sampling and careful aggregation of work products than others. Since cost-efficiency is important for IR evaluation methodologies, one might imagine that the relevance judgment collection process is amenable to reinforce learning techniques that can model these asymmetries and collect annotator labels with greater efficiency. Despite this intuition, I am not aware of any work that thoroughly explores the application of reinforcement learning for relevance judgments.

One reason for this might be the lack of *signal* available in traditional relevance judgment collections. Typically, we only collect a single number (i.e., the relevance judgment) from workers for each document. More recently, some work has provided new ways to derive additional signal from relevance judgment tasks that might prove useful for both supervised and reinforcement learning models. [15] propose the user of *annotator rationales*, or text excerpts from a document provided alongside a labeling decision, as a source of supervision for text classification. [11] apply this concept directly to the problem of crowdsourcing relevance judgments and illustrate numerous additional benefits of this approach that are specific to crowdsourcing, such as increased transparency and task difficult, which in turn discourage cheating and increase the average quality of worker answers. In ad-

dition, [11] have made their data available (i.e., relevance judgments and rationales). I will use this dataset for training and evaluating RL models for relevance judgment collections, with particular interest paid to the additional signal provided by rationales.

More generally, prior work has attempted to obtain signal from non task-specific sources for supervision. [9] and [10] provide an overview of techniques that have been developed for intelligent aggregation of answers from multiple workers, ranging from the simplest technique, majority vote, to more sophisticated techniques, such as EM-based worker models which weight the answers of individual workers according to some estimation of their quality. The RL methods proposed in this work will learn when to stop collecting judgments for a particular task and subsequently apply one or more of these classic aggregation techniques to the results.

3. METHODOLOGY

This section formalizes the problem of crowdsourcing relevance judgments and discusses different approaches for formulating this as an RL problem.

3.1 The Problem

The classical **relevance judgment problem** consists of a set of *search queries* and, for each query, a set of *documents*. The goal is to determine, for each {*query*, *document*} pair, whether the document is relevant or not relevant to the specified search query. Traditionally these determinations are made by an "expert" annotator, but in its crowdsourcing formulation, a *requester* is responsible for collecting one or more judgments from *crowd annotators* and aggregating them to arrive at an informed relevance decision.

As outlined in Section 2, when using crowdsourcing to tackle the relevance judgment problem, one typically relies on a scheme of static repetition and aggregation: a requester collects some predetermined number of judgments from different crowd annotators for each and aggregates them in some intelligent manner to arrive at a single decision. This helps combat noise in the collected labels resulting from uneven worker expertise, subjectivity, or worker malice. However, there are downsides to this approach. It is **redundant** and collects many judgments for a single task. This not only undermines the cost-efficiency of crowdsourcing, but it also unclear how to reason about what static number of judgments for each document would suffice across the dataset. Instead, the decision about how many judgments to collect is often made arbitrarily or by dividing total budget among all tasks. In this sense, the approach is also **naive**. By collecting the same number of judgments for each task, it implicitly assumes all tasks are equally as difficult and all answers equally as noisy.

Ideally, we would like to collect the minimum number of judgments required to make a correct relevance decision *for a specific document*, which may be subject to both task characteristics and noisiness of the labels.

3.2 RL Formulations

To navigate this ideal scenario, we can adopt RL techniques, which are well-suited to model and optimize rewards in such complex state spaces. I first present a simple *tabular approach* that vastly simplifies the state space and allows us to evaluate current collection approaches. I then develop a *function approximation approach* which incorporates task-specific features and feedback from worker responses to provide a more accurate reflection of the state space and thus, theoretically, more efficient collection.

3.2.1 Tabular Formulation

A simple tabular formulation of the relevance judgment problem is as a finite, episodic Markov Decision Process (MDP), in which states are entirely defined by the number of judgments that the agent has collected at discrete points in time. At any point in time, the agent may choose from two actions: COLLECT an additional judgment from a random crowd annotator; or DECIDE on the relevance of a document for a particular query given the judgments it has collected. A single episode represents the entire collection and decision-making formulation for a single document, and ends when an agent takes the DECIDE action. Since our goal is to optimize the cost-efficiency of crowdsourcing relevance judgments, the reward function would generate negative rewards for each judgment collected, and a positive or negative reward for a correct or incorrect relevance decision, respectively.

