Task Clustering on Mechanical Turk Tasks

Elena Agapie Juho Kim

ABSTRACT

Mechanical Turk is a crowdsourcing platform used to outsource simple human intelligence tasks (HITs) to human workers (Turkers) at a low monetary reward. Understanding what types of tasks are posted and completed is an important question surrounding the landscape of the marketplace. This paper applies clustering techniques to identify task groups that share common properties. The results show that certain features are important for clustering the data, such as the pricing or how long a hit is available. The LDA algorithm identifies clusters that have the potential of showing interesting topics in the data. These identified task clusters in Mechanical Turk can contribute to building more robust and accurate predictors for output quality and automatic controllers for task parameters.

INTRODUCTION

Recently, many crowdsourcing platforms have been introduced to mediate human computation tasks between requesters (employers in the traditional market context) and workers (employees). Amazon's Mechanical Turk is the largest and arguably the most popular marketplace. Human computation tasks commonly found on Mechanical Turk are simple micro-size tasks that are easy to solve for humans but are difficult for even complex computer programs. Example human intelligence tasks (HITs) include simple classification, information gathering, object recognition, and transcription.

A typical task life-cycle on Mechanical Turk starts with a requester posting a HIT group with preset task parameters. The parameters include the reward amount given to workers, the number of tasks available (or Turkers needed) for a certain task type, how long the task will be available for, task description, result type, etc. Once the task is posted, anonymous workers search for and pick up the HIT they want to work on, and their results are automatically stored and sent to the requester.

As with any other functioning markets, one challenge is to predict the outcome of tasks from the requester-specified task parameters. Accurate prediction of the result quality and the rate of task completion helps requesters optimize their actions to allocate their budget more efficiently. Furthermore, better prediction systems can guide the design of an optimization agent dynamically adjusting the parameters. The applications with higher problem complexity call for basic building blocks, and one such critical component is how the tasks are clustered and distributed.

This paper attempts to address the following questions regarding the tasks posted on Mechanical Turk:

- What kind of tasks are posted and completed?
- What are notable clusters of tasks and how do they differ?
- What is the task distribution in terms of reward, complexity, and size?

We first present related work that attempts to understand the task structure on Mechanical Turk and build agents that automatically design tasks. Then clustering algorithms used in the experiment are introduced, followed by the experiment design and results. We discuss the effect of algorithms on the results and ways to improve the clustering performance. The paper concludes with potential implications of having such task clusters and future work.

RELATED WORK

Ipeirotis [6] analyzed various dimensions of the tasks posted on Mechanical Turk with a large database of crawled tasks. His task analysis includes the top requesters, their representative task types, and frequent HIT keywords ranked by various metrics. One objective of this work is to understand how the task clustering results extend and verify the analytic findings of Ipeirotis.

Researchers have developed different ways to reach the goal of modeling and optimizing the requester's behavior. Using simple regression models, Huang et al. [5] observe that a task takes longer to be completed if the amount of work assigned per dollar paid is higher. They also note that the quality prediction is difficult because variance is high even for the same task design, although they report only preliminary results in automatic design with simple image labeling tasks.

Turkit [8] is a toolkit for organizing task completion cycles. It attempts to realize purely algorithmic human computation by embedding program models that use voting and iteration. Dai et al. [2] propose TurKontrol, an autonomous agent controlling the workflow with a decision-theoretic optimization technique. The agent makes decisions based on the expected utilities of each action, taking into account the trade-off between quality and cost for certain types of workflows. Our clustering approach can provide a basis for deciding which workflow to model in order to support useful work types.

QUESTIONS / HYPOTHESES

The main question that has guided our work is determining clusters of data for the Mechanical Turk HITs. More specifically, we focus on the following questions:

- what are strong predictors of a cluster? (task complexity, certain keywords, etc.)?

- how can we identify topics from the hits content?

We approached one crawl at a time, and plan to follow up with analyzing how parameters that cluster data might differ in time, or what other features we can add to better cluster the data. We will have a discussion about these ramifications to our questions.

