LABORATORY DIRECTED RESEARCH & DEVELOPMENT

SAND20XX-XXXXR
LDRD PROJECT NUMBER: 173667
LDRD PROJECT TITLE: Active Learning in the Era of Big Data
PROJECT TEAM MEMBERS: Kevin Jamieson, Warren L. Davis IV

ABSTRACT:

Active learning methods automatically adapt data collection by selecting the most informative samples in order to accelerate machine learning. Because of this, real-world testing and comparing active learning algorithms requires collecting new datasets (adaptively), rather than simply applying algorithms to benchmark datasets, as is the norm in (passive) machine learning research. To facilitate the development, testing and deployment of active learning for real applications, we have built an open-source software system for large-scale active learning research and experimentation. The system, called NEXT, provides a unique platform for real-world, reproducible active learning research. This paper details the challenges of building the system and demonstrates its capabilities with several experiments. The results show how experimentation can help expose strengths and weaknesses of active learning algorithms, in sometimes unexpected and enlightening ways.

INTRODUCTION:

We use the term “active learning” to refer to algorithms that employ adaptive data collection in order to accelerate machine learning. By adaptive data collection we mean processes that automatically adjust, based on previously collected data, to collect the most useful data as quickly as possible. This broad notion of active learning includes multi-armed bandits, adaptive data collection in unsupervised learning (e.g. clustering, embedding, etc.), classification, regression, and sequential experimental design. Perhaps the most familiar example of active learning arises in the context of classification. These active learning algorithms select examples for labeling in a sequential, data-adaptive fashion, as opposed to passive learning algorithms based on preselected training data.

The key to active learning is adaptive data collection. Because of this, real-world testing and comparing active learning algorithms requires collecting new datasets (adaptively), rather than simply applying algorithms to benchmark datasets, as is the norm in (passive) machine learning research. In this adaptive paradigm, algorithm and network response time, human fatigue, the differing label quality of humans, and the lack of i.i.d. responses are all real-world concerns of implementing active learning algorithms. Due to many of these conditions being impossible to faithfully simulate active learning algorithms must be evaluated on real human participants.

Adaptively collecting large-scale datasets can be difficult and time-consuming. As a result, active learning has remained a largely theoretical research area, and practical algorithms and experiments are few and far between. Most experimental work in active learning with real-world data is simulated by letting the algorithm adaptively select a small number of labeled examples from a large labeled dataset. This requires a large, labeled data set to begin with, which limits the scope and scale of such experimental work. Also, it does not address the practical issue of deploying active learning algorithms and adaptive data collection for real applications.

To address these issues, we have built a software system called NEXT, which provides a unique platform for real-world, large-scale, reproducible active learning research, enabling

- **machine learning researchers** to easily deploy and test new active learning algorithms;
- **applied researchers** to employ active learning methods for real-world applications.

At the heart of active learning is a process for sequentially and adaptively gathering data most informative to the learning task at hand as quickly as possible. At each step, an algorithm must decide what data to collect next. The data collection itself is often from human helpers who are asked to answer queries, label instances or inspect data. Crowdsourcing platforms such as Amazon's Mechanical Turk or Crowd Flower provide access to potentially thousands of users answering queries on-demand. Parallel data collection at a large scale imposes design and engineering challenges unique to active learning due to the continuous interaction between data collection and learning.

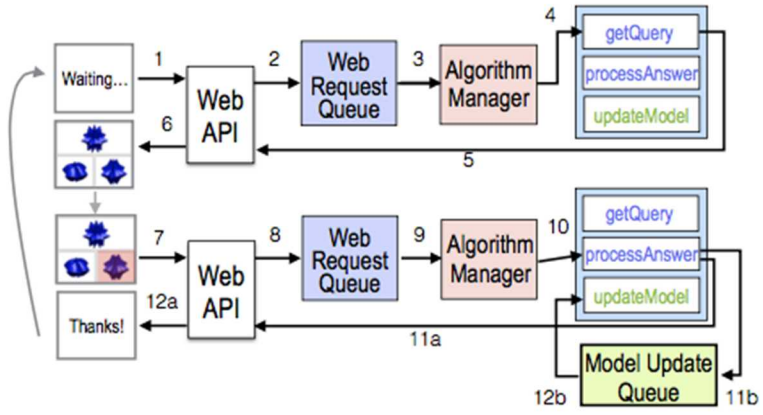
This report details the challenges of building active learning systems and our open-source solution, the NEXT system (available at https://github.com/kgjamieson/NEXT). We also demonstrate the system's capabilities for real active learning experimentation. The results show how experimentation can help expose strengths and weaknesses of well-known active learning algorithms, in sometimes unexpected and enlightening ways.

