Alloy: Clustering with Crowds and Computation

Joseph Chee Chang, Aniket Kittur, Nathan Hahn
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
{josephcc, nkittur, nhahn}@cs.cmu.edu

ABSTRACT

Crowdsourced clustering approaches present a promising way to harness deep semantic knowledge for clustering complex information. However, existing approaches have difficulties supporting the global context needed for workers to generate meaningful categories, and are costly because all items require human judgments. We introduce Alloy, a hybrid approach that combines the richness of human judgments with the power of machine algorithms. Alloy supports greater global context through a new "*sample and search*" crowd pattern which changes the crowd's task from classifying a fixed subset of items to actively sampling and querying the entire dataset. It also improves efficiency through a two phase process in which crowds provide examples to help a machine cluster the head of the distribution, then classify low-confidence examples in the tail. To accomplish this, Alloy introduces a modular "*cast and gather*" approach which leverages a machine learning backbone to stitch together different types of judgment tasks.

Author Keywords

Computer Supported Cooperative Work (CSCW); World Wide Web and Hypermedia; Database access / Information Retrieval; Empirical Methods, Quantitative

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

Clustering, or pulling out the patterns or themes among documents, is a fundamental way of organizing information and is widely applicable to contexts ranging from web search (clustering pages) to academic research (clustering articles) to consumer decision making (clustering product reviews) [18]. For example, a researcher may try to pull out the key research topics in a field for a literature review, or a Wikipedia editor may try to understand the common topics of discussion about a page in order to avoid or address previous conflicts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI 2016, May 7–12, 2016, San Jose, California, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-3362-7/16/05 ...$15.00.
http://dx.doi.org/10.1145/2858036.2858411

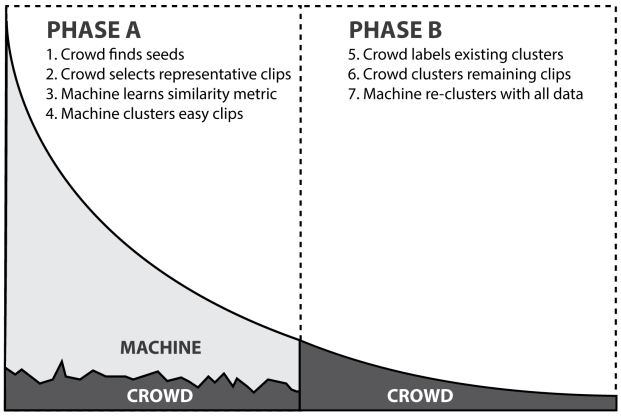

Figure 1. A conceptual overview of the system. In the first phase, crowd workers identify seed clips to train a machine learning model, which is used to classify the "head" of the distribution. In the second phase, crowd workers classify the more difficult items in the "tail". A machine learning backbone provides a consistent way to connect worker judgments in different phases.

Doing so involves complex cognitive processing requiring an understanding of how concepts are related to each other and learning the meaningful differences among them [2, 24, 29].

Computational tools such as machine learning have made great strides in automating the clustering process [4, 10, 6]. However, a lack of semantic understanding to recognize the important differences between clusters leaves the difficult task of identifying meaningful concepts to the human analyst [11]. This reflects an inherent advantage for humans over machines for the complex problem of understanding unstructured data beyond merely measuring surface similarity, and a corresponding opportunity for research in combining human and computational judgments to process complex information [14, 25, 16].

One such promising avenue of research harnesses the power of crowds to identify categories and cluster rich textual data. Crowdsourcing approaches such as Cascade, Deluge, and Crowd Synthesis [9, 5, 1] have demonstrated the power of splitting up rich, complex datasets into small chunks which can be distributed across many human coders. However, all of these approaches must grapple with a fundamental problem: since each human coder is seeing only a small part of the whole dataset, a lack of global context can lead to incoherent results. For example, if the items sampled are too similar, the worker might create overly fine-grained clusters. On the

other hand, if the items sampled are too dissimilar, the worker might create overly broad clusters. Clusters found in many worker segmentation sets may give rise to redundant clusters, while clusters whose items are sparsely split among segmentation sets may never be realized at all. As an example, [1] cite redundancies in Cascade's top level clusters having both "green" and "seafoam green", "blue" and "aqua", as well as the encompassing category of "pastels". While Crowd Synthesis used an iterative approach to address these redundancy problems, it trades this off with lowered robustness as issues with early workers' categories can cascade throughout subsequent workers' judgments. This suggests the design space of approaches for crowd clustering may be being critically limited by the assumption of splitting up the dataset into small, fixed pieces that prevent workers from gaining a more global context.

Another challenge with current crowd clustering approaches is that using human judgments to label each piece of data is costly and inefficient. Deluge addresses some issues with efficiency, improving on Cascade by reducing the number of human judgments elicited as the rate of new category generation slows [9]. However, these crowd clustering algorithms still require human judgments for every item, which is costly. In the real world data often follows a long-tailed distribution in which much of the data is captured by a small number of categories in the head of the distribution [35]. For such data in which many items in the head of the distribution are likely to be highly similar, once humans have identified the meaningful categories and representative examples it would be more efficient if a machine could classify the remaining items in those categories. A danger with such an approach is that the sparse categories in the tail of the distribution with few examples may be difficult to train a machine to recognize, and so human judgments may have another important role in "cleaning up" low frequency categories.

This paper describes Alloy, a hybrid approach to text clustering that combines the richness of human semantic judgments with the power of machine algorithms. Alloy improves on previous crowd clustering approaches in two ways. First, it supports better global context through a new "*sample and search*" crowd pattern which changes the crowd's task from classifying a fixed subset of items to actively sampling and querying the entire dataset. Second, it improves efficiency using initial crowd judgments to help a machine learning algorithm cluster high-confidence unlabeled items in the head of the distribution (prominent categories), and then uses later crowd judgments to improve the quality of machine clustering by covering the tail of the distribution (edge cases and smaller categories). To achieve these benefits, Alloy introduces a novel modular approach we call "*cast and gather*" which employs a machine learning backbone to stitch together different types of crowd judgment tasks. While we provide a particular instantiation of the cast and gather approach here (with a hierarchical clustering backbone which gathers three types of crowd tasks, or "casts"), the general framework for modularizing multiple types of human judgments with a common machine-based backbone may inspire application to other contexts as well.