TASK CLUSTERING

Clustering is an unsupervised learning method used to detect clusters within a data set that share common properties [7]. From multiple patterns embedded in the data, the user starts by specifying a set of features that are believed to influence the formation of clusters. The features represent patterns in different scales and types, which in turn are used to compute the distance between any two patterns. Clustering algorithms then make the clustering decision based on a similarity (or distance) measure between two distinct data points or patterns; if they are similar enough, they are considered to belong to the same group.

K-means

K-means [9] is the simple clustering method employing a squared error criterion. Starting with k randomly assigned clusters, the algorithm assigns each pattern to the closest cluster. It finds the centroid for each cluster and computes squared error between each point belonging to a cluster and the centroid of that cluster. Reassignment of clusters continues until a convergence criterion (no revision to assignment or decrease in squared error is below a threshold) is met.

X-means

While simple, K-means is limited in that scalability is difficult to achieve and the user should provide the number of clusters, K. To resolve such drawbacks of K-means, X-means [10] improves K-means by making local splitting decisions after each K-means run. The Bayesian Information Criterion (BIC) guides the decision of the cluster locations and the number of clusters. The gain is both in running time and quality.

EM

Another way of looking at the clustering problem is to think of patterns in the data set to be drawn from a mixture of distributions. Then clusters can be identified by solving for parameters for each distribution, and also the number of such clusters can be inferred. The most common method is based on learning a mixture of Gaussians with continuous variables. This problem becomes parameter estimation where a general-purpose maximum likelihood algorithm such as Expectation Maximization (EM) is suitable for [3]. EM begins with an initial, random estimate of parameters and measures likelihood of being drawn from certain components for each pattern. The computed likelihood indicates where a given point should belong to, and a density estimation for each cluster follows easily.

LDA

In LDA [1], each item of a collection of documents is modeled as a mixture over an underlying set of topics. The documents are represented as a random mixture of latent topics, and topics are characterized by a distribution over words. LDA assumes that the words are generated by topics, through fixed conditional distributions. The topics are considered exchangeable in a document. The words of a document are drawn from a multinomial distribution. In this mixture model, each document is generated by choosing a topic *z*, and then generating N words independently from a conditional multinomial distribution. Over the entire corpus, the word distributions are considered representations of topics and each document can exhibit several topics.

LDA uses the approximate empirical Bayes estimates for the LDA model. Given a corpus of documents

D= $\{w_1, \ldots, w_{M\}}$ we wish to find the parameters *a* and *b* that maximize the likelihood of the data:

$$l(a, b) = \sum_{d=1}^{M} \log p(w_d | a, b)$$

LDA alternates variational EM procedure to maximize the lower bound with respect to variational parameters γ and ϕ, that model a Dirichlet, respectively a multinomial distribution. These paramenters are used to represent the variational distribution and we won't go into details about them. For fixed values of the variational parameters maximize the lower bound with respect to the model parameters *a* and *b*, which generate the topic of the documents in the corpus.

For the task clustering problem presented in this paper, we will apply all of the clustering algorithms mentioned above to better explore the relative strength and weakness of each.

METHOD

The data set we used in our experiment is collected through Mechanical Turk Tracker [http://mturk-tracker.com/]. The tracker crawls the entire marketplace every hour, taking snapshots of all available tasks at that time. Our data contains tasks posted between January 2009 and April 2010. Each HIT entry in the data consists of the following information:

- **Crawl related**: the time of the crawl, and whether successfully completed

- **HIT status related**: group id, Turker qualification, reward, time allotted, number of HITs available, requester name, requester id, page number in the task list, position in the task list page

- **HIT time related**: expiration date, time allotted once the Turker accepts the HIT, time started, time finished, duration

- **HIT content related**: title, description, keyword, the actual HTML content of the HIT page.