Any deterministic policy learned from this tabular formulation will effectively establish an optimal number of judgments n to collect for each document on average. While this approach ignores episode-specific characteristics that might further improve efficiency, it still serves as a useful model of traditional approaches based on collecting predetermined numbers of judgments. In this sense, even this simple model can inform future decision-making when selecting how many judgments to collect and form a baseline when evaluating more sophisticated techniques.

3.2.2 Function Approximation

To incorporate task-specific characteristics and feedback from worker responses into a more effective model of the environment, we can use function approximation. Incorporating these elements into a model in a tabular fashion would be impractical, both because of the sheer size of the state space and characteristics of crowdsourcing (e.g., high-latency, finite worker pools, resource constraints like time and money, or subjective task complexity) which make it difficult to collect or simulate large numbers of examples in practice.

Notably, these same constraints also make the problem less susceptible to representation learning approaches (e.g., deep learning), so my evaluation relies on manual feature engineering informed by prior art. In particular, I explore two classes of features for linear function approximation: *task features* and *work features*. A complete list of features used throughout this paper is outlined in Table 1. Each feature listed is a binary feature. I also investigated non-binary features, but found that they were less effective. This is likely related to the availability of training data.

Together, these two classes of features are meant to establish a more accurate approximation of the relevance judgment problem environment. To my knowledge, this is the first relevance judgment model, RL or otherwise, to jointly incorporate both task and response characteristics, thus en-

coding both noisiness in labels and some sense of the inherent difficulty or noisiness of the specific task at hand.

Task Features

Task features are derived from the query or document being evaluated. These features are inspired by previous work. For instance, [12] and formalize classes of user goals predominate throughout web search and builds a taxonomy of query types based on three high-level classes: *navigational*, in which the user wants to navigate to a known website with which they are already familiar; *informational*, in which the user wants to learn something by reading web pages; and *resource*, in which the user wants to obtain some non-informational resource, such as a program or audiovisual media.

Work Features

Work features are derived from characteristics of the annotator judgments and/or rationales which have been collected by an agent at any point in time. These features can be rationalized as a way of encoding knowledge about the noisiness of worker labels into the model of the state space.

The relevance of some of these features to the state space is obvious: e.g., collecting many judgments or seeing unanimous agreement among many judges should surely be correlated with reliable judgments, since the repetition and aggregation scheme is based on these principles. On the other hand, features derived from annotator rationales may be less obvious. These features are based on [11], which shows that aggregating annotator judgments based on the similarity of their rationales can produce more accurate judgments. The intuition is that workers who simply agree on an answer may agree by chance or through ignorance, but that workers who agree on both an answer and their way of supporting it have probably formed a coherent argument and reliable answer.

4. EXPERIMENTAL SETUP

4.1 Dataset

Due to budget constraints, the experiments are carried out "in vitro" using an existing crowdsourcing dataset. I adopt the dataset made publicly available by [11], which consists of five relevance *and* rationales for many documents across a variety of search queries to simulate online use of RL methods. These relevance judgments and rationales were collected from workers using Amazon Mechanical Turk and represent the only relevance judgment dataset to incorporate annotator rationales, which provide a richer signal from which to model our state space using RL techniques, as mentioned in Section 2.3. To produce the document-specific *task features* from Table 1, I crawl the URLs associated with each document in the dataset. To produce the query type of each query, I manually categorized each URL using the taxonomical definitions set forth in [12].

4.2 Implementation

I implemented the problem representations outlined in Section 3.2 as well as all learning algorithms on top of the RLPy framework [5]. RLPy is an object-oriented RL software package focused on value-function methods using linear function approximation. RLPy is ideal for such experiments due to its separation of concerns: it entirely separates agents, domains, policies, and representations from one another, making it easy to provide custom implementations of any of them without affecting other parts of the RL system.

