Mixed-Initiative Clustering

Yifen Huang

CMU-10-005

April 2010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Tom M. Mitchell, Chair
Jaime G. Carbonell
William W. Cohen
Adam Cheyer, Siri

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Keywords: Mixed-Initiative Learning, Human Computer Interaction, Text Clustering

For my parents, Li-Chuan Kuo and Ju-Ding Huang

Abstract

Mixed-initiative clustering is a task where a user and a machine work collaboratively to analyze a large set of documents. We hypothesize that a user and a machine can both learn better clustering models through enriched communication and interactive learning from each other.

The first contribution of this thesis is providing a framework of mixed-initiative clustering. The framework consists of machine learning and teaching phases, and user learning and teaching phases connected in an interactive loop which allows bi-directional communication. The bi-directional communication languages define types of information exchanged in an interface. Coordination between the two communication languages and the adaptation capability of the machine's clustering model is the key to building a mixed-initiative clustering system.

The second contribution comes from successfully building several systems using our proposed framework. Two systems are built with incrementally enriched communication languages – one enables user feedback on features for non-hierarchical clustering and the other accepts user feedback on hierarchical clustering results. This achievement validates our framework and also demonstrates the possibility to develop machine learning algorithms to work with conceptual properties.

The third contribution comes from the study of enabling real-time interactive capability in our full-fledged mixed-initiative clustering system. We provide several guidelines on practical issues that developers of mixed-initiative learning systems may encounter.

The fourth contribution is the design of user studies for examining effectiveness of a mixed-initiative clustering system. We design the studies according to two scenarios, a learning scenario where a user develops a topic ontology from an unfamiliar data set, and a teaching scenario where a user knows the ontology and wants to transfer this knowledge to a machine. Results of the user studies demonstrate that mixed-initiative clustering has advantages over non-mixed-initiative approaches in terms of helping users learn an ontology as well as helping users teach a known ontology to a machine.

Acknowledgments

It is a great pleasure to be Tom Mitchell's student. He has taught me how to do good research and, more importantly, demonstrated me a great example of excellent thinkers. His insights have shaped up this thesis topic and kept me focused on important issues in the big picture. His patience has supported me through the down times and his encouragement has made me not afraid of pursuing difficult research questions. He has also painstakingly guided me how to properly present my work. I couldn't imagine having a more ideal advisor.

I would also like to thank my thesis committee members: Jaime Carbonell, William Cohen, and Adam Cheyer. They have provided many helpful comments and suggestions on related research ideas, theoretical methodologies, evaluation approaches, and practical issues in intelligent assistent systems that make this thesis more complete. Though she was not a part of my committee, I want to thank Carolyn Rose for guiding me through the user study design and result analysis.

Thanks to Sophie Huang and Dinesh Govindaraju. Without their hard work, we couldn't have accomplished the study on activity extraction, which directly stimulated this thesis work. I am grateful to have the chance to work with members of the CALO project, espeically Richard Wang, Einat Minkov, and Vitor Carvalho. They have provided insightful discussions and exchanged with me great research ideas.

My dear officemates, Andy Schlaikjer, Justin Betteridge, Jae Dong Kim, Yang Rong, and Jie Lu, are awesome! They have made offices in both NSH 4533 and GHC 6411 a great place for study and for fun. My dear landlords, Etty Reut and Barbara Milch, are not only landlords. Their cheerful family makes me feel like home in Pittsburgh. I also owe many thanks to my best friend, Mei-Hui Lin, for her amazing friendship and support.

Last, it is always not enough to express my appreciation to my parents for their unconditional love, and my partner, Ming-Yu Robert Chen, for his warm companion.

Contents

Chapter 1

Introduction

Historically, text clustering has been approached using fully autonomous algorithms. We consider here mixed-initiative clustering involving an interactive loop between a machine and a user. To illustrate the idea, consider the problem of organizing a large collection of research papers into a set of research categories that is meaningful for you. A fully autonomous clustering algorithm will produce a clustering that optimizes certain statistical properties, but it is unlikely to match your own understanding of research sub-areas because the the statistical distributions of words do not capture the semantic subtleties you have in mind. Furthermore, manually organizing these papers is also unattractive because it is both tedious to assign thousands of papers to categories and also because you may not know the optimal categories until you spend enough effort going through the actual data.

This thesis studies this new type of clustering – mixed-initiative clustering – as an interactive learning process between a machine and a user to solve the clustering problem together. We want to emphasize that mixed-initiative clustering focuses on not only the machine clustering perspective, but also on helping a user understand the current clustering result and modify the result easily to a better revision according to the user's evolving understanding of the data. Furthermore, we consider this study on mixed-initiative clustering as a case study of a more general class of mixed-initiative leaning problems.

The idea of studying mixed-initiative clustering comes from an early work of extracting user activities on a personal workstation.

1.1 Activity Extractor

In 2003, I joined a research project called CALO, which was abbreviated for "Cognitive Assistant that Learns and Organizes" [25]. The project spanned across several machine learning research areas in an attempt to build an intelligent assistant for workstation users [8]. My participation to the CALO project was to extract user activities on a workstation and identify what activity the user is working on so the intelligent assistant could provide activity-specific information or services to the user.

In this research [38], we attempted a combinatorial approach consisted of clustering emails based on their text content and social network cliques, and constructing a structured representation of each cluster (activity), associating calendar meeting and person names with the activity. Figure 1.1 is an example of our activity exaction results. We obtained some good activity descriptions by this combinatorial approach even from noisy clustering as long as the majority of documents in a cluster were associated with the same activity.

- **Activity Name**: CALO ActivityExtractor
- **Keywords (omitting person names)**: ActivityExtractor, TFC, IRIS, clustering, heads, emails, collected, clusters, SRI, ...
- **Person Names**: Adam Cheyer (0.49), Ray Perrault (0.36), Hung Bui (0.32), Melissa Beers (0.30), James Arnold (0.28), Jack Park (0.26), Sophie Wang (0.25), Tomas Uribe (0.25), jeanne ledbetter (0.24), Leslie Pack Kaelbling (0.24), . . .
- **Meetings**: CALO TFC telecon (0.59), CALO phone call (0.55), SRI Meeting - Chicago (0.48), SRI TFC Telecon and quarterly rpt (0.47), SRI visit. Bill and Ray. Call Melissa Beers when arriving (0.47), CALO annual mtg at SRI (0.45), . . .
- **Primary Senders**: tom.mitchell@cmu.edu (75), sophie.wang@cs.cmu.edu (20), adam.cheyer@sri.com (16), perrault@ai.sri.com (14), tgd@eecs.oregonstate.edu (13), . . .
- **Primary Recipients**: tom.mitchell@cmu.edu (94), adam.cheyer@sri.com (40), william.cohen@cs.cmu.edu (35), perrault@ai.sri.com (19), . . .
- **Emails**: [email125], [email72], . . . (245 emails in total)
- **User Activity Fraction**: 245/2822=0.086 of total emails
- **User Involvement**: user authored 30% of email (default 31%)

Figure 1.1: An example of an activity description created automatically from clustering and information extraction.

The lesson learnt was that incorporating unsupervised clustering and information ex-

traction techniques enhances a user's comprehension of autonomously generated results. However, the overall performance was still far from perfection, which we suspected was what an average user would expect from an intelligent assistant. By investigating possible reasons behind the huge gap between an autonomous clustering result and a user's ideal result, we identified three major problems (the clustering mismatch problem, the user teaching problem, and the user learning problem) in the unsupervised clustering approaches. The continuing research to tackling these problems led to this thesis work on mixed-initiative clustering.

1.2 From Unsupervised Clustering to Mixed-Initiative Clustering

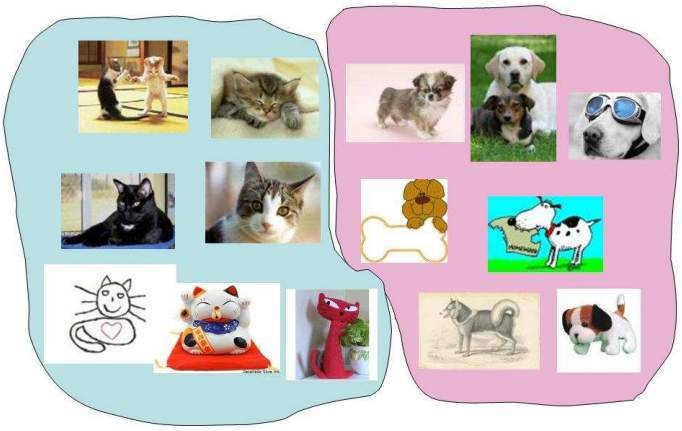
1.2.1 Clustering Mismatch Problem

The first and obvious problem of unsupervised clustering is the difference between a machine's autonomous clustering result and a user's ideal result. We call this difference "clustering mismatch." An unsupervised clustering algorithm typically produces a clustering that optimizes certain statistical properties of data according to its model assumption, which is unlikely to match a user's understanding of major topics in the data. In practice, an autonomous clustering generated by an unsupervised clustering algorithm is rarely perfect to users.

Let's use a fabricated task of sorting a set of pictures into two clusters to visualize clustering mismatch. Figure 1.2.1(a) represents a possible autonomous clustering for this task, while Figure 1.2.1(b) represents that a user named Maureen wants to have a cluster of cats and a cluster of dogs, in which two semantic meaningful topics are applied as her clustering model for this sorting task. From Maureen's point of view, there are several errors in the autonomous clustering (Figure 1.2.1(a)) that contradict her clustering model (Figure 1.2.1(b).)

