A Method to Accelerate Human in the Loop Clustering

Anni Coden[*] Marina Danilevsky[†] Daniel Gruhl[‡] Linda Kato[§] Meena Nagarajan[¶]

Abstract

Data analysis tasks often require grouping of information to identify trends and associations. However, as the number of elements rises to the hundreds and thousands the cost of having a person perform the groupings unassisted quickly becomes prohibitive. Previous approaches have combined traditional clustering techniques with manual interaction steps, yielding human-in-the-loop clustering algorithms that incorporate user feedback by reweighting features or adjusting a similarity function. But in the real world, many grouping tasks lack both a feature set and a well-defined (dis)similarity metric, having only a subject matter expert with an implicit understanding of the correct relationships between elements based on the domain and the task at hand.

We present a refine-and-lock clustering interaction model and demonstrate its effectiveness for cognitive-assisted human clustering over other interaction models such as split/merge and must-link/can't-link. Our approach offers effective automatic clustering assistance even in the absence of clear features or a definitive similarity metric; ensures that every cluster has final user approval; and exhibits at least a 3.94x improvement over other interactive clustering approaches in time to completion.

Keywords: ad-hoc clustering, human-in-the-loop, interactive clustering

1 Introduction

High-value, ad-hoc, task-specific data exploration problems are ubiquitous, arising in the early stages of analysis when a clear goal has not yet been defined. Often, data must be combined into some logical groupings in order to facilitate analysis. For example, consider a company wishing to understand the types of concerns voiced on social media with a newly launched product. This sort of question often yields hundreds or thousands of unique phrases which must be abstracted into a handful of coherent categories to facilitate effective investigation. While some of these categories may be easy to elucidate (overheating problems), others may be more subjective (aesthetics), or require external domain knowledge (problems with a particular part, which the human knows is sourced from an external supplier). The category taxonomy may also vary depending on the downstream analysis: a marketing study may focus on the sentiment towards a product whereas a product analysis may target what sub populations the product is being used by. The same data now requires different categories.

Subject matter experts (SMEs) performing such ad-hoc groupings greatly benefit from machine support. Traditionally, clustering algorithms have served this purpose and has been widely studied and used in a variety of application domains. A pairwise similarity measure is required, along with either the number of desired clusters, or an appropriate threshold (if agglomerative clustering is used). The efficacy of a clustering algorithm depends heavily on specifying the appropriate similarity measures, the features underlying this measure and/or the number of desired clusters.

However, specifying the similarity measure based on some features may conceptually not be feasible. Consider a clustering task for which anyone can be a subject matter expert: organizing your kitchen. Each cabinet's contents represents a cluster, implying that similar items are placed in the same cabinet. However, where one SME may place a mixing bowl in the same cabinet as salad bowls (both are bowls), another SME will co-locate it with the cookie sheet (he uses both primarily for baking). Both SMEs generate 'correct' clusterings, and neither one will be able to find the mixing bowl in the other's kitchen, nor convince each other to reorganize their kitchen.

Visual clustering approaches attempt to address cases where similarity measures are infeasible, bypassing the automated clustering step entirely, and simply displaying elements in a graphical interface, leaving the user to do the work of moving elements into the right clustering. However, these systems generally do not scale well for larger datasets as even 100 elements are challenging to display on a single interface and may take users a very long time to manually cluster.

1.1 Task Definition To formalize our task definition, consider a collection of elements to be clustered. The correct number of clusters is unknown; there may

[*]IBM T.J. Watson Research Center, anni@us.ibm.com
[†]IBM Almaden Research Center, mdanile@us.ibm.com
[‡]IBM Almaden Research Center, dgruhl@us.ibm.com
[§]IBM Almaden Research Center, kato@us.ibm.com
[¶]IBM Almaden Research Center, meenanagarajan@us.ibm.com

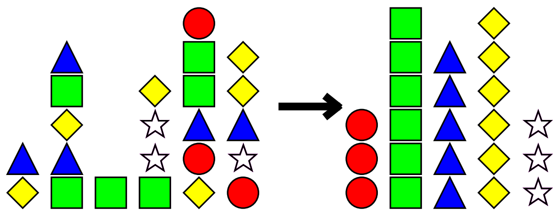

Figure 1: An initial clustering (left) must be transformed into the final clustering (right). Elements that end up in the same cluster are referred to as having the same 'color'. Note that the initial and final number of clusters may differ, and that the final clusters are not required to be of similar size.

be no well-defined features; and there is no similarity metric that can be reliably specified. On the other hand, the SME performing the clustering has an inherent understanding of the relationships between the elements, and knows a correct, or invalid cluster when he sees it.

