Helping Users Sort Faster with Adaptive Machine Learning Recommendations

Steven M. Drucker, Danyel Fisher, Sumit Basu

Microsoft Research, 1 Microsoft Way, Redmond, WA. 98052
{sdrucker; danyelf; sumitb}@microsoft.com

Abstract. Sorting and clustering large numbers of documents can be an overwhelming task: manual solutions tend to be slow, while machine learning systems often present results that don't align well with users' intents. We created and evaluated a system for helping users sort large numbers of documents into clusters. iCluster has the capability to recommend new items for existing clusters and appropriate clusters for items. The recommendations are based on a learning model that adapts over time – as the user adds more items to a cluster, the system's model improves and the recommendations become more relevant. Thirty-two subjects used iCluster to sort hundreds of data items both with and without recommendations; we found that recommendations allow users to sort items more rapidly. A pool of 161 raters then assessed the quality of the resulting clusters, finding that clusters generated with recommendations were of statistically indistinguishable quality. Both the manual and assisted methods were substantially better than a fully automatic method.
Keywords: Mixed initiative interactions, adaptive user interfaces, information interfaces, interactive clustering, machine learning.

1 Introduction

A project manager at a software corporation is faced with tens of thousands of pieces of feedback from beta users. She wishes to divide the feedback into groups to ensure that each item will be addressed. Going in, she does not know what the appropriate categories for the data will be. The categorization she creates is organic and personal – she splits up the work with her team in mind, creating appropriate categories according to their abilities. A different project manager with a different team would surely create a different clustering for the same task. Existing tools for this type of personalized sorting are limited in their ability to help a user with the task – today, the program manager would likely use a tool such as Excel, but would have to do the categorization manually.

This need exists across many domains. A graduate student organizing related work for his thesis, or a human resources manager organizing resumes, faces a similar challenge: they have very large amounts of data to sort, and their particular choices are unique to their own needs and interests. Indeed, a different graduate student, or different manager, would come up with a distinct clustering.

Using Machine Learning to Speed Clustering

Numerous algorithms are available to help automatically cluster documents. These algorithms make implicit or explicit assumptions about what items should be grouped together, which may not be well aligned with a user's own notion of appropriate groupings. Furthermore, users may have different notions of what items should go together, and may not be able to explain why, at a high level, certain items should or should not go together. We wish to take advantage of the assistive power of machine learning, while allowing users to express their needs naturally.

Methods of combining machine learning with manual clustering have met with mixed results. Both the Implicit Queries for Data Mountain [8] and Scatter-Gather [1] research found that machine learning made sorting and retrieval tasks, respectively, *slower* for users, though they did help users to recall the locations of items and better understand the target corpus.

We differ from these past approaches. Starting with machine-generated clusters can have two undesirable outcomes. Primarily, it can diminish trust in the system if automatic clusters do not correspond to the users' schematization. Additionally, we are concerned that automatically-generated clusters might bias users, guiding them away from creating clusters that are meaningful to their interests. While fully automated clustering can be very effective for datasets in which cluster centers are well-separated (such as disambiguating search results as in [18]), we show that automatic clusters are less successful for ambiguous datasets. We want to allow users to grow clusters organically, yet still gain the benefits of machine assistance.

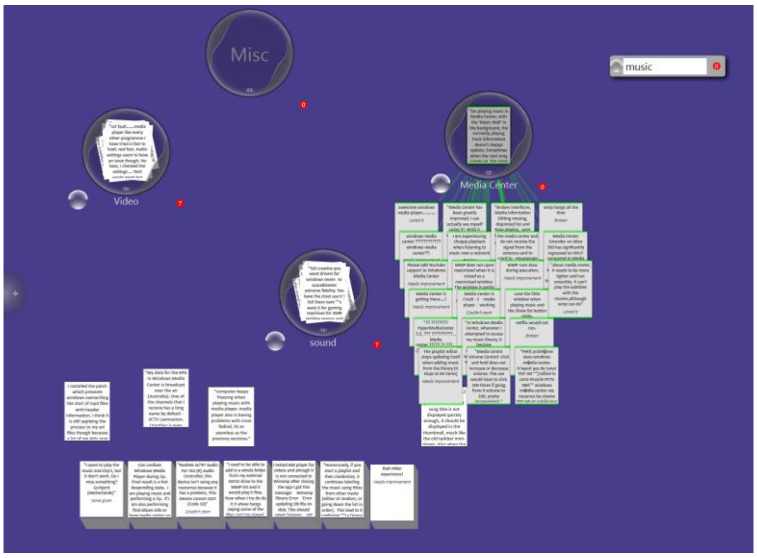


Figure 1: The iCluster interface. The user has asked for suggestions of documents that might fit the "Media Center" cluster.

The iCluster System

Our system, called iCluster, assists users in organizing a set of items after observing their actions; it learns a model of each individual cluster as well as a global distance metric. The system makes recommendations to the user; these recommendations improve as the user organizes more data. As the system is training on the work users have done so far, users' individual clustering habits will lead to personalized recommendations.

The primary contributions of our work are an adaptive, interactive system to assist people in sorting items, and an evaluation of how recommendations affect the sorting task. We explored time and accuracy tradeoffs using iCluster in comparison with both a manual system and fully automated clustering.