Furthermore, different users may have different preferences of meaningful topics about the same data. Given the same set of pictures as in the above mentioned task of separating them into two groups, Figure 1.2.1(b)(c) illustrates two different users' clustering models: Maureen wants to highlight the difference between animal species while Joe wants to emphasize the difference between real animals and animated ones. As a consequence,

(a) A possible clustering result generated by a computer on this set of pictures.



(b) Maureen's preference of splitting the same set of pictures is by animal species.

(c) Joe's preference is to separate real animals to animated ones.

Figure 1.2: A fabricated task of sorting a set of pictures into two clusters visualizes the clustering mismatch problem. Due to diversified user preferences on what are meaningful topics for this clustering task, the clustering mismatch for Maureen, the difference between (a) and (b), is different from the clustering mismatch for Joe, the difference between (a) and (c). User involvement is important for a machine to correct clustering errors and learn the right set of topics for each user.

even in the perfect situation where an unsupervised clustering algorithm can produce a clustering result of no clustering mismatch, without knowing the user is Maureen or Joe, a machine still doesn't know how to choose between more than one possible user clustering model. User involvement is important for a machine to correct clustering errors and learn the right preference for each user. The most popular user involvement is requesting a user to label some examples for each target cluster. In machine learning literatures, the study of using a small amount of labeled examples to improve clustering quality for a majority of unlabeled examples is called semi-supervised clustering [39][7][26].

1.2.2 User Teaching Problem

Once a user is involved in a clustering task, the user teaching problem asks what is a good communication language a user can use to teach a machine. The teaching-by-example approach in semi-supervised clustering can be considered as one specific way of communicating a user's clustering model to a machine. We believe it is not the only way. For example, in the fabricated task, Maureen wants a cluster of cats and a cluster of dogs. Using the teaching-by-example approach, Maureen is restricted to teach a machine some examples of cats and dogs, or correct errors one by one in the mismatched clustering result, e.g., Figure 1.2.1(a). Wouldn't it be more intuitive for her to teach a machine "I want a cluster of cats and a cluster of dogs" directly?

The challenges for addressing the user teaching problem include (1) enriching intuitive types of user feedback so that a user can communicate abstract ideas about her clustering model to a machine, and (2) developing machine clustering algorithms so a machine can adapt to new user feedback types.

1.2.3 User Learning Problem

The user learning problem addresses a even more fundamental problem in a clustering task: when a user is not familiar with data and still needs time to figure out what are meaningful topics in the data, can a clustering system provide enough information to assist the user finding these meaningful topics? In other words, it is necessary to enrich a machine's communication to a user so the user can understand the data and identify meaningful topics in the data quickly. For example, keywords in Figure 1.1 help a user figure out which activity this cluster is about better than reading content of 245 emails clustered to this

cluster. Solutions to this problem are critical because a user who resorts to clustering techniques usually wants to discover new topics in the data rather than teaching a machine a clustering model she already knows. The user learning problem is often not addressed in semi-supervised clustering, because typically, semi-supervised clustering assumes its examples are labeled by an oracle user, meaning the user understands targeting categories a priori.

We propose to solve the user learning problem through mixed-initiative clustering. We believe when a user cannot be an oracle knowledge provider to a machine, she can still be a good collaborator as long as a machine and a user are able to provide useful guidance to each other.

1.3 Thesis

The goal of this research is to move unsupervised and semi-supervised clustering forward to mixed-initiative clustering in order to solve the clustering mismatch problem, the user teaching problem, and the user learning problem. In order to achieve this goal, mixed-initiative clustering should enhance the communication between a user and a machine so they can learn and teach each other efficiently, and be able to interact with each other in real time. *We hypothesize that mixed-initiative clustering can help a user achieve better clustering results than non-mixed-initiative approaches due to the enriched communication and the interactive learning and teaching between a user and a machine.*

Contributions of this thesis include a framework for mixed-initiative clustering, successfully building systems with enriched communication languages and real-time interaction, and design of user studies for examining effectiveness of a mixed-initiative clustering system. Results of the user studies prove our hypothesis is correct – mixed-initiative clustering indeed has advantages over non-mixed-initiative approaches.

The remaining of this thesis is organized as follows. Chapter 2 introduces our framework for mixed-initiative clustering. Chapter 3 and Chapter 4 describe the details of enriching communication in mixed-initiative clustering systems and practical issues encountered when building an interactive system. In order to test the hypothesis of this thesis, Chapter 5 proposes a user study design and its experimental results to examine the effectiveness of mixed-initiative clustering. The last chapter concludes this thesis study.

Chapter 2

Framework of Mixed-Initiative Clustering

Mixed-initiative clustering consists of an interactive loop between a machine and a user in which the machine and the user work as partners. They need to learn from each other and teach each other about their up-to-date model revisions, which are expected to converge to each other. The communication between the two are carried out through an interface specifically designed for mixed-initiative clustering. Figure 2.1 depicts this relationship.

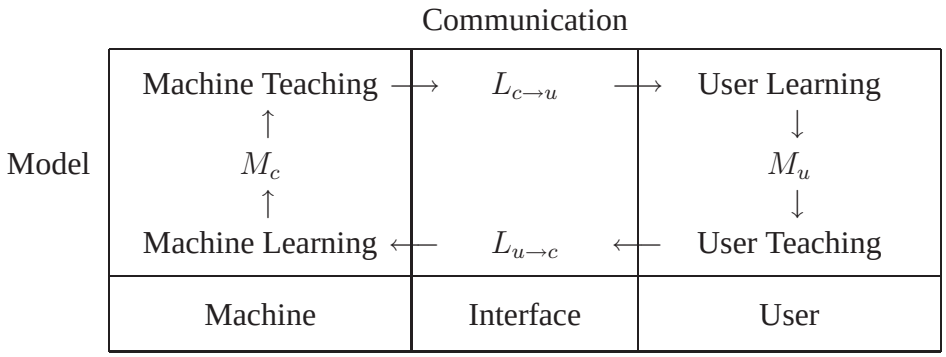


Figure 2.1: The framework of mixed-initiative clustering. A mixed-initiative clustering system consists of a machine and a user. They communicate with each other through an interface. The interactive loop iterates through the machine learning phase, the machine teaching phase, the user learning phase and the user teaching phase.

Let's start to examine the interactive loop from the machine's side. A machine learns its clustering model, M_c, from statistical distributions of words in the text corpus and from all previously accumulated user feedback. Then the machine needs to teach its current model to its user partner. Since most clustering models are not in a human-readable form, a machine needs to extract human-readable properties of M_c such as a graph depicting its hierarchy of clusters and keywords associated with each cluster in the hierarchy. In addition to properties extracted from M_c, information of instances and target clusters is also necessary for a user to gain basic understanding of a clustering task. Different property types describe information about the clustering task and M_c from different aspects to a user. The computer-to-user language, $L_{c \to u}$, is defined by all types of properties that the computer uses to communicate to the user.

The user side also has a learning phase followed by a teaching phase. In the learning phase, a user develops her own clustering model, M_u, from reading the machine-proposed properties in the interface. A user's clustering model is an ontology of meaningful topics discussed in the data. [1] The user then teaches her ontology to the machine through confirming good properties and correcting inappropriate properties. This is also known as user feedback, and all user feedback types allowed in the interface define the user-to-computer language, $L_{u \to c}$. As this process iterates, the agreement between the user and the machine on all properties increases. The machine's clustering model, which is learned from these properties, and the user's clustering model, which the user applies in modifying inappropriate properties during the process, can hopefully converge.

2.1 Active Learning as a Special Case of Mixed-Initiative Clustering

Active learning[48][10] is a subfield in machine learning that also consists of an interactive loop between a machine and a user. In the interactive loop of active learning, the machine asks an oracle user a query most likely to decrease overall uncertainty, and the oracle user answers the machine by providing a label for the query. A typical query posted by an

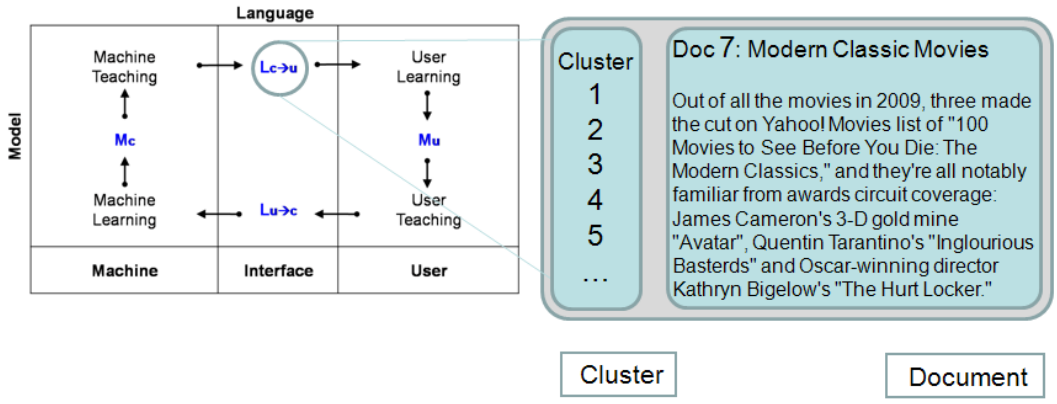
[1] We use "user clustering model" and "user ontology," interchangeably in this thesis. "User clustering model" emphasizes that it is the counter-part of the machine's clustering model in the mixed-initiative clustering framework. "User ontology" emphasizes that a user excerpts from her real world knowledge some meaningful topics into a specialized hierarchical ontology in order to solve a clustering task.

active learner is one unlabeled data instance, for example, a document in text clustering. Through the interaction, an active machine learner is expected to progressively shrink the plausible region of its model hypothesis space.