Each element can be thought of as having a latent 'color' attribute, representing the SME's knowledge, with respect to the rest of the elements. Elements of the same 'color' are destined to be placed in the same group by the SME. The task is therefore to provide the SME with an assistive mechanism, to start from some initial clustering (see Figure 1) and modify it to the correct final clustering, using as few 'moves' as possible.

The challenge lies in defining a reasonable estimate of a pairwise element distance metric, given the lack of a traditional 'feature space.' In order to provide the SME with algorithmic assistance, not merely a visual clustering interface, these estimates must be updated and propagated with every 'move' the SME makes. This results in a constantly improving distance metric and thus hopefully element clustering. Finally, it is essential that every cluster is directly finalized by the SME, since his inherent knowledge of element 'color' cannot be formalized, and the algorithm's distance estimate is likely to be imperfect.

To address these challenges, we present `Grouper`, a novel approach that combines the strength of algorithmic clustering with the usability of the visual clustering paradigm. By allowing direct user interaction with the element clusters, and employing a variant of the triangle inequality to help propagate information about the relationships between elements, we observe at least a 3.94$\times$ acceleration of the ad-hoc grouping task.

2 Related Work

Many real-world clustering tasks do not have a single correct answer. This may be because the feature space is unavailable or impossible to express, a similarity metric cannot be defined, or the clustering of the same dataset must change depending on the SME and her task. Previous work has explored various human-in-the-loop approaches to interactive clustering[4, 9, 16]. Setting aside the purely visual systems with no underlying algorithmic assistance, much of the work relies on a user specifying or guiding a criterion or feature space on which to base the clusters. In contrast, we do not require the availability of a feature space that may be conceptually challenging to derive and agree upon for some tasks. Our intuition for similarity is derived from the elements themselves, since we recognize that a different feature space may be relevant for every clustering task, or even for individual clusters.

Interactive clustering approaches generally start with an initial cluster which the SME can them modify using operators for split or merge [7, 2] or must-link and cannot-link paradigms [18]. Other available levers are at the feature level where SMEs can focus on a subset of features to affect later clustering [3]. Prior work therefore limits human involvement to after the clustering has been performed (e.g., to subsequently merge or split clusters). In our work, human involvement albeit minimal, is present throughout the process until the SME reaches satisfactory clusters.

There has also been work in alternative clustering where the system constructs multiple clusterings and allows the SME to select between them let the user select between them [8]. Multiple clusterings can be constructed in many ways, for example by re-weighting features or changing the objective functions; however, such approaches by design require well-defined features, which may not always be attainable.

3 Grouper

`Grouper` comprises a graphical user interface (UI) for displaying and interacting with the clustering, a pairwise element distance metric with a naïve triangle inequality constraint propagation assist, and a set of actions the user can perform to affect the clustering and/or the element distance. The overall approach is as follows. An initial clustering is presented to the user in a graphical interface. For each cluster the user can 'lock' the cluster, indicating she is satisfied with it and it should no longer be changed, or 'refine' the cluster by adding or removing elements, or changing the pairwise distance of elements within a cluster[1]. After a user performs several actions using the user interface, the user

[1]Other actions are supported, such as 'search for all elements containing a string, check the correct ones and add them to the cluster', 'find me the 20 unlocked items closest to this cluster', etc., but these are out of the scope of this paper.

requests a reclustering from `Grouper`, and the process is repeated until no unlocked clusters remain. It is important to note that while every action taken by the user with the UI impacts the distance metric, only 'locking' results in permanent changes to the clustering. We refer to this paradigm as 'Refine and Lock'.

The high-level approach of `Grouper` is presented in Algorithm 1, with the rest of this section devoted to the details of the implementation.

Algorithm 1 Grouper

```
1: while Not All Clusters Locked do
2:     COMPUTE DISTANCE
3:     Cluster unlocked elements
4:     Display suggested clustering to user
5:     Cognitive assist: propagate user actions to current clustering and to distance measures
```

3.1 Refine and Lock The 'Refine and Lock' paradigm describes the user interaction aspect of `Grouper`. The clustering is refined by adding elements to or removing elements from a cluster (the latter by moving them to a generic holding cluster.) As noted in [4], users desire deterministic control over the clustering, in that they do not want an algorithmic assist to destroy correct clusters. The locking half of the paradigm caters to this important requirement: 'locking' a cluster indicates that it can no longer be modified by the cognitive assist step[2].

After the user has performed a number of refine and lock actions, she can initiate the cognitive assist step to algorithmically recluster the unlocked elements. The distance matrix is then recomputed, as described in Section 3.3, and the elements outside of the locked clusters are reclustered. The implication is that as the approximation of element similarity improves, as derived, from the user's actions the clusters will become more accurate and rapidly convergence to locked states.