RELATED WORK

Document and short text classification are well researched topics in natural language processing and machine learning. With enough labeled training data, state-of-the-art algorithms can often produce good results that are useful in real world applications. Yet building such systems often requires expert analysis of specific datasets both to manually design an organization scheme and to manually label a large set of documents as training data. Unsupervised approaches, or clustering, aim to discover structures on-demand and without expert preparation [17, 15, 32]. While these data mining approaches may discover dimensions (features) that provide a good separation of the dataset, the inferred categories can be difficult for a human to interpret, and many of them may not capture the most meaningful or useful structure in a domain due to high dimensionality or sparseness in the word vector space [2, 24]. To deal with these issues, researchers have explored ways to automatically discover topical keywords that can help identify useful categories in unstructured data such as TF-IDF, latent semanic analysis, and latent Dirichlet allocation [28, 20, 12, 4]. However, even with these improvements, automatic methods often still perform poorly, especially when the number of document is small, the lengths of the documents are short, or when the information is sparse.

More recently, researchers have begun to use crowds to organize datasets without predefined categories. Cascade [9] attempts to address abstraction and sampling problems by first having multiple workers generate categories for each item and then later having workers choose between them. By providing limited context to each worker (8 items or 1 item with 5 categories), it suffers from categories that can have varying levels of specificity. As a follow up study, Deluge [5] produces comparable results, but with significantly lower cost by optimizing its workflow using machine algorithms. In another line of research, Crowd Synthesis [1] showed that providing more context by simply showing more items can lead to significant better categories, suggesting that global context is one of the key elements for crowd clustering algorithms. In general, most current systems provide context by showing a small sample of items, hoping that they captures the distribution of information in the larger dataset. We propose an alternative approach that builds up workers' mental models by asking them to repeatedly sample for new items, identify discriminative keywords, and search the dataset for similar items, taking advantage of people's capacity of information foraging [30].

A complementary set of approaches to crowd clustering research has focused on addressing the scaling problem through computation, applying approaches such as partial clustering [37], learning similarity metrics through triad-wise comparisons [33], or using matrix completion to reduce the number of labels needed from workers [38]. While these approaches have shown to be powerful on simple information such as images or travel tips, synthesizing more complex information can be difficult without providing novice crowdworkers with richer context or opportunities to deeply process the data.

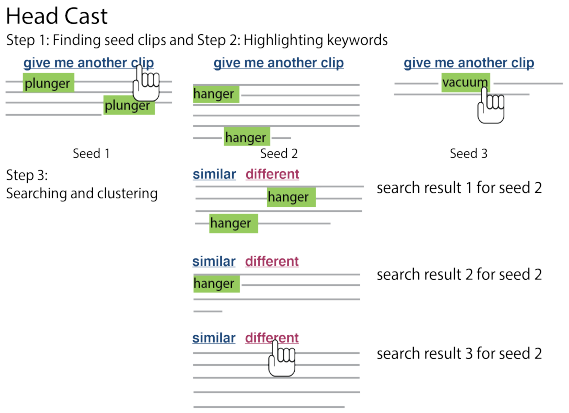

Figure 2. The interface and steps of the Head Cast HIT.

ALLOY

The Alloy system clusters a collection of clips, or short text descriptions (Figure 3), using a machine learning backbone that gathers various judgments from human workers. In our terminology, each human task is a "Cast" for human judgements which are then "Gathered" together with the machine learning backbone. Alloy enables Casts (here, crowdworker tasks) of different types and in different orders to be fused together by calling a Gather after each one. In each Cast stages, arbitrary number of workers can be hired for better robustness or lower cost. In this paper we present three types of Casts with different purposes as well as one type of Gather. At a high level, the "Head Cast" is aimed at finding common categories in the head of the distribution, while the "Tail Cast" is aimed at classifying categories in the tail of the distribution for which machine clustering has low confidence. The "Merge Cast" aims to clean up existing categories by combining highly similar categories. We also describe a Gather Backbone that fuses the judgements from multiple crowdworkers, and connects multiple casts to form complete workflows. For ease of exposition we introduce each component in the context of a typical workflow: the Head Cast, the Gather, the Merge Cast, and the Tail Cast.

The Head Cast

The Head Cast aims to identify salient keywords to uncover the most common categories in the head of the distribution. Doing so involves challenges in providing workers sufficient context to know what a good category is, and also in how to structure their work process in order to train a machine learning algorithm to take over the classification of categories based on human-identified seeds and keywords. Previous studies show that presenting multiple items from a collection can help provide context to human workers [13], increasing the likelihood of obtaining better clusters. However, it can be difficult to determine how much context is sufficient and how to produce a good sample that captures the distribution of information of the whole dataset. Therefore, we introduce a new crowd-pattern we call "*sample and search*" for providing global context through active sampling and searching with keywords. We ask crowdworkers to identify coherent categories by presenting with four random items, but allowing them to replace each item by random sampling from the entire dataset until they are confident that the items will be in different categories in the final output. This requirement gives them the motivation to build up better global understanding of the dataset through repeated sampling. After obtaining the four seed items, we ask crowdworkers to identify keywords in each clips to search for related items in the dataset. This process takes advantage of people's capacity of finding new information [30]. To create a familiar experience, we allow the workers to freely change their query terms and update the search results in real time. This way they can refine their searches based on the results, the same way as when conducting online information foraging tasks [19]. As shown in Figure 2, the Head Cast HIT interface consists of three steps:

1. **Finding seeds**: Four random seed clips are presented to each crowdworker. Over each clip, there is a button that allows them to replace the clip with another random clip from the dataset. They are then asked to replace any clips that are too similar to the other seed clips. The workers repeatedly replace the seed clips until the four clips at hand belong to four different answer categories.
2. **Highlighting keywords**: The crowdworker is then instructed to highlight one to three unique keywords from each of the four seed clips that best identify their topics.
3. **Search and label**: For each seed clip, we automatically search for similar clips from the entire corpus based on the highlighted keywords and TF-IDF cosine similarity. The crowdworker is asked to label the top nine search results as *similar* to or *different* from their seed clips.

In Step 1, the crowdworkers need some understanding of the global context before they can confidently judge that the seeds belong to different categories in the final output. Previous work usually address this problem by presenting multiple items to each crowdworker, in hopes of sampling both similar and dissimilar items to give some sense of the global context. In reality it could be difficult to judge how many items is sufficient for different datasets, and overly small size could lead to bad samples that are unrepresentative of the global distribution. We took a different approach by presenting fewer items at first, but allowing workers to replace the seeds with random clips from the dataset. This provide them both the mechanism and motivation to explore the dataset until they have enough context to find good seed clips.

The intuition behind Step 2 is that people are already familiar with picking out good keywords for searching documents related to a concept via their online information seeking experiences. In addition, requiring them to highlight unique keywords in the seeds first, further ensures that they are familiar with the concepts in the seed clips, before they search for similar items. In Step 3, the crowdworkers can still change and refine their highlights from Step 2, and the system will refresh the search results in realtime. This gives the crowdworkers both the motivation and mechanism to extract better keywords that lead to better search results to label. In Figure 3, we show two example clips from the datasets collected using the two questions: *How do I get my tomato plants to produce more tomatoes?* and *What does a planet need to*

support life? The highlighted words in each clips are the keywords selected by one of the crowdworkers, showing that workers are finding useful words for classification.