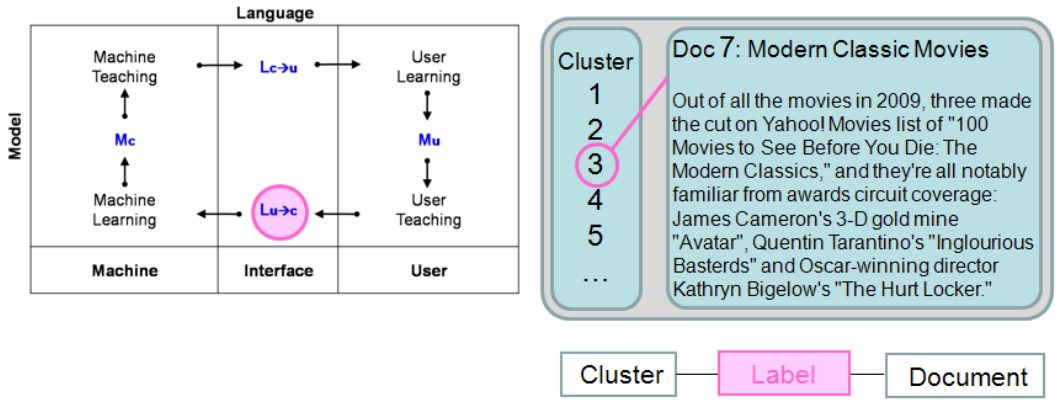
Active learning can be considered as a special case of mixed-initiative clustering. The research of active learning mainly focuses on the machine learning phase in our framework. Strategies are proposed to decide which query is the best query to ask a user. The machine teaching phase presents the list of target clusters and the query selected by the active learner. No additional information is extracted to further explain the query to the user. Since the user in active learning is assumed to be an oracle user, which means the user already establishes a clustering model of the target clusters, the user learning phase is neglected in active learning. The user identifies the cluster label of the query in the user teaching phase by applying the existing user model to the query.

Figure 2.2 illustrates the idea of applying our framework to analyze communication languages in active learning. Although communication in active learning study is mostly simulated, we visualize how an active learning interface for a text classification task should look like on the right hand side of the figure. The computer-to-user language, $L_{c \to u}$, summarizes what types of properties a machine discloses to a user through the interface. In the active learning case, its computer-to-user language contains only the basic clustering task information of target clusters and (unlabeled) instances one at a time. We present this computer-to-user language as two unfilled rectangles, one for target clusters and one for queried documents, underneath the simulated interface. There is no edge between the unfilled cluster and document rectangles because the document-to-cluster property type is not included in this $L_{c \to u}$. In the other communication direction, the user-to-computer language, $L_{u \to c}$, defines what types of user feedback are allowed in the interface In active learning, $L_{u \to c}$ contains only one feedback type, giving an instance a cluster label. In other words, the labeling feedback adds new document-to-cluster properties to share with a machine. Beneath the interface illustration, we represent the user-to-computer language by drawing a new edge between clusters and instances and highlighting the labeling feedback type in a colored rectangle. The communication languages in active learning are quite primitive because an oracle user doesn't need to learn and always knows how to answer.

Proactive learning study [14] addresses the unrealistic oracle user assumption in active learning and argues the importance to learn from imperfect oracle users who make individual errors, and sometimes are reluctant to give feedback. However, this study still



(a) In an active learning interface, the computer-to-user's communication, $L_{c \to u}$, contains target clusters and one unlabeled instance.



(b) The user reads the document and then labels the document to one of target clusters. The user-to-computer's communication, $L_{u \to c}$, contains one feedback type, giving an instance a cluster label.

Figure 2.2: An illustration of communication languages used in active learning.

assumes that a user's role is a teacher and a machine's role is a learner so it suggests a machine-learning centric approach – combining teaching efforts from multiple imperfect oracle users. Mixed-initiative clustering, on the other hand, suggests a user-learning centric approach to drop the unrealistic oracle user assumption. It treats a non-oracle user as a collaborator who can learn from and teach a machine. In order to achieve this goal of collaboration, mixed-initiative clustering needs enriched computer-to-user communication so a user can learn effectively and efficiently from a machine, and enriched user-to-

computer communication so a user can teach a machine in intuitive ways. The primitive communication languages in active learning would fail to assist a collaborative user in both communication directions.

2.2 Enriching Communication Languages

Good communication between a user and a machine is the key to the success of a mixed-initiative system. However, although a person can communicate easily with another person through common natural languages, and a machine can communicate with another machine by following protocols, communication between a machine and a user is not easy due to the fundamental differences between natural languages and computer protocols.

If we want to embed natural language communication in mixed-initiative clustering, it is necessary to design a dialogue system capable of discussing a clustering task, but that would divert the main focus of this study. On the other hand, a human user is more intelligent than a machine. She can understand what a machine says more easily than vice versa. Thus, we choose to define the **computer-to-user language**, $L_{c \to u}$, as property types that a machine knows how to exact, and the **user-to-computer language**, $L_{u \to c}$, as user feedback types that an interface is designed to support. Given the definitions of communication languages, **communication enrichment** is equivalent to adding new property types into the computer-to-user language, $L_{c \to u}$, and adding new user feedback types into the user-to-computer language, $L_{u \to c}$.

2.2.1 Computer-to-User Communication Language

The goal of the computer-to-user communication is to introduce the clustering task and describe its current clustering model, M_c, to a user. Each property type included in the computer-to-user language, $L_{c \to u}$, should contribute to this communication goal.

Formally speaking, a machine in a mixed-initiative clustering task tries to group N instances $\{X_1, \ldots, X_N\}$ into $\widehat{C}$ clusters $\{\widehat{S}_1, \ldots, \widehat{S}_{\widehat{C}}\}$, and a user also tries to group the same set of instances into C clusters, $\{S_1, \ldots, S_C\}$. The content of an individual instance, $X_i \in \{X_1, \ldots, X_N\}$, is a fundamental *single instance content* property that can be shown in an interface. The label of each individual machine-learned cluster, $\widehat{S}_j \in \{\widehat{S}_1, \ldots, \widehat{S}_{\widehat{C}}\}$, is a *single cluster label* property for a clustering task.

Let X_i indicate the i^{th} instance, and $\widehat{Y}_i$ indicate the cluster to which X_i is assigned using the machine's clustering model. $\widehat{Y}_i$ is derived from the following function, $M_c(X_i) = \widehat{Y}_i \in \{\widehat{S}_1, \ldots, \widehat{S}_{\widehat{C}}\}$, where M_c is the machine's clustering model that predicts the cluster association of each instance. Similarly, we use Y_i to refer to a user's clustering result of X_i, $M_u(X_i) = Y_i \in \{S_1, \ldots, S_C\}$, where M_u is the user's clustering model. An optimal mixed-initiative clustering is achieved when each machine-learned cluster corresponds to one topic in a user's clustering ontology, $\forall \widehat{S}_j \in \{\widehat{S}_1, \ldots, \widehat{S}_{\widehat{C}}\}, \widehat{S}_j \in \{S_1, \ldots, S_C\}, \widehat{C} = C$, and each instance-to-cluster property agrees with a user's ideal clustering result. $M_c(X_i) = \widehat{Y}_i = Y_i = M_u(X_i)$.

Each machine's prediction, $M_c(X_i) = \widehat{Y}_i$, can be shown as an *instance-to-cluster property* in the interface. In some clustering models, especially similarity-based models, an additional *instance-to-instance property* type, $M_c(X_i) = M_c(X_i')$ or $M_c(X_i) \neq M_c(X_i')$, can be derived to indicate if two instances should or should not belong to a same cluster.

For the machine learning purpose, each instance, X_i, is typically represented as a features vector, $\langle F_1(X_i), \ldots, F_M(X_i)\rangle$, where $F_m(X_i)$ is a feature function that extracts a specific piece of information from an instance. For example, the bag-of-word representation in text clustering consists of a set of feature functions, each feature function counts the number of times a particular word appears in the content of X_i. Often, multiple types of features can be extracted from instances. While representing an instance as a feature vector is good for machine learning, representing a cluster by key features is good for user learning because it tells a user what features contribute greatly to the machine's instance-to-cluster predictions, and hint the difference between the target cluster and remaining clusters. We call key features of a cluster *feature-to-cluster properties*.

In addition, hierarchical clustering provides an *cluster-to-cluster property* type that depicts relationships between two clusters. For example, if a cluster is a child cluster of another cluster in a machine's proposed hierarchy, it tells a user that the machine thinks the parent cluster corresponds to a more general topic, and the child cluster corresponds to a sub-topic of the general topic.

Below, we give a detailed description for each of the above mentioned property types that a computer-to-user language, $L_{c\rightarrow u}$, may include and use to communicate with a user in the interface. We need to point out that this is not an exhausted list.

