MindMiner: Quantifying Entity Similarity via Interactive Distance Metric Learning

Xiangmin Fan[1], Youming Liu[1], Nan Cao[2], Jason Hong[3], Jingtao Wang[1]

[1]Computer Science and LRDC, University of Pittsburgh, Pittsburgh, PA 15260, USA {xiangmin, jingtaow}@cs.pitt.edu

[2]IBM T.J. Watson Research, Yorktown Heights, NY 10598, USA nancao@us.ibm.com

[3]HCII Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA jasonh@cs.cmu.edu

ABSTRACT

We present MindMiner, a mixed-initiative interface for capturing subjective similarity measurements via a combination of new interaction techniques and machine learning algorithms. MindMiner collects qualitative, hard to express similarity measurements from users via *active polling with uncertainty* and *example based visual constraint creation.* MindMiner also formulates human prior knowledge into a set of inequalities and learns a quantitative similarity distance metric via *convex optimization*. In a 12-participant peer-review understanding task, we found MindMiner was easy to learn and use, and could capture users' implicit knowledge about writing performance and cluster target entities into groups that match subjects' mental models.

Author Keywords
Mixed-Initiative Interface; Clustering; Visualization; Convex Optimization; Machine Learning.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g., HCI): User Interfaces; I.2.6. Artificial Intelligence: Learning Descriptors.

INTRODUCTION

Cluster analysis [1, 9, 10] is a common task in exploratory data mining, and involves combining entities with similar properties into groups. Clustering is desirable in that it is unsupervised and can discover the underlying structure of data without *a priori* information. However, most clustering techniques face one key challenge when used in real world applications: clustering algorithms expect a quantitative, deterministic distance function to quantify the similarity between two entities. In most real world problems, such similarity measurements usually require subjective domain knowledge that can be hard for users to explain. For example, a human instructor may easily find that the writing styles of two students are very similar to each other by reviewing their writing samples. Whereas such perceived similarities may not be reflected accurately in the distance measurement between two corresponding feature vectors.

IUI'15 Companion, Mar 29 - Apr 01, 2015, Atlanta, GA, USA
ACM 978-1-4503-3308-5/15/03.
http://dx.doi.org/10.1145/2732158.2732173

To address these challenges, we created a mixed-initiative interface, MindMiner (Figure 1), to capture users' subjective similarity measurements. MindMiner makes contributions in both interaction design and machine learning algorithms. MindMiner captures prior knowledge from users through *active polling with uncertainty* and *example based visual constraint creation*. *Active polling with uncertainty* enables users to specify their subjective opinion on the *global* importance of a feature (including the value "not sure") which improves the accuracy and speed of the clustering results. *Example based visual constraint creation* allows users to directly express their *a priori* domain knowledge via six types of constraints on the data samples being visualized. The constraint management interface allows users to browse existing examples, investigate the impact of each constraint, and discover conflicting conditions.

MindMiner also provides interface level support that uses active learning to provide optional hints as to which examples might be more helpful for clustering. We also report how inequalities are formulated to capture *a priori* knowledge from users, and how the inequalities are used in a convex optimization process to extract the "mental model" of entity similarity from users in the form of the Mahalanobis distance metric.

MINDMINER IN ACTION

We present a scenario giving an overview of MindMiner. MindMiner was originally designed for grading tasks, but can also be used in other scenarios in which capturing and quantifying users' domain knowledge on similarity is needed.