This process is repeated until all clusters are locked, i.e., the user has validated the clustering fully. The various actions available via the UI all impact the constraints the system has on the distance estimates between items. (see Table 1).

3.2 Distance Measure `Grouper` employs a very flexible distance metric - it is, in fact, two distance matrices, one that holds the current estimate of the maximum pairwise distance, and one the minimum. We track an estimate range rather than a single number estimate for two reasons. First, user actions in the UI can be mapped to either a decrease in the maximum or increase in the minimum (as described in Table 1) in a way that is more flexible than perturbing a single metric. The second is that it allows us to naïvely employ the triangle inequality to propagate the constraints imposed by a user action throughout the distance matrices. This use of the triangle inequality is naïve as there is no reason the pairwise distances should exist in a 'space,' and there is no deterministic interpretation of their value. However, the assumption of transitivity it embodies (e.g, if A and B are similar, and A is very dissimilar to C, then B is probably also dissimilar to C) empirically seems to hold for most tasks. A best approximation of pairwise distance is easily obtained by averaging the max and min matrices.

[2]Locking is also when the cluster is assigned a label, and potentially a description.

3.3 Distance Computation Algorithm In practice, `Grouper` works on three $n \times n$ matrices (for a set of n elements to group). The first matrix is the maximum distance between two elements (that is, D_{ij}^{max} is the maximum distance between element i and element j). This matrix is uniformly initialized to 9 (an arbitrary choice, but yielding an intuitive scale of 1–10 for 'how similar are these' in the user interface via subtraction as it has been shown in the literature that user perceived similarity has the same properties as a distance function[15].) Self similarity applies so $diag(D^{max}) = 0$.

The second matrix is, likewise, the minimum distance between elements , and is initialized to $D_{ij}^{min} = 0$. In most cases D^{max} and D^{min} will also be symmetric.

The third matrix is a measure of semantic similarity, used as a tie breaker when two elements are otherwise identical. We choose the popular `word2vec`[14] measure for this and refer to the matrix as *semantic*, whose values are typically in the range of $[0, 1]$ from least to most similar. The similarity values themselves depend on the corpus on which `word2vec` was trained (we show results based on `word2vec` trained on multiple different corpora in Section 4). Our models are trained on unigrams; hence, we define similarity between $phrase1$ and $phrase2$ as the average of the distances between each word in $phrase1$ and each word in $phrase2$. Words which are not part of the `word2vec` model are assigned a default of 0.5 to every other word. Note, this merging of multiple 'points' in a `word2vec` space results in a collection of point-wise similarity measures, rather than discrete points in an initial space.

While we use `word2vec` for semantic similarity, there is no reason any other similarity measure could not be used instead (e.g., ontological distance if all the elements to be clustered are in a known ontology[12]). The more representative the semantic similarity measure is of the SME's envisioned element relationships, the fewer refinements will need to be done. Of course, the semantic similarity matrix can also be ignored (set uniformly to 0.5), shifting more work to the SME. The `Grouper` initialization is captured in Algorithm 2.

Algorithm 2 Grouper Initialization

```
1: function INITIALIZE
2:     $maxDistance_{ij} \leftarrow 9$
3:     $minDistance_{ij} \leftarrow 0$
4:     if $i == j$ then
5:         $maxDistance_{ij} \leftarrow 0$
6:     $semantic_{ij} \leftarrow word2vec(word_i, word_j)$
```

Every clustering iteration requires a single estimate of the distance between points, referred to as a 'working distance'. We compute this working distance matrix as the average of *minDistance* and *maxDistance*, plus an offset from *semantic* (Equation 3.1). Since the *semantic* similarity is 0 for none and 1 for exact, we subtract it from one to get a distance and then scale it appropriately[3] to reduce its impact.

$$WD = \frac{1}{2}(D^{min} + D^{max}) + K_s(1 - semantic) \tag{3.1}$$

These working distances are then used by either a `k-median`[11] (if there are a hundred or less elements to cluster) or median-based agglomerative preliminary clustering approach[13].[4] Note that the distance computation gives us a pairwise distance measure but does not define a feature space. Hence only clustering algorithms which can operate on distance measures alone can be applied, thus our choice of `k-median` and a median based agglomerative clustering.

Once the initial clustering is completed it is presented to the user via the UI for refinement. Each action the user takes has an impact on D^{min} and D^{max} as shown in Table 1. This list is illustrative, and can be extended as additional UI components are added.

We refer to the set of user refinements performed during an iteration i as R_i and the concatenation of all user refinements as $Refine()$. `Grouper` keeps track of these refinements and applies them every time a reclustering is requested.

At every iteration, we compute the current state of D^{min} and D^{max} as shown in Algorithm 3. This implements the Naïve Triangle Inequality constraint

[3]We set $K_s = 0.25$, an arbitrary value, but one we found to work well in practical application.