DETAILED DESCRIPTION OF EXPERIMENT/METHOD:

System Functionality - A data flow diagram for NEXT is presented in Figure 1. Consider an individual client among the crowd tasked with answering a series of classification questions. The client interacts with NEXT through a website which requests a new query to be presented from the NEXT web API. Tasked with potentially handling thousands of such requests simultaneously, the API will enqueue the query request to be processed by a worker pool (workers can be thought of as processes living on one or many machines pulling from the same queue). Once a worker accepts the job, it is routed through the worker's algorithm manager

(described in the extensibility section below) and the algorithm then chooses a query based on previously collected data and sufficient statistics. The query is then sent back to the client through the API to be displayed.

After the client answers the query, the same initial process as above is repeated but this time the answer is routed to the 'processAnswer' endpoint of the algorithm. Since multiple users are getting queries and reporting answers at the same time, there is a potential for two different workers to attempt to update the model, or the statistics used to generate new queries, at the same time, and potentially overwrite each other's work. A simple way NEXT avoids this race condition is to provide a locking queue to each algorithm so that when a worker accepts a job from this queue, the queue is locked until that job is finished. Hence, when the answer is reported to the algorithm, the 'processAnswer' code block may either update the model asynchronously itself, or submit a 'modelUpdate' job to this locking model update queue to process the answer later synchronously (see Section 4 for details). After processing the answer, the worker returns an acknowledgement response to the client.



1. Request query display from web API
2. Submit job to web-request queue
3. Asynchronous worker accepts job
4. Request routed to proper algorithm
5. Query generated, sent back to API
6. Query display returned to client
7. Answer reported to web API
8. Submit job to web-request queue
9. Asynchronous worker accepts job
10. Answer routed to proper algorithm
11a. Answer acknowledged to API
12a. Answer acknowledged to Client
11b. Submit job to model-update queue
12b. Synchronous worker accepts job

Figure 1: NEXT Active Learning Data Flow

```
def getQuery( context ):
    // GET suff. statistics stats from database
    // use stats, context to create query
    return query
def processAnswer( query, answer ):
    // SET (query, answer) in database
    // update model or ENQUEUE job for later
    return "Thanks!"
def updateModel( ):
    // GET all (query, answer) pairs S
    // use S to update model to create stats
    // SET suff. statistics stats in database
```

Figure 2: Example algorithm prototype that active learning researcher implements. Each algorithm has access to the database and the ability to enqueue jobs that are executed in order, one at a time.

Of this data flow, NEXT handles the API, enqueueing and scheduling of jobs, and algorithm management. The researcher interested in deploying their algorithm is responsible for implementing getQuery, processAnswer and updateModel. Figure 2 shows pseudo-code for the functions that must be implemented for each algorithm in the NEXT system.

A key challenge here is latency. A getQuery request uses the current learned model to decide what queries to serve next. Humans will notice delays greater than roughly 400 ms. Therefore, it is imperative that the system can receive and process a response, update the model, and select the next query within 400 ms. Accounting for 100-200 ms in communication latency each way, the system must perform all necessary processing within 50-100 ms. While in some applications one can compute good queries offline and serve them as needed without further computation, other applications, such as contextual bandits for personalized content recommendation, require that the query depend on the context provided by the user (e.g. their cookies) and consequently, must be computed in real time.

