Using Bandits to Efficiently Source Training Data

SI 699

Aaron Miller, Andrew Vande Guchte, Bryan Romas

Executive Summary

Our project focused on combining reinforcement learning, specifically multi-armed bandits, and crowdsourcing through Amazon Mechanical Turk to generate novel annotations for a class-unbalanced dataset. Although limited by a small budget we are able to show through synthetic experiments that our proposed system will converge on the optimal query in a query pool, thereby providing a more efficient and cost effective method for annotating a corpus, especially when the classes of interest are highly imbalanced.

Introduction & Motivation

Supervised learning models—including powerful deep learning systems—require labeled data from which to train. Sometimes, such labeled data is easy to come by because it is generated automatically throughout everyday life (such as the hashtags assigned to given tweets, or the number of people to visit a given website). However, for many tasks, no labeled data exists. A popular and efficient remedy for this problem is to use crowdsourced labeling to produce large quantities of labeled data quickly. Efficiency notwithstanding, crowdsourcing labels for training data carries its own host of issues. In particular, when the classes in the data that are being labeled are heavily imbalanced (i.e. the dataset is said to be "class-imbalanced", as per Jamison et al. [2014][1]), using crowdsourcing to label the dataset becomes much less efficient. This is because if data is sampled randomly to be labeled, an extraordinarily large corpus of data will need to be labeled to get a large enough sample of each class that a model could learn the fundamental patterns for all of the unbalanced classes of the data. Embarking on such an endeavor can quickly become infeasible if monetary or temporal resources are limited.

In our project, we propose and implement a system of reinforcement learning using multi-armed bandits that optimizes the selection of unlabeled data to present to Amazon Mechanical Turk (AMT, a crowdsourcing tool) workers to maximize the number of rare classes that are labeled. The goal is to produce enough labeled instances of the rare class of data that a supervised model can be trained to identify both the common and rare classes of data. Note that a related issue that is not fully discussed in this project is the discovery of new formulations in which the rare classes could manifest (see the Past Work section). Our project does not have the monetary resources to meaningfully implement a reinforcement learning algorithm that identifies new morphologies of rare classes.

The specific task that we set out to accomplish is identifying "directive" statements (i.e. where someone is telling someone else to do something) within a transcription data. Such a task is useful for personal-assistant artificial intelligence, e.g. to identify tasks that need to be completed from a business meeting. Directive statements, while present in meeting transcripts are extremely rare (often only a few instances of directives in an entire meeting transcript). We, therefore, use our reinforcement learning system to identify and label training data for the classifier that we built to identify directive statements. Since the scope of our training data is small, we did not expect our final model to perform well. As such, we also built a synthetic data pipeline that generates labelers and different ways to query a synthetic dataset to show the viability of our reinforcement learning approach to identifying rare classes from unlabeled data.

In the following report, we first briefly summarize previous work that has inspired our approach to reinforcement learning. Following that, we describe the data we are using and the annotation

[1] Jamison, E., & Gurevych, I. (2014). Needle in a Haystack: Reducing the Costs of Annotating Rare-Class Instances in Imbalanced Datasets. *PACLIC*.

rules that we gave to the AMT labelers. Then, we discuss our methods and the details of our multi-armed-bandit approach. Finally, we describe our experiments, both synthetic and actual, and offer some conclusions and takeaways from this project.

Past Work

Reinforcement Learning

Reinforcement learning is a rich area of study focused on designing a system that learns from its actions to realize some goal, usually framed as “maximizing a reward”. As such, the ultimate challenge of reinforcement learning is optimizing the tradeoff between exploration and exploitation, put succinctly:

> To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future.[2]

Our system is focused on determining which combination of queries in the query pool will result in the maximum reward. Our main focus, therefore, is on providing a variety of queries for the pool and establishing a viable and valuable reward for the system to pursue.

ReQ-ReC

In a study titled “ReQ-ReC: high recall retrieval with query pooling and interactive classification”[3], the authors outline what they call a “double-loop retrieval system” that has served as a primary inspiration for our work. On a high-level, ReQ-ReC (ReQuery-ReClassify) iteratively searches a corpus of documents by selecting a subset of documents through a query, obtaining labels from a human annotator for the selected documents, re-training a classifier on all documents labeled up to that point, and eventually refining the query so it moves closer to related documents and away from non-related documents in a vector space. In the study, the authors explored several versions of ReQ-ReC, however, the best performing version based on R-precision and mean average precision was one called “Diverse Active” which selected lower-ranked, positive instances in an attempt to expand the search space into promising new areas that might otherwise be overlooked if only high-ranked instances were used. While Req-Rec was focused on providing high recall and high precision query results through a search service, for instance in medical or legal research, our focus is on discovering

---

[2] *Sutton, R.S., & Barto, A.G. (1988). Reinforcement Learning: An Introduction. IEEE Transactions on Neural Networks, 16, 285-286.*

[3] Li, C., Wang, Y., Resnick, P., & Mei, Q. (2014). ReQ-ReC: high recall retrieval with query pooling and interactive classification. *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval.*

rare instances of a class in an unlabeled and possibly unexplored corpus. Instead of relying on the ranked results returned by a query for evaluation purposes, we rely on the agreement between human annotators to approximate, in a fashion, the accuracy of the query.