- **single instance content**: This property type refers to showing the whole content of one instance, X_i, in the interface, e.g., displaying the text content of a document for

text clustering. Contents of data instances, $\{X_1, \ldots, X_N\}$, are basic information for a clustering task.

- **single cluster label**: One single cluster label, $\widehat{S}_j$, corresponds to one target cluster in the clustering task. This property type assumes the existence of the cluster. In active learning or semi-supervised clustering tasks, the number of target clusters, C, is fixed and cluster labels are pre-determined. However, the best value of C, and the best set of cluster labels may not be known in advance for a mixed-initiative clustering task. In addition, a system can apply an automatic naming algorithm to generate meaningful cluster labels than cluster indices.
- **instance-to-cluster property**: This property is obtained by a clustering predicted by the machine's clustering model, $M_c(X_i) = \widehat{Y}_i \in \{\widehat{S}_1, \ldots, \widehat{S}_{\widehat{C}}\}$. An instance-to-cluster property shows which cluster label an instance is assigned to, $\widehat{Y}_i = \widehat{S}_j$.
- **instance-to-instance property**: An instance-to-instance property shows a pair of instances in the interface and tells a user that a machine thinks these two instances belong or don't belong to a same cluster. This property type is based on the model prediction of two instances: $\widehat{Y}_i = \widehat{Y}_j$ or $\widehat{Y}_i \neq \widehat{Y}_j$.
- **feature-to-cluster property**: A feature-to-cluster property indicates that a feature is a key feature for a cluster. Given instance feature vectors, $\langle F_1(X_i), \ldots, F_M(X_i)\rangle$, and a clustering of $M_c(X_i) = \widehat{Y}_i,\ \ \widehat{Y}_i \in \{\widehat{S}_1, \ldots, \widehat{S}_{\widehat{C}}\}$, a feature selection algorithm, $FS(F_m, \widehat{S}_j | \{X_i, \widehat{Y}_i\})$, measures how representative a feature F_m is to a cluster $\widehat{S}_j$. Top K features with the highest representative scores for cluster $\widehat{S}_j$ become the cluster's feature-to-cluster properties.
- **cluster-to-cluster property**: This property type, which is introduced by hierarchical clustering, considers relationships between two clusters in a hierarchy. A hierarchy of clusters shown to a user consists of several parent-cluster-to-child-cluster properties in the form of $Parent(\widehat{S}_i) = \widehat{S}_p$, where cluster $\widehat{S}_p$ is a parent cluster of cluster $\widehat{S}_i$. A child-cluster-to-parent-cluster property, $\widehat{S}_i \in Child(\widehat{S}_p)$, is a reverse property to $Parent(\widehat{S}_i) = \widehat{S}_p$. A symmetric cluster-to-sibling-cluster property, $\widehat{S}_i \in Sibling(\widehat{S}_j)$, can be also derived from two parent-cluster-to-child-cluster properties if $Parent(\widehat{S}_i) = \widehat{S}_p \wedge Parent(\widehat{S}_j) = \widehat{S}_p$. We refer to all these properties in a hierarchy as the cluster-to-cluster property type.

Many information visualization techniques can be applied to show properties in an

interface. For example, a list of feature-to-cluster properties can be shown as tag cloud.

2.2.2 User-to-Computer Communication Language

The user-to-computer communication can be considered as the user's response/critique to the computer-to-user communication. A user teaches a machine her updated topic ontology by confirming machine-proposed properties that fit her ontology, and correcting inappropriate machine-proposed properties that contradict her clustering ontology. In future revisions, a machine can learn to anchor confirmed properties, not to reproduce the original properties before correction, and to generate the corrected properties.

Most property types included in a computer-to-user language, except the intrinsic single instance content property type, can be confirmed, or corrected by a user, and there can be more than one method to correct a property. For example, disapproval/removal feedback corrects the wrong existence assumption of a property, addition feedback corrects the wrong non-existence assumption of a property, and the moving feedback corrects a wrong association relationship to a right one. A user feedback type corresponds to a specific confirming/correcting method on a specific property type. The user-to-computer language, $L_{u \to c}$, includes all user feedback types that a mixed-initiative interface provides for its users.

Table 2.1 lists several possible user feedback types to be included in an mixed-initiative clustering interface for each property type mentioned in Section 2.2.1. It also describes in which situations a user wants to give these types of feedback, and how to translate these feedback types into property anchoring and property modification for machine learning. The same as the property type list, this feedback type list is not an exhausted list.

Table 2.1: This table lists possible feedback types according to their target property types, describes situations in which a user may find these feedback types useful, and illustrates implication of machine learning in terms of property anchoring and modification.

$L_{c\to u}$	$L_{u\to c}$
single cluster label: $\widehat{S}_j$	**confirm** When a user thinks a cluster corresponds to a topic in her clustering model, $\widehat{S}_j \in \{S_1 \dots S_C\}$, she can confirm $\widehat{S}_j$ so this property can be anchored in future revisions of $\{\widehat{S}_1 \dots \widehat{S}_{\widehat{C}}\}$. **disapprove** When a cluster doesn't relate to any topic in a user's clustering model, $\widehat{S}_j \notin \{S_1 \dots S_C\}$, she can disapprove/remove $\widehat{S}_j$ so it won't appear again in future revisions, e.g., $\{\widehat{S}_1 \dots \widehat{S}_{\widehat{C}}\} \setminus \widehat{S}_j$. **add** When a user finds a topic in her clustering model is missing, $S_j \notin \{\widehat{S}_1 \dots \widehat{S}_{\widehat{C}}\}$, she can add a cluster to represent this topic. In future revisions, a machine should propose $\{\widehat{S}_1 \dots \widehat{S}_{\widehat{C}}\} \cup \{S_j\}$ as its single cluster label properties. **modify** A user can modify a cluster label which she is not satisfied with. This feedback makes the following property modification: $Label(\widehat{S}_j) =$ 'unsatisfying label' $\Rightarrow$ $Label(\widehat{S}_j) =$ 'good label'. This feedback also implicitly confirms the cluster, $\widehat{S}_j \in \{S_1 \dots S_C\}$.

instance-to-cluster property: $\widehat{Y}_i = \widehat{S}_j$	**confirm** When a user agrees with a machine's clustering of an instance, $M_u(X_i) = Y_i = \widehat{Y}_i$, she can confirms the corresponding instance-to-cluster property, so future clustering revisions learn to keep this confirmed property. **disapprove** When a user disagrees with a machine's clustering of an instance, $M_u(X_i) = Y_i \neq \widehat{Y}_i$, she can give disapproval feedback on the corresponding instance-to-cluster property. This feedback gives the following property constraint in future revisions: $M_c(X_i) = \widehat{Y}_i \in \{\widehat{S}_1 \dots \widehat{S}_{\widehat{C}}\} \setminus \widehat{S}_j$. **label** When an instance is not assigned to any cluster by a machine (null property), a user can add an instance-to-cluster property, $\widehat{Y}_i = \widehat{S}_j$, according to the content of a single instance. **move** This feedback type allows her to move the instance from the inappropriate machine-assigned cluster, $\widehat{Y}_i = \widehat{S}_j$, to her desirable cluster, $\widehat{Y}_i = \widehat{S}_{j'} \in \{S_1 \dots S_C\}$.
instance-to-instance property: $\widehat{Y}_i = \widehat{Y}_j$ or $\widehat{Y}_i \neq \widehat{Y}_j$	**confirm** When a user agrees with a must-link ($Y_i = Y_j$ agrees with $\widehat{Y}_i = \widehat{Y}_j$) or cannot-link ($Y_i \neq Y_j$ agrees with $\widehat{Y}_i \neq \widehat{Y}_j$) instance-to-instance property, she can confirm it. This confirmed property should be kept in future revisions. **disapprove** When a user disagrees with a must-link ($Y_i \neq Y_j$ contradicts $\widehat{Y}_i = \widehat{Y}_j$) or cannot-link ($Y_i = Y_j$ contradicts $\widehat{Y}_i \neq \widehat{Y}_j$) instance-to-instance property, she can disapprove it. In future revisions, $\widehat{Y}_i \neq \widehat{Y}_j$ should be generated instead of the disapproved $\widehat{Y}_i = \widehat{Y}_j$ property and vice versa.

feature-to-cluster property: top K F_ms with the highest scores of $FS(F_m, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\})$	**confirm** When a user thinks a machine-proposed key feature, F_m, is representative to a cluster, $\widehat{S}_j$, she can confirm this feature-to-cluster property. High $FS(F_m, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\})$ score should be kept in future clustering revisions. **disapprove** When a user thinks a machine-proposed key feature, F_m, doesn't fit the topic of a cluster, $\widehat{S}_j$, she can disapprove this feature-to-cluster property. The score of $FS(F_m, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\})$ should be tuned down greatly in future revisions. **move** If a user thinks a key feature, F_m, of a cluster is better suitable for another cluster, a user can move the key feature from its original cluster, $\widehat{S}_j$ to the other cluster, $\widehat{S}_k$. This feedback implies that any future clustering obtained by a retrained model, M_c, shouldn't have high $FS(F_m, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\})$ score, but should have high $FS(F_m, \widehat{S}_k \mid \{X_i, \widehat{Y}_i\})$ score. **re-rank** When a user agrees with a machine that two key-features, F_m and F_n, are both representative to a cluster, she may weigh the importance of these two key-features differently than what a machine proposes. A re-rank feedback type allows a user to adjust the importance order of key-features. This feedback makes the following property modification: $FS(F_m, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\}) > FS(F_n, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\}) \Rightarrow FS(F_m, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\}) < FS(F_n, \widehat{S}_j \mid \{X_i, \widehat{Y}_i\})$.