Alice is an instructor for a philosophy course that has 23 students. There are four writing assignments, and the essays submitted by students were graded via three features (accuracy, clarity, and insight). The grading was done by herself, the TA, or "double-blind" peer-review by students [4]. Even with the help from her TA and peer-review, Alice still feels it is tedious and time consuming to get a clear

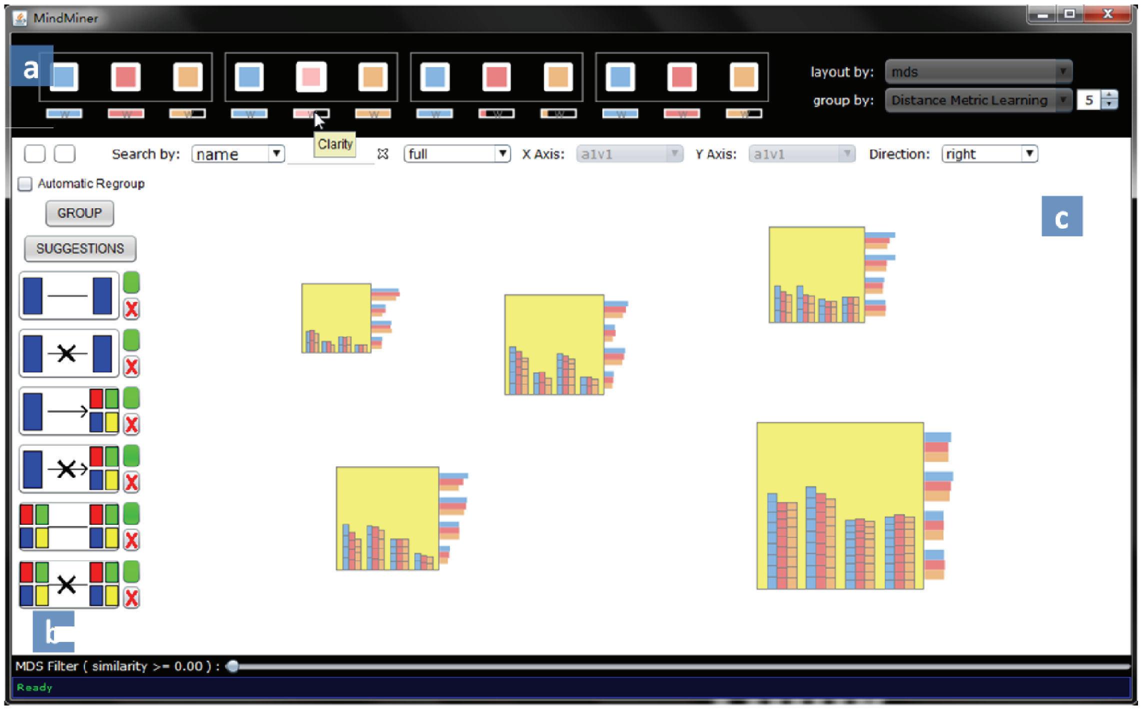


Figure 1. The primary UI of MindMiner, showing 23 students in a college-level philosophy class grouped into five clusters based on their performance (accuracy, clarity, and insight) in four writing assignments using six example constraints specified by an instructor. MindMiner consists of three parts: (a) The *Active Polling Panel* allows users to optionally indicate the importance for each measurement dimension; (b) The *Constraints Management Sidebar* displays example-based constraints collected; (c) The *Interactive Visualization Workspace* lets a user see detailed information about entities, create example-based constraints, split and combine groups, examine and refine clustering results and examine personalized groups.

picture of the overall performance of the whole class. Alice also wants to identify students with similar writing problems so she can provide customized feedback to them. Alice can use MindMiner to achieve a balance between workload and feedback accuracy.

After logging into MindMiner, Alice retrieves student performance data from a remote server. Alice believes that writing accuracy is the most important factor she cares about and clarity a close second. She is not sure about the importance of insight. Therefore, she uses the *Active Polling Panel* (Figure 1.a) to make a choice for each feature. She chooses “very important” for accuracy, “important” for clarity and “not sure” for insight.