5. EVALUATION

For evaluation, I use the dataset described in Section 4.1 to *simulate* online crowdsourcing. The dataset includes five relevance judgments and rationales for each {query, document} pair, as well a "correct" judgment for each.

As described in Section 3.2, an episode pertains to the collection and decision process for a single {query, document} pair, so at the beginning of an episode, a pair and its associated judgments/rationales is selected from the dataset at random for simulation. The agent begins with no knowledge of the judgments which are available in the dataset. A COLLECT action thus results in the agent acquiring knowledge of the relevance judgment and rationale of a single worker (of the five present in the dataset) at random which it has not already collected. This collection is reflected by an updated environment, either in the simple state change in the tabular chase or through an updated feature vector in the approximation case. When the agent chooses a DECIDE again, the judgments which have been collected by the agent so far for this episode are aggregated into a single decision and compared to the "correct" judgment, which is also provided by the dataset. I use the simple Majority Vote scheme for aggregation, though Section 5.4 expands agent decision-making beyond this simple technique.

For the majority of the experiments, I assign the following static values for the reward function as described in 3.2: {+100 for a DECIDE action that results in a correct judgment, -100 for a DECIDE action which results in an incorrect judgment, -10 for a COLLECT action}. I ran various experiments before landing on these values, and I discuss the implications of tuning these parameters in Section 5.5.

I first establish a baseline using the tabular problem formulation described in Section 3.2.1, and compare this baseline with the space of predetermined policies (i.e., those which collect an equivalent number of judgments for all documents). I then present results from different RL algorithms operating in the function approximation formulation and discuss and evaluate implications and possible modifications to the formulation presented thus far.

5.1 Establishing a Baseline

Section 3.2.1 describes a tabular approach for representing the relevance judgment problem which poses the task as a finite, episodic MDP. Recall that typical approaches to relevance judgment collection collect the same predetermined number of judgments for every single document in the test collection. As previously discussed, this tabular formulation will result in a fixed policy equivalent to a predetermined strategy, and is thus convenient because it allows to evaluate this family of approaches typically employed in the field and converge on one that would yield the best results for our dataset. Thus, it provides a great baseline when comparing more sophisticated methods.

Recall that the tabular representation defines a state space entirely dependent on how many judgments have been col-

Feature	Type	Description
QTYPE=Navigational	Task	The query is a *navigational* query.
QTYPE=Informational	Task	The query is an *informational* query.
QTYPE=Resource	Task	The query is a *resource* query.
QLENGTH-SHORT	Task	The query is two words or less.
QLENGTH-LONG	Task	The query is longer than two words.
DLEN=50	Task	The total amount of text on the web page is 50 or more words.
DLEN=500	Task	The total amount of text on the web page is 500 or more words.
J=0	Work	0 judgments have been collected.
J=1	Work	1 judgment have been collected.
J=2	Work	2 judgments have been collected.
J=3	Work	3 judgments have been collected.
J=4	Work	4 judgments have been collected.
J=5+	Work	5 or more judgments have been collected.
ASIM<0.25	Work	The average pairwise similarity of rationales for this task is below 0.25
ASIM=0.25	Work	The average pairwise similarity of rationales for this task exceeds 0.25.
ASIM=0.50	Work	The average pairwise similarity of rationales for this task exceeds 0.50.
ASIM=0.75	Work	The average pairwise similarity of rationales for this task exceeds 0.75.
MSIM=0.25	Work	The highest pairwise similarity of any two rationales for this task exceeds 0.25.
MSIM=0.50	Work	The highest pairwise similarity of any two rationales for this task exceeds 0.50.
MSIM=0.75	Work	The highest pairwise similarity of any two rationales for this task exceeds 0.75.
QPRESENT	Work	One or more query keywords is present in at least one annotator rationale.
UNANIMOUS	Work	At least two judgments have been collected and all are identical.