cluster-to-cluster property: $Parent(\widehat{S}_i) = \widehat{S}_p$ $Child(\widehat{S}_p) = \{\widehat{S}_i\}$ $Sibling(\widehat{S}_i) = \{\widehat{S}_j\}$	**move** When a user finds the topic of a cluster is not a sub-topic of its parent cluster's topic, she can correct this property by moving the child cluster to another place in the hierarchy that fits her ontology for this clustering task. $Parent(\widehat{S}_i) = \widehat{S}_p \Rightarrow Parent(\widehat{S}_i) = \widehat{S}_{p'}$ **merge** If a user finds a cluster, $\widehat{S}_i$, duplicates the topic of another cluster, $\widehat{S}_j$, she can merge $\widehat{S}_i$ into $\widehat{S}_j$. **split** If a user thinks a cluster, $\widehat{S}_i$, is a mixture of multiple topics in her ontology, she can split the cluster into several clusters. This feedback makes the following property modification: $Child(\widehat{S}_i) = \emptyset \Rightarrow Child(\widehat{S}_i) = \{\widehat{S}_{new})\}$. **unify** If a user thinks a subtree of clusters is about a same topic, she can unify this subtree of clusters. This feedback makes the following property modification: $Child(\widehat{S}_p) = \{\widehat{S}_i\} \Rightarrow Child(\widehat{S}_p) = \emptyset$.

In terms of interface design, one may want to enable the multiple-selection mode so a user can select multiple properties of a same type, and give feedback on selected properties altogether.

Conceptual Properties

When a user works on a clustering task, she does not only group instances based on their contents solely, she also applies her existing real world knowledge to the clustering task. For example, a user categorizes a news article about a baseball game as sports news not only because the specific article explicitly mentions "sports" several times, but also because she knows conceptually that "baseball is a type of sport." If there is another baseball game article in which the word "sports" is absent, a user can still apply the same conceptual knowledge to categorize the second baseball article into the sports news cluster. There are two alternative properties that can represent this conceptual understanding: (1) a cluster-to-cluster property where "baseball" is a child cluster of a "sport" cluster,

or (2) a feature-to-cluster property where "baseball" is a keyword of "sports". In other words, cluster-to-cluster properties and feature-to-cluster properties correspond to a user's conceptual understanding more closely than instance-to-cluster properties. We hypothesize that including conceptual property types, e.g., feature-to-cluster and cluster-to-cluster properties, to the computer-to-user language can help a user understand clustering results better and proceed to teach a machine more easily than without conceptual property types.

Communication Robustness of Multiple Property and Feedback Types

A user's clustering model, M_u, is a hidden model to a machine, and may not be fully developed in the beginning of a clustering task. We hope that showing multiple and diversified types of properties ensure user comprehension like redundancy in natural languages ensure the robustness of human-to-human communication. In addition, by allowing multiple types of user feedback, even a user finds some types of properties are harder to give feedback upon, she may find it is easy to confirm properties or correct inappropriate properties of other types.

Figure 2.3 illustrates the idea of showing multiple types of properties and allowing multiple types of user feedback in an interface. In this example, if a machine only presents instance-to-cluster properties such as "Cluster 3 contains Email 1, 5, 6, 8, . . . " in the interface, a user cannot learn a topic of Cluster 3 from this information. By adding key-person-to-cluster properties (Jaime, William and Adam [2] are primary recipients of Cluster 3,) and key-word-to-cluster properties ("Mixed-initiative" and "thesis" are keywords of Cluster 3,) a user learns that Cluster 3 corresponds to her thesis activity. The redundant clues of key persons and keywords build up robustness in machine teaching because machine-proposed properties often contain errors, and a user typically doesn't pay meticulous attention to every property shown in the interface. Once a user understands the topic of a cluster, she can teach a machine through multiple types of user feedback. However, a user may be too lazy to confirm good properties or to correct inappropriate properties, e.g., the user's reaction to the key-person-to-cluster properties in this example. By allowing multiple user feedback types, we hope that at least some feedback, especially feedback on conceptual properties, is intuitive enough for a user to give, such as confirming the keyword "thesis" that fits precisely the user's idea that Cluster 3 is about her thesis activity.

[2] They are committee members of this thesis

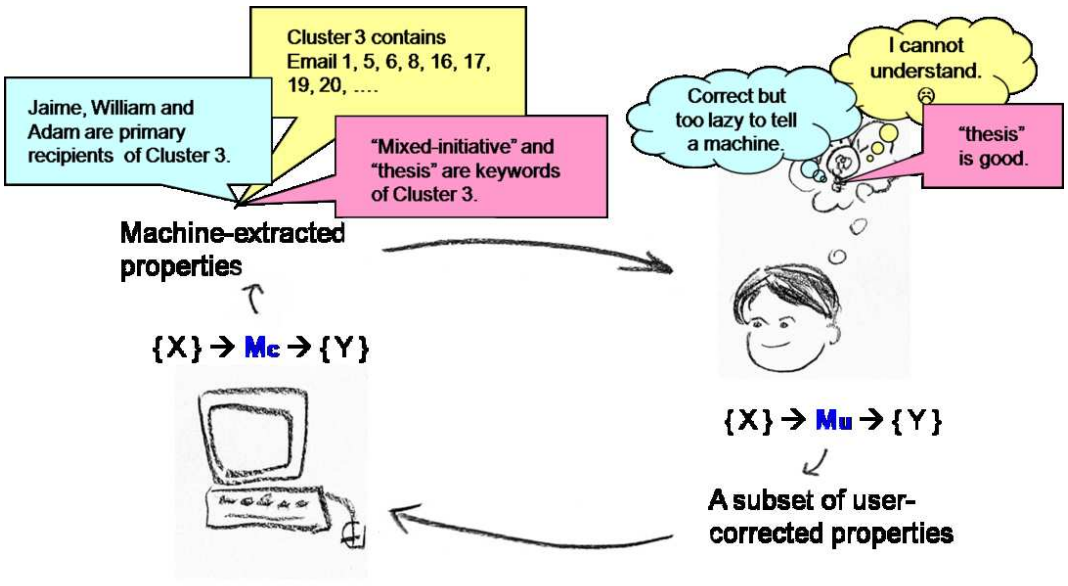


Figure 2.3: An illustration of showing multiple types of properties and allowing multiple types of user feedback in an interface.

It is important to include various property types and user feedback types in the communication languages, $L_{c \to u}$ and $L_{u \to c}$. From a user's point of view, the more natural the communication is, the more efficiently a user can learn from machine-proposed properties and proceed to teach a machine. From a mixed-initiative clustering system's point of view, with accelerated user learning and user teaching phases, the system is likely to elicit additional interactive loops from a user.

However, arbitrary machine-extracted property types or user feedback types cannot be added to the communication languages due to some fundamental constraints in mixed-initiative clustering.

2.3 Constraints and Coordination

There are mainly three constraints that restrict the enrichment of communication languages.

First, the computer-to-user language and user-to-computer language share the same interface. The shared interface has limited display space, which means the number of property types in the computer-to-user language cannot grow without bound because they

need to fit in the display space of an interface. The interface also introduces the feedback design limitation. Only feedback types that can be implemented are good for user-to-computer languages.

The second constraint comes from learning capability of a computer's clustering model, M_c. The limit of M_c's learning capability depends on whether a machine learning algorithm can be developed to adjust M_c according to each feedback type's specific property anchoring or property modification such as some examples listed in Table 2.1. Given different model assumptions, each clustering model has its own set of learnable feedback types. In order to save user efforts, we should either exclude un-learnable feedback types from the user-to-computer language, or switch to a new clustering model whose set of learnable feedback types include new types of user feedback a mixed-initiative clustering system wants to include.

The third constraint comes from a user's learning capability. A user may not be able to understand every type of properties a machine provides. For example, in image clustering, values of individual pixels may not be meaningful to a user. In text clustering, this constraint is not obvious because each word in text has its own semantic or functional meaning. Features that are not self-explanatory should be excluded from the computer-to-user language.

According to these constraints, developers of a mixed-initiative clustering system have to choose its computer-to-user language, user-to-computer language, and machine's clustering model properly so they work together. **Coordination** between these three components in mixed-initiative clustering is like balancing and leveling three legs of a tripod. The coordinated computer-to-user language, user-to-computer language and machine's clustering model can be considered as the signature of a mixed-initiative clustering system.

2.4 Related Work

Generally speaking, a mixed-initiative learning system is a system where a machine and a user collaborate in order to achieve the user's goal. The research directions of mixed-initiative systems are quite diversified. Nevertheless, the framework of mixed-initiative clustering proposed in Figure 2.1 can be conceptually used to categorize different research directions.