Multiple Queries as Bandit Arms

The multi-armed bandit problem is a reinforcement learning problem. In these types of problems, a limited set of resources can be allocated between alternative choices in a way that maximizes their expected gain. In our case, our limited labeling resources were allocated across a select number of queries of our data to maximize the number of positively-identified directive statements. Each choice's properties are only partially known at the time of allocation and become better understood as resources are allocated to different choices over time. In Li et al. (2016)[4], researchers developed a document retrieval system where multiple queries were kept simultaneously active and assigned turns to those queries using a multi-armed bandit. As with any multi-armed bandit problem, a reward is given from a probability distribution specific to the query currently being "pulled". The objective is to maximize the sum of rewards earned by "pulling" the arm with the highest expected value. Specific to our work, we utilized this system to explore what combination of queries would return the best results. The focus of Li et al. (2016) was to develop a better document retrieval system, as existing retrieval systems rely on a single active query to pull documents. Using a multi-arm bandit algorithm enabled them to develop a system that explored which queries return more relevant results. Two methods were proposed by Li et al. to maximize query efficiency. The first corresponded to a scenario where the result space was well-known and well-defined, such that the requester knows exactly what kind of documents they are looking for. In this scenario, the requester may use "query pools," or collections of pre-defined queries that the requester suspects should return good results. In this case, the multi-armed bandit algorithm simply explores the "query pool" to determine which query provides the best results. The other scenario was when the requester does not know the full parameters of the query space. In such a scenario, the requester simply starts with a single query and new queries are automatically generated from the results of that query. New queries are formed from the positive samples in the result set that were ranked lowly, thus hopefully producing a new query that captures something that the old query was missing. Then, the multi-armed bandits are performed on the ever-expanding query set.

Finding Rare Classes

One of the focal points of this research is the challenge of identifying rare-classes in unlabelled data. Surprisingly, the problem of document discovery coupled with classification has received little attention despite its importance and broad relevance. In 2013 a group of researchers addressed this challenge using a joint-active learning approach. In the twenty-fifth edition of

---

[4] Li, C., Wang, Y., Resnick, P., & Mei, Q. (2016). Multiple Queries as Bandit Arms. *Proceedings of the 25th international ACM SIGIR conference on Information and Knowledge Management.*

IEEE Transactions on Knowledge and Data Engineering[5], researchers Hospedales et. al address the challenges of joint discovery and classification through active learning methodology which adaptively accounts for the success of both tasks. Moreover, they developed a generative-discriminative model because in their words "*generative models naturally provide good discovery criteria and discriminative models naturally provide good classifier learning criteria.*"

The novelty of Hospedales et. al's research is in its dual-objective purpose (discovery and classification) and how it adaptively selects criteria. More commonly, "exploration" will be preferred while there are easily discoverable classes which have not been found, and "exploitation" is used to refine decision boundaries. However, Hospedales et. al discovered that this isn't a universal best approach for all datasets. Their model can adapt to situations where it is useful to return to searching for the rarest classes after learning to classify easier ones, or where one query is consistently the best.

In defining a reward function, the team developed an algorithm which rewarded the discovery of new classes to jointly optimize classification and discovery. The first part of the algorithm rewards discovery of a new class, before rewarding an increase in multiclass entropy after labeling a point, i. They also added a weighting parameter, α , which is the weighting prior for discovery vs. classification. This approach was tested on nine datasets of varying composition, consistently adapting query criteria as more data was obtained, and outperforming other contemporary approaches.

Datasets & Data Collection

CHiME-5

The Computational Hearing in Multisource Environments (CHiME) dataset is part of a challenge coordinated through the University of Sheffield, United Kingdom. The challenge is focused on signal processing and machine listening however it proved compatible with our purpose as it featured fairly rare instances of directives. We explored the frequency of bigrams and trigrams (Figure 1) within the data set that could signal the presence of a directive within a sentence. We believe that the frequency of "directive bigrams" reinforces the rarity of directives, while also providing evidence for their existence within the dataset.

[5] T. M. Hospedales, S. Gong and T. Xiang, "Finding Rare Classes: Active Learning with Generative and Discriminative Models," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 2, pp. 374-386, Feb. 2013.

“Directive” Bigram	Count
“Will you”	4
“Do that”	600
“Take that”	68
“Could you”	144

Fig. 1: Counts of “directive bigrams” present within the CHIME-5 data set.

The transcribed dinner party conversations are inherently messy with any given fragment including background noise, overlapping conversations, and natural disfluencies. Several typical examples are provided below. Any directives in the examples are underlined.

“Use one hand and then use the other hand for the other one. K. Cuz you don't wanna get the [inaudible 0:10:47.25] Yeah yeah yeah Fold it over. Call it a day. Mm. [noise] Ah What? It's.”

“Brownie? No Brownie brownies. Aw [laughs] Okay Yes I have uh I have my pope-mobile. [noise] Can I get the power to get Jessica? [laughs] [laughs] Um When you level up. K. Actually [laughs] d- d- doubles.”

“I'll help. [inaudible 0:03:56.23] It's gonna soak it up. Oh [inaudible 0:03:58.31] you can just say it. But get a Hold on. Get a thingamajig. Here. Yeah All right. Cuz like Cuz I don't think Ibs has.”

The data set is approximately 57 MB and consists of 197,410 sentences of varying lengths. As seen in Figure 4, more than 20 percent of all sentences are ten or fewer words.