[4]We set k to target 6 elements per cluster, and merge any single element clusters in. Additionally, we break up clusters larger than 12 elements. These size choices are arbitrary but seem to work well for the tasks we have studied.

Refine Action	**Effect**
Remove an item i from a cluster J	$\forall j \in J \mid minDistance_{ij} \leftarrow min(9, minDistance_{ij} + K_r)$
Lock a cluster J	$\forall i, j \in J \mid maxDistance_{ij} \leftarrow min(K_l, maxDistance_{ij})$
Move an item i into a cluster J	$\forall j \in J \mid maxDistance_{i,j} \leftarrow max(0, maxDistance_{ij} - K_a)$
Thumbs up a cluster J	$\forall i, j \in J \mid maxDistance_{ij} \leftarrow min(K_t, maxDistance_{ij})$
Drag an item i out of a cluster J	$\forall j \in J \mid minDistance_{ij} \leftarrow min(9, minDistance_{ij} + K_d)$

Table 1: Grouper refine actions and effect on distance measure. We set $K_r = 2, K_l = 2, K_a = 2, K_t = 5, K_d = 1$. Locking also has a side effect on clustering, as locked clusters are no longer considered by the machine assisted clustering

propagation. There is no reason to expect that the distances created by $Refine()$ will obey the triangle inequality, but by (naïvely) assuming they do, we can propagate information through the matrices.

It is important to note that `Grouper` does not try to construct a consistent distance function, and is agnostic to the user's reasoning for choosing her particular clustering; the only goal is for the working distance to come to closely approximate the element relationships according to the user.

Formally, the Triangle Inequality says that for any three points A, B, and C:

$$\mid D_{(A,B)} - D_{(B,C)} \mid \leq D_{(B,C)} \leq (D_{(A,B)} + D_{(B,C)}) \tag{3.2}$$

Thus, $D_{A,B}$ and $D_{A,C}$ together define an upper and lower bound on $D_{B,C}$. The upper is straightforward:

$$D^{max}_{(B,C)} \leq D^{max}_{(A,B)} + D^{max}_{(B,C)} \tag{3.3}$$

but the lower is more complicated as we need to allow for potentially overlapping uncertainty ranges:

$$D^{min}_{(B,C)} \geq max(D^{min}_{(A,C)} - D^{max}_{(AB)}, D^{min}_{(A,B)} - D^{max}_{(A,C)}, 0) \tag{3.4}$$

Note that Algorithm 3 only performs necessary (non-negative) computations to update D^{min}, which allows it to scale well even as the dataset grows.

3.4 Triangle Inequality Violation Algorithm 3 assumes that the data is noise free with respect to a distance function in a space, which does not always hold true with real data. In cases where the constraint is violated it appears in the distance matrices as a 'min' that is greater than a 'max'. We address this by adding a Step 15 to Algorithm 3, which simply flips the min and max values:

Algorithm 3 Grouper Distance Computation With Naïve Triangle Inequality Propagation

```
1: function COMPUTE DISTANCE
2:     REFINE($D^{min}$,$D^{max}$)
3:     while refinements made do
4:         $\forall i, j \mid D^{max}_{ij} \leftarrow min(D^{max}_{ij}, min(D^{max}_{i*} + D^{max}_{*j})$
5:         $v_{ij} \leftarrow 0$
6:         $\forall i, j \mid v'_{ij} = max(D^{min}_{i*} - D^{min}_{*j})$
7:         if $v'_{ij} > 0$ then
8:             $v_{ij} \leftarrow v'_{ij}$
9:         else
10:            $\forall i, j \mid v'_{ij} = max(D^{min}_{*j} - D^{min}_{i*})$
11:            if $v''_{ij} > 0$ then
12:                $v_{ij} \leftarrow v''_{ij}$
13:        $D^{min}_{ij} \leftarrow max(v_{ij}, D^{min}_{ij})$
```

$$\begin{aligned} &\forall i, j \mid minDistance_{ij} > maxDistance_{ij} \rightarrow \\ &[minDistance_{ij}, maxDistance_{ij}] \\ &\leftarrow [maxDistance_{ij}, minDistance_{ij}] \end{aligned} \tag{3.5}$$

3.5 Complexity and Requirements The worst case scenario for `Grouper` is when every element has to be moved (e.g., when the original clustering puts all elements into one cluster and the users desired output is to have each element in their own cluster). Such cases are unlikely, indicating that there is no similarity between any elements in the set. It has been shown in the literature[1] that agglomerative clustering (each element in its own cluster) algorithms have a complexity $O(n2log(n))$ and divisive clustering (all elements in a single cluster) algorithms have a complexity of $O(2n)$ which, when no assumptions are made on the initial clustering, give an upper bound on the complexity of our algorithm.