The body of this paper is organized as follows. We first examine our work in the context of systems for organizing documents spatially, past work on clustering in the machine learning literature, and systems that use interactive learning to search and organize information. We then describe the design of the system, including an observational study and subsequent iterative development as well as the backend algorithmic support. Next, we discuss the evaluation of the system; we find that assisted clustering can be substantially faster than unassisted clustering while maintaining the quality of the clusters. Last, we discuss our findings and future work.

Related Work

Manual Clustering

A number of research projects have examined the ways that people sort their data, both looking at personal data and shared information. Dourish *et al* [10] examine shared files in a government office. While the government workers nominally sort items with a defined, global taxonomy, the sorting and clustering process is based on local needs and information. Malone [12] examines office workers' organization of their desks; Whittaker and Hirschberg [17] look at how people manage personal paper archives. Both studies find that users use local, idiosyncratic organizations, moving things into categories that make sense for their own tasks. Similarly, they find the now classic breakdown between 'filers' and 'pilers', and that people very naturally arranged items in piles. While these piles reflect the users' interests, Whittaker and Hirschberg also show that it can be difficult to find papers once they have been piled, or to get an overview of what a pile represents.

Techniques and Interfaces for Adaptive Clustering

The machine learning community has a history of work on unsupervised (i.e., fully automatic) clustering of data by various means, from the well-known k-means algorithm to the more modern approach of Latent Dirichlet Allocation [5]. More recently, there has also been some work on "interactive clustering," in which an algorithm is presented with pairs of items that *must* or *cannot* belong together. The algorithm then optimizes a distance metric which best satisfies these constraints, and items are clustered using k-means (e.g. [7]). However, the work in the machine learning community has been theoretical in nature; there has been very limited work on incorporating interactive clustering into an interactive *system*, and none measuring its effect on human behavior. The closest such work is that of desJardin et al. [9], which proposes an interface in which users could specify these constraints. This work contained only a simulation of human behavior and did not study how or whether the system helped humans with the task.

Basu *et al.* [3] propose an algorithm to support online interactive clustering that works somewhat differently: instead of reclustering the entire dataset, their method provides online recommendations for new items that could be appropriate for a given cluster. That work was not incorporated into a final system nor evaluated with real users. In this work, we adapt Basu *et al.*'s technique with a spatial interface.

Spatial Layout for Document Collections

There has been quite a bit of research into how people use spatial organization in real life to help manage their collections of documents. Malone [12] explores how people organize their office desks, and Whittaker and Hirschberg [17] look at how people manage personal paper archives. They find the now classic breakdown between 'filers' and 'pilers', and that people very naturally arranged items in piles. However, they show that it can be difficult to find papers once they have been piled, or to get an overview of what a pile represents. Our interface uses a spatial layout to build on these observations while trying to alleviate the difficulties.

User interfaces have been exploring spatial organization since the early 1980's. Work at Apple expanded on simple 2D spatial layouts with piles [13] that allowed users to group related documents into stacks. More recently, DataMountain [15], Bumptop [1] and DynaPad [4] all expand from the simple 2D metaphor by adding 3D, physics, or zooming. Our contributions are orthogonal to this work – we could augment any of these systems with the techniques described in this paper. Other interfaces allow users to cluster items spatially: for instance, Bubble Clusters [16] make clustering an implicit operation based on spatial distance between items. While our approach has a more explicit containment model, our interface could be adapted to work with the bubble-clustering approach.

Czerwinski et al. [8] adds "implicit queries" to the DataMountain system. DataMountain [14] allows users to spatially layout web thumbnails in a pseudo-3D plane; their project extends the work by highlighting related documents. Related documents are computed with a fixed similarity metric (unlike our dynamic learning system), based on either a previous subject's categorizations, or a cosine distance between document word vectors. The authors evaluate categorization time, number of categories and subsequent re-retrieval; they found that users using implicit queries took longer to place items but were more able to recall where they were.

Scatter/Gather [15] uses a search interface to build clusters of related objects; users can then dynamically recluster based on a selection. Scatter/Gather uses this for interactive refinement; it was designed for question-answering tasks, rather than information sorting tasks. As such, it was used and evaluated in a very different manner than our system.

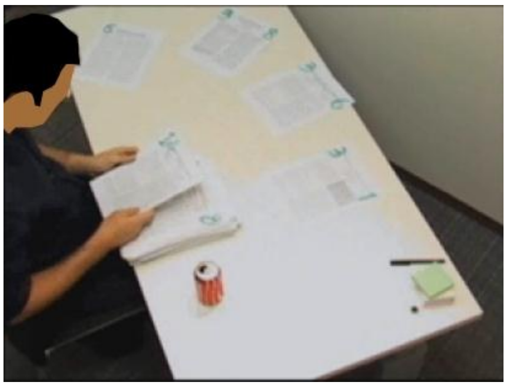


Figure 2: A user clusters 60 CSCW papers by hand during the observational study.

The iCluster System

We developed iCluster to address the challenges of clustering large sets of data. Our design was based first on an observational study of clustering behavior. In this section, we discuss first the observational study, then the prototype we constructed.

Clustering Paper Documents Manually

In order to better understand how people currently go about the task of clustering a large number of papers, we observed six people who were given the task of organizing 60 papers from the last few years of the ACM CSCW conference. The subjects were given one hour to get through as many documents as possible. Subjects were asked to group the papers according to their topics and addressed problems. They were given print-outs of the papers, sticky notes, paper-clips, pens, highlighters, and folders; users were permitted to organize the papers in any way they wished (Figure 2).