Then Alice teaches MindMiner her subjective knowledge on performance similarity of students by showing the system some example constraints. Alice reviews detailed information of the students by mousing over the nodes. MindMiner automatically selects the most potentially informative pairs and highlights the suggestions with dashed lines. After checking two students involved in a constraint suggestion, she finds that they performed similarly. So, she drags them together, which creates a must-link constraint between the two students, telling MindMiner that these students should be grouped together. A corresponding symbol for this constraint then appears in the *Constraints Management Sidebar* (Figure 1.b). She later creates a cannot-link between dissimilar students by right clicking and dragging from one to the other. Every time Alice adds a new constraint, the distance metric learning algorithm runs a convex optimization algorithm to derive the optimized solution. The bars in the *Active Polling Panel* show the updated weights in real-time.

MindMiner also checks if there are any conflicts caused by new constraints. If so, it gives a warning by highlighting the corresponding constraints in the *Constraints Management Sidebar* using a red background. Alice checks the conflicting constraints and finds that one of the previous example constraints she created is not correct so she deletes it. Each constraint item in the *Constraints Management Sidebar* is double-linked with corresponding students via mouse hovering, so it is easy for Alice to diagnose the cause when a conflict is reported by MindMiner.

Alice clicks the “group” button located on the top of the *Constraints Sidebar* to see whether the examples provided by her are sufficient for grouping students together in a useful manner. MindMiner applies the updated distance metric using a k-means clustering algorithm, and then displays the resulting groups. Alice then checks the results and finds that the groups are not as good as she expected. She adds a few more constraints and then she checks “automatic regroup”. In this mode, once there is a new constraint, MindMiner’s learning algorithm executes and the system automatically regroups the students based on the most updated distance metric. Alice can continue this iterative process by adding new constraints, deleting existing constraints or adjusting importance levels of the features, until she gets satisfactory clustering results.

RELATED WORK

Previous efforts have been made by researchers to improve the quality of clustering using both algorithmic [10, 11] and user interface [2] approaches. For example, various semi-supervised clustering algorithms have been proposed by researchers in the machine learning community, either by adapting a similarity measure via user specified constraints or by modifying the process of determining intermediate cluster centers. However, most existing work focuses on *theoretical feasibility*: they assume users can provide sufficient, unambiguous, and consistent information to facilitate clustering before the algorithms start.

Researchers in HCI and Information Visualization have also explored the use of interactive applications for guided machine learning, in tasks such as image retrial [6], clustering [1, 9], graph navigation [3], and improving classification results [7]. However, most existing interactive clustering systems focus on conceptual demonstration and assume users can provide distinct, representative, and consistent examples with little guidance. In comparison, MindMiner provides both algorithm and interface level support for addressing the real world problems, such as handling inconsistent, ambiguous domain knowledge for exploratory clustering tasks, managing users' collected *a priori* knowledge, and achieving better clustering results with more representative constraint examples.

DESIGN OF MINDMINER

Visualization Design

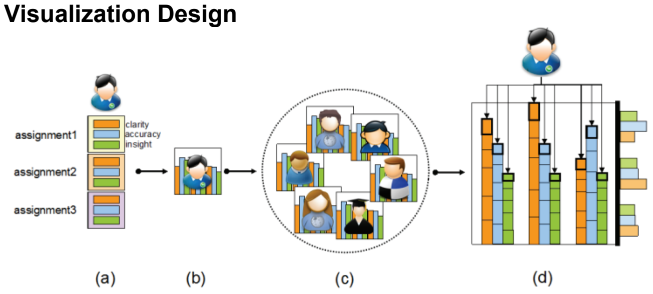


Figure 2. Visualization design of MindMiner.

We use interactive stacked bar charts in MindMiner to visualize clusters of data with multivariate features. Figure 2 illustrates an example of our design in which a student dataset is visualized. As shown in Figure 2.a, we use different background colors to illustrate different assignments, and different foreground colors to represent the different features. A student's feature vector is represented as a bar chart (Figure 2.b) in which the sizes of the bars represent the corresponding review scores. Similarly, we represent a clustered group of students (Figure 2.c) by packing all of the students' review scores together into a stacked bar chart, categorized by assignments (Figure 2.d). We also represent the averaged student feature scores of each assignment as another grouped bar chart attached to the group.