Tomato seedlings will need either strong, direct sunlight or 14-18 hours under grow lights. Place the young plants only a couple of inches from florescent grow lights. Plant your tomatoes outside in the sunniest part of your vegetable plot.
In its astrobiology roadmap, NASA has defined the principal habitability criteria as "extended regions of liquid water, conditions favourable for the assembly of complex organic molecules, and energy sources to sustain metabolism

Figure 3. Example clips from two datasets with crowd keywords.

To learn a similarity function between clips, we use the crowd labels and keywords to train a classifier that predict how likely two clips to be labeled as similar. Although the judgments from workers via the HIT interface about which clips go together provide valuable training information, we need to leverage these judgments to bootstrap similarity judgments for the clips that they did not label and to resolve potentially conflicting or partial category judgments. To do so we trained an SVM classifier in real-time to identify the set of keywords that are most indicative of categories and predict whether two clips in the dataset belonged to the same cluster. The training events are all possible pairwise combinations of clips in the clusters obtained with the HIT interface, which may include both positive (similar) and negative (different). The feature dimensions are all the keywords highlighted by the crowdworkers, and the value of each dimension is the product of the number of times that keyword occurred in the two clips. In general, the keywords labeled by the crowdworkers contain little irrelevant information compared to all words in the clips, but there could still be some highlighted words that are not indicative of a category. For example, one crowdworker worked on the dataset for "*How do I unclog my bathtub drain?*" labeled "*use*", "*a*", and "*plunger*" as three keywords. Even though *plunger* is a very indicative feature for clustering this dataset, the first two highlighted words seem too general to be useful. Using a linear kernel to estimate the weights for the different dimensions (i.e., keywords) seems well suited for our purpose [7, 36]. Further, if the same keyword is used by different crowdworkers but lead to very different labels, the linear SVM model will give lower weight to the corresponding dimention and thus lower the effects of keywords that are less indicative of the categories. We use LIBSVM which implements a variant of Platt scaling to estimate probability [27, 31]. The overall intuition is that the SVM classifier is doing a form of feature selection, weighting those words in clips that could maximally distinguish clips amongst clusters.

In a preliminary experiment, we tested using all words in the clips as features to train the SVM model. The intuition is machine algorithms might do a better job at identifying keywords that can outperform keywords identified by crowdworkers. However, the results show that using all words as features did not yield better results, and having much higher feature dimensions increases the training time significantly.

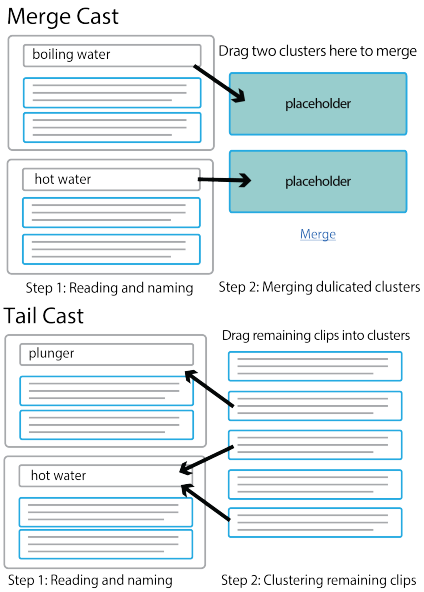

Figure 4. The HITs for Merge Cast: Naming and merging existing clusters and Tail Cast: Clustering remaining clips.

Finally, with the probability output of the SVM model as a similarity function between clips and a stopping threshold of 0.5 probability, we use a hierarchical clustering algorithm that serves as the Gather Backbone to capture head clusters.

Gather Backbone: Hierarchical Clustering

Using a multiple-stage approach with different types of microtasks can make it difficult to fuse together the different crowd judgements to form a coherent result. A key element to our approach in *casting* for category judgments in different ways is that we have a unifying mechanism to *gather* them back together. For example, throughout our process we cast for human category judgments in very different ways, including having people identify seed clusters (the Head Cast), merge duplicated categories (the Merge Cast), and classify the tail of the distribution (the Tail Cast). Instead of creating ad-hoc links between these judgments we propose using a unifying gathering mechanism composed of a machine learning backbone which translates the different *casted* judgments into similarity strengths used as the basis of clustering. We believe this *Cast and Gather* pattern may be useful as a way to conceptualize the relationship between machine algorithms and crowd judgments for a variety of tasks.

To build a complete clustering workflow with multiple casts, we use a hierarchical clustering algorithm as the backbone that connects different casts. More specifically, the backbone algorithm fuses the judgements from different crowdworkers working on the same cast into clusters, which, in turn, become the shared context transferred to the next cast of the workflow.

With a clip similarity function from the prior cast and a stopping threshold, the hierarchical clustering method initially treats each clip as a cluster by itself, and iteratively merges the two most similar clusters until a threshold is reached. The result is a partially clustered dataset with clusters and single-

tons. When the backbone is used after the last cast in the workflow, each singleton is then merged into the most similar cluster. The similarity between two clusters is defined as:

$$ClusterSim(\omega_1, \omega_2) = \frac{1}{|\omega_1||\omega_2|} \sum_{t_j \in \omega_1} \sum_{t_k \in \omega_2} ClipSim(t_j, t_k) \quad (1)$$

where ω_1 and ω_2 are the two clusters, t_j and t_k are each of the clips in ω_1 and ω_2, respectively, and the $ClipSim()$ function is the given similarity function between clips.

The Merge Cast

While the Head Cast is designed to find the large clusters in the head of the distribution, since each crowdworker works independently, some of those clusters may actually be different subsets of the same larger category or the same categories based on different keywords (e.g., *sunlight* vs *natural lighting*). The Merge Cast is designed to consolidate existing clusters by merging duplicated categories. The input to this cast is a set of clusters that may or may not cover the entire dataset, and the output is fewer or equal number of clusters each with a list of ranked short descriptions. The challenge with detecting duplicate categories is that people need to understand what is in each category first. We start by presenting a set of existing clusters, and asking crowdworkers to name each of them. This acts as a defensive design[21] that ensures the crowdworkers understand the current context (scope and abstraction level), and also to obtain short descriptions for each of the clusters. Crowdworkers are then asked to merge identical categories by dragging them into the placeholders on the right (Figure 4).

If there are too many head clusters to fit into a microtask, the Merge Cast can be run recursively by first running on disjoint sets of existing clusters to consolidate them independently. Then, run another sets of Merge Cast on the output of each initial Merge Casts, and recurse until the output reduces to a set of clusters that could be presented in a global Merge Cast to ensure consistency. The assumption here is that the set of clusters in the final output of Alloy should be manageable by a single person to be useful. We also wanted to point out that the number of clusters is likely to scale much slower than the size of the dataset for many real-world data.