Consistent with Malone *et al.*'s results [12], all of the participants organized their documents into piles. Subjects spread their piles on the table, and labeled them with sticky notes. Some people put papers that they were not yet certain of into a temporary "holding" pile, while others left those papers in the initial pile. Several users refactored their piles when a new paper made them realize that they had mis-categorized a past paper.

In general, the lessons learned from this preliminary study were the following:

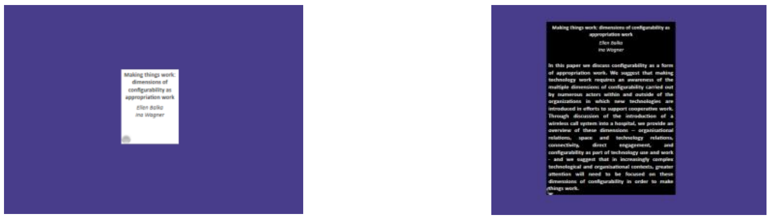
- The desktop was used as a holding area; papers were then piled in distinct stacks. People used spatial memory to re-find piles, although this could be slow and error-prone.
- Labels on piles helped users return to them more easily; unlabeled piles were confusing.
- People often did not put items in a cluster until several related items were found.

- People seemed to use different organizational principles to cluster the information (agreeing with Dourish *et al.*'s findings [10]). For instance, some grouped methodological techniques, while others grouped based on a particular technology.

iCluster Prototype

We used the feedback from the observational study in order to build a prototype for users to cluster papers online. The iCluster prototype uses spatial design concepts: documents and clusters are arranged on the desktop, and can be dragged into new locations. Documents can be dragged into clusters; users can request recommendations of what documents might fit into a given cluster. Figure 1 shows the interface. The prototype can be seen in action in the supplemental video figure.

iCluster represents each document as a small "trading card" (Figure 3). For instance, in a corpus of textual content, we include the first few lines of the text on the front, and the entire statement on the back. For conference papers, we put title and authors on the front and the abstract on the back. Initially, the cards are stacked in piles (seen at bottom, Figure 1); users place the cards into clusters.



Making things work: dimensions of configurability as appropriation work

Ellen Balka

Ina Wagner

Making things work: dimensions of configurability as appropriation work

Ellen Balka

Ina Wagner

In this paper we discuss configurability as a form of appropriation work. We suggest that making technology work requires an awareness of the multiple dimensions of configurability carried out by numerous actors within and outside of the organizations in which new technologies are introduced in efforts to support cooperative work. Through discussion of the introduction of a wireless call system into a hospital, we provide an overview of these dimensions – organisational relations, space and technology relations, connectivity, direct engagement, and configurability as part of technology use and work - and we suggest that in increasingly complex technological and organisational contexts, greater attention will need to be focused on these dimensions of configurability in order to make things work.

Figure 3: The front (left) and back (right) of a single document in the iCluster interface.

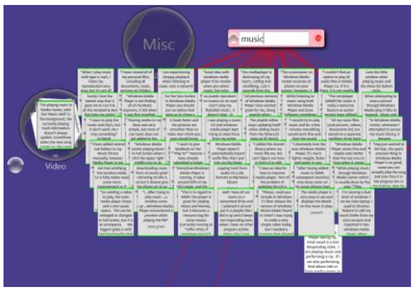


Figure 4: Full-text search. The user has searched for the word "collaboration," matching documents are brought to the foreground

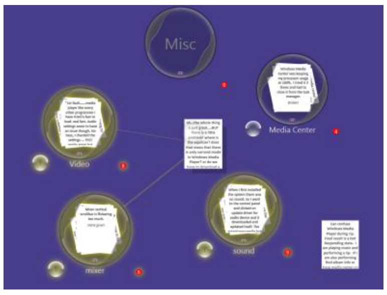


Figure 5: Item-to-group recommendation. The recommendation system believes that this document would fit better in the "mixer" group (lower left, thick line) than the "video" or "sound" group (thinner lines to left and bottom right), and not at all to the media center group

We experimented with various forms of loose spatial layout (such as DataMountain and bubble clusters), but found that our users liked the ability to hide information in a container. Users create clusters in iCluster with a button on the left side of the screen which creates an empty cluster that can be moved around the screen. Users can caption clusters with a click and by entering a name on the keyboard.

A text entry box on the screen enables keyword search. When a keyword is entered, up to 70 matching items are spread out underneath the search query box in columns which allows for effective use of space for the revealed documents. The items are highlighted, and lines are drawn to the search box, to make the items easier to find. Figure 4 illustrates a text search in progress. Based on user feedback, we also highlight any groups that contain items that match the search.

The novel contributions of iCluster are the ways in which the system can help users to make clustering decisions. Assistance is based on items within clusters, and so does not begin until users have placed items into clusters. Recommendations can be triggered in two ways. When an item is selected, lines of different weights are drawn to clusters, showing which ones might be most closely related to the selected item. A thicker line indicates a more likely match. We refer to this as an item-to-group recommendation. Figure 5 shows one such recommendation.

The second form of recommendation is the group-to-item recommendation. When a user clicks on a cluster, the system computes which twenty documents would be the best match for the cluster, based on entries already placed in clusters. It then lays out those documents using the same column strategy as before, underneath the selected group. Group-to-item recommendation is illustrated in Figure 1. As with search results, there is no "accept" button for group-to-item recommendations; rather, the user simply drags the object they are interested in into the relevant group. Indeed, a user may even choose to drag a recommended item into a different group.