Two other common human guided clustering approaches are split/merge[7], where the user either selects a cluster for the system to automatically split or selects two clusters to merge, and must-link/can't-link[18], where a user indicates pairs of elements that must or can't be in the same final cluster.

The advantage of `Grouper` over both of these approaches is that it makes no assumptions regrading the stability of the clusters[1] nor does it require a precomputation of an average-linkage tree. Further, the user is able to explicitly refine both the clustering and the dynamically computed distance metric, rather than merely specifying a split/merge or must-link/can't-link operation. Another differentiation is that `Grouper` does not require the similarity function to have a particular property (e.g. the stability property in [1]), nor an *a priori* specification of a similarity function. Clearly `Grouper` always converges, meaning that the clustering as desired by the user is always achieved and generally in significantly fewer steps than using visual clustering (split/merge or must-link/can't-link algorithms) alone.

bacon	beef carnitas
beef teriyaki	beer
brownies	cake
cheese burger	chianti
chicken enchiladas	chicken noodle soup
chicken nuggets	chicken parmesan
chicken tikka masala	chocolate milk
chocolate shake	churros
clam chowder	coffee
fish tacos	flan
french fries	garlic bread
general tso's chicken	ginger ale
green tea	grilled salmon
horchata	house salad
ice cream	italian soda
macaroni and cheese	milk
naan	nachos
onion rings	oysters
pancakes	peking raviolis
pizza	ribeye steak
saag paneer	samosa
scrambled eggs	split pea soup
steak sandwich	sweat and sour pork
tandoori chicken	tea
thai iced tea	tiramisu
toast	waffles
won ton soup	

Figure 2: Food lexicon for the Menu and Food Court clustering tasks. The foods are drawn from a large corpus of common foods and selected for commonality to both tasks.

4 Experiments

We evaluate `Grouper`'s effectiveness at assisting users with two clustering tasks performed on a single group of foods (see Figure 2). For each task there is no 'right' answer, rather, each user is working towards a clustering that reflects their view of the (culinary) world.

We further implement models of synthetic users representing `Grouper` and comparable approaches to simulate transforming an initial grouping provided by the system to each of the target groupings provided by a user (the outcomes of the two aforementioned clustering tasks). We evaluate the number of moves required to reach each user's clustering, with the intuition that a successful system is one which assists users in arriving at their ideal clustering more rapidly.

4.1 Task The two task charge statements were:

Food Court You are given a set of food items and asked to create a set of food shops containing those menu items for a food court. Each menu item must belong to exactly one food shop. You can have as many or as few food shops as you want.

Menu You are given a set of food items and asked to create a set of menu categories containing those items for a restaurant menu. Each menu item must belong to exactly one category. You can have as many or as few menu categories as you want.

Primed with the above charge statements, users employed `Grouper`'s UI to create their preferred clustering for each task. Five users each performed both clustering tasks (three starting with the Menu task and two starting with the Food Court task, and then moving on to the other task). This produced 10 'gold standard' clusters, five for each task.

4.2 Synthetic Users Obtaining accurate measures of 'how much effort' an approach takes can be difficult without a large scale user study. Since we wish to explore different clustering approaches we instead observed a few users and then implemented models of synthetic users to compare the effort it took each synthetic user to obtain each 'gold standard' clustering. Each synthetic user represents an embodiment of an human-in-the-loop clustering approach.

For evaluation we count each refinement made to the clustering as a 'move'. To compare fairly, 'creation' of a final cluster does not count as a move (thus, if the initial clustering presented by the system is perfect we consider that this result took zero moves to achieve).

For each synthetic user, an initial `k-median` clustering is performed using one of the `word2vec` *similarity* matrices. To examine sensitivity to this initial clustering we tried matrices built off the UKWAC corpus[10](computed two different ways for missing terms[5]), the Google Billion Word corpus[5] and a 2GB medical crawl by the Blekko search engine[6]. For each synthetic user we perform the ten experiments using `word2vec` built from each of the above four corpora to populate our the distance matrix (the *semantic* distance metric for Grouper). In the following subsections we describe the implementation of each synthetic user, to compare a manual baseline, split/merge, must-link/can't-link, and `Grouper`.

[5]When computing phrases one can treat missing terms as having a distance of .5 to all other terms, or decide that in the presence of a missing term the whole phrase has a distance of .5 to all other terms.

[6]https://en.wikipedia.org/wiki/Blekko

4.2.1 Manual Assignment This baseline approach models taking index cards with elements on them and moving them into piles one at a time Placing each card into its final pile is a 'move', though for fair comparison to the other approaches, the first element in a new pile does not 'count'.