Sentence Parts-Of-Speech	Frequency
['INTJ', 'PUNCT']	20,916
['PUNCT', 'NOUN', 'PUNCT']	19,284
['INTJ']	4,068
['INTJ', 'PUNCT', 'INTJ', 'PUNCT']	1,902
['PROPN', 'PUNCT']	1,614
['ADV', 'PUNCT']	1,038
['ADJ', 'PUNCT']	936
['INTJ', 'INTJ', 'PUNCT']	924
['PUNCT', 'ADJ', 'NUM', 'PUNCT']	840
['PRON', 'PUNCT']	832
['VERB', 'PUNCT']	810
['PUNCT', 'ADJ', 'PROPN', 'PUNCT']	794
['NOUN', 'PUNCT']	772
['PROPN']	734
['PRON', 'VERB', 'PUNCT']	622

Fig. 2: Frequency of the 15 most common parts-of-speech combinations within the CHIME-5 data set.

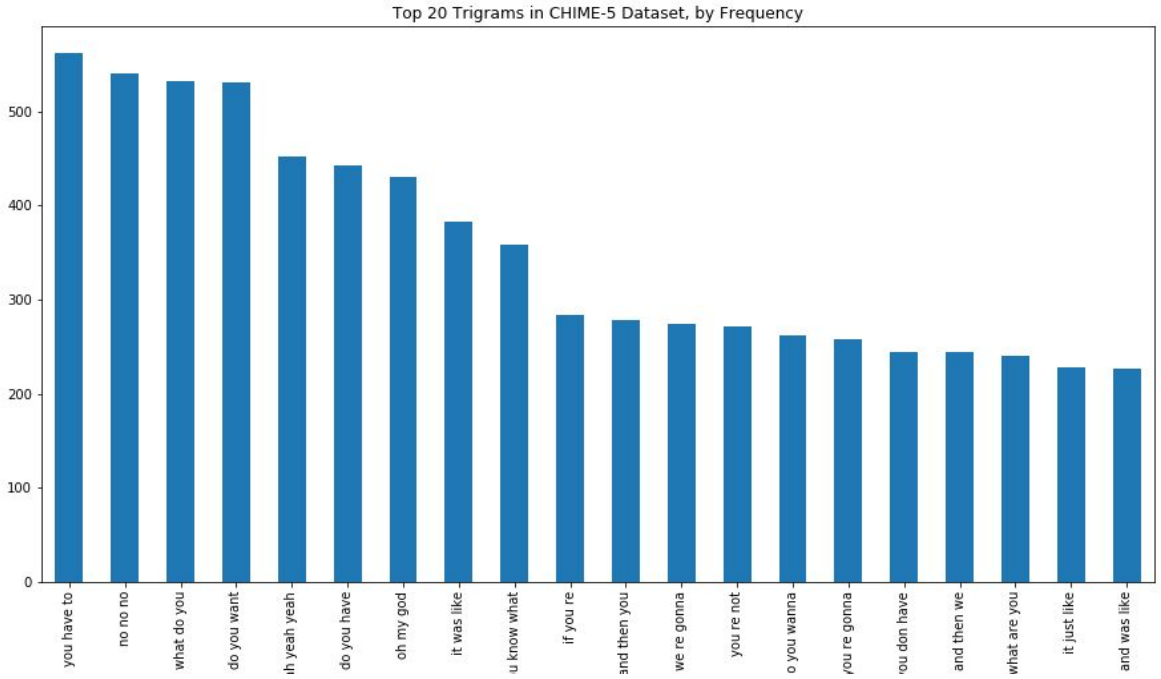

Fig. 3: The top 20 trigrams found in the CHIME-5 data set, ranked by the frequency of each trigram.

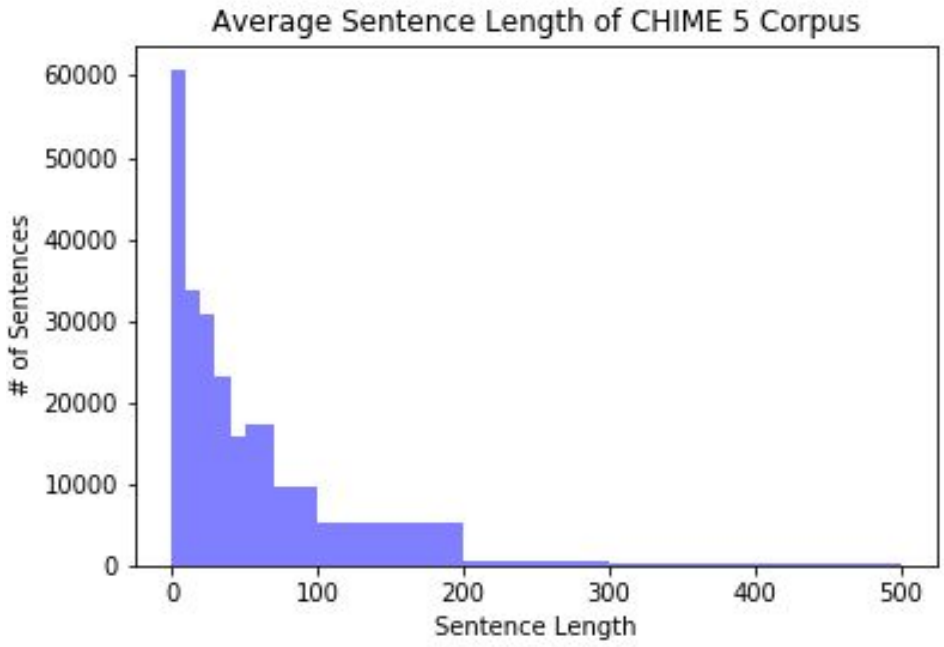

Fig. 4: A binned distribution of sentence length of text in the CHIME-5 data set. More than 20 percent of sentences are ten words or less.