Mixed-initiative systems are often designed for users who are domain experts but not

knowledge engineers [34] [17]. In this case, a mixed-initiative system needs to enrich a user's teaching language so the user can contribute her domain knowledge to the machine without being a knowledge engineer. Several techniques developed in the machine learning literature can be considered as new communication types between a machine and a user. For example, CueFlik [18] allows a user to create a library of concept rules by giving a few positive and negative image examples of a target concept, and apply learned concept rules to re-rank image search results. For text clustering specifically, various new clustering property types and their corresponding feedback have been proposed such as learning from labeled features [32] [19] [45] [16], learning from pair-wise must-link or cannot-link relationships [55] [3], and re-arranging information elements through spatial representations [28]. The communication direction from a machine to a user is also important for a human teacher in terms of knowing whether the teaching is sufficient. The work of socially guided machine learning [52] represents a robot's confidence level as a learner to its human teacher through the robot's gaze behaviors.

When users are not domain experts nor knowledge engineers, mixed-initiative learning systems need to help users learn task-specific knowledge. For example, showing diverse examples [44] helps users establish their preferences. This study also demonstrates that an explanation interface inspires users' trust. One study [56] proposes a guided machine learning approach that generates sophisticated ontologies of manually-built quality and with less manual efforts. Another study [1] introduces an interactive corpus exploration tool called InfoMagnets for topic analysis of human tutoring dialogues. Other work [23] [49] emphasizes the importance of a combination of multiple types of clustering properties and corresponding rich user feedback, so a user can understand a machine's current clustering better and and correct errors more easily.

In cases where more than one reasonable user ontologies are available, a user chooses the best ontology based on her preference. Some early work [37] [33] has demonstrated that a digital personal assistant with a machine learning component gradually learns a user's preferences. A recent survey [40] depicts three types of interactive systems along this research direction that learn preferences of users including recommender systems, personal assistant agents, and personalized user interfaces. There are also studies investigating how to enrich user teaching languages of expressing a user's preference. For example, *DD-PREF* [12] is a language that expresses user preferences abstractly. It allows a user to specify the degree to which individual features should be varied (diversity) or focused on

particular values (depth).

In addition to communication enrichment, interaction is another key component for mixed-initiative learning. Horvitz's work [21] provides several design considerations for activating the interactive loop. In order to decide whether an automatic calendar assistant should be invoked, this paper proposes a utility function according to uncertainties in knowing a user's actual intention, a user's attention cost and automation benefits. Scatter/Gather is an interactive clustering system designed for browsing large document collections [9] or navigating retrieval results [20]. The system presents its user a "cluster digest" of keywords and titles of the most typical documents, allows users to select a few clusters of interests (gather), and re-clusters the selected clusters into finer granularity (scatter). We can consider the Gather/Scatter system enriches its computer-to-user language of multiple types of properties and has feedback types of cluster merging and splitting in its user-to-computer language. User studies on the Scatter/Gather system [43] show that although this interactive system may not be an effective tool for finding relevant documents, it can be useful for exploring large unknown data collections.

Mixed-initiative learning is also useful for transfer learning. A military intelligent service [51] learns knowledge-based rules interactively from experienced domain experts. Once the rules are learnt, the same service is able to tutor incoming domain apprentices.

In terms of understanding users' interaction with a mixed-initiative system, Stumpf et al. [50] investigate users' willingness to interact with machine learning reasoning, and what kinds of feedback users may give to machine learning systems. They find rule-based explanations are the most understandable to users, keyword-based explanations are the next, but similarity-based explanations have serious understandability problems. Nevertheless, the diversified preference of the winning explanation paradigm among their participants suggests the necessity of combining multiple paradigms of explanations. Their findings support our ideas of communication enrichment with multiple types, especially conceptual types, of properties and user feedback. There are also user studies on the faceted product search and recommendation focusing on different mixed-initiative interface design [44]. Their work suggests several useful guidelines such as showing example options and diverse examples in order to help users gain preference fluency and establish personal preference, explaining the computation behinds ranking results and compromises made in partially satisfied results, and providing trade-off assistance in addition to the search function. Another user study [53] observes how humans want to teach robots in

an interactive robot training process. Its results show users treat the robot in a social way, tending towards positive feedback, guiding the robot, and adjusting their training behavior according to the robot's learning limitation. It suggests that a robot communicates more of its internal state to the human teacher, and augmenting the human reward channel because a human teacher has various communication intents (feedback, guidance, and motivation.) These suggestions for reinforcement learning are similar to our idea of enriching communication languages for mixed-initiative clustering. In the data visualization and analysis domain, an exploratory tool [42], SocialAction, integrates various statistical measurements on social network analysis and allows flexible user selection on these measurements. We can consider each statistic measurement as one type of properties in its computer-to-user language. Four long-term case studies examine experts' use of this tool [41]. The outcomes of case studies confirm the advantage of integrating statistics and visualization and also show that experts of different domains may find different statistical measurements useful.

All research directions mentioned above are highly related to one or some problems in mixed-initiative learning. However, research that solves communication enrichment and real-time interaction jointly is hugely under-explored. Also, user studies for understanding user involvement in any mixed-initiative learning system are necessary.

We hope our study on mixed-initiative clustering provides a complete case study to the research community of mixed-initiative learning. Chapter 3 and Chapter 4 discuss how to build a successful mixed-initiative clustering system that integrates communication enrichment, coordinated machine learning algorithms, and real-time interaction. In Chapter 5, we introduce our user studies to examine the effectiveness of mixed-initiative clustering. Our user teaching study indicates the mixed-initiative style helps users teach efficiently, which is is similar to the result in [22][56]. To our knowledge, the user studies conducted in this thesis are the first to investigate the user learning and user teaching scenarios separately in the context of mixed-initiative text clustering. The study on the initialization of the interactive loop is also unprecedented.

Chapter 3

Communication Enrichment and Coordination

Developers of a mixed-initiative clustering system need to enrich communication languages, $L_{c \to u}$ and $L_{u \to c}$, of the system, while coordinating the machine's clustering model, M_c, to learn from enriched user-to-computer communication (user feedback.) In this chapter, we present two detailed examples of building mixed-initiative clustering systems. The first mixed-initiative clustering system enables user feedback on features for non-hierarchical clustering by introducing a new clustering algorithm, the SpeClustering model. On top of the first system, the second mixed-initiative clustering system focuses on how to build a "hierarchical" mixed-initiative clustering system that presents cluster-to-cluster properties in the interface and accepts hierarchical user feedback.

While focusing on communication enrichment and coordination, only one iteration of the interactive loop in mixed-initiative clustering is experimented with in this chapter. The experiment setting consists of an initial autonomous clustering with extracted properties, a long user feedback session, and a machine retraining session. We leave the practical issues of building fully interactive systems to the next chapter.

3.1 Mixed-Initiative Clustering with Feedback on Features

The first mixed-initiative clustering system we introduce in this chapter is built to improve the quality of activity extraction according to user guidance. As described in Sec-

tion 1.1, the automatic activity extractor generates a list of activity descriptions by clustering text content of a user's email, and extracting features such as keywords and primary senders/recipients for each cluster. We notice that machine-extracted features greatly help a user identify some of her activities. Given the lesson learnt, this mixed-initiative clustering system presents extracted features (feature-to-cluster properties) in addition to the initial clustering result (instance-to-cluster properties) in its interface and seeks user feedback on all types of properties.

3.1.1 Enriching Communication Languages for Non-Hierarchical Clustering

Figure 3.1 shows a snapshot of the interface of our first mixed-initiative clustering system and annotates property types used in the computer-to-user communication language. Using the combo-box at the top-left corner, a user selects which cluster she wants to investigate. Each selectable cluster is associated with a numerical cluster label, which means the computer-to-user language includes the property type of single cluster labels. Given the selected cluster, the top right panel of the interface displays a list of documents assigned to the cluster according to the machine's clustering model, M_c. Each document in this panel corresponds to one instance-to-cluster property. When the user selects one document, the text content of the selected document (single instance content property) shows in the bottom right panel. The left side of the interface displays a list of keywords and key-persons of the selected cluster. Each keyword corresponds to one feature-to-cluster property and so does each key-person. Through this interface, a machine communicates four types of properties with regard to the clustering task and its clustering model to a user. The left side of Figure 3.3 illustrates the computer-to-user language, $L_{c \to u}$ with these four property types. The feature-to-cluster properties actually contain two types of key features, keywords and key-persons, but the drawing simplifies this difference.