Realtime Computing - Research in active learning focuses on reducing the sample complexity of the learning process (i.e., minimizing number of labeled and unlabeled examples needed to learn an accurate model) and sometimes addresses the issue of computational complexity. In the latter case, the focus is usually on polynomial-time algorithms, but not necessarily realtime algorithms. Practical active learning systems face a tradeoff between how frequently models are updated and how carefully new queries are selected. If the model is updated less frequently, then time can be spent on carefully selecting a batch of new queries. However, selecting in large batches may potentially reduce some of the gains afforded by active learning, since later queries will be based on old, stale information. Updating the model frequently may be possible, but then the time available for selecting queries may be very short, resulting in suboptimal selections and again potentially defeating the aim of active learning. Managing this tradeoff is the chief responsibility of the algorithm designer, but to make these design choices, the algorithm designer must be able to easily gauge the effects of different algorithmic choices. In the NEXT system, the tradeoff is explicitly managed by mod- ifying when and how often the updateModel command is run and what it does. The system helps with making these decisions by providing extensive dashboards describing both the statistical and computational performance of the algorithms.

Reproducible research - Publishing data and software needed to reproduce experimental results is essential to scientific progress in all fields. Due to the adaptive nature of data collection in active learning experiments, it is not enough to simply publish data gathered in a previous experiment. For other researchers to recreate the experiment, the must be able to also reconstruct the exact adaptive process that was used to collect the data. This means that the complete system, including any web-facing crowd sourcing tools, not just algorithm code and data, must be made publicly available and easy to use. By leveraging cloud computing, NEXT abstracts away the difficulties of building a data collection system and lets the researcher focus on active learning algorithm design. Any other researcher can replicate an experiment with just a few keystrokes in less than one hour by just using the same experiment initialization parameters.

Expert data collection for the non-expert - NEXT puts state-of-the-art active learning algorithms in the hands of non-experts interested in collecting data in more efficient ways. This includes psychologists, social scientists, biologists, security analysts and researchers in any other field in which large amounts of data is collected, sometimes at a large dollar cost and time expense. Choosing an appropriate active learning algorithm is perhaps an easier step for non-experts compared to data collection. While there exist excellent tools to help researchers perform relatively simple experiments on Mechanical Turk (e.g. PsiTurk [1] or AutoMan [2]), implementing active learning to collect data requires building a sophisticated system like the one described in this paper. To determine the needs of potential users, the NEXT system was built in close collaboration with cognitive scientists at our home institution. They helped inform design decisions and provided us with participants to beta-test the system in a real-world environment. Indeed, the examples used in this paper were motivated by related studies developed by our collaborators in psychology.

NEXT is accessible through a REST Web API and can be easily deployed in the cloud with minimal knowledge and expertise using automated scripts. NEXT provides researchers a set of example templates and widgets that can be used as graphical user interfaces to collect data from participants.

Multiple Algorithms and Extensibility - NEXT provides a platform for applications and algorithms. Applications are general active learning tasks, such as linear classification, and algorithms are particular implementations of that application (e.g., random sampling or uncertainty sampling with a C-SVM). Experiments involve one application type but they may involve several different algorithms, enabling the evaluation and comparison of different algorithms. The algorithm manager in Figure 1 is responsible for routing each query and reported answer to the algorithms involved in an experiment. For experiments involving multiple algorithms, this routing could be round-robin, randomized, or optimized in a more sophisticated manner. For example, it is possible to implement a multi-armed bandit algorithm inside the algorithm manager in order to select algorithms adaptively to minimize some notion of regret.

Each application defines an algorithm management module and a contract for the three functions of active learning: getQuery, processAnswer, and modelUpdate as described in Figure 2. Each algorithm implemented in NEXT will gain access to a locking synchronous queue for model updates, logging functionality, automated dashboards for performance statistics and timing, load balancing, and graphical user interfaces for participants. To implement a new algorithm, a developer must write the associated getQuery, processAnswer, and updateModel functions in Python; the rest is handled automatically by NEXT. We hope this ease of use will encourage researchers to experiment with and compare new active learning algorithms. NEXT is hosted on Github and we urge users to push their local application and algorithm

RESULTS:

NEXT is capable of hosting any active (or passive) learning application. To demonstrate the capabilities of the system, we look at just one simple application motivated by cognitive science studies. The collected raw data along with instructions to easily reproduce these examples, which can be used as templates to extend, are available on the NEXT project page.