Algorithm 4 Synthetic Manual Assignment

```
1: $moves \leftarrow 0$
2: while there are still cards to assign do
3:     Draw a card
4:     if it belongs in a pile already on the table then
5:         Add it to that pile and $moves \leftarrow moves + 1$.
6:     else
7:         Begin a new pile
```

Note that this is a degenerate case of Split/Merge, with every element initially in its own cluster, and the user performing one merge operation per element. Thus, with 53 elements in the Food Lexicon, it takes 53 'moves' to assign every element to its final cluster, less the number of final clusters.

4.2.2 Split/Merge Split/Merge is one of the more popular approaches for human-in-the-loop clustering[7]. The user requests that particular clusters be split or merged (relying on the existing distance metric to perform splitting), until the correct clustering is achieved.

Algorithm 5 Synthetic Split/Merge

```
1: $moves \leftarrow 0$
2: while some clusters are not monochromatic do
3:     if two clusters are the same color then
4:         Merge them; $moves \leftarrow moves + 1$.
5:     else
6:         Find the smallest cluster with the least number of colors and request it be split; $moves \leftarrow moves + 1$.
```

When a split is performed, the system takes the two items most separated by the distance metric and 'clustering' the rest of the elements in the cluster with them as centroids. In the case of distance ties the item is randomly assigned.

4.2.3 Must-link/Can't-link Another popular approach involves the user marking pairs of elements that either *must* or *can't* be in the same cluster[18]. A problem with this approach is that in the absence of a known

Algorithm 6 Synthetic Must-Link/Can't-Link

```
1: $moves \leftarrow 0$
2: while some clusters are not monochromatic do
3:     if there is an item in a cluster and another item of the same color is in a different cluster then
4:         Mark the two as must-link
5:         $moves \leftarrow moves + 1$.
6:     if there are two items in a cluster of different colors then
7:         Mark the two as can't-link
8:         $moves \leftarrow moves + 1$.
9:     Recluster by creating clusters of 1 element.
10:    Merge all must-linked clusters
11:    Merge all clusters that are not prohibited by a can't-link to the cluster closest to them using average distance
```

k there needs to be $\sum_{n=1}^{k} n$ *can't-link*s defined to prevent overmerging. In fact, it has been shown in the literature that users become exhausted when they must mark a large number of must-link or can't-link pairs [6].

4.2.4 Grouper The `Grouper` approach is outlined in Algorithm 7). It is worth noting that this synthetic user does not use the "grow cluster" option, nor the "search and add" options of the Refine and Lock interface. Additionally, they do not remove, say, two elements from an otherwise good group and then lock it. In short, the results obtained by the synthetic approach should be considered a lower bound for the efficiency of `Grouper` when used by actual people.

4.3 Scale up Effectiveness at large scale is an important question, as efficiency is less of an issue with smaller clustering tasks. However, real-world tasks, especially those that deal with extracted text, can have high variability (e.g., misspellings, abbreviations, nicknames) and thus run to thousands of phrases in a single dataset. We selected two large datasets, each of which have been clustered by an SME: a collection of FDA drugs (889 elements) and a collection of adverse drug reactions (3,145).

The FDA drugs (FDA) set includes mentions of various medications from clinical records, including a variety of misspellings, abbreviations, trade names from various countries, etc. The task was to group the elements into classes of drugs for clinical use in a general practitioner setting. These drugs could also be grouped in many other ways (e.g., method of action, method of clearing, expense, availability of over the counter alternatives, etc.), and `Grouper` could be used equally well to achieve clusterings for such other tasks.

The Adverse Drug Reaction (ADR) set includes 'in their own words' reports of patients about their experiences taking various prescribed drugs. There are a few hundred semi-hierachical categories that are typically used for side-effect labeling; for example, 'pain' is a side effect, and so is 'facial pain', 'lip pain', 'neck pain', etc. The task was to group the few thousand phrases into appropriate categories of side effects. If there are very few elements in a more specific phrase (e.g., 'neck pain') it may make more sense to group it with a broader phrase (e.g., 'pain') but this is fairly subjective. These phrases also show extremely high variability (e.g., 'multiplicative effect on my craziness' gets grouped with 'abnormal feeling').

We run Synthetic Split/Merge and Synthetic Grouper to reach the clusterings specified by the SMEs, in order to compare the efficiencies of the approaches at large scale.