During the cluster investigation, a user can decide to remove the cluster if it doesn't represent any of her activities, or annotate this cluster by giving it a text description if it represents one of her activities. This text description of the cluster is attached to the end of the numerical cluster label. Once a cluster is kept as one of the user's activities, a user can further confirm or remove some email initially clustered to the activity, and confirm or remove some keywords or key-persons with regard to the activity she has in mind. Also,

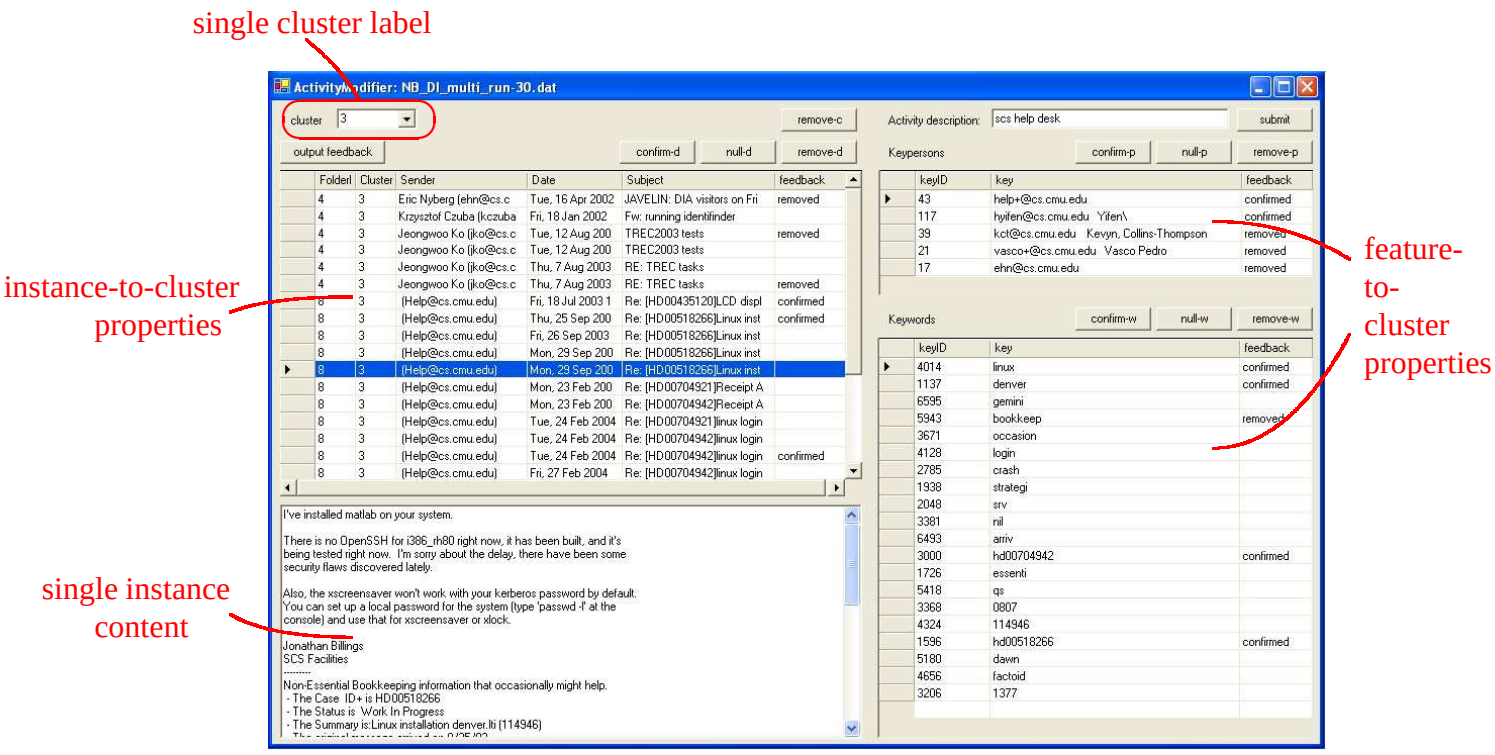


Figure 3.1: A snapshot of the user interface of the first mixed-initiative clustering system with annotations for its computer-to-user communication language. This interface displays a selected cluster, a list of documents in the selected cluster, content of a selected document, keywords and key-persons of the selected cluster as a computer's communication to a user.

the user can go back and forth between clusters to keep her overall feedback consistent. Figure 3.2 shows the same interface as Figure 3.1 but annotates the interface with feedback types used in the user-to-computer communication language. The user-to-computer communication of the system includes feedback types of confirming and removing single cluster label properties, instance-to-cluster properties, and feature-to-cluster properties. Most of these feedback types can be provided by users explicitly through the interface except the cluster confirmation. The cluster confirmation is assumed implicitly when a user gives a short text description of an activity cluster or confirms some documents or key features for a cluster. In addition, each word in the cluster description is considered a keyword for the cluster. The right side of Figure 3.3 depicts the user-to-computer language, $L_{u \to c}$, of this system as two user feedback types, confirmation and removal, in each

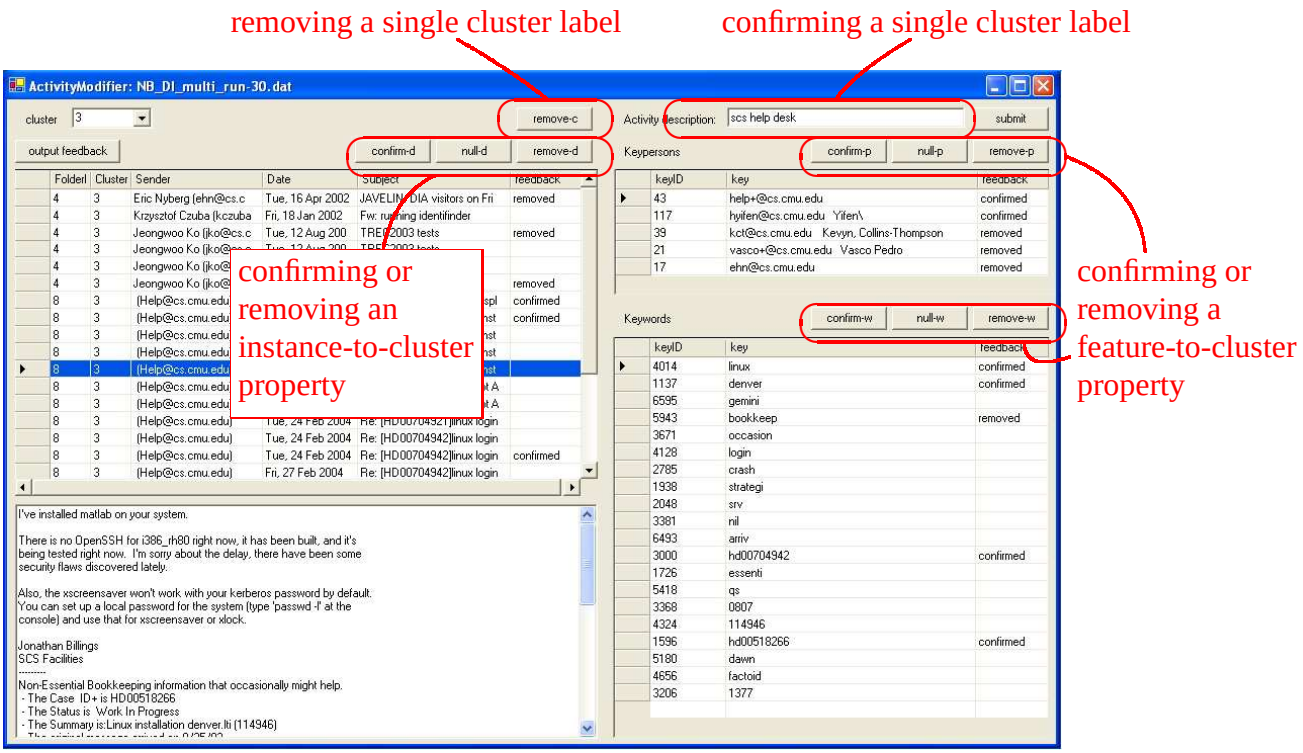


Figure 3.2: A snapshot of the user interface of the first mixed-initiative clustering system with annotations for its user-to-computer communication language. This interface allows various user feedback types such as removing a cluster, implicitly confirming a cluster by giving the cluster an activity description, confirming/removing a document, or confirming/removing a keyword/key-person. A user can go back and forth between clusters to keep her overall feedback consistent.

colored box attached to the three property types in the computer-to-user language, $L_{c \to u}$.

By comparing Figure 3.3 and Figure 2.2 (the primitive communication languages of active learning,) one can see why we claim this mixed-initiative clustering system enriches the communication between a machine and a user. We consider the most important communication enrichment in this non-hierarchical mixed-initiative clustering system is the presentation of key features and user feedback on them. This is because, as discussed in Section 2.2, the feature-to-cluster property type is one of two ways to represent a user's conceptual understanding of her clustering ontology.

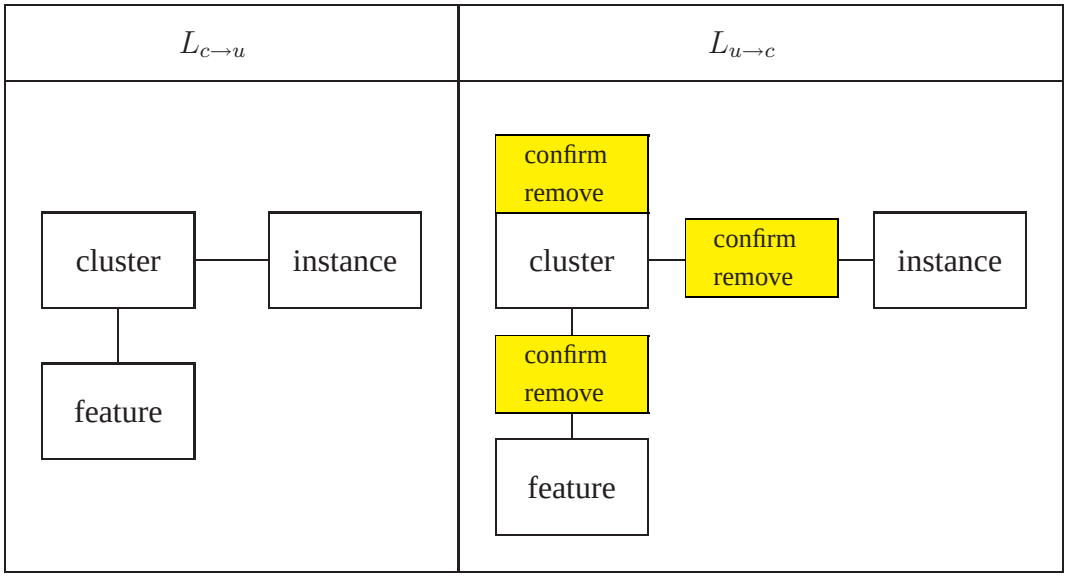


Figure 3.3: The communication languages of our non-hierarchical mixed-initiative clustering system. The computer-to-user language of this system includes property types of single clusters, instance-to-cluster properties, and feature-to-instance properties. The user-to-computer language includes feedback types of confirming and removing properties of all three types defined in the computer-to-user language.