Annotation Rules

Establishing an annotation guideline that clarified the task at hand and codified the intuition and rules behind what constitutes a directive was foundational to our work. To that end, we iteratively documented a set of rules and manually labeled five-sentence subsets from our corpus. The format and content of our annotation guide were largely inspired by Pustejovsky and Stubbs[6]. A relevant concept that helped solidify our annotation rules was that of "grammatical mood" which is how a speaker signals their intentions and beliefs. Our search for directives in a corpus was effectively a search for instances of a speaker expressing the imperative (primary use), prohibitive, directive, or conditional mood. A challenge here, and one of the reasons for our low initial agreement, is other moods could be construed as indicating a directive, primarily the hortative and imperative (secondary use focused on inviting, permitting, or wishing). Key content from our annotation guide was provided to AMT workers to help ensure consistency.

Our most recent version of the annotation guide is included in the appendix.

[6]Pustejovsky, J., & Stubbs, A. (2013). *Natural language annotation for machine learning*. Beijing: O'Reilly.

Amazon Mechanical Turk

As previously mentioned, Amazon Mechanical Turk (AMT) is a crowdsourcing tool used to perform on-demand tasks (such as identifying the emotion expressed in a picture). Our group asked AMT workers to undergo the same labeling task which our research group has gone through. The process is as follows:

1. A worker is given five sentences from transcribed text data (this will be approximately five sentences; some of the sentences are too long due to transcription error so we use word count instead of sentence count).
2. The worker is asked to identify if a directive was given in those five sentences.
3. If a directive was identified, the user is asked to provide which words indicated the directive.

Instructions to AMT workers included a summary, detailed instructions, and examples of positive and negative classes. The summary section informs AMT workers that the goal of the task is to determine if a group of sentences contains a directive or not. Detailed instructions provide greater depth to the task and reflect language from our annotation guide, including:

- "*A directive is any statement that includes a reference to a person/group of people, a command/instruction/order, and a goal. Both the person/group of people and the goal may be implied.*"
- "*As a note to help with your labeling: If you ignore niceties such as "please" or "thanks", does the sentence seem like a question or a statement? If it seems like a question, it is not a directive.*"

Methods

Our approach to identifying and labeling as many rare classes as possible marries the document retrieval optimization techniques presented in Li et al. (2014, 2016) with the AMT crowdsourcing platform. At a high level, we employ multi-armed bandits to iteratively query a corpus of unlabeled documents using a pool of search rules defined in our "query pool." The returned documents from the selected query are passed to AMT, and the total number of rare classes that were identified during crowd-sourced labeling is used as a reward function for the multi-armed bandit to determine the next query. After many iterations, we can use the labeled data gathered from AMT to train a supervised classifier to identify the rare classes automatically. Fig. 5 describes our system schematically.