K-means, X-means, and EM

Our first experiment was designed to understand the overall task cluster structure of the entire database. The sample data set is based on the 100 randomly picked crawl points across more than 30,000 collected ones. Because there are many factors (e.g. time of day, day of week, month) that could potentially lead to biased sampling, we decided to pick random crawl points. This set consisted of the total of 90,490 HIT groups (average 904.9 HITs per each crawl) from 1,307 unique requesters. For the purpose of clustering, only numerical attributes were applied: string distance for nominal features does not inform us of anything significant (e.g. requester name A is closer to B or C?). Euclidean distance was used as the distance function. The features used are as follows:

- **HITs available**: the number of HITs posted by the requester of this HIT group

- **Page number**: the page number this HIT is displayed in the Turker's interface

- **Reward**: money paid to Turkers for work

- **HIT content length**: word count of the HIT content page

- **duration**: how long it took for all the HITs in this HIT group to be completed by Turkers.

We used the Weka [4] package, an open-source collection of machine learning algorithms for K-means, X-means, and EM.

LDA

We used the LDA implementation that accompanied Blei's paper [1]. The code is available at http://www.cs.princeton.edu/~blei/lda-c/. The data used included content of the actual HITS posted on MTurk. This included HTML code. This code was parsed to eliminate HTML code, but even after cleaning content, remainders of the scripting were still present. We will discuss this the effect of the state of the data on the topics discovered by LDA in the results. Each document in LDA is equivalent with the content of a HIT. The format of the data that LDA used included counts of words per document and numbers of distinct words per document. We used the HITs available in one crawl of MTurk, with the intention to illustrate the topics available at one point in time.

RESULTS

K-means

Simple K-means is prone to overfitting as there is no simple way to determine the number of clusters. The following graph shows that squared errors diminish as K increases. This does not necessarily mean that there are more clusters for the data set. That is why we turn to X-means to make sense out of the clustering results.

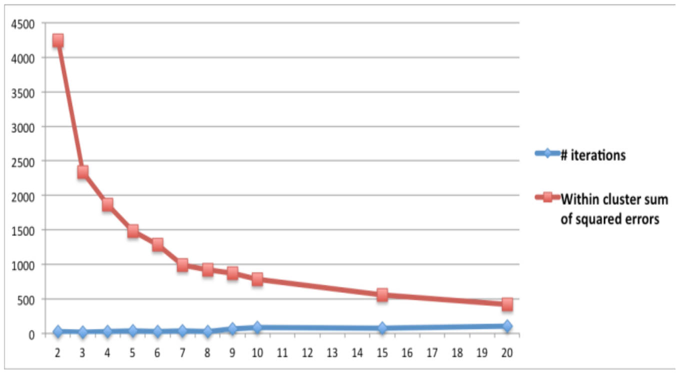

Figure 1. K-means performance (X-axis: the number of clusters K, Y-axis: squared error and the number of iterations)

X-means

	Instances	# HITs	page no.	reward	content len.	completion time
Cluster 0	29710 (33%)	1.04	85.26	0.35	2418.18	483.71
Cluster 1	38438 (42%)	163.37	31.65	0.27	1498.06	429.8
Cluster 2	6952 (8%)	1.42	70.52	0.18	2074.69	5808.96
Cluster 3	15390 (17%)	44.16	22.57	0.09	1466.72	5103.51
Total / Mean (std. dev.)	90490	76.77 (1414.62)	50.72 (32.09)	0.26 (0.86)	1840.66 (4247.61)	1650.39 (2324.98)

Figure 2. X-means clusters with mean values for each attribute

Distortion: 16384.12, BIC-Value : 243417.84.

As shown in Figure 2, X-means returns 4 clusters: high availability + high payment, high availability + low payment, low availability + high payment, and low availability + low payment. Cluster 0 is a high-paying job with low availability that has long descriptions and many input fields to fill in, therefore more work. Although it appears very late in the search page, Turkers complete it reasonably fast. Cluster 2 is similar to Cluster 0 in that its availability is low, it appears late in the HIT list, and it has long content. But it pays half the money of Cluster 0, therefore taking more than 10 times longer than Cluster 0 to complete. This finding is in support of literature that showed positive correlation between the amount of reward and task completion time. Cluster 1 includes 42% of all the tasks available in the marketplace, characterized as highly available and accessible with shorter content. The only difference with Cluster 3 is that it pays three times more, therefore finishes more than 10 times faster. The clusters suggest that there is a clear difference between high and low availability tasks. Furthermore, 2~3 times more investment accelerates the completion rate by 10+ times.