We consider is a pure-exploration problem in the dueling bandits framework [3], based on the New Yorker Caption Contest (data provided by Robert Mankoff at *The New Yorker*). Each week New Yorker readers are invited to submit captions for a cartoon, and a winner is picked from among these entries. We used a dataset from the contest for our experiments. Participants in our experiment are shown a cartoon along with two captions. Each participant's task is to pick the caption they think is the funnier of the two. This is repeated with many caption pairs and different participants. The objective of the learning algorithm is to determine which caption participants think is the funniest overall as quickly as possible (i.e., using as few comparative judgments as possible). In our experiments, we chose an arbitrary cartoon and n = 25 arbitrary captions from a curated set from the New Yorker dataset (the cartoon and all 25 captions can be found in [0]). The number of captions was limited to 25 primarily to keep the experimental dollar cost reasonable, but the NEXT system is capable of handling arbitrarily large numbers of captions (arms) and duels.

Dueling Bandit Algorithms

There are several notions of a "best" arm in the dueling bandit framework, including the Condorcet, Copeland, and Borda criteria. We focus on the Borda criterion in this experiment for two reasons. First, algorithms based on the Condorcet or Copeland criterion generally require sampling all 25-choose-2 = 300 possible pairs of arms/captions multiple times [4, 3]. Algorithms based on the Borda criterion do not necessarily require such exhaustive sampling, making them more attractive for large-scale problems [5]. Second, one can reduce dueling bandits with the Borda criterion to the standard multi-armed bandit problem using a scheme known as the Borda Reduction (BR) [5], allowing one to use a number of well-known and tested bandit algorithms.

The algorithms considered in our experiment are: random uniform sampling with BR, Successive Elimination with BR [6], UCB with BR [7], Thompson Sampling with BR [8], and Beat the Mean [9] which was originally designed for identifying the Condorcet winner (see [0] for more implementation details).

Experimental Setup and Results

We posted 1250 NEXT tasks to Mechanical Turk each of which asked a unique participant to make 25 comparison judgments for $0.15. For each comparative judgment, one of the five algorithms was chosen uniformly at random to select the caption pair and the participant's decision was used to update that algorithm only. Each algorithm ranked the captions in order of the empirical Borda score estimates, except the Beat the Mean algorithm, which used its modified Borda score [9]. To compare the quality of these results, we collected data in two different ways. First, we took union of the top-5 captions from each algorithm, resulting in 8 "top captions," and asked a different set of 1497 participants to vote for the funniest of these 8 (one vote per participant); we denote this the plurality vote ranking. The number of captions shown to each participant was limited to 8 for practical reasons (e.g., display, voting ease).

Caption	Plurality vote	Thompson	UCB	Successive Elim.	Random	Beat the Mean
My last of...	0.215 ± 0.013	0.638 ± 0.013	0.645 ± 0.017	0.640 ± 0.033	0.638 ± 0.031	0.663 ± 0.030
The last g...	0.171 ± 0.013	0.632 ± 0.017	0.653 ± 0.016	0.665 ± 0.033	0.678 ± 0.030	0.657 ± 0.031
The women'...	0.151 ± 0.013	0.619 ± 0.026	0.608 ± 0.023	0.532 ± 0.032	0.519 ± 0.030	0.492 ± 0.032
Do you eve...	0.121 ± 0.013	0.587 ± 0.027	0.534 ± 0.036	0.600 ± 0.030	0.578 ± 0.032	0.653 ± 0.033
I'm drowni...	0.118 ± 0.013	0.617 ± 0.018	0.623 ± 0.020	0.588 ± 0.032	0.594 ± 0.032	0.667 ± 0.031
Think of i...	0.087 ± 0.013	0.564 ± 0.031	0.500 ± 0.044	0.595 ± 0.032	0.640 ± 0.034	0.618 ± 0.033
They promi...	0.075 ± 0.013	0.620 ± 0.021	0.623 ± 0.021	0.592 ± 0.032	0.613 ± 0.029	0.632 ± 0.033
Want to ge...	0.061 ± 0.013	0.418 ± 0.061	0.536 ± 0.037	0.566 ± 0.031	0.621 ± 0.031	0.482 ± 0.032