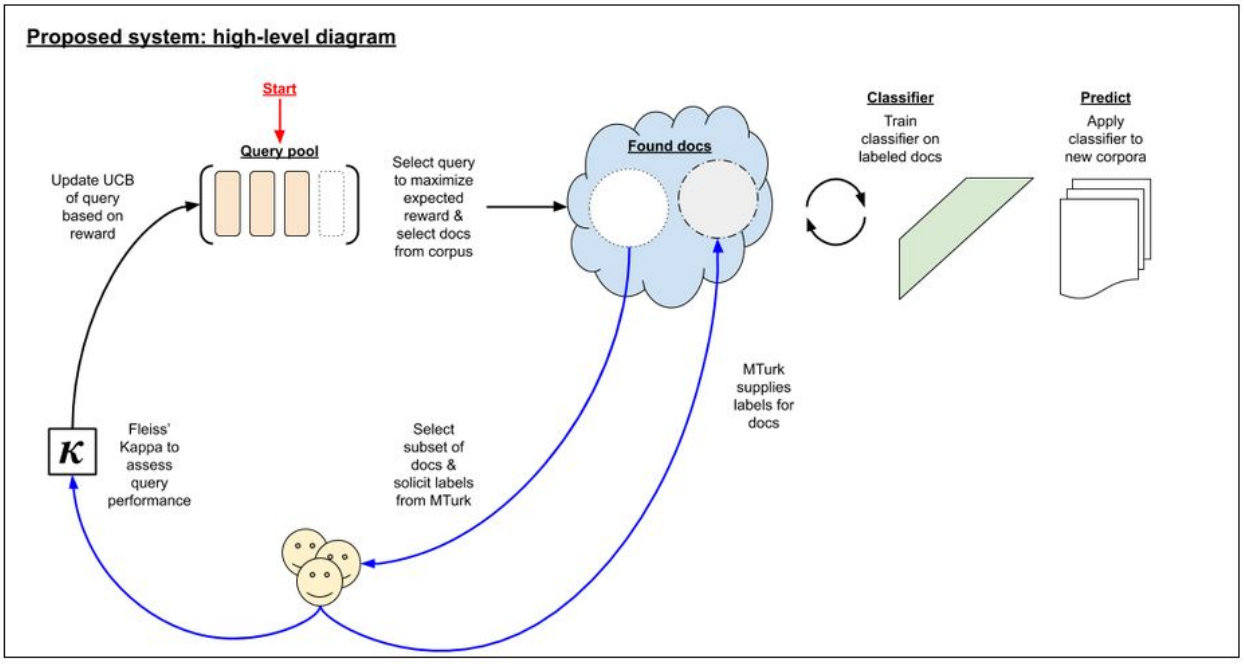

Fig. 5: A high-level overview of our reinforcement learning system to efficiently source balanced, labeled, training data for a classifier that is intended to classify unbalanced data.

Multi-Armed Bandit Algorithm

Our multi-armed bandit algorithm is similar to what has been proposed in both Li et al. (2014) and Li et al. (2016), with a couple of key differences. We intend to use only pre-defined queries (i.e. queries that we have come up with on our own, not anything that was automatically generated) akin to the “query pool” method of Li et al. (2016).

With our method, we subsample our dataset given one of the queries from our query pool. That subsample is sent to AMT for labeling. The results of that query are assessed, and then the next query is chosen via the commonly-used “Upper-Confidence Bound” (UCB) method (Auer et al., 2002[7]). We chose this method because the UCB method is guaranteed to converge to minimum regret given sufficient “plays” of the algorithm. The UCB is calculated via:

$$UCB = X_i + \frac{log(t)}{N_i}$$

where t is the number of times the algorithm has been performed, N_i is the number of times query i has been selected, and X_i is the expected reward of query i (calculated as the mean reward from all times that query i has been selected).

[7] Auer, P., Cesa-Bianchi, N. & Fischer, P. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 235–256 (2002). https://doi.org/10.1023/A:1013689704352

In both Li et al. [2014] and Li et al. [2016], the result sets from the queries are ranked in terms of textual relevance as a reward. However, we will take advantage of an alternative approach to determine our reward, because our target throughout the querying is not a specific topic, but rather a language construct (a “directive”). Our reward parameter will be the number of positive samples labeled from the result set from the query. If a particular document is labeled by different labelers as both a directive and not a directive, then we will consider the reward from that document R_d to be

$$R_d = \frac{L_p}{L_p + L_n}$$

where L_p is the number of times the document was labeled as a directive and L_n is the number of times the document was labeled as not a directive.

Synthetic Pipeline

In order to test the validity of our system, we performed a proof-of-concept simulation. Synthetic “levers” were created, each returning 100 “documents” to be labeled. With real data, we don’t know the ground-truth of unlabeled documents, however, in this simulation we programmed each “lever” to return documents from the following uniform distribution:

$$X \sim U(\mp)$$

where μ is the mean of the distribution, α is half of the width of the distribution, and X is the fraction of documents that belong to a rare class. For our simulations, α was set at 0.1, but μ varied among the “levers” so that there would be some pattern for the multi-armed-bandit to identify and an ideal “lever” for it to converge toward. Similarly, synthetic “labelers” were created with various accuracies for each “labeler.” An accuracy of 1.0 would always label each document the same as its ground truth. An accuracy of 0.5, because of the binary nature of the labeling task, corresponded to a “labeler” that was randomly assigning labels to documents.

Query Pool

For the production run of our pipeline with real data (i.e. to retrieve training data for our directive-detection classifier), we designated our query pool to consist of four queries. In a much larger production (one where we were not as severely financially limited for our crowdsourcing), we may have created a method of developing new queries from the resulting query pool, however, given our limited resources, we chose to only use four. The first three queries use regular expressions to find specific word patterns. If there was a match, the sentence and its surrounding context sentences (approximately five sentences in total) were retrieved and saved in a file for the labelers to assess. We had a query for “will”, “go”, and “get”. The words selected

as foci for the regular expressions were chosen because they seemed common from visual inspection of directive statements that we hand-labeled.

The final query retrieved sentences that contained a participle immediately before a verb. Similarly, this part-of-speech pairing was discovered to be common amongst directives when they were visually inspected.

Classifier

Using the annotations from AMT we created training and test data sets to assess the performance of a “downstream” binary classification task. In order to rectify the, potentially, divergent annotation results we labeled a sample as containing a directive if at least 50% of the annotators assigned it the positive label, otherwise we assigned it the “not directive” label. After controlling for gold samples used across multiple experiments we obtained 284 ground truth samples.

We generated additional features by creating indicator columns which simply indicated whether the text for the sample contained certain tokens or not. For instance, whether the text contained a question mark or “could”. We also created features based on readability and the Flesch reading ease score using the textstat library. Finally, we generated a tf-idf statistic based on the text in the samples.

We created DummyClassifier, Logistic Regression, and SVM classifiers and performed hyperparameter tuning using grid search. Our best performing model was the Logistic Regression with an F1 score of 0.41. However, this was not much better than our baseline, the DummyClassifier. A summary of model performances is below.

Model	**Precision**	**Recall**	**F1**
Dummy classifier with stratified strategy	0.35	0.43	0.38
Logistic Regression	0.54	0.33	0.41
SVM with linear kernel	0.47	0.33	0.39

Table 1: A summary of the results from our directive classifiers.