We now describe the learning algorithm that drives both the item-to-group and group-to-item recommendations.

Backend learning algorithm

The recommendation algorithm is based on Basu *et al.* [3]; we provide an overview here for completeness. The algorithm is a hybrid approach that incorporates aspects of both distance metric learning and per-cluster classification. The algorithm dynamically learns both types of models and reweights their relative contributions. In so doing, it can leverage both approaches in modeling the users' intent. The algorithm was shown to make good recommendations after fairly few items were rated, and can be applied to arbitrary types of data, as long as each item can be described by a set of candidate features (continuous or categorical) and the distances between pairs of items can be described by a set of candidate distance measures.

Given a set of features and distances, and the list of items which belong to each cluster, the algorithm optimizes a distance metric between items that best satisfies these constraints. Unlike previous work (such as the fomulations in Cohn and Caruana [7] or desJardins et al.[9]) that only optimized the weights on the candidate distance measures, this method also trains a classifier for each cluster and converts it into a new distance measure; details of the formulation are in [3]. In this way, this method is able to take advantage of the benefits of both metric learners and classifiers. To recommend the closest items to a cluster, the algorithm computes the mean distance using this learned measure to all the elements in the cluster. The items are then ranked according to this quantity.

We extended Basu *et al.*'s algorithm to get recommendations of the closest cluster to an item. To do so, we compute the rank of the item for each cluster, then sort the clusters based on this rank. We use rank rather than distance because distances from different clusters are not calibrated with respect to each other.

For our data, the per-item features were produced by combining all the text from the text fields and representing this via standard TFIDF features, which were further compressed via principal components analysis (PCA) to yield the top 50 features per item. The 25 distance measures between items came from computing the absolute distances between the top 25 features. Because we had some concerns about the model overfitting when only a small number of items had been labeled, we made the small extension of scaling the number of features according to the data available. Specifically, we limited the number of features available to the classifier to five times the total number of clustered items so far; we found this led to better recommendation behavior for the first few filed items.

Basu *et al.* [3] validated their algorithm's recommendations through simulation: five expert humans manually clustered 300 papers from the SIGIR, CSCW, or SIGGRAPH conferences, depending on their area of expertise. The authors then simulated the recommendations that the system would have generated based on the data input, playing back the clusters that the users formed in random order. With only three examples as labels, more than 25% of the recommended items were correct suggestions on average; with eight examples, nearly 40% were correct. Furthermore, the algorithm suggested items at a significantly higher rate of precision than either a classifier or a metric learning approach on its own.

This relatively high level of precision even with small numbers of examples led us to believe that the algorithm would be an appropriate choice for the engine of the iCluster system during the sorting task.

System Validation

In order to explore both the need for and the effects of using iCluster, we performed several different evaluations. First, we wanted to establish that people do indeed cluster items in different ways from each other. Second, we wanted to investigate whether using iCluster would allow users to generate clusters more rapidly. This speed should not be at the cost of quality: the clusters they generate should be of similar quality to ones generated without machine assistance; and that the clusters should be of higher quality than purely-automatic clusters.

In order to compare the effects of recommendations, we created a version of iCluster that does not contain either item-to-group recommendations or group-to-item recommendations. We refer to this limited version as BASELINE, and to the full version as ASSIST.

We also wanted to make sure that the iCluster system could be generalized to work for several different problem domains.

We explore the following four hypotheses:

- **H1: Individual users will develop different and distinct clusters.**
- **H2: Users will cluster more items in the ASSIST condition than in BASELINE.**

- **H3: Clusters created in the ASSIST condition be of comparable quality to those generated in the BASELINE condition.**
- **H4: Both ASSIST and BASELINE clusters will be of higher quality than automatically-generated clusters.**

We tested these hypotheses through several studies. The first study did not use any assistance whatsoever, but rather had people use the base system to cluster several corpora and we subsequently examined the overlap of items to establish that people do indeed use different strategies to cluster documents. This was used to evaluate H1.

The second study was a within subjects study to explore how people perform with and without assistance from the iCluster system. It allows us to address H2.

Finally, a third study was performed to evaluate the quality of the clusters generated in the previous studies (and to an automated clustering process) to address hypotheses H3 and H4.

Study 1: Generating Clusters Baseline

For our first study, we recruited four sorters who were familiar with the SIGCHI literature, and another four sorters who were generally familiar with computing. CHI sorters were given a dataset based on a selection of 400 recent CHI and UIST papers. The other sorters were assigned to the FBK condition, which consisted of a dataset of 300 items of free-text feedback about a piece of software. In both cases, papers were stacked in 5 initial piles in random order. SIGCHI papers had the title and authors on the front, and the abstract on the back, and so averaged 138 words of content. The FBK dataset, in contrast, had no abstract; messages averaged just 31 words of content. Users started with no clusters.

We asked CHI sorters to imagine that they were on the program committee of a conference and needed to sort the different documents into separate tracks. FBK sorters were to sort the feedback items into functional groups that could be addressed by hypothetical product teams. They were asked to get through as many items as possible in 45 minutes while still creating useful groups.

These sorters were all assigned to the BASELINE condition of the program where no assistance was given. Sorters were able to use text-search to find documents by keyword.