Table 1: Function Approximation Features

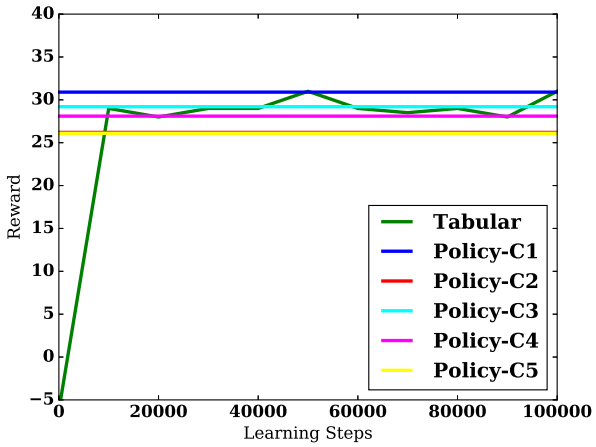

Figure 1: Baseline Performance

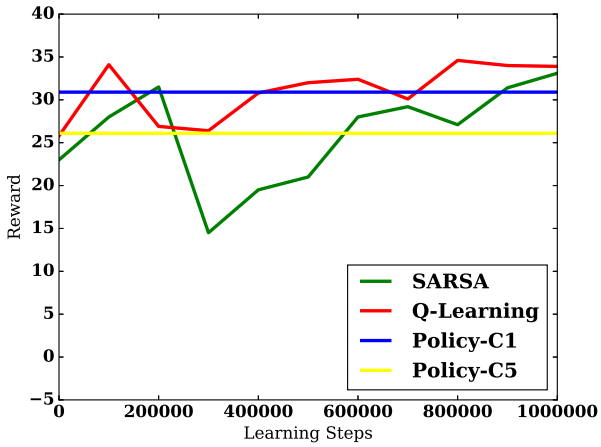

Figure 2: RL Performance

lected by the agent. Since the dataset I am using only includes five judgments per document (see Section 4.1), the full tabular state space consists of $\{I, C_1, C_2, C_3, C_4, C_5, D\}$, where I is the initial state, D is the terminal state reached after executing a DECIDE action, and C_i states corresponding to the agent having collected i judgments.

Figure 1 shows the results of a Q-Learning agent learning in this tabular formulation. The straight lines represent the average reward that would be achieved with the five possible predetermined strategies (i.e., collecting one judgment for every document, two judgments for every document, etc.) over the entire dataset. Interestingly, the Q-Learning agent converges to a policy of selecting one judgment per document, suggesting that repetitious judgments (i.e., the basis of quality control in crowdsourcing) are a waste for this dataset. This is discussed further in Section 5.5.

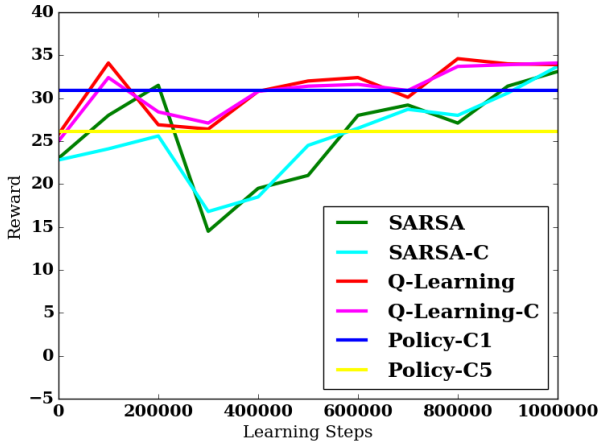

Figure 3: RL Performance - Aggregation

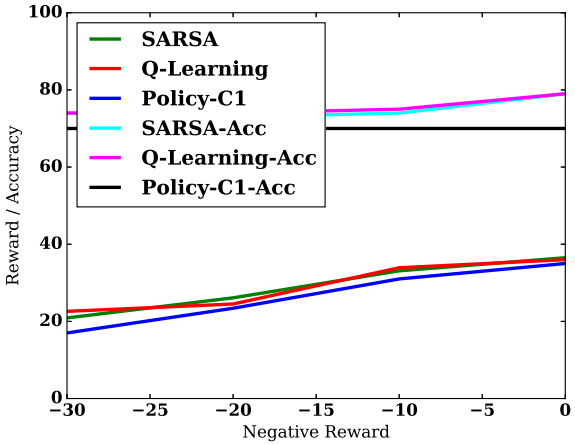

Figure 4: Effect of Reward Function

Rank	Action: Collect	Action: Decide	Action: Collect	Action: Decide-MV	Action: Decide-HSIM
1	J=0	UNANIMOUS	J=0	UNANIMOUS	UNANIMOUS
2	MSIM<0.25	QPRESENT	MSIM<0.25	QPRESENT	MSIM>0.75
3	ASIM<0.25	ASIM=0.75	ASIM<0.25	ASIM>0.75	ASIM>0.75