Knowledge Collection Interfaces

MindMiner offers two novel knowledge collection techniques, *active polling with uncertainty* and *example-based constraints collection*, to make it easier for end-users to externalize their implicit mental models of entity similarity. We also introduce an active learning [5] heuristic to help users provide similarity examples that are more informative to the follow-up learning algorithms.

Active Polling with Uncertainty

MindMiner lets users specify their perceived importance of each feature via *Active polling with uncertainty* (Figure 1.a). Available choices are – "not important", "important", "very important" and "not sure". This step is optional and the default choice is "not sure". These choices correspond to different parameter search spaces in the convex optimization stage. As we illustrate later, expressing *subjective certainty* can reduce the number of examples needed in the next step and improve clustering quality.

Example-based Constraints Collection

MindMiner allows users to specify their knowledge on entity similarity via examples. This approach is supported by a psychology theory [8] which suggests that people represent categories through examples or prototypes.

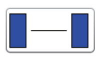
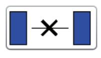
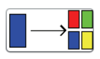
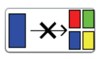
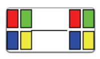
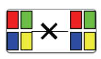
Symbol	Name	Details
	Must-link	Lets user specify that two entities should be grouped together. Leads to a new entry in equation (2)
	Cannot-link	Lets user specify that two entities should not be grouped together. Leads to a new entry in equation (3).
	Must-belong	Lets user specify that one entity should be included in a specific group. Leads to multiple must-links, and added as multiple entries in equation (2)
	Cannot-belong	Lets user specify that one entity should not be included in a specific group. Leads to multiple cannot-links, and added as multiple entries in equation (3)
	Similar-groups	Lets user specify that two existing groups should be put together. Leads to multiple must-links, and added as multiple entries in equation (2)
	Dissimilar-groups	Lets user specify that no items in the two existing groups should be put into the other group. Leads to multiple cannot-links, and added as multiple entries in equation (3)

Table 1: Symbols and descriptions of constraints supported.

MindMiner supports six types of constraints. All six constraints can be specified by users in the primary interface via mouse-based direct manipulation operations (Table 1). Constraints created are shown in the *Constraint Management Sidebar* (Figure 1.b). This sidebar allows users to browse, remove, or check the impact of each constraint created. Conflicting constraints are also highlighted in red. These constraints are used in the inequality generation step later.

DISTANCE METRIC LEARNING ALGORITHMS

Once MindMiner collects uncertainty and similarity information from users, it converts such information into a set of inequalities, formulates a convex optimization problem, and learns a distance metric from user provided similarity information.

Inequality Generation

The similarity measurement $d\,(s_i, s_j)$ between entity s_i and entity s_j is defined as:

$$\boldsymbol{d}\,(\boldsymbol{s_i}, \boldsymbol{s_j}) = \sqrt{(\boldsymbol{s_i} - \boldsymbol{s_j})\boldsymbol{W}(\boldsymbol{s_i} - \boldsymbol{s_j})^T} \qquad \textbf{(eq. 1)}$$

We transform the problem of learning meaningful distance metrics to a convex optimization problem [11]:

$$min_W \sum\nolimits_{(s_i, s_j)\in S} d^2(s_i, s_j) \qquad \textbf{(eq. 2)}$$

$$\text{s.t.} \quad \sum\nolimits_{(s_i, s_j)\in D} d\,(s_i, s_j) \geq 1 \qquad \textbf{(eq. 3)}$$

$$for\ each\ w_k:\ w_k \geq 0\ (1\leq k\leq n) \qquad \textbf{(eq. 4)}$$

Each sum item in **eq.2** corresponds to a positive constraint collected, while each sum item in **eq.3** corresponds to a negative constraint collected. It can be proven that the optimization problem defined by **eq.2** – **eq.4** is convex, and the distance metric W_{raw} can be solved by efficient, local-minima-free optimization algorithms.