The results of the experiment are summarized in the table. Each row corresponds to one of the 8 top captions and the columns correspond to different algorithms. Each table entry is the Borda score estimated by the corresponding algorithm, followed by a bound on its standard deviation. The bound is based on a Bernoulli model for the responses and is simply $\sqrt{\frac{1}{4k}}$, where k is the number of judgments collected for the corresponding caption (which depends on the algorithm). The relative ranking of the scores is what is relevant here, but the uncertainties given an indication of each algorithm's certainty of these scores. In the table, each algorithm's best guess at the "funniest" captions are highlighted in decreasing order with darker to lighter shades.

Overall, the predicted captions of the algorithms, which generally optimize for the Borda criterion, appear to be in agreement with the result of the plurality vote. One thing that should be emphasized is that the uncertainty (standard deviation) of the top arm scores of Thompson Sampling and UCB is about half the uncertainty observed for the top three arms of the other methods, which suggests that these algorithms can provide confident answers with 1/4 of the samples needed by other algorithms. This is the result of Thompson Sampling and UCB being more aggressive and adaptive early on, compared to the other methods, and therefore we

recommend them for applications of this sort. We conclude that Thompson Sampling and UCB perform best for this application and require significantly fewer samples than non-adaptive random sampling or other bandit algorithms. The results of a replication of this study can be found in [0], from which the same conclusions can be made.

DISCUSSION:

The entire NEXT system was designed with machine learning researchers and practitioners in mind rather than engineers with deep systems background. NEXT is almost completely written in Python, but algorithms can be implemented in any programming language and wrapped in a python wrapper. We elected to use a variety of startup scripts and Docker for deployment to automate the provisioning process and minimize configuration issues. Details on specific software packages used can be found in [0]

Many components of NEXT can be scaled to work in a distributed environment. For example, serving many (near) simultaneous 'getQuery' requests is straightforward; one can simply enlarge the pool of workers by launching additional slave machines and point them towards the web request queue, just like typical web-apps are scaled. This approach to scaling active learning has been studied rigorously [15]. Processing tasks such as data fitting and selection can also be accelerated using standard distributed platforms and machine learning packages (see next section).

A more challenging scaling issue arises in the learning process. Active learning algorithms update models sequentially as data are collected and the models guide the selection of new data. Recall that this serial process is handled by a model update queue. When a worker accepts a job from the queue, the queue is locked until that job is finished. The processing times required for model fitting and data selection introduce latencies that may reduce possible speedups afforded by active learning compared to passive learning (since the rate of 'getQuery' requests could exceed the processing rate of the learning algorithm).

If the number of algorithms running in parallel outnumber the number of workers dedicated to serving the synchronous locking model update queues, performance can be improved by adding more slave machines, and thus workers, to process the queues. Simulating a load with stress-tests and inspecting the provided dashboards on NEXT of CPU, memory, queue size, model-staleness, etc. makes deciding the number of machines for an expected load a straightforward task. An algorithm in NEXT could also bypass the locking synchronous queue by employing asynchronous schemes like [16] directly in processAnswer. This could speed up processing through parallelization, but could reduce active learning speedups since workers may overwrite the previous work of others.

ANTICIPATED IMPACT:

There have been some examples of deployed active learning with human feedback; for human perception [11, 17], interactive search and citation screening [18, 19], and in particular by research groups from industry, for web content personalization and contextual advertising [20, 21]. However, these remain special purpose implementations, while the proposed NEXT system provides a flexible and general-purpose active learning platform that is versatile enough to develop, test, and field any of these specific applications. Moreover, previous real-world deployments have been difficult to replicate. NEXT could have a profound effect on research reproducibility; it allows anyone to easily replicate past (and future) algorithm implementations, experiments, and applications.