After being introduced to the system and allowed to try it out on briefly on a test data set, sorters were given 45 minutes to cluster as many documents as they could. We logged all interactions with the system as they clustered in that time period, including the number of items sorted.

Study 2: Within Subjects Generating Clusters with and without Assist

Because of large variations in individual style and familiarity with the corpus, it was clear that a within subjects study was the only way to allow us to meaningfully compare sorting time or quality (H2 and H3). In addition, our system was vulnerable to unusually strong learning effects: once a user has clustered any sample from a corpus, they have a good intuition for the clusters that they will use for another sample.

We recruited 16 new sorters; we used as our corpora software feedback in two comparable but different areas (Network and Multimedia). We balanced the order of presentation (with and without assistance) as well as the order of the corpus. Sorters were introduced to the features of the interface between trials, and given a chance to try the interface. They were then given an unspecified amount of time to cluster as many items as they could into 'high quality' groups. After 20 minutes, they were interrupted and the task was restarted on a different corpus using the other condition of the interface.

Study 3: Evaluating Cluster Quality

Sorters could easily bias the cluster speed test, of course, merely by dropping items into clusters randomly. To address H3 and H4, we wanted to establish whether the clusters generated groups of high quality.

Judging cluster quality in an objective way has traditionally been quite difficult. A rater, presented with a grouping and asked for an assessment of quality, will search for some rational explanation of how the items fit together. Recent work by Chang *et al.* [6] presented a clever new way to judge cluster quality that alleviates this problem. The basic idea was to present some items from a cluster and one outlier; the accuracy with which raters could correctly detect the outlier was their measure of quality.

For our work, we adapt and expand this model by measuring the quality of a clustering with two properties:

Coherence: A cluster is coherent if a user can recognize that its members go together (and so identify an outlier).

Distinctness: A set of clusters is distinct if a user can recognize that elements from different clusters do *not* go together (and so pick out a matching element for a set).

In an Outlier test, adapted from [6], raters are presented with four items from one cluster, and one from a second, both chosen from a single sorter. The rater must identify the outlier that was chosen from the second cluster. If raters are unable to identify the outlier, then the cluster is likely to be a weak one; if raters are successful across many trials, the cluster is more likely to be coherent.

Choose the item that does not belong.

- HeartBeat: an outdoor pervasive game for children; *Remco Magielse, Panos Markopoulos*
- A comparative study of speech and dialed input voice interfaces in rural India; *Neil Patel, Sheetal Agarwal, Nitendra Rajput, Amit Nanavati, Paresh Dave, Tapan Parikh.*
- Design influence on social play in distributed exertion games; *Florian Mueller, Martin Gibbs, Frank Vetere*
- O' game, can you feel my frustration?: Improving user's gaming experience via stresscam. *Chang Yun, Dvijesh Shastri, Ioannic Pavlidis, Zhigang Deng*
- Designing digital games for rural children: a study of traditional village games in India. *Matthew Kam, Akhil Mathur, Anuj Kumar, John Canny*

Figure 6: A sample "Outlier" question. The second item in the list is different from the others.

Choose the item that goes with both of these

- Collaborative interaction with volumetric display; Tovi Grossman, Ravin Balakrishnan
- MightyTrace: multiuser tracking technology on lc-displays; Ramon Hofer, Patrick Kaplan, Andreas Kunz

From among these

- Bonfire: a nomadic system for hybrid laptop-tabletop interaction; Shaun Kane, Daniel Avrahami, Jacob Wobbrock, Beverly Harrison, Adam Rea, Matthai Philipose, Anthony Lamarca
- Codex: a dual screen tablet computer; Ken Hinckley, Morgan Dixon, Raman Sarin, Francois Guimbretiere, Ravin Balakrishnan.
- Stirring up experience through movement in game play: effects on engagement and social behavior; *Sin Lindley, James Le Couteur, Nadia Berthouze*
- More than face-to-face: empathy effects of video framing; *David T. Nguyen, John Canny*

Figure 7: A sample "Match" question. The second item matches the first two

We augmented Chang *et al.*'s work with Match questions, which are meant to measure distinctness. We prompt the rater with two items chosen from one cluster. The rater selects one matching from the same cluster from among four distractor items chosen from other clusters (Figure 7). If raters can tell which item matches, then the clusters are more likely to be distinct from each other.

We created and deployed a web-based study within our organization to test the quality of the clusters generated in the first user study. Questions were selected randomly from among the ASSIST, BASELINE, or AUTO conditions (described below). After a training round, raters were given a series of 21 outlier and 21 match questions in randomized order.

Automated Condition. A third condition was added to explore how well a fully-automated clustering algorithm could perform in comparison to human-sorted clusters. A brief note on our automatic clustering approach. We use Latent Dirichlet Allocation (LDA) instead of k-means since the task presented to users was to determine the underlying topics of the collection, assigning appropriate documents to each cluster according to how well they belong: this is precisely what LDA was designed to do [4]. LDA computes the underlying topics and estimates for each document the "responsibility" of each topic in explaining the information in that document; we interpret the topics as clusters and use the most responsible topic for a given document as its assigned cluster. The top documents belonging to each topic are those that had the highest value for this entry; those with the lowest values were left uncategorized. To best align with the clusters created by our subjects, we set the number of topics to be the average number of clusters from the subjects. We left uncategorized as many documents as the average number of uncategorized items from our subjects. We refer to this as the AUTO clustering condition. To produce four clusterings from this condition, we randomized the order of the items four times and ran LDA separately on each set.