With the labels from the crowdworkers, we will again use the Gather Backbone to combine the judgements. The goal is to merge existing clusters if more than half of the crowdworkers also merged them in their solutions. Since in the Merge Cast workers can not break up existing clusters or reassign clips, we can formulate the clip similarity function as:

$$ClipSim(t_1, t_2) = \frac{1}{N} |\{\omega : t_1, t_2 \in \omega \text{ and } \omega \in \Omega\}| \quad (2)$$

where t_1, t_2 are the two clips, N is the total number of crowdworkers, Ω is the set of all clusters created by all crowdworkers, and ω is any cluster that contains both clips. This function is robust against a few workers doing a poor job. For example, if one crowdworker assigned every clip in the dataset to a single, general cluster (e.g., *answers*), the effect to the similarity function would be equivalent to having one less crowdworker and applying Laplacian smoothing. It is a common concern for crowd-based clustering methods that novice workers may create overly abstract categories (e.g., *solutions* or *tips*), that covers all items in the datasets. With our approach, it would require more than half of the workers to merge all items into a single cluster to generate a single cluster in the output.

From the output of the Gather Backbone, we rank the short descriptions associated with each cluster. Since clips are labeled by multiple crowdworkers, each cluster is associated with multiple descriptions via its clips. We use the F1 metric to rank these names to find the most representative description for each cluster, where the precision of a name label is defined as the number of clips in the cluster that it associates with divided by the size of the cluster, and recall as divided by the total number of clips associated with it.

The Tail Cast

The Tail Cast is designed to clean up the remaining singleton clips by classifying them into existing clusters or creating new clusters. The intuition is that even though machine learning techniques can produce high performance output, sometimes it is achieved at the expense of sacrificing the border cases. Human-guided "clean up" is often necessary for data produced by a machine learning model. The input of this cast is a set of existing clusters (with or without short descriptions) and a set of remaining clips. The output is a set of clusters with short descriptions.

We use an interface similar to the Merge Cast (Figure 4), and asked crowdworkers to review or name each of the existing clusters first, so that they build up better global understanding of the dataset before they organize the remaining clips. If Merge Cast was performed previously, their names are presented to lower cognitive load. The crowdworkers are then instructed to cluster the unorganized clips shown on the right by assigning them into existing clusters, creating new clusters, or removing uninformative clips. If there are too many remaining clips to fit into a single microtask, they are partitioned into groups of 20 items. Even though we may be dividing the remaining clips into partitions, all workers in the Tail Cast starts with learning the same global context that is the set of existing clusters from the Head Cast.

Finally, we use the Backbone Gather again to combine the multiple solutions from the crowdworkers. The goal is analogous to the goal of the Merge Cast: if two clips are assigned to the same category by more than half of the crowdworkers, they should be in the same cluster in the combined solution. For the similarity function, we simply replace the variable N in Equation 2 by the degree of redundancy.

EVALUATION METRIC

Unlike evaluating a classification task, which would typically be based on the precision and recall of pre-defined classes, evaluating clusters is not as straightforward due to the potentially different number of classes in the gold-standard and the system output. For example, high precision can be achieved by simply having more clusters in the output and the mapping between them. To address this, we use the normalized

Dataset	# sources	# workers	# clips	bad clips	# clusters
Q1:*How do I unclog my bathtub drain?*	7	16	75	25%	8
Q2:*How do I get my tomato plants to produce more tomatoes?*	18	13	100	10%	8
Q3:*What does a planet need to support life?*	19	19	88	31%	7
Q4:*What are the best day trips possible from Barcelona, Spain?*	12	12	90	18%	16
Q5:*How to reduce your carbon footprint?*	20	11	160	14%	11
Q6:*How do I unclog my bathtub drain?*	17	23	159	14%	11
Wiki: Talk page sections for the Wikipedia *Hummus* article	N/A	N/A	126	0%	13
CSCW: Abstract sections of CSCW 2015 accepted papers	N/A	N/A	135	0%	45

Table 1. Datasets used for evaluation

mutual information metric (NMI), which is a symmetric measurement sensitive to both the number of clusters, and the precision of each cluster. Specifically, it compares all possible cluster mappings to calculate the mutual information, and normalizes by the mean entropy so that the numbers are comparable between different datasets:

$$NMI(\Omega, C) = \frac{I(\Omega, C)}{0.5 * [H(\Omega) + H(C)]} \quad (3)$$

where Ω is the output clusters and C is the gold-standard clusters. The mutual information I is defined as:

$$I(\Omega, C) = \sum_k \sum_j P(\omega_k \cap c_j) log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} \quad (4)$$

where ω_k and c_j denotes each of the clusters in Ω and C, respectively. The probability $P(\omega)$ of item set ω is defined as $|\omega|/N$, where N is the number of total items. Finally, the mutual information is normalized by the mean entropy of Ω and C, so that the scores are comparable across datasets. To give some intuition, given w that maps to a gold-standard cluster c, we can calculate the precision by $P(w \cap c)/P(w)$ and recall by $P(w \cap c)/P(c)$, and the metric considers both with $P(w \cap c)/P(w)P(c)$. However, in reality it may be difficult to obtain such mappings, and the metric simply sums up scores of all possible mappings weighted by probability $P(w \cap c)$.

We use NMI for it is widely found in the literature for clustering evaluation. A more recent study found that it might favor datasets with more clusters, and proposed a variant that adjusts for randomness (AMI, [34]). We acknowledge this is a potential limitation, but found that the number of clusters Alloy produced were quite close to the gold-standard (average 10.3 vs 10.2), suggesting the concerns may be minimized. To be on the safe side, we also measured Alloy's performance using AMI on two datasets and found similar results.

DATASETS

In order to evaluate Alloy, we compared it to other machine learning and crowdsourcing clustering approaches in three different contexts: information seeking, Wikipedia discussions, and research papers. These contexts all involve rich, complex data that pose challenges for automated or existing crowd approaches. Below we describe each dataset and how we either generated or collected gold-standards.

Information Seeking Datasets

We picked five questions asked on popular Q&A forums (e.g., Quora, reddit, and Yahoo! Answers) that covered a diverse range of information needs. We then posted these questions to Amazon Mechanical Turk (AMT), and asked each crowdworker to find 5 webpages that best answered the questions in Table 1. The top sources were sent to workers to highlight clips that would help answer the question via an interface similar to that described in [22]. The first four datasets (Q1 to Q4) collected consist of 75 to 100 clips, extracted from 7 to 19 webpages using 12 to 19 crowdworkers. In addition, we also collected two datasets with more than 150 clips (Q5 and Q6) by gathering more clips from the sources.