There exist many sophisticated libraries and systems for performing machine learning at scale. Vowpal Wabbit [22], MLlib [23], Oryx [24] and GraphLab [25] are all excellent examples of state-of-the- art software systems designed to perform inference tasks like classification, regression, or clustering at enormous scale. Many of these systems are optimized for operating on a fixed, static dataset, making them incomparable to NEXT. But some, like Vowpal Wabbit have some active learning support. The difference between these systems and NEXT is that their goal was to design and implement the best possible algorithms for very specific tasks that will take the fullest advantage of each system's own capabilities. These systems provide great

libraries of machine learning tools, whereas NEXT is an experimental platform to develop, test, and compare active learning algorithms and to allow practitioners to easily use active learning methods for data collection. NEXT is a system designed to give machine learning researchers and applied data scientists the tools necessary to understand the best ways to adaptively collect data and to actively learn.

In the crowd-sourcing space there exist excellent tools like PsiTurk [1], AutoMan [2], and Crowd- Flower [26] that provide functionality to simplify various aspects of crowdsourcing, including automated task management and quality assurance controls. While successful in this aim, these crowd-programming libraries do not incorporate the necessary infrastructure for deploying active learning across participants or adaptive data acquisition strategies. NEXT provides a unique platform for developing active crowdsourcing capabilities and may play a role in optimizing the use of human- computational resources like those discussed in [27].

Finally, while systems like Oryx [24] and Velox [28] that leverage Apache Spark are made for deployment on the web and model serving, they support very specific types of models because they were optimized for that purpose, limiting their versatility and applicability. They were also built for an audience with a greater familiarity with systems and understandably prioritize computational performance over, for example, the human-time it might take a cognitive scientist

or active learning theorist to figure out how to actively crowd-source a large human-subject study using Amazon's Mechanical Turk.

At the time of this submission, NEXT has been used to ask humans hundreds of thousands of actively selected queries in ongoing cognitive science studies. Working closely with cognitive scientists who relied on the system for their research helped us make NEXT predictable, reliable, easy to use and, we believe, ready for everyone.

LDRD
LABORATORY DIRECTED RESEARCH & DEVELOPMENT

Communication - Publications

Non-stochastic Best Arm Identification and Hyperparameter Optimization, Kevin Jamieson and Ameet Talwalkar, *Submitted*, 2015.

Sparse Dueling Bandits, Kevin Jamieson, Sumeet Katariya, Atul Deshpande, and Robert Nowak, *AISTATS*, 2015.PDF

Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting, Kevin Jamieson and Robert Nowak, *CISS*, 2014. PDF

lil' UCB : An Optimal Exploration Algorithm for Multi-Armed Bandits, Kevin Jamieson, Matt Malloy, Robert Nowak, and Sebastien Bubeck, *COLT*, 2014. PDF

On Finding the Largest Mean Among Many, Kevin Jamieson, Matt Malloy, Robert Nowak, and Sebastien Bubeck,*Asilomar*, 2013. PDF

NEXT: A System for Real-World Development, Evaluation, and Application of Active Learning Kevin Jamieson, Lalit Jain, Chris Fernandez, Nick Glattard, Robert Nowak, *NIPS,* 2015 PDF

Communication – Presentations

NIPS 2014 Human Propelled Machine Learning Workshop - Invited talk (12/13/2014)□Introduction of the Next.Discovery computational framework and system that simplifies the deployment and evaluation of adaptive learning algorithms that use actual human feedback.

AMP Lab UC Berkeley - Invited talk (12/2/2014)□An overview of the methods and challenges of adaptive sampling algorithms.

COLT 2014 - Oral paper presentation (6/15/2014)□A presentation of the paper "lil' UCB : An Optimal Exploration Algorithm for Multi-Armed Bandits"

CISS 2014 - Invited talk (3/17/2014)□An overview of active ranking (see my publications) and how adaptive sampling with pairwise comparisons differs from function evaluations.

Communication – Open Source

The open-source project is on github: https://github.com/kgjamieson/NEXT with tutorials on how to use it.

CONCLUSION:

This research has provided considerable insights into the nature of active learning, producing numerous published results. In addition, these ideas and breakthroughs have been realized in the NEXT system, which has been deployed as an open source project.

The analyst-in-the loop approach to labeling data in production systems, while extremely useful for data analysis, can be prohibitively expensive. This research drastically reduces the cost by enabling real-world experimentation for active learning in a flexible, extensible framework. We anticipate this will lead to new understandings and breakthroughs, just as it has for passive learning.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.