We did not expect the models to perform well given the small amount of “ground truth” available, the inconsistency of the annotations, and the fact that convergence of purely synthetic data, as shown below, requires hundreds of experiments.

Results

Simulations/Synthetic Data

A suite of simulations was performed to test our approach. For these simulations, four "levers" were used—**a**, **b**, **c**, and **d**—with μ of 0.2, 0.3, 0.4, and 0.5, respectively. Each simulation utilized six synthetic "labelers." As a baseline to compare against, a "all_random" simulation was performed with all of the "labelers" labeling randomly. In addition, a "two_0.9" simulation was performed with four of the "labelers" labeling randomly and two of them labeling with 0.9 accuracy. Many other experiments were performed, but will not be discussed explicitly here. A summary of the experiments performed is in Table 2.

Name	Labeler Acc	At Least 1	At Least 5	Confidence Ratio
all_random	0.5, 0.5, 0.5, 0.5, 0.5, 0.5	49,219	5,568	0.11
all_0.6	0.6, 0.6, 0.6, 0.6, 0.6, 0.6	48,621	6,359	0.13
all_0.7	0.7, 0.7, 0.7, 0.7, 0.7, 0.7	46,909	10,334	0.22
all_0.8	0.8, 0.8, 0.8, 0.8, 0.8, 0.8	43,264	16,099	0.37
all_0.9	0.9, 0.9, 0.9, 0.9, 0.9, 0.9	36,524	21,771	0.60
one_0.9	0.5, 0.5, 0.5, 0.5, 0.5, 0.9	49,073	4,964	0.10
two_0.9	0.5, 0.5, 0.5, 0.5, 0.9, 0.9,	48,680	6,435	0.13
three_0.9	0.5, 0.5, 0.5, 0.9, 0.9, 0.9,	47,679	9,228	0.19
four_0.9	0.5, 0.5, 0.9, 0.9, 0.9, 0.9	45,706	13,657	0.30
five_0.9	0.5, 0.9, 0.9, 0.9, 0.9, 0.9	42,455	18,343	0.43

Table 2: Results from the suite of simulations performed as a proof-of-concept test of our system. "Name" is the moniker of the simulation, "Labeler Acc" are the accuracies of the synthetic labelers in the experiment, "At Least 1" is the number of documents that received at least one rare-classification, "At Least 5" is the same but for five rare-classifications, and "Confidence Ratio" is the ratio of "At Least 1" and "At Least 5."

The "all_random" experiment never converged on a specific "lever," but rather employed each of **a**, **b**, **c**, and **d** equally throughout the simulation. Fig. 6 shows the percentage of each "lever" being selected by the algorithm over time. Over the course of the simulation, 5,568 "documents"

received a rare-classification from at least 5 “labelers.” In contrast, 49,219 “documents” received a rare-classification from at least 1 “labeler.” Unsurprisingly, this means that only 11.3% of documents that received at least one classification as rare had at least 5 of the 6 “labelers” agree that it was a rare class (this ratio is known as the “confidence ratio” for the rest of this section).

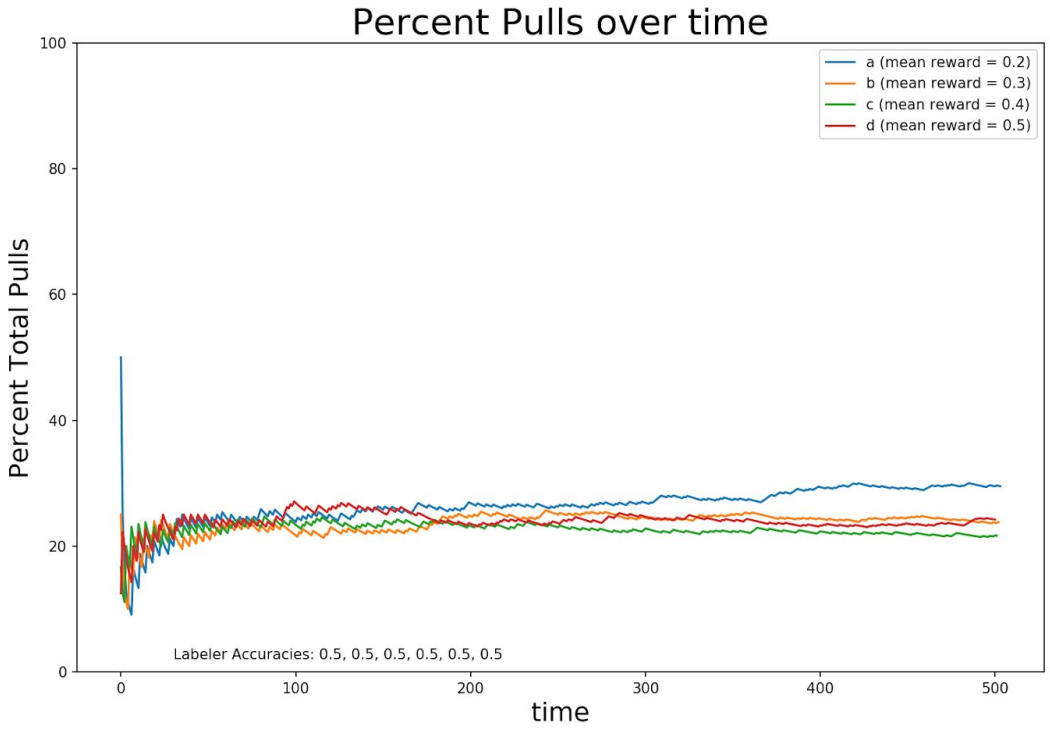

Fig. 6: Percentage of total “pulls” of each “lever” over time in the “all_random” experiment.