Algorithm 7 Synthetic Grouper

```
1: $moves \leftarrow 0$
2: while not all clusters are locked do
3:     if a monochrome cluster is the same color as a locked group then
4:         merge it in
5:         $moves \leftarrow moves + 1$
6:     else
7:         if there are any monochrome clusters then
8:             lock one
9:         else
10:            pick the largest cluster of the least colors
11:            Remove one of the minority elements.
12:            if it is now monochrome then
13:                if it is the same color as a locked group then
14:                    merge it in
15:                    $moves \leftarrow moves + 1$
16:                else
17:                    lock it
18:            else
19:                if the removed element matches a locked group then
20:                    merge it in
21:                    $moves \leftarrow moves + 1$
22:                else
23:                    lock it
24:    Recompute working distance matrix
25:    Recluster the unlocked elements
```

5 Results

For the small scale experiments the food lexicon in Figure 2 was clustered in two ways as outlined in Section 4.1. Each of the ten clusterings were different.

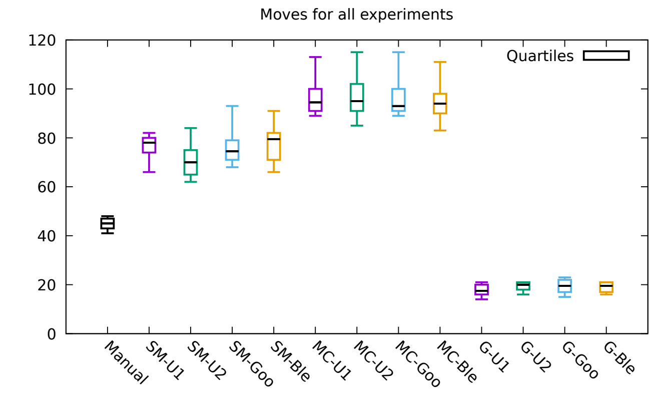

Figure 3: Whisker diagram of moves needed to transform initial clusterings to each of the 10 user-created clusterings. Split/Merge (SM), Must-link/Can't-link (MC) and `Grouper` (G) are each shown with four different semantic distance metrics - UKWAC 1 (**U1**), UKWAC 2 (**U2**), Google Billion Words (**Goo**), Blekko Medical (**Ble**). Manual assignment (Manual) is also shown for comparison. For each experimental setting, the ten clusterings are shown as a single candlestick; the top and bottom bars are the maximum and minimum moves required, the middle bar is the median, and the box spans the second and third quartiles.

Table 2 presents the Adjusted Rand Index [17] calculation between the ten generated clusterings by the five users (denoted U1-U5), demonstrating that while there is more agreement on shared tasks, it is clear each user had a different idea of the 'right' answer, including the number of clusters (menu sections or food shops) they believed to be correct.

As described in Section 4.2 we used these 10 user-specified clusterings as targets and constructed synthetic user models to perform each of the four clustering approaches (manual, split/merge, must-link/can't-link and `Grouper`). For each approach we employed one of the 4 different *similarity* matrices to cluster them. The results can be seen in the whisker plot in Figure 3; the ten experimental endpoints for each approach and *similarity* is represented by a single candlestick. The top and bottom represent the maximum and minimum moves required, the center line is the median and the box spans the second and third quartiles. This gives a good sense of how sensitive the methods were to differences in the final state of the clustering (compact candlesticks represent more robust techniques).

As noted, for each of the sets we tried four different `word2vec` models explore the sensitivity of these models to the choice of semantic similarity measure. As can be see, these had some but not much impact on the results, although again robustness can seen by the amount of variability in the candlesticks representing each *similarity*.

On average, `Grouper` outperformed split/merge by a factor of 3.94× and must-link/can't-link by a factor of 4.92×. For this simplified use of `Grouper` we find that it took an average of 0.361 'moves' per element in the initial set to fully cluster them. Although we explored the sensitivity of each model to the semantic similarity measure choice, it does not seem to have great impact on any of the approaches, with `Grouper` in particular being robust to the choice of semantic similarity function.

5.1 Scale up

We performed two large-scale clustering experiments comparing `Grouper` against Split/Merge, as described in Section 4.3. The results are summarized in Table 3.

For the Drugs set, the moves per element increased to .552 and for ADR to .535. In both these cases the human user employed additional tools in `Grouper` such as searching for candidates (e.g., searching for "cillin" found many antibiotics in one step), as well as "find me the most similar to the current locked cluster" which worked very well for the ADR case. Thus the clustering actually took the humans less time than the synthetic user would indicate. Additionally, it is worth noting that the ratio is more or less unchanged from hundreds to thousands, indicating even in challenging large cases `Grouper` is an effective approach.