Result Regularization

In order to avoid trivial solutions (especially when the number of constraints is small), we regularize W_{raw} by using Weights Bounds (WB) which is derived by *active polling with uncertainty*. Then we get a W that conforms to all the prior knowledge we collected from end-users. We apply W to the distance metric function and get the relevant distance metric. Then the distance metric W is used in k-means clustering algorithm to generate meaningful clusters.

Active Learning Heuristic

Not all user-specified examples are equally helpful. Therefore, we adopted concept of active learning [5], which allows MindMiner to identify and suggest ambiguous entity relationships that are most informative in improving the quality of distance metric learning. The informative entity pairs discovered via active learning are marked with dashed lines in the main interface.

PILOT USER STUDY

We conducted a 12-participant (5 female) user study to understand the performance and usability of MindMiner. There were two controlled tasks and one free exploration task in the study. These tasks evaluated the feasibility and usability of MindMiner in general, as well as the efficacy of the *active learning* interface and the *active polling with uncertainty* interface in specific. We defer the in-depth analysis of the experimental results to future work.

MindMiner was able to learn distance metrics and generate meaningful clustering results from all participants in all the tasks. Most participants found the MindMiner interface easy to learn and use. Sample comments include: "*visualization of students' multi-dimensional scores makes it much easier to understand,*" "*giving example-based constraints is a good way to express one's ideas, especially when the mental mind is not very clear,*" and "*it could pick the exact same important dimensions as I think.*"

CONCLUSION AND FUTURE WORK

We presented MindMiner, a mixed-initiative interface to capture domain experts' subjective similarity measurements via a combination of new interaction techniques and machine learning algorithms. MindMiner collects qualitative, hard to express similarity measurements from users via *active polling with uncertainty*, *example based visual constraint creation* and *active learning.* MindMiner also formulates human prior knowledge into a set of inequalities and learns a quantitative similarity distance metric via *convex optimization*.

We plan to further investigate the behaviors of MindMiner via quantitative analysis, improve the scalability of MindMiner from both interface and algorithm perspectives, and deploy MindMiner via longitudinal studies in the wild in the near future.

This research is in-part supported by an RDF from the Learning Research and Development Center (LRDC) at Pitt.

REFERENCES

1. Basu, S., Banerjee, A., Mooney, R., Active semi-supervision for pairwise constrained clustering. In *Proc. SDM 2004.*
2. Cao, N., Gotz, D., et al, DICON: Interactive Visual Analysis of Multidimensional Clusters, In *Proc. IEEE Infovis 2011.*
3. Chau, D., Kittur, A., Hong, J. and Faloutsos, C., Apolo: making sense of large network data by combining rich user interaction and machine learning, In *Proc. CHI 2011.*
4. Cho, K., Schunn, C., Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers and Educations*, 48(3):409-426.
5. Cohn, D., Atlas, L., Ladner, R., Improving generalization with active learning, *Machine Learning*, 15(2); 201-221, 1994.
6. Fogarty, J., Tan, D., Kapoor, A. and Winder, S., CueFlik: interactive concept learning in image search, In *Proc. CHI 2008.*
7. Kapoor, A., Lee, B., et al, Interactive optimization for steering machine classification, In *Proc. CHI 2010.*
8. Rosch, E., Mervis, C., Family resemblances: Studies in the internal structure of categories, *Cognitive Psychology*, 7(4):573-605.
9. Seo, J., Shneiderman, B., Interactively exploring hierarchical clustering results. *IEEE Computer*, vol. 35, 2002.
10. Wagstaff, K., Cardie, C., et al, Constrained K-Means clustering with background knowledge. In *Proc. ICML 2001.*
11. Xing, E., Ng, A., Jordan, M., Russell, S., Distance Metric Learning with Application to Clustering with Side-Information. In *Proc. NIPS 2002.*