In the “two_0.9” experiment, the algorithm quickly converges on **d** as the lever that provides the highest reward. By the end of the simulation, nearly 70% of the “pulls” were from lever **d** (Fig. 7). The number of “documents” that received 5 or more rare-classifications was 6,435, and 1 or more was 48,680 for a confidence ratio of 13.2%. Despite the quick convergence of the algorithm, since most of the “labelers” were still labeling randomly, the confidence ratio was still low.

In order to increase the confidence ratio, the average accuracy of the “labelers” must be increased. Fig. 8 shows the relationship between confidence ratio and average accuracy of the synthetic labelers. This relationship is intuitive, however, it has meaningful implications for when applying our system to real data: If the confidence ratio or agreement between the crowd-sourced labelers is low, it is reasonable to expect that the accuracy of the judgments by the labelers as a collective is also low. Therefore, agreement between the crowd-sourced labelers is critical to ensure that the documents being labeled as directives are truly directives.

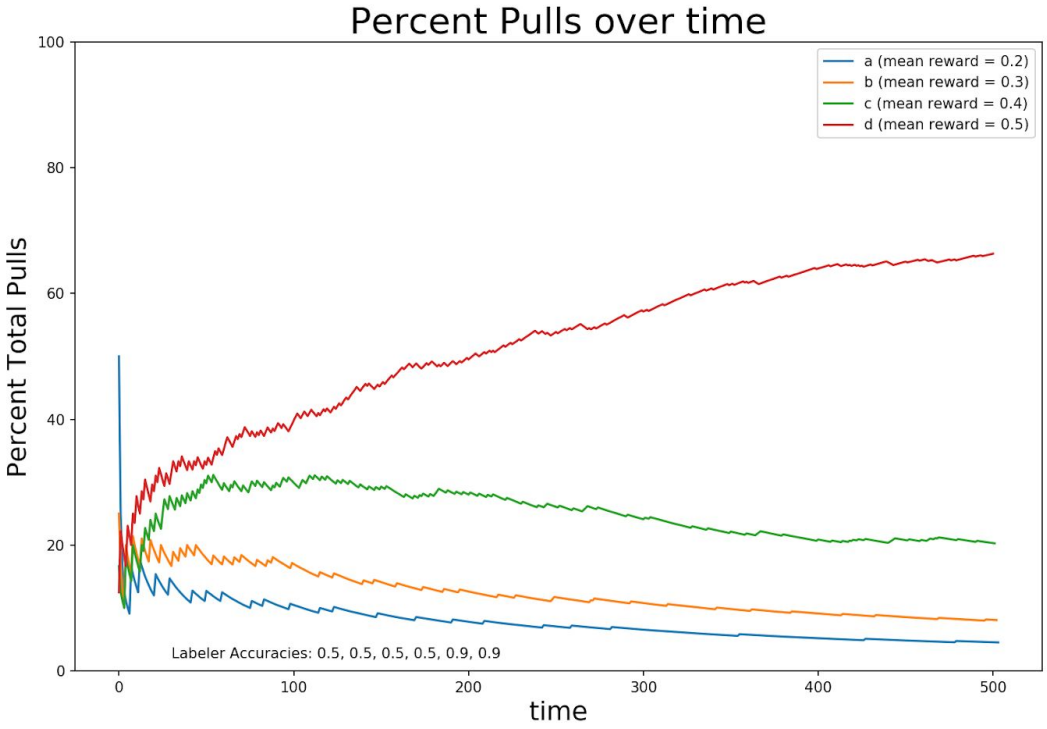

Fig. 7: Percentage of total “pulls” of each “lever” over time in the “two_0.9” experiment.

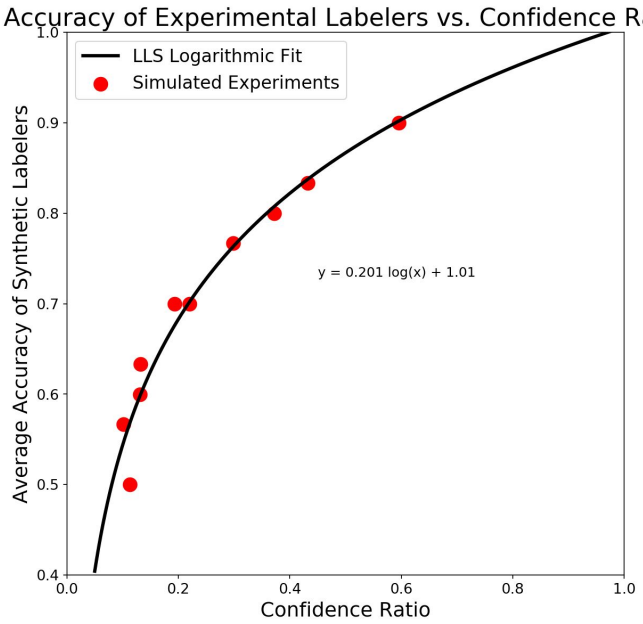

Fig. 8: Average accuracy of the synthetic labelers vs. the confidence ratio for the 10 synthetic experiments performed in this study.