Table 2: Feature Analysis

5.2 Comparison of RL Algorithms

Having established that the policy of collecting one judgment before a decision is optimal *in the average case* and observed the space of possible predetermined policies, we can turn our attention to methods with can dynamically cater to more complex interpretations using function approximation.

I evaluated the function approximation representation presented in 3.2 using two RL algorithms: Q-Learning and SARSA(λ). I chose these two algorithms because of their prominence in the RL community and their coverage of both off-policy (Q-Learning) and on-policy (SARSA) learning. The learning algorithms operate under an ϵ-Greedy policy, with an initial learning rate of 0.1. Both are undiscounted. These hyperparameters were set using a minimal hyperparameter search. Though these parameter settings provided the highest performance among those searched, all parameter settings which were evaluated were relatively unstable during learning. Further tuning of these parameters might yield better performance.

Figure 2 shows the performance of SARSA and Q-Learning against the baseline fixed policies for collecting only one judgment and five judgments. At each data point shown, the policy is evaluated over 1,000 episodes. SARSA and Q-Learning appear to be relatively unstable during learning, but do converge upon strategies which outperform Policy-C_1, the optimal predetermined baseline.

Interestingly, neither of the algorithms exhibits the learning curves that are typically seen in many problems, with large gains in reward in the early stages of learning. Instead, they exhibit more of an oscillating learning behavior. Perhaps this is not surprising, since, for example, the algorithms are not guaranteed to converge when combined with linear function approximation. I believe these learning curves are ultimately a consequence of this dataset, which consists of extremely high-quality judgments, tempering even the worst-case scenarios during learning and making it difficult to make substantive learning gains over simple strategies. This point is discussed further in Section 6.

5.3 Feature Analysis

Since the use of linear function approximation here operates only on binary features, it is easy to rank the importance of features in determining actions by their resultant weights after training. The left half of Table 2 shows the top 3 binary features for the two actions available to the agent: COLLECT and DECIDE. I included only the top 3 features both for clarity of discussion and because the relative weights of less-important features tended to make them irrelevant.

The strongest features indicating that an agent should collect additional judgments are having collected no judgments and poor similarity between collected annotator rationales. These make sense: if the agent has not collected any judgments from the crowd, the best it can do at that point is randomly guess the relevance of the document. Alternatively, if the agent has collected judgments, but their corresponding rationales are not similar, then that suggests that either the workers are lazy and have provided sub-par rationales or that the task is inherently difficult or subjective and workers have simply not reached a consensus. In either case, additional crowd judgments would likely be desirable.

On the other hand, unanimous agreement among judgments that have been collected, the presence of query keywords in annotator rationales, and very high similarity between

collected annotator rationales were learned to be strong indicators that the agent should stop collecting and make a decision about the relevance of the document. Again, these seem to fit intuition: the presence of query keywords is probably a strong indicator (though not a sufficient condition!) of document relevance, and unanimous judgments or highly similar rationales indicate that the crowd annotators have likely converged on sound reasoning.

Interestingly, all of the most relevant features for *work features*, and not explicitly features of the task itself. One explanation for this is that the *task features* I selected (shown in Table 1) are simply poorly formed or not correlated with task complexity. However, this may also be a result of the bias of the dataset. Over the 50 search queries included, 90% of them turned out to be *short*, *informational* queries giving the agent poor learning coverage to the other features.

5.4 Selection of Aggregation Method

The action space up to this point has consisted of exactly two actions: COLLECT an additional judgment from the crowd and DECIDE on a judgment by aggregating those which have been collected according to the majority vote aggregation scheme. Notably, since quality control is such a pervasive issue in crowdsourcing, a vast and spectral body of aggregation techniques has been developed. Ordinarily, when novel aggregation techniques are introduced, they are statically used over every data point in a test collection and compared to existing techniques.