To generate gold standards, two graduate students clustered each dataset independently. Raters were blind to Alloy's clusters, and no discussion on clustering strategies nor predefined categories were made prior to the process. Raters initially read every item in the dataset to build global understanding before they started organizing. Conflicts between raters were resolved though discussion. The first author participated in labeling two (out of the seven) datasets, but was always paired with another annotator outside of the research group. To measure inter-annotator agreement, we used the symmetric NMI metric as described in the previous section.

The agreements between raters are shown in Table 2. The datasets for "*How do I unclog my bathtub drain?*", "*How do I get my tomato plants to produce more tomatoes?*" and "*What are the best day trips possible from Barcelona?*" had high agreement between the two annotators of 0.7 to 0.75 NMI. For the "*What does a planet need to support life?*" dataset, the agreement was significantly lower (0.48). We kept this dataset to show the limitations of the proposed method, and we will discuss further in later sections. For the two larger datasets Q5 and Q6, the agreement scores were around 0.6.

Research Papers

Since some of the questions in the above dataset were about common daily life problems, an open question is whether crowd judgements were based on workers' prior knowledge or the context we provided them. To evaluate the system using more complex data where workers would likely have little prior knowledge we turned to research papers from the 2015 CSCW conference. For this dataset we used the official conference sessions as the gold standard for evaluation. The intuition is that conference organizers would place similar papers together in the same session. We acknowledge that the objectives of organizing conference sessions are not entirely the same as Alloy; most notably, conference session planning requires schedule conflict resolution and fixed size sessions. However, session co-occurrence represents valuable

judgments from experts in the community about which papers belong to a common topic, and even though each cluster is smaller in size (e.g., 3-4 papers per session) we can look at whether papers put together by experts are also put together by Alloy and the other baselines [8].

Wikipedia Editor Discussion Threads

Wikipedia relies on its editors to coordinate effectively, but making sense of the archives of editor discussions can be challenging as the archives for a single article can consist of hundreds or thousands of pages of text. We use as a dataset the discussion archives of the *Hummus* article, a popular yet controversial article, and use the discussion threads as the set of documents. The talk page consists of 126 discussion threads about various issues of the main articles that spans over the past 10 years (Table 1). Two annotators read the main article and the full talk threads before they started the labeling process. The NMI score between the two annotators was *.604*, which is comparable to the two other large datasets Q5 and Q6.

Wikipedia data can be more difficult to organize than previously mentioned datasets, because it can be organized in very different ways, such as topics, relations to the main article sections, and mention of Wikipedia guidelines [1]. The annotators also had a hard time coming up with a gold standard through discussion, and found both their categorization solutions to be valid. Therefore, instead of creating a single gold standard, we report the NMI scores between Alloy's output and each of the annotators.

EXPERIMENT OVERVIEW

In the following sections, we will describe three experiments and their results. Two workflows that uses the Gather to connect the different Casts are tested. The first experiment is an external evaluation that compares Alloy with other approaches. We use the full workflow that consists of the Head Cast, the Merge Cast, and the Tail Cast to cluster the six information seeking datasets (Q1-Q6), and compare with previous crowd-based methods and four machine algorithm baselines. The second experiment is an internal evaluation that tests the robustness of Alloy by using different number of workers in the Head Cast and the Tail Cast. Finally, in our last experiment, we test Alloy's performance on two different types of datasets: Wikipedia editor discussions and research papers.

EXPERIMENT 1: EXTERNAL VALIDATION

We first look at how Alloy compares with machine algorithms, other crowd algorithms, and inter-expert agreements. In the Head Cast, crowdworkers highlight keyword and cluster similar clips via searching, and in the Tail Cast another set of crowdworkers organizes all remaining clips.

We compare this Workflow 1 to three baselines that are commonly used in the clustering literature: latent Dirichlet Allocation (LDA) [4], latent semantic analysis (LSA) [12], and TF-IDF [28, 20]. We also compare against a hybrid baseline that uses human-identified keyword vectors from the Head Cast. This aims to test the value of the approach beyond the human identification of keywords by trying to cluster using only the keywords. In addition to comparing against automatic methods, we also compare Alloy to a popular crowd based method. The evaluation conditions are summarized below:

- *Workflow1*. The workflow with ten crowdworkers each for the Head Cast and the Tail Cast for Q1-Q4. An additional five workers for the Merge Cast for Q5-Q6. Each HIT costs 1 USD.
- *TF-IDF*. Weighted cosine similarity as the similarity function for the Gather. No human-computation was employed.
- *Crowd keywords*. Cosine similarity based on worker-highlighted keywords from the Head Cast as the similarity function for the Gather.
- *LSA*. The LSA model is used as the similarity function for the Gather. No human-computation was employed.
- *LDA*. The LDA topic model is used as the similarity function for the Gather. No human-computation was employed.
- *Cascade*. A version of Cascade with only one recursion using the default parameters as described in the paper.

Results

Alloy introduces a novel approach for providing context in the microtask setting with the sampling mechanism in the Head Cast. We captured crowdworkers' behavior during the tasks and found that nearly all (97.5%) workers used the sampling mechanism to gain context beyond the initial four items. On average, each worker sampled 15.1 items , and more specifically, 11.3% sampled more than 25 items, 23.8% sampled 15~24 items and 62.5% sampled 5~14 items.

Comparing with Machine Algorithms

On average, the proposed method performed significantly better and more consistent than all machine baselines (Table 2). In the worst case, Alloy clusters measured 0.058 NMI lower than the inter-annotator agreement, while the baseline systems measured more than 0.1 NMI lower in most cases. In a few cases some baselines also performed well (e.g., LSA performed slightly better on Q5), but none of them produced good results consistently across all datasets. Compared to the gold-standard clusters, Alloy produced clusters about as close to the gold-standard clusters as the two human annotators were to each other, despite the judges' advantages of having a global view of the datasets and multiple rounds of reading, labeling, and discussion. In addition, worker-identified keywords consistently outperformed TF-IDF, showing that the crowdworkers are extracting keywords in the Head Cast that are salient for identifying clusters each dataset. On the two larger datasets (Q5 and Q6), Alloy achieved similar performance as the four smaller datasets; better and more consistent comparing to the baseline systems, and near experts agreement comparing to the gold-standard.

Note that for every machine algorithm baseline we explored multiple parameters for each of the four questions, (hyper-parameters, number of topics, stopping threshold), and report the highest scores. The results of the baseline algorithms are likely over-fitting to the data, but we wanted to compare Alloy to these algorithms under their best possible settings [10].