Results

In this section we first evaluate the hypotheses and then delve deeper into a single user's results to better understand the interaction of search and recommendations.

H1: Individual users will develop different and distinct clusters.

We base our analysis for this hypothesis on the first user study. We wished to develop a measure of cluster similarity: that is, given two users, how similar are their clusters?

We pairwise compared every user's clustering to that of every other user. To measure the degree of agreement between two users, we took all pairs of items that were in the same cluster for the first user (where both those items were also filed by the second user); we then computed the fraction of those pairs which were in the same cluster for the second user. The agreement measure thus is between 0 and 1.

If the natural clusters of a dataset are the primary driver of users' clusters, then we would expect different users to create similar clusters to each other (with an agreement of near 1.0); conversely, if users' individual styles or biases have a strong effect, we would expect substantial disagreement. Users had a mean overlap of 0.49 (standard deviation 0.15) of their items. This low overlap shows that clusters were very different from each other. We consider H1 to be verified: different users created distinct clusters from the same items in a dataset.

H2: Users will cluster more items in the ASSIST condition than in BASELINE.

For each user, we counted the number of items that they successfully placed into a cluster. Figure 8 shows the number of items filed under each condition; Figure 9 shows the relative speedup between BASELINE and ASSIST. Both illustrate that ASSIST made a significant difference to filing speed, offering as much as a doubling in speed.

We used Wilcoxon signed-rank test to compare the number of items filed in the ASSIST and BASELINE conditions. There was a significant difference between ASSIST and BASELINE, z=-1.966, p=.049.

Figure 9 breaks down the *relative* speedup per user—that is, the number of items sorted in ASSIST divided by BASELINE. 12 of the 16 users were faster with assistance; 5 were 30% or more faster than in BASELINE. (Later, we look at the use of suggestions for one individual's run, which suggests that greater gains are possible.)

We consider H2 to be verified: users clustered more items in ASSIST than in BASELINE.

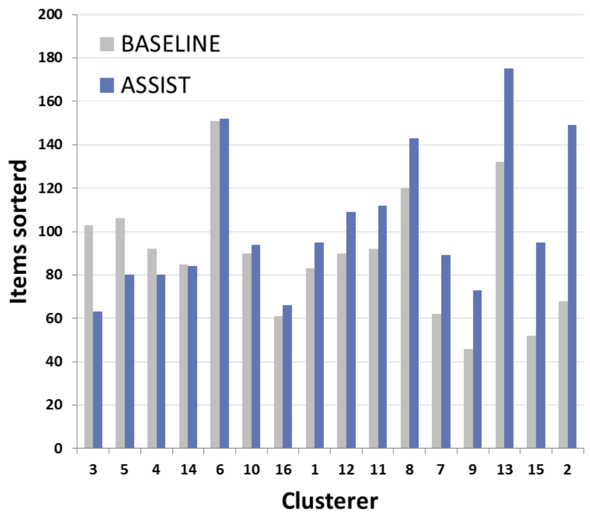


Figure 8: Number of items filed under ASSIST and BASELINE for the second (within subjects) study. Subjects are sorted by absolute difference between their ASSIST and BASELINE runs.

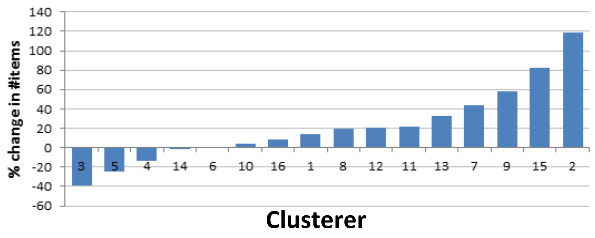


Figure 9: Relative (percentage) speedup from the BASELINE to the ASSIST condition. One user was more than twice as fast with ASSIST.

H3: ASSIST clusters will be of comparable quality to BASELINE clusters.

Our definition of quality is based on multiple raters' evaluations of clusters. Each rater answered a sample of Match and Outlier questions about the different sorters' clusters; the quality of a given sorting of a dataset is then the percentage of questions for that sorter that were judged correctly. As in H2, we base our result on the within-subjects study.

Some sorters produced clusters that the raters found easy to interpret; others produced clusters that were more difficult to understand. A set of clusters from a single user is scored as the percentage of questions (as described earlier) correctly answered by raters, and so ranges from 0.0 to 1.0. As raters selected from five possible options for Outlier questions and four options for Match questions, we place chance at 0.225.

A Wilcoxon Ranked Samples test was conducted to compare average quality of clustering in ASSIST and BASELINE conditions. There was no significant difference between ASSIST and BASELINE; z=-1.221, p =0.222.

H3 was verified: users with assistance created clusters not distinguishably worse from users without assistance.