Interestingly, RL algorithms enable *dynamic* selection of aggregation techniques: we can simply incorporate more than one aggregation technique into our action space and then theoretically learn *when* to dispatch different techniques. To evaluate this idea, I evaluate the same Q-Learning and SARSA algorithms using the same function approximation formulations with slightly expanded action spaces: {COLLECT, DECIDE-MV, DECIDE-HSIM}. The DECIDE-MV action corresponds to aggregating the collected judgments according to the simple majority vote scheme, and the DECIDE-HSIM action aggregates judgments according to the high rationale similarity filter proposed by [11].

Figure 3 shows the results of these experiments compared to the action space from the previous experiments in Section 5.2. The SARSA-C and Q-LEARNING-C plots correspond to the algorithms operating on the expanded action space, where **C** stands for "choose" (i.e., choosing the aggregation scheme). In these experiments, the expanded action space and ability to choose among these aggregation schemes was not very helpful in increasing reward. Again, this is likely a consequence of the dataset: we have seen that, on average, an agent may only collect a single judgment; in this majority of cases, the aggregation scheme is irrelevant.

However, we can still gleam interesting insights from these experiments. The right half of Table 2 shows the top 3 ranked features for the DECIDE-MV and DECIDE-HSIM actions. Interestingly, they do not weight the same features heavily. In particular, the *maximum pairwise similarity* of rationales is more important the average pairwise similarity in the case of DECIDE-HSIM. This actually makes a lot of sense: the aggregation technique proposed by [11] is capable of selecting the minority crowd opinion over the majority if members of the minority strongly agree on their rationales. In this sense, the maximum pairwise similarity should indeed be more important than the average similarity.

These details suggest that RL methods are capable of intelligent dispatch of aggregation methods, and that in a more complete evaluation, we might see additional performance gains from doing so.

5.5 Reward Function

It is now imperative to return to one of the key representation decisions of these experiments, the definition of the reward function. Previously, I highlighted one of the key difficulties with even simple collection policies which collect a static number of judgments for every document: it is unclear how we should reason about *how many* judgments should be collected. In theory, such reasoning *should be* predicated on many variables: available, budget, size of the crowd workforce, latency of task completion, quality of workers, and ultimately, on how important different accuracy thresholds are for evaluating IR systems. For instance, if collecting 5x as many judgments provides only 5% better accuracy, is it worth it? Unfortunately, little work has been done to provide adequate answers to this question, a hindrance to the wide-scale adoption of crowdsourcing for this task.

Nevertheless, by exploring different reward functions, we can build a general idea of how these RL techniques would perform across a spectrum of cost-efficiency scenarios. All of the experiments thus far use the following reward function: {+100 for a correct judgment, -100 for an incorrect judgment, -10 for collecting an additional judgment}. Table 4 shows the reward *and accuracy* (in terms of correct judgments) of the final policies learned by SARSA and Q-Learning, as well as our baseline policy, evaluated over the dataset for different negative reward values for collecting additional judgments. Notably, SARSA and Q-Learning are always able to outperform the naive baseline. This suggests that regardless of which reward function is most representative of evaluation needs in practice, RL techniques should be considered. It is also interesting to note that across the entire spectrum, the differences in accuracy are less pronounced than differences in reward. In other words, all policies may approach similar accuracy, but at different levels of cost-efficiency dependent on requirements. RL techniques are once again useful to navigate such constraints.

6. DISCUSSION

In this report, I applied RL techniques to the crowdsourced relevance judgment problem. In practice, crowdsourcing is a bit of a Rube Goldberg machine: quality control is handled by a hodgepodge of arbitrary design decision, asking multiple workers to complete the same task, and aggregating the answers according to some academic scheme. Indeed, it is often difficult or impossible to know how many repetitious answers should be collected or which of the wide body of aggregation techniques should be used in these scenarios.