Dataset	Inter.annot.	Workflow1	Workflow2	TF-IDF	Keywords	LSA	LDA	# of clusters	
								Alloy	expert
Q1	.734	**.759*** σ=.033	.550* σ=.093	.510	.647	.512	.478	7	8
Q2	.693	**.687*** σ=.016	.467* σ=.046	.534	.562	.537	.506	8	8
Q3	.477	**.468**	.425	.390	.440	.467	.442	7	7
Q4	.750	**.727**	.633	.673	.676	.704	.603	14	16
Q5	.630	.576	-	.568	.508	**.582**	.551	16	11
Q6	.588	**.588**	-	.462	.492	.497	.456	10	11
Average	.645	**.634**	-	.523	.554	.550	.503	10.3	10.2
CSCW	-	**.748**	-	.584	.652	.691	.725	23	45

Table 2. Evaluation Results. * indicates mean of 11 runs using different workers.[1]

Expert	Alloy	Cascade Single Pass
Hot Water	Hot Water / Clearing a drain with hot water	problem/symptom (21) drain clean (55) slow drain (36) plumbing (52) comments on why solutions are not working (8) how do i unclog a drain (46) bathroom (39) clean drain (42) chemical solution (16) drano (11) mechanisms that unclog drain (28) steps in unclogging the drain (45) manual solution (32) internet forum suggestion (33) how do i unclog my bath tub (49) drain water (31) help unclog the drain (47) helpful tip (45) what should i do with a plunger cup to open a drain (9) plunge (15) help (37) what to do with a drain clog (43) helpful response (35) technical advice (26) drain (50) reasons why drains become clogged (15) reasons why a drain becomes clogged (13) uncategorized (2)
Plunger	Plunge / Vigorous Plunging	
	Plunger / Dealing with Overflow	
Plumbing Snake	Snake the Drain / Plumbing Snake	
Remove Cover	Remove the Drain Cover / Check drain cover	
Chemicals	Drain Cleaner / Use drain cleaner for hair clogs	
Bent Hanger Wire	Remove Hair Clusters / Using a bent wire to clear a drain	
Call a Plumber		
Shop Vacuum		

Figure 5. Categories comparison for Q1

Comparing with Previous Crowd Methods

We compare Alloy with Cascade using datasets Q1-Q4, a popular crowd-based method for discovering taxonomies in unstructured data based on overlapping crowd clusters [9]. We implemented a simplified version of Cascade using the parameters described in the paper, but with only one recursion. We acknowledge that fine tuning and multiple recursion might improve Cascade's performance, but the numbers from our evaluation are consistent with the results reported in the Cascade paper based on the same metric and similar datasets.

On average, 84% of categories generated with Alloy were shared with clusters in the gold standard, versus 50% for Cascade. Cascade produced soft clusters where child clusters did not necessarily have all the items included in their parents, which breaks the assumptions of using NMI. To produce a direct comparison, we use the gold standard to greedily extract best matching, overlapping clusters that cover all items, and evaluated them using the average F1. In essence, this simulates an omniscient "oracle" that gives Cascade the best possible set of cluster matches, and so is perhaps overly generous but we wanted to err on the conservative side. The average F1 scores for each questions using Alloy are .72, .54, .48, .52, and using Cascade are .50, .48, .42, .39, showing a consistent advantage across questions. Furthermore, Alloy achieved this better performance at a lower cost (average $20 for Alloy vs $71 for Cascade), suggesting that machine learning can provide valuable scaling properties. We show categories created by experts and elicited from the two systems in Figure 5 to give a better sense of the datasets and the output.

EXPERIMENT 2: ROBUSTNESS

In this section, we examine the robustness of Alloy by varying the number of crowdworkers employed in the Head and the Tail Cast on datasets Q1-Q4. We start with having only 1 worker in the Head Cast, and evaluate performance as we hire more workers until we have 20. To test the two phase assumption, in a second condition, we switch to the Tail Cast after hiring 10 workers in the Head Cast, and continue to hire 1 to 10 more workers. This way, we can characterize the cost/benefit trade-offs in hiring different amount of human judgments. Further, by omitting the Tail Cast completely in the first condition, we can verify the two phase assumption by comparing the performance of a two-phase process (Head Cast and Tail Cast) with a one-phase control (Head Cast only) while equaling the number of workers:

- *Workflow1*. The workflow with ten crowdworkers each for the Head Cast and the Tail Cast. Each HIT costs 1 USD.
- *Workflow2*. The workflow with twenty crowdworkers and the Head Cast only. Each HIT costs 1 USD.

In addition, to test how robust Alloy is to the variance of crowdworkers on Amazon Mechanical Turk, we also hired eleven sets of ten different crowdworkers (a total of 440) for each Head and Tail Casts for Q1 and Q2.

Results

In Figure 6, we show the performance of employing different number of workers in the Head and the Tail Cast. Initially, increasing the number of workers in the Head Cast shows significant performance improvements. However, after gathering training data from around 10 workers, the performance gain from hiring additional crowdworkers decreases notably. Instead, performance improved significantly even with only a few additional crowdworkers in the Tail Cast to refine the clusters. Overall, having 10 crowdworkers in each of the Head and Tail Cast consistently outperformed having all 20 crowdworkers in the Head Cast across all four questions (Table 2), suggesting there is significant value in the Tail Cast.

[1]We also evaluated Q1 and Q2 using the AMI metric that accounts for randomness. The inter-annotator agreements are .674 and .643, respectively, and Alloy performed .674 and .609, respectively. See the Evaluation Metric Section for detail.

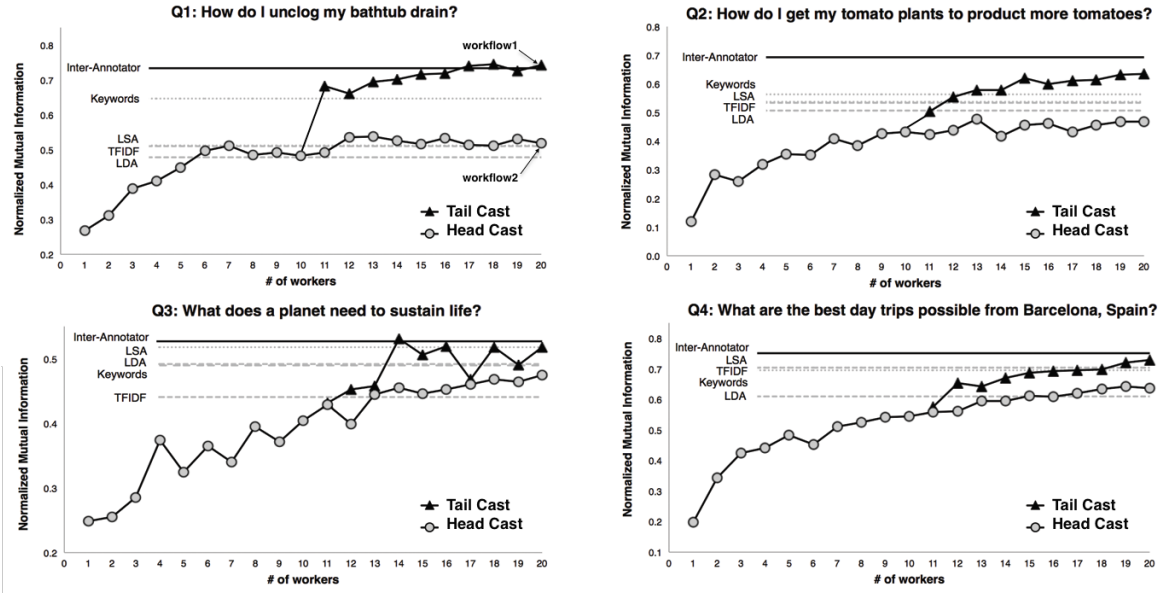

Figure 6. Performance comparison of using different number of crowdworkers in the Head Cast and the Tail Cast.