EM

The EM algorithm in our experiment decides the number of clusters by 10-fold cross validation. This significantly increases the running time of EM. EM only returned two clusters as shown below.

EM returns only two clusters, where Cluster 0 has 76% of all the tasks. The result of EM is difficult to analyze because there are conflating evidence of its clustering. For instance, when reward is higher for Cluster 1, it shows even higher completion time. Figure 3 shows the clustering results.

LDA

We ran LDA with a fixed input value for *a*=1 and attempted to identify the effect of varying the number of topics on the results we obtained. The algorithm keeps a parameter limiting the number of EM maximum iterations. There is a high variety of task contents on MTurk. Some of the hits require text tasks where entire paragraphs might be present without being connected to what the hit requires.

We expect to identify types of tasks, along with the topics yielded throughout the corpus of documents. We ran the algorithm on 1349 documents, containing 23365 distinct terms. The topic clusters that we

	Instances	# HITs	page no.	reward	content len.	completion time
Cluster 0	68465 (76%)	1.227	58.84	0.20	1225.44	1429.71
Cluster 1	22025 (24%)	309.43	25.7021	0.43	3735.61	2330.09
Total	90490	76.77	50.72	0.26	1840.66	1650.39

Figure 3. EM clusters with mean values for each attribute. Log likelihood: -27.0855

identified have a lot of overlapping words, however we could identify clusters that might reflect topics of relevance for our data. Some examples are below. Topic 010 might suggest a topic that includes mostly MTurk instructions and key words that are characteristic to how the instructions would be phrased (declining, mturk, reviewers, describe, verify, done, etc). Topic 014 might be characteristic to instructions that require turkers to perform text related tasks (fiction, mark, slides, etc):

- topic 010: *done, Problems*, sitting, Rockefellers, *declining, largely*, murray, *VERIFY, MTurk, fiction*, 06_42, *describe, Reviewers*, inappropriately, repairing

- topic 014: *done*, 06_42, Rockefellers, 96818, *largely*, Butt, *open*), *VERIFY, Problems, Reviewers*, id, 4GB, *workspace*, sitting, *Difference, fiction, mark, slides*

However other topics might make less sense in natural language, although some relevant topic might be associated with the following cluster:

- topic 011: *newness*, murray, 506, natural, 248, *concise*-, porcelain, 254-8530, *referral, done*, high-rise), *VERIFY, initiates*, TXSAQJ4M8W4ZYYN0TK3Z, Hobby

We ran LDA for generating 5, 10, 15, 20 and 40 topics. From our observations, with more topics we had more variety in the words associated with the topics. A lot of the words repeat throughout topics but the 20 and 40 topic run generated topic groups like the ones presented above, which might cluster objects of interest. This is however to be explored in future work. As an immediate follow up to this we plan to experiment with different values for the other starting parameters, such as starting with a different *a* or with something other than a random distribution. We have limited our analysis to the basic steps to first identify if LDA might be a method that yields relevant cluster. We believe that after some more proper data cleaning the results would be more representative to the corpus.

DISCUSSION

Algorithms

K-means is a very simple model that has advantages in isolated and compact clusters. While its simplicity is attractive, it has clear limitations when applied to noisy and complex data as in our case. EM displays better responses to complexity. Determining the number of clusters is a non-trivial problem in our case because the space is not well-defined. X-means and EM present useful information by optimizing the number of clusters. LDA is also an algorithm in which the topics found do not necessarily capture a precise structure of the data. One measure that we will use for LDA is the perplexity (a measure of the mean per word likelihood), which is a measure of how well the model will generalize. The expected results are that as the number of topics increases, the data will be overfit and it will reflect in the perplexity value decreasing. This is a measure through which we can compare LDA to other probabilistic models.