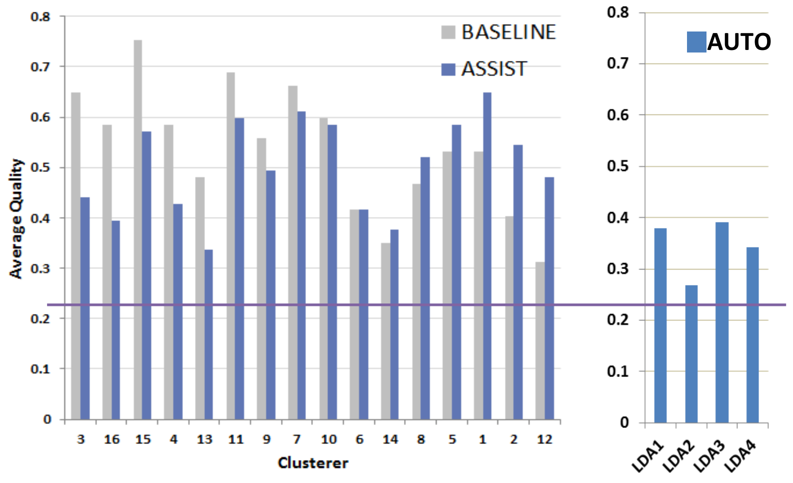


Figure 10: Support for H3 and H4. (left): Relative quality of ASSIST, and BASELINE. Bar height is the average quality for a given user. The bars are ordered by absolute difference in quality. Chance is 0.225 shown by the purple line. Figure 10 (right): Quality measurements of AUTO (4 different runs of an LDA clustering algorithm).

H4: Both ASSIST and BASELINE clusters will be of higher quality than automatically-generated clusters.

As an additional condition for the quality evaluation study, we included clusters that were automatically generated to compare with both the passive and baseline conditions. We conducted a 3×1 ANOVA on mean rater scores and found F(2,21) = 15.56, $p < 7.17e\text{-}5$. Unlike the results in H3, this test is **not** paired samples, so it is not surprising that we have weaker p-values for the ASSIST-BASELINE condition. Figure 11 illustrates the three conditions; Table 4 shows the p values for the tests.

We consider H4 to be verified: Both BASELINE and ASSIST were significantly different than AUTO, and not from each other.

Table 4: (left) Mean and standard deviation for quality of within-subjects groups. (right) Significance values for ANOVA tests between conditions. Starred entries signify significant differences (i.e., the two conditions' scores are different from each other, thus ASSIST and BASE both differ significantly from AUTO, but do not different significantly from each other.)

	Mean (sd)
ASSIST	0.58 (0.117)
BASE	0.652 (0.197)
AUTO	0.348 (0.081)

Test (difference between conditions)	p	
ASSIST-AUTO	0.001	*
BASE-AUTO	8.53e-5	*
ASSIST-BASE	0.73	

How Suggestions Helped

In order to better understand how sorters used recommendations, we can look at a sample user's track over time (Figure 12). This user, from the between-subjects study, was in the FBK ASSIST condition.

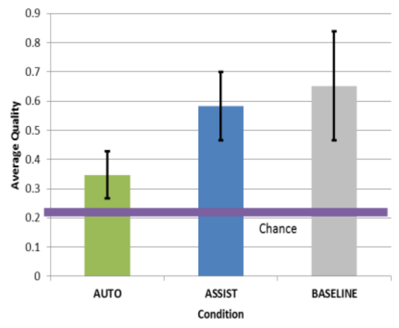


Figure 11: The relative quality of all groups across AUTO, ASSIST, and BASELINE. Bar height is the average quality; error bars represent the std. dev. ASSIST and BASELINE are each significantly different from AUTO, and not from each other.

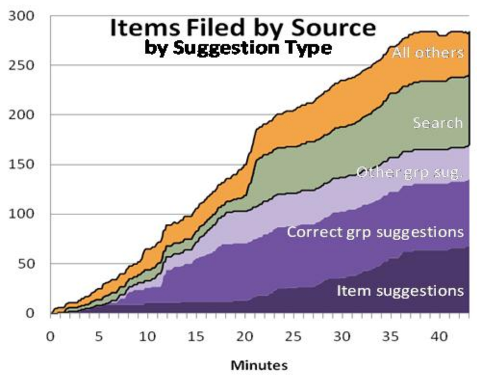


Figure 12: Stack chart showing numbers of items clustered over time for one sorter, separated by item source. Note the large proportion of items from suggestions.

A user can place documents into a cluster from three sources: they could take documents from the piles or from the desktop, carry out a full-text search, or get a suggestion from either the item suggestion or the group suggestion. We track the time at which the user dropped each item into a cluster, and the source of the item. Thus, if a user did a full-text search that revealed a set of documents, but only sorted two of them into clusters, then the "search" line would increment twice, for each of those two documents.

We can see from Figure 12 that this sorter sorted far more documents from recommendations than from search or from the pile. Text search never provided more than half of the sorted documents at any point: indeed, text search provided most of its value in one particularly fruitful search (around minute 22).

The suggestions are further subdivided into three parts. When the user clicks on a group and asks for suggestions, we distinguish between when a user places the item in the recommended group ("correct") and when a user places the item in another group ("other"). In addition, item-to-group suggestions are always available; we count these only if the user follows the suggestion.

Note that item suggestions grew rapidly over the course of the study, especially after the half-way point: at this point, the suggestions are increasingly useful and the user is able to very rapidly sort items into groups.

Discussion

Comparing iCluster to DataMountain Implicit Queries

It is interesting that we found a significant improvement in the speed of sorters in comparison to the implicit query work on DataMountain (DM-IQ) [8]. The previous work had found that sorters were slowed by recommendations; we found the opposite.

We believe that the differing structure of the learning algorithm may help explain the distinction. DM-IQ was based on a static recommendation, and did not adapt as the user placed their items. If a user's mental model matched the system's model, the suggestions would be helpful; however, if the user had a different sort in mind, then the highlighting would be distracting. There is some evidence for this distraction in the original DM-IQ work: the paper reports that users "often considered whether or not to follow the system recommendation" [8].