For Q1 and Q2, we also ran Alloy eleven times using different crowdworkers, and compared the results against the gold-standard labels and also with each other. Comparing to the gold-standards, which have inter-annotator agreements of .734 and .693 for Q1 and Q2 respectively, Alloy produced an average NMI of .759 (SD=.016) and .687 (SD=.016), respectively. Further, the average pair-wise NMI score of the 11 runs are .819 (SD=.040), and .783 (SD=.056), respectively, suggesting Alloy produces similar results using different crowdworkers on the same datasets.

EXPERIMENT 3: OTHER DATASETS

In this experiment, we use the same distributed workflow to test Alloy using the Wiki and CSCW datasets as described in the Dataset Section, in order to test how Alloy generalizes to other types of data. These datasets contain long academic documents or editorial discourses that are infeasible to present multiple items to the crowdworker in one HIT. Instead, we show a small portion of each item in the datasets to the crowdworkers. For each item in the Wiki dataset, we display the thread-starter post and the first two replies. For the CSCW dataset, we present the abstract section of each paper, and compare results with the official conference sessions. Machine baselines were however given access to all of the text of the paper and the full discussion threads in order to provide a strong test of Alloy's approach.

Results

For the CSCW dataset, Alloy outperformed all machine baseline systems with *.748* NMI score using conference sessions as the gold standard Table 2. The Keyword baseline outperformed the TF-IDF baseline (*.652* vs *.584*), showing that the crowdworkers are extracting valuable keywords in the Head Cast, despite that research papers may be difficult or impossible for crowdworkers to understand. On the other hand, Alloy produced 24 categories out of 135 abstracts, more than all other datasets. One possible assumption is that it may be more difficult for novice workers to induce abstract categories when organizing expert dataset, leading to higher number of more lower level categories in the outcome.

For the Wiki dataset, the NMI score between annotators was *.604*, which is comparable to the two other large datasets Q5 and Q6. Comparing to the two sets of expert labels independently, Alloy's output measured *.528* and *.507*. Compared to all previous results, Alloy seemed to perform less favorably on this dataset. As mentioned in the Dataset Section, the raters found the this dataset the most difficult to organize, as there are many different valid structures that the two annotators were unable to reach an agreement also hints that the space of valid solutions may be larger on this dataset. In addition, we only showed the first three comments of each discussion to the crowdworkers, whereas the annotators and the machine baselines have access to the full discussion. We acknowledge length of items is a limitation, and will discuss in detail in the Discussion Section.

DISCUSSION

In this paper, we took a step towards tackling the problem of clustering high-dimensional, short text collections by combining techniques from natural language processing and crowdsourcing. By using a two-phase process connected by a machine learning backbone, our proposed method compensates for the shortcomings of crowdsourcing (e.g., lack of context, noise) and machine learning (e.g., sparse data, lack of semantic understanding). As part of the system we introduced an approach aimed at providing greater context to workers by transforming their task from clustering fixed subsets of data to actively sampling and querying the entire dataset.

We presented three evaluations that suggest Alloy performed better and more consistently than automatic algorithms and

a previous crowd method in accuracy with 28% of the cost (Exp.1), is robust to poor work with only 20 workers (Exp.2), and is general enough to support different types of input (Exp.3). Qualitatively, we noticed Alloy often produced better names for categories than machine algorithms would be capable of, including names not in the text (e.g., a cluster including items about *smart thermostats* and *solar panels* was named "*Home Improvements*" which was not in the actual text).

One potential concern might be whether Alloy's tasks take too long to be considered microtasks. While Alloy deploys HITs that take more than a few seconds to finish, we think they are still comparable to other complex microtask systems such as Soylent [3] and CrowdForge [23]. Specifically, based on a total of 281 HITs, the median run-time for the Head Cast HITs is 7.5 minutes (M=8,3, SD=4.1), for Merge Cast 8.3 minutes (M=16.2, SD=15.6), and for Tail Cast 11.4 minutes (M=13.2, SD=6.1). Despite having less workers doing longer tasks, Alloy performed consistently across different sets of workers on the same datasets.

During development, some assumptions, both explicitly and implicitly, were made about the input of the system: 1) there are more clips than categories. 2) the categories follow a long-tailed distribution. 3) clips belong to primarily one cluster. 4) there is a small set of gold-standard clusters. 5) workers can understand the content enough to cluster it. Note that we do not assume the crowdworkers can understand the semantics of the content, but just enough to identify ideas that are salient and common in the dataset. Thus they may be able to cluster complex topics such as machine learning without understanding those topics if enough relational context is embedded in the clips. For example, an abstract of a research paper may say "this paper uses POMDP machine learning approaches to cluster text", they might put it in a "clustering" cluster without knowing what a POMDP is.

One obvious limitation to our approach is clustering long documents. This is a common limitation for crowd-based systems that rely on workers reviewing multiple items for context (either from random selection or active sampling). It becomes infeasible to fit multiple items in a single HIT if the length of each item is long. Another related limitation is organizing documents that describe multiple topics. Lab studies in a past work [22] showed that individuals are able to decompose long documents into short clips of single topics during information seeking tasks. One way to expand the proposed method to overcome the length limitation could be splitting documents into short snippets, either with the crowds or machine algorithms, and create topical clusters using Alloy.

Another limitation is organizing datasets that are inherently difficult to structure categorically. For example, concepts in Q3 (*planetary habitability*) have causal relationships without clear categorical boundaries (e.g., *distance to sun*, *temperature* and *liquid water*). As a result, all approaches had significant trouble, including low agreement between human annotators. On the other hand, some dataset can be organized categorically in multiple ways. In Q4 (*Barcelona*) we found that some categories fit a *place* schema (e.g., *Sitges*, *Girona*) while other categories fit a *type* schema (e.g., *museums*, *beaches*). One approach for addressing this could be trying to cluster workers to separate the different kinds of schemas; however, upon inspection we found that individual workers often gave mixtures of schemas. This interesting finding prompts further research to investigate what cognitive and design features may be causing this, and how to learn multiple schemas.