Real data

Based on the results of our 14 experiments we can roughly assess agreement within and across the experiments by calculating the simple average agreement between annotators. While assessing agreement using Fleiss' Kappa would be ideal and more rigorous, the results from AMT were inconsistent in terms of the number of annotators per HIT and made calculating Fleiss' Kappa problematic. This approach to assessing agreement is not rigorous for two related reasons: the number of annotators who labeled a HIT is not taken into consideration and the role of chance is not considered. For instance, the probability of two annotators agreeing on a binary labeling task by chance is 0.5*0.5 = 0.25. The figure below shows the average agreement per each experiment (indicated by a unique file name). For each sample in the experiment the percentage of labelers agreeing on the label is calculated and then the average across all samples is calculated. The results here seem promising but, as mentioned above, neither the number of annotators nor the role of chance agreement is included in this calculation. This challenge highlights how important it is to maintain a consistent number of annotators across labeling tasks.

File	Proportion of agreement
will_mturk_results_3_filtered.csv	0.857
get_mturk_results_1_filtered.csv	0.844
get_mturk_results_3_filtered.csv	0.783
will_mturk_results_5_filtered.csv	0.843
go_mturk_results_1_filtered.csv	0.899
get_mturk_results_2_filtered.csv	0.955
pos_mturk_results_2_filtered.csv	0.933
will_mturk_results_1_filtered.csv	0.847
go_mturk_results_2_filtered.csv	0.795
go_mturk_results_4_filtered.csv	0.827
will_mturk_results_4_filtered.csv	0.773
go_mturk_results_3_filtered.csv	0.770
pos_mturk_results_3_filtered.csv	0.913
POS_mturk_results_4_filtered.csv	0.849

Fig. 9: Table showing simple average agreement between annotators across experiments.

Conclusions

One issue with our approach is that, while better than randomly selecting samples for labeling, it still requires potentially hundreds of iterations in order for it to converge on an optimal solution. This can vary somewhat depending on how imbalanced the classes are, the quality of the labelers, and the accuracy of the queries, however it is still an area for potential improvement. A success of our approach is that the overall system is flexible enough to be applied to almost any labeling task where a query pool, a reward, and a classification or regression task can be defined.

Given our limited resources, we were unable to perform multiple iterations of re-assessing the annotation guide after seeing how well the AMT labelers agree on documents being directives. If we had more time, we would use the agreement between the AMT labelers to inform how well our annotation guide defines our task. If necessary, we would then update the annotation guide until we could achieve reasonable agreement between the workers. From there, we would use AMT to retrieve 10,000 - 100,000 labeled documents, as opposed to the 1,800 we did here.

We would also like to refine our approach with AMT, or another crowdsourcing platform, to ensure the results we recieve can better conform to Fleiss' Kappa measurement standards. In our current approach with AMT, we are not able to ensure that we have an equal amount of labelers that are also labeling an equal amount of HITs. Ideally, we could easily control the number of labelers for each query, and ensure we receive the same amount of labeled HITs across labelers to calculate Fleiss' Kappa and other metrics of interest. At this time, AMT cannot guarantee those results in our implementation.

As a final thought, this project taught us how to handle situations where the data is not ideal and the resources even less so. We had access to unlabeled data that did not explicitly fit the original task we outlined, very few resources to crowdsource labels with, and a task that was challenging to define well enough for others to understand it easily. Therefore, we needed to work through parts of a project pipeline that are often skipped in class. Throughout this experience, we learned how to develop an annotation guide, build multi-armed bandits from scratch, and implement simulations to test the validity of our solution.

Appendix

Annotation Guide

The following guide was created but not shared with AMT workers due to lack of space on the platform. It was primarily referenced by us in our own labeling. A related abbreviated version was provided to AMT workers.

The goal of this task is to determine whether a text contains a directive or not. If there is a directive in the text you will be asked to copy and paste it into a designated space as well. A directive is any statement that includes a reference to a person/group of people, a command/instruction/order, and a goal. Both the person/group of people and the goal may be implied. Directives must be longer than one word.

The examples are from transcribed conversations and multiple conversations may take place simultaneously within an example, and multiple people may be speaking to each other. As a note to help with your labeling: If you ignore niceties such as "please" or "thanks", does the sentence seem like a question or a statement? If it seems like a question, it is not a directive. If it seems like a statement, it may be either a directive or not, depending on the context. If you are still unsure, it is likely not a directive.

"They will eat salad." -> NOT A DIRECTIVE

"Let's eat salad." -> NOT A DIRECTIVE

"You should eat salad." -> NOT A DIRECTIVE

"If there's any left, try the salad." -> DIRECTIVE

"Matthew and Lilly, eat your salad." -> DIRECTIVE

"Will you pass the lemonade?" -> NOT A DIRECTIVE

"Can you pass the lemonade?" -> NOT A DIRECTIVE

"Pass the lemonade, please." -> DIRECTIVE

“Pass the lemonade.” -> DIRECTIVE

“Try it if you want.” -> NOT A DIRECTIVE

“If you want, try it.” -> NOT A DIRECTIVE

“Will you please try it.” -> NOT A DIRECTIVE

“We’ll try it.” -> NOT A DIRECTIVE

“Try it.” -> DIRECTIVE