Improving the Clustering Performance

There are many ways to improve the clustering performance. More attributes will yield clusters and topics that make more sense. More sophisticated HTML and text parsing and numerically encoding such text-based features can improve the accuracy of clustering. Also, attributes related to the Turker's work quality can make the clustering process much more informed. In addition to only the task duration, features such as task approval rate or correctness can be added to the parameters. Finally, Mechanical Turk data as is contains a lot of noise and outliers. We only filtered out null or empty group ID and obviously invalid values in other fields, but smarter ways to detect and remove noise and outliers are desired.

For LDA, several reasons might have contributed to the fact that a lot of our clusters did not seem representative of a topic: the data still needs more cleaning - similar words (correction, corrections are distinct now but could be processed and represent the same word); the words that do not belong to a dictionary could be thrown out, since we might get a lot of noise from words that have html left overs; the words that occurred only once in the entire corpus could be thrown out of the analysis. Also, a very wide variety of information is posted on MTurk and it might actually be difficult to determine what are the topics that appear frequently. The Blei et al. paper [1] dealt with a subset of 16000 documents, 28414 distinct words. Although the number of documents is higher, we could argue that our corpus is comparable in topics because of the variety of HITS posted on MTurk.

Evaluation

As with many other unsupervised learning methods, an objective evaluation of clustering is often difficult because there is no single correct answer. Clustering is an exploratory process of understanding the data set, hugely dependent on the practitioner's expertise level, intention, and hypothesis. The same applies to the Mechanical Turk tasks; there is no ground truth in how task clusters are structured. One of the things we have touched upon earlier is to compare the performance of different clustering algorithms.

Implications and Future Work

As discussed earlier, automatic task design is a huge challenge in crowdsourcing. The clustering approach can enhance the robustness of statistical predictors by allowing different parameter estimation for each cluster, which means more customized parameters for each cluster of tasks. Another implication of clustering is that we can infer a more informed distribution of the tasks available in the marketplace. It can lead to a better understanding of market dynamics, suggesting how to improve the design of failing tasks with unreasonable design parameters. 'Debugging' the design failure would be easy because the cluster information will tease apart the problem of certain clusters as opposed to others.

In addition to this, we plan to explore additional features that might characterize our data. This could include characteristics that might tell us how soliciting a task is: length of the task, number of fields that need to be filled in (text boxes, buttons, etc), how much of the task is dominated by images or text, etc. We believe this would add an additional dimension to the features that cluster the given data.

Another interesting venue of future work is to develop ways to integrate topics retrieved from LDA with clusters from X-means. This is challenging but very useful in practice from both algorithm and analytic perspectives.

REFERENCES

[1] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research* 3: *pp.*993–1022.

[2] P. Dai, Mausam and D. S. Weld (2010). Decision-Theoretic Control of Crowd-Sourced Workflows. Twenty-Fourth National Conference on Artificial Intelligence (AAAI-10). Atlanta, GA, July 2010.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin (1977). Maximum Likelihood from Incomplete Data via the EM algorithm. Journal of the Royal Statistical Society, Series B, vol. 39, 1:1-38.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten (2009). The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[5] E. Huang, H. Zhang, D. C. Parkes, K. Z. Gajos, and Y. Chen (2010). Toward automatic task design: A progress report. In Proceedings of KDD-HCOMP'10. ACM.

[6] P. Ipeirotis (2011). Analyzing the Amazon Mechanical Turk Marketplace, ACM XRDS (Crossroads).

[7] A. K. Jain, M. N. Murty, and P. J. Flynn (1999). Data clustering: a review. ACM Comput. Surv. 31, 3 (September 1999), 264-323.

[8] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller (2010). TurKit: Human Computation Algorithms on Mechanical Turk. UIST 2010.

[9] J. B. MacQueen (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297.

[10] D. Pelleg and A. W. Moore (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. Seventeenth International Conference on Machine Learning, 727-734.