Looking forward, we identified a set of patterns that may be useful to system designers aiming to merge human and machine computation to solve problems that involve rich and complex sensemaking. The hierarchical clustering backbone we use to integrate judgments from a variety of crowdworker tasks allows us to *cast* for different types of crowd judgments and *gather* them into a coherent structure that iteratively gets better with more judgments. We also introduce useful new patterns for improving global context through self-selected *sampling* and keyword *searching*. One important consideration these patterns bring up is that while previous ML-based approaches to crowd clustering have focused on minimizing the number of judgments, we have found it is at least as important to support the rich context necessary for doing the task well and setting up conditions that are conducive for crowdworkers to induce meaningful structure from the data.

We hope the patterns described in this paper can help researchers develop systems that make better use of human computation in different domains and for different purposes. For example, the *sample and search* pattern could potentially be adapted to support other tasks such as image clustering, where crowdworkers could use the sampling mechanism to get a sense of the variety of images in the dataset, highlight discriminative objects, and label images queried based on features extracted from the highlighted regions. Furthermore, the *cast and gather* pattern may provide a useful framework for combining crowds and computation that is both descriptive and generative. For example, Zensors [26], a crowd-based real-time video event detector, could be considered a form of the cast and gather pattern which uses a classification algorithm instead of a clustering algorithm as a backbone, and casts for human judgements whenever its accuracy falls below a threshold (e.g., if an environmental change lowers precision), with the classifier backbone retrained with the new human labels. While we used a clustering backbone in this work, future system designers might consider other machine learning backbones (e.g., classification or regression algorithms) for different tasks. Overall, we believe this approach takes a step towards solving complex cognitive tasks by enabling better global context for crowd workers and providing a flexible but structured framework for combining crowds and computation.

ACKNOWLEDGEMENTS

The authors would like to thank to Andrew Peters for the valuable discussions. This work was supported by NSF grants IIS-1149797, IIS-0968484, IIS-1111124, Bosch, and Google.

REFERENCES

1. Paul André, Aniket Kittur, and Steven P Dow. Crowd synthesis: Extracting categories and clusters from complex data. In *Proc. CSCW 2014*.
2. Richard Ernest Bellman. 2003. *Dynamic Programming*. Dover Publications, Incorporated.
3. Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proc. UIST 2010*. ACM, 313–322.
4. David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3 (2003), 993–1022.
5. Jonathan Bragg, Daniel S Weld, and others. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *First AAAI conference on human computation and crowdsourcing*.
6. Allison June-Barlow Chaney and David M Blei. 2012. Visualizing Topic Models.. In *ICWSM*.
7. Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM 2011 TIST* 2, 3 (2011), 27.
8. Lydia B Chilton, Juho Kim, Paul André, Felicia Cordeiro, James A Landay, Daniel S Weld, Steven P Dow, Robert C Miller, and Haoqi Zhang. 2014. Frenzy: Collaborative data organization for creating conference sessions. In *Proc. CHI 2014*. ACM, 1255–1264.
9. Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proc. CHI 2013*. ACM, 1999–2008.
10. Jason Chuang, Christopher D Manning, and Jeffrey Heer. 2012. Termite: Visualization techniques for assessing textual topic models. In *Proc. of the International Working Conference on Advanced Visual Interfaces*. ACM, 74–77.
11. Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proc. CHI 2012*. ACM, 443–452.
12. Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS* 41, 6 (1990), 391–407.
13. M. M. Schaffer D.L. Medin. 1978. Context theory of classification learning. *Psychological review* 85, 3 (1978), 207.
14. Jerry Alan Fails and Dan R Olsen Jr. Interactive machine learning. In *Proc. IUI 2003*. ACM, 39–45.
15. John A Hartigan. 1975. *Clustering algorithms*. John Wiley & Sons, Inc.
16. Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Machine learning* 95, 3 (2014), 423–469.
17. Anil K Jain and Richard C Dubes. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.
18. A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data clustering: a review. *ACM Comput. Surv.* 31 (1999), 3.
19. Bernard J Jansen, Danielle L Booth, and Amanda Spink. 2009. Patterns of Query Reformulation During Web Searching. *Journal of the American society for information science and technology* 60, 7 (2009), 1358–1371.
20. Karen Sprck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 28 (1972), 11–21.
21. Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *Proc. CHI 2008*. ACM, 453–456.
22. Aniket Kittur, Andrew M Peters, Abdigani Diriye, Trupti Telang, and Michael R Bove. Costs and benefits of structured information foraging. In *Proc. CHI 2013*. ACM, 2989–2998.
23. Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut. Crowdforge: Crowdsourcing complex work. In *Proc UIST 2011*. ACM, 43–52.
24. Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM TKDD* (2009).
25. Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. Structured Labeling for Facilitating Concept Evolution in Machine Learning. In *Proc. CHI 2014*. 3075–3084.
26. Gierad Laput, Walter S Lasecki, Jason Wiese, Robert Xiao, Jeffrey P Bigham, and Chris Harrison. Zensors: Adaptive, Rapidly Deployable, Human-Intelligent Sensor Feeds. In *Proc. CHI 2015*. ACM, 1935–1944.
27. Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. 2007. A note on Platts probabilistic outputs for support vector machines. *Machine learning* 68, 3 (2007), 267–276.
28. Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Vol. 1. Cambridge university press Cambridge.
29. Douglas L Medin and Marguerite M Schaffer. 1978. Context theory of classification learning. *Psychological review* 85, 3 (1978), 207.
30. Peter Pirolli and Stuart Card. 1999. Information foraging. *Psychological review* 106, 4 (1999), 643.
31. John Platt and others. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.
32. Michael Steinbach, George Karypis, Vipin Kumar, and others. 2000. A comparison of document clustering techniques. In *KDD workshop on text mining*, Vol. 400. Boston, 525–526.
33. Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. 2011. Adaptively learning the crowd kernel. In *In ICML11*.
34. Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary?. In *Proc. ICML 2009*. ACM, 1073–1080.
35. Ryen W White, Mikhail Bilenko, and Silviu Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *Proc. SIGIR 2007*. ACM, 159–166.

36. Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research* 5, 975-1005 (2004), 4.
37. Jinfeng Yi, Rong Jin, Anil K Jain, and Shaili Jain. 2012a. Crowdclustering with sparse pairwise labels: A matrix completion approach. In *AAAI Workshop on Human Computation*, Vol. 2.
38. Jinfeng Yi, Rong Jin, Shaili Jain, Tianbao Yang, and Anil K Jain. 2012b. Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In *Advances in Neural Information Processing Systems*. 1772–1780.