As our Hypothesis 1 suggests, different users' clustering were very different from each other (which [8] also finds), which would reduce the utility of static suggestions.

Starting with a Blank State

The iCluster interface begins with a blank slate: users do not get an overview of the dataset. This differs from systems like Scatter/Gather, which provide an initial structure. We received mixed feedback on this: several users preferred to start from scratch; others wanted more of an overview about the dataset as a whole.

The current form closely models the real-world task of dealing with a pile of data. However, it suffers from a slow start, with weak recommendations until clusters contain several items. The drawback to an overview is that it requires a naïve distance metric to generate the initial view; this might introduce a level of bias and possibly distrust. We expect the tradeoff between these approaches is problem- and user-dependent, and hope to further explore this in our future work.

Current Use

The system has been in use in several different groups at our organization. Each of these groups collects user feedback on software products; their interest is in organizing the feedback into groups that can be assigned to particular debugging and development teams. During periods of intensive feedback collection, iCluster has been used for triaging larger collections of data.

Scaling Up

In Basu *et al.*'s original algorithm [17], suggestions improve as users sort more items (from 25% accuracy after three examples to 40% accuracy after eight). As clusters get larger, and the user creates more of them, suggestions should become more relevant, allowing users to evaluate them more rapidly and to accept a larger percentage of suggestions. We hope to add automatic features to allow a user to rapidly accept all recommendations for a cluster, and to cluster thousands of items in the background. This should help speed the system more as users work longer.

Qualitative Feedback

After the study, several sorters asked for the system to sort their email. One former program committee member said that the system would be "awesome for a Paper Committee sorting task." Several sorters noted that even when the recommendations for a cluster were not quite right, the suggestions were often appropriate for another cluster and so were useful. We attribute this to the learner building a model of items the user was likely to cluster.

Conclusion

iCluster implements an interactive clustering algorithm which learns as users add more data. While it becomes increasingly accurate as clusters get larger, it begins to show effects almost immediately, allowing users to rapidly train the model and get useful recommendations. With the help of our adaptive tool, users were faster than without it, and produced clusters of equal quality. Clusters produced with our tool were significantly better than clusters created automatically on the same datasets. We've been able to apply this tool to a wide variety of domains, ranging from conference paper sorting, to user feedback, and are currently exploring using it to cluster medical patient progress graphs.

REFERENCES

1. Agarawala, A. and Balakrishnan, R. 2006. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proc. CHI '06.*
2. Anil K. Jain. 2010. Data clustering: 50 years beyond K-means. Pattern Recogn. Lett. 31, 8 (June 2010), 651-666.
3. Basu, S., Fisher, D., Drucker, S., Lu, H. 2010. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. *AAAI 2010*.
4. Bauer, D., Fastrez, P., and Hollan, J.. 2005. Spatial Tools for Managing Personal Information Collections. In *Proc. HICSS'05*
5. Blei, D. M., Ng, A., Jordan, M. I., and Lafferty, J. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3: 993–1022, 2003.
6. Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., and Blei, D. Reading Tea Leaves: How Humans Interpret Topic Models. In *Proceedings of NIPS 2009*.
7. Cohn, D., and Caruana, R. 2000. Semi-Supervised Clustering: Incorporating User Feedback to Improve Cluster Utility. In *Proceedings of the Conf. on Artificial Intelligence*. AAAI Press.

8.Czerwinski, M., Dumais, S., Robertson, G., Dziadosz, S., Tiernan, S., van Dantzich, M. 1999. Visualizing implicit queries for information management and retrieval. In *Proc CHI '99.*

9.desJardins, M., MacGlashan, J., and Ferraioli, J. 2007. Interactive Visual Clustering. In *Proc. of the Int'l Conf. on Intelligent User Interfaces*. ACM Press.

10.Dourish, P., Lamping, J. and Rodden, T. 1999. Building Bridges: Customisation and Mutual Intelligibility in Shared Category Management. *Proc. GROUP '99*.

11.Jones, W. and Dumais, S. 1986. The spatial metaphor for user interfaces: experimental tests of reference by location versus name. *ACM Trans. Inf. Syst.* 4, 1 (January 1986)

12.Malone, T. How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. Office Info. Sys.*, 1(1):99–112, 1983.

13.Mander, R., Salomon, G., and Wong, Y. Y. 1992. A "pile" metaphor for supporting casual organization of information. In *Proc. CHI '92.*

14.Pirolli, P., Schank, P., Hearst, M., and Diehl, C. Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection. In *Proc. CHI '96*

15.Robertson, G., Czerwinski, M., Larson, K., Robbins, D. C., Thiel, D., and van Dantzich, M. 1998. Data mountain: using spatial memory for document management. In *Proc. UIST '98.*

16.Watanabe, N., Washida, M., and Igarashi, T. 2007. Bubble clusters: an interface for manipulating spatial aggregation of graphical objects. In *Proc. UIST '07.*

17.Whittaker, S. and Hirschberg, J. The character, value, and management of personal paper archives. *ACM* Trans. on Comp.-Human Int., 8(2):150–170, 2001.

18. Zeng, H.-J., He, Q.-C., Chen, Z., Ma, W.-Y., and Ma, J. Leaning to Cluster Web Search Results. In *Proceedings of SIGIR 2004.*