Corpus	Size	Clusters	S/M	G	Ratio
FDA	889	62	1,467	491	2.99×
ADR	3,145	207	4,745	1,682	2.82×

Table 3: Grouper(G) vs. Split/Merge(S/M) efficacy on large real-world clustering tasks. The 'Ratio' column indicates the speed-up factor of Grouper over Split/Merge

6 Conclusion

`Grouper` is a novel approach that combines the strength of algorithmic clustering with the usability of the visual clustering paradigm. By allowing direct user interaction with the element clusters, and employing a variant of the triangle inequality to help propagate information about the relationships between elements, we observe at least a 3.94× acceleration of the ad-hoc grouping task. The approach gives the user more control to interact with the clusters unlike the traditional Split/Merge or Must-link/Cant-link interaction paradigms. As the scale of the data increases, `Grouper` remains a superior method, tested up to clustering a set of over three thousand elements. It performs this without explicit features, and without a distance metric that is real or stable.

We hope that this paper illustrates the opportunity for 'human-centric' designs of human-in-the-loop clus-

		Food Court					**Menu**				
		U1	*U2*	*U3*	*U4*	*U5*	*U1*	*U2*	*U3*	*U4*	*U5*
Food Court	*U1*	1	0.63	0.25	0.33	0.42	0.02	0.06	0.08	0.05	0.18
	U2	0.63	1	0.35	0.55	0.58	0.04	0.11	0.12	0.07	0.18
	U3	0.25	0.35	1	0.51	0.17	0.27	0.28	0.39	0.31	0.42
	U4	0.33	0.55	0.51	1	0.31	0.15	0.15	0.19	0.21	0.33
	U5	0.42	0.58	0.17	0.31	1	-0.01	0.05	0.07	0.03	0.1
Menu	*U1*	0.02	0.04	0.27	0.15	-0.01	1	0.52	0.59	0.48	0.39
	U2	0.06	0.11	0.28	0.15	0.05	0.52	1	0.6	0.47	0.44
	U3	0.08	0.12	0.39	0.19	0.07	0.59	0.6	1	0.65	0.4
	U4	0.05	0.07	0.31	0.21	0.03	0.48	0.47	0.65	1	0.47
	U5	0.18	0.18	0.42	0.33	0.1	0.39	0.44	0.4	0.47	1

Table 2: Adjusted Rank Index between groupings for both tasks between 5 users *(U1-U5)*. Note the top left and bottom right quadrants are more highly correlated, indicating greater agreement between users when performing the same task.

tering systems, from the distance algorithm to the user interface. In the future, we plant to further refine the approach, improve the distance computation (through linear programming optimization) and enhance the synthetic user models to better evaluate all the features of the `Grouper` system.

7 Acknowledgements.

We are grateful to Dr. Joseph Terdiman and Steve Welch for providing the manual large scale groupings, and Neal Lewis for providing the `word2vec` models.

References

[1] P. Awasthi, M.-F. Balcan, and K. Voevodski. Local algorithms for interactive clustering. In *ICML'14*, pages 550–558, 2014.

[2] M.-F. Balcan and A. Blum. *Clustering with Interactive Feedback*, pages 316–328. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[3] R. Bekkerman, H. Raghavan, J. Allan, and K. Eguchi. Ijcai. In *IJCAI'07*, pages 684–689, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[4] S. Chang, P. Dai, L. Hong, C. Sheng, T. Zhang, and E. H. Chi. Appgrouper: Knowledge-based interactive clustering tool for app search results. In *IUI'16*, pages 348–358, New York, NY, USA, 2016. ACM.

[5] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[6] J. Chuang, Y. Hu, A. Jin, J. D. Wilkerson, D. A. McFarland, C. D. Manning, and J. Heer. Document exploration with topic modeling: Designing interactive visualizations to support effective analysis workflows. In *NIPS'13 Workshop on Topic Models: Computation, Application, and Evaluation*, 2013.

[7] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *SIGIR'92*, pages 318–329, New York, NY, USA, 1992. ACM.

[8] S. Dasgupta and V. Ng. Towards subjectifying text clustering. In *SIGIR'10*, pages 483–490, New York, NY, USA, 2010. ACM.

[9] S. M. Drucker, D. Fisher, and S. Basu. Helping users sort faster with adaptive machine learning recommendations. In *INTERACT'16*, pages 187–203, Berlin, Heidelberg, 2011. Springer-Verlag.

[10] A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *WAC-4'08*, pages 47–54, 2008.

[11] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[12] L. Jing, L. Zhou, M. K. Ng, and J. Z. Huang. Ontology-based distance measure for text clustering. In *SDM Workshop on Text Mining*, 2006.

[13] C. D. Manning, P. Raghavan, and H. Schütze. Hierarchical clustering. *Introduction to information retrieval*, pages 378–401, 2008.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] R. N. Shepard et al. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

[16] A. Srivastava, J. Zou, R. P. Adams, and C. Sutton. Clustering with a reject option: Interactive clustering as bayesian prior elicitation. In *ICML WHI '16*, 2016.

[17] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *ICML '09*, pages 1073–1080, New York, NY, USA, 2009. ACM.

[18] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *ICML'01*, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.