Reinforcement learning techniques are a powerful mechanism for dealing with these concerns, as they are fundamentally formulated to deal with such optimization problems. Here, I presented two classes of RL representations for the

relevance judgment problem. The first, a tabular approach, vastly simplifies the state space, but provides a powerful way of evaluating classes of policies based on collecting an identical number of judgments across all documents. Typical approaches in practice exist in this class.

I also developed a linear function approximation approach which draws features from both the task itself and feedback from the responses (i.e., the changing environment) as they are collected. I found that SARSA and Q-Learning outperform policy baselines when learning in this representation. They provide automatic learning of and thus useful insight into features that can affect or simplify judging and increase efficiency. Furthermore, I found that these models can be extended to consider other unexplored problems within the domain, such as intelligent dispatch of aggregation techniques, rather than defaulting to the same technique in every situation. These results suggest that RL techniques provide a powerful and *flexible* framework for the relevance judgment problem, one which can encode the particular accuracy and domain needs of different applications (e.g., ad-hoc search, question answering, social search) elegantly through modifications to reward functions, action spaces, and features; and then operate autonomously online.

6.1 Future Work

In light of these interesting results, there are a number of areas for improvement and future work. Most obviously would be the further tuning of hyperparameters, as the results seemed sensitive to these selections. Additionally, even the final parameters which were used yielded unstable learning curves for both Q-Learning and SARSA. Another obvious area would be to explore a wider body of RL learning algorithms.

Additionally, the dataset used here from [11] is quite abnormal in the crowdsourcing space: the data features significantly higher quality of judgments than prior literature. At the same time, this dataset is the only one to feature additional signal, such as annotator rationales, which made the application of RL techniques appealing. Though I believe these peculiarities affected many of the results in this study, such as the marginal gains in accuracy and reward via RL and trivial and counter-intuitive optimal policy, I believe these techniques would be even more useful for more noisy collections.

Related to this is the concern of training data. Tough I did not explicitly explore the rate of convergence in this study, in practice this is a major issue. Budget constraints are a crucial concern in crowdsourcing applications, and with more noisy datasets, it may take an impractical amount of data for an agent to adapt to a particular application space. To simplify learning, combing of features, or exploration of more useful features which were not explored in this study, may be necessary.

7. REFERENCES

[1] O. Alonso, D. E. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. In *ACM SigIR Forum*, volume 42, pages 9–15. ACM, 2008.

[2] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there. In *European Conference on Information Retrieval*, pages 16–27. Springer, 2008.

[3] P. Dai, C. H. Lin, D. S. Weld, et al. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.

[4] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[5] A. Geramifard12, C. Dann, R. H. Klein, W. Dabney, and J. P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16:1573–1578, 2015.

[6] A. Goel, S. Khanna, and B. Null. The ratio index for budgeted learning, with applications. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 18–27. Society for Industrial and Applied Mathematics, 2009.

[7] S. Guha and K. Munagala. Approximation algorithms for budgeted learning problems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 104–113. ACM, 2007.

[8] C.-J. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, volume 12, pages 45–51, 2012.

[9] N. Q. V. Hung, N. T. Tam, L. N. Tran, and K. Aberer. An evaluation of aggregation techniques in crowdsourcing. In *International Conference on Web Information Systems Engineering*, pages 1–15. Springer, 2013.

[10] G. Kazai and I. Zitouni. Quality management in crowdsourcing using gold judges behavior. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 267–276. ACM, 2016.

[11] T. McDonnell, M. Lease, M. Kutli, and T. Elsayed. Why Is That Relevant? Collecting Annotator Rationales for Relevance Judgments. In *Proceedings of the 4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2016.

[12] D. E. Rose and D. Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, pages 13–19. ACM, 2004.

[13] E. M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information processing & management*, 36(5):697–716, 2000.

[14] E. M. Voorhees, D. K. Harman, et al. *TREC: Experiment and evaluation in information retrieval*, volume 1. MIT press Cambridge, 2005.

[15] O. Zaidan, J. Eisner, and C. D. Piatko. Using" annotator rationales" to improve machine learning for text categorization. In *HLT-NAACL*, pages 260–267. Citeseer, 2007.