3.1.2 Coordination with the SpeClustering Model

We still need a clustering algorithm that can learn from all types of user feedback to establish the coordination of mixed-initiative clustering. One commonly used probabilistic model for text clustering is the multinomial naive Bayes model described in [39], which models a document as a vector of words with each word generated independently by a multinomial probability distribution conditioned on the document's class (i.e., conditioned on which cluster it belongs to). However, although the naive Bayes model can learn from user feedback on single cluster label properties and instance-to-cluster properties, it cannot learn from user feedback on feature-to-cluster properties.

There are some works that propose methods to adapt a machine's clustering model from user feedback on features. For example, some work [27] uses a few user-supplied keywords per class and a class hierarchy to generate preliminary labels to build an initial text classifier. Another work [32] proposes a technique in which they ask a user to identify interesting words among automatically selected representative words for each class of documents, and then use these user-identified words to re-train the classifier as in [27].

Researchers working on active learning have also studied using feedback about key features. For example, a user-recommended feature can be converted into a mini-document in order to help train an SVM classifier [19]. An alternative approach [45] utilizes the key feature information by adjusting the SVM weights associated with these key features to a pre-defined value in binary classification tasks. A recent work of generalized expectation criteria [35] provides a method to incorporate preferences about model expectations into parameter estimation objective functions and it has been applied to learning feedback on feature-to-cluster properties [16].

In this thesis, we propose and use the SpeClustering model for a machine to learn user feedback on feature-to-cluster properties.

Basic ideas of the SpeClustering model

The SpeClustering model [23] is a novel probabilistic model that uses an idea similar to popular topic models [5]. It stands for "Specific Clustering", and refers to the fact that the probabilistic model estimates a latent boolean variable for each feature to determine whether it is relevant to or independent of a specific cluster. Put another way, the model assumes that only *some* of the words in the document are conditioned on the document's cluster, and that other words follow a more general word distribution that is independent of which cluster the document belongs to. To see the intuition behind this model, consider a cluster of emails about skiing. There will be some words (e.g., "snow") that appear in this cluster of emails because the topic is skiing, and there will be other words (e.g., "contact") that appear for reasons independent of the cluster topic. The key difference between the standard multinomial naive Bayes model and our SpeClustering model is that our model assumes each document is generated by a mixture of two multinomials – one associated with the document's cluster, and the other shared across all clusters.

Formally speaking, the SpeClustering model extends the standard multinomial model in two ways. The first modification is to add a G topic variable that is intended to capture general topics not related to the cluster. The second modification is to introduce a boolean decisive variable, D, associated with each word X in a document. This boolean variable decides whether the word is generated from the specific topic or the general topic. If $D = 1$, the observation word X is generated by the cluster-specific topic S, and if $D = 0$, the observation X is generated by a general topic G. Throughout this paper we simplify the model by assuming there is only one general topic instead of multiple topics. Figure 3.4

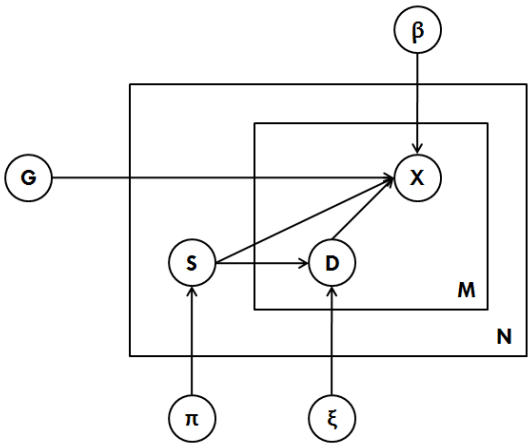


Figure 3.4: Graphical representation of SpeClustering model. S is a variable representing the cluster-specific topic associated with a document, X represents one of the M observed words in one of the N observed documents, and D is a boolean decisive variable that indicates whether word X is generated conditioned on the cluster S or whether it is generated according to a cluster-independent general distribution of words G.

shows the graphical model representation of the model. Here the outer rectangle (or plate) is duplicated for each of the N documents, and the inner plate is duplicated for each of the M words, X, and associated decisive variables, D. Note the general topic G is constant across all documents and words, whereas the cluster topic S is different for each document.

The Speclustering model has four sets of parameters:

$$\pi_c = P(S = c)$$

$$\xi_c = P(D = 1|S = c)$$

$$\beta_{cv} = P(X = v|S = c)$$

$$\beta_{gv} = P(X = v|G = g)$$

where $c \in \{1, 2, ..., |S|\}$, $g \in \{1\}$ for the simplified case and $v \in \{1, 2, ..., |X|\}$. $|S|$ is the number of clusters. We also use θ to refer to a specific SpeClustering model with all four sets of parameters.

Most importantly, the SpeClustering model can derive a new probability directly from its parameters:

$$P(D = 1|S = c, X = v; \theta) = \frac{\xi_c \beta_{cv}}{\xi_c \beta_{cv} + (1 - \xi_c)\beta_{gv}}$$

This probability describes whether a particular feature v is generated by a cluster's specific topic c, as apposed to the general topic g. When the value of this probability is closer to one, the feature v is more likely to be a key feature of the cluster c. We can apply this characteristic to learn a user's feedback on feature-to-cluster properties. When a user confirms a feature v as a key feature for the cluster c, this feedback teaches a machine that the value of the probability $P(D = 1|S = c, X = v; \theta)$ should be one. Oppositely, when a user removes feature v from the key feature list of cluster c, this feedback is equivalent to setting the probability $P(D = 1|S = c, X = v; \theta)$ to zero.

Corpus likelihood

Given a corpus $\mathbb{C}$ that contains n instances $\mathbb{C} = \{x_1, x_2, ..., x_n\}$, and x_i is represented as a vector of observations $\{x_{ij}; j \in \{1, 2, ..., m_i\}\}$, we use the notation s_i to indicate the value of the hidden S variable for instance x_i and d_{ij} to indicate the value of the hidden D variable associated with observation x_{ij}. The likelihood of corpus $\mathbb{C}$ given a SpeClustering model, θ, is defined as follows:

$$P(\mathbb{C}|\theta) = \prod_{i=1}^{n} \sum_{s_i=1}^{|S|} P(s_i) \cdot \prod_{j=1}^{m_i} [P(d_{ij} = 1|s_i)P(x_{ij}|s_i) + P(d_{ij} = 0|s_i)P(x_{ij}|g)]$$

which can be written in terms of the model parameters as follows:

$$P(\mathbb{C}|\theta) = \prod_{i=1}^{n} \sum_{s_i=1}^{|S|} \pi_{s_i} \cdot \prod_{j=1}^{m_i} [\xi_{s_i} \beta_{s_i x_{ij}} + (1 - \xi_{s_i})\beta_{g x_{ij}}]$$

Parameter estimation with unobserved variables

In the unsupervised case where only a set of documents is given (variable X is observed) but cluster assignments and key features are not available (variable S and D are unobserved), we use an EM process [11] for parameter estimation. The EM algorithm is commonly applied to find a (local) maximum-likelihood estimate of the parameters in

situations when the observable data is incomplete and the model depends on unobserved latent variables. Given $\mathcal{X}$ and $\mathcal{Y}$ as the incomplete and complete data, the algorithm iterates through two steps: in the E step, we evaluate $Q(\theta|\theta^t) = E[\log P(\mathcal{Y}|\theta)|\mathcal{X}, \theta^t)]$, and in the M step, we obtain new estimation of parameters $\theta^{t+1} = \arg\max_\theta Q(\theta|\theta^t)$. In our SpeClustering model, the incomplete data is $\mathcal{X} = \{x_{ij}\ \forall i \in \{1, ..., n\}\ j \in \{1, ..., m_i\}\}$ and the complete data is $\mathcal{Y} = \{s_i, d_{ij}, x_{ij}\ \forall i \in \{1, ..., n\}\ j \in \{1, ..., m_i\}\}$. The exact estimation for each parameter in the M step is listed below.

$$\pi_c^{t+1} = \frac{\sum_{i=1}^n \phi_i^t(c)}{n}$$

$$\xi_c^{t+1} = \frac{\sum_{i=1}^n \phi_i^t(c) \sum_{j=1}^{m_i} \psi_{ij}^t(c)}{\sum_{i=1}^n \phi_i^t(c) \cdot m_i}$$

$$\beta_{cv}^{t+1} = \frac{\sum_{i=1}^n \phi_i^t(c) \sum_{j=1}^{m_i} \delta(x_{ij} = v)\psi_{ij}^t(c)}{\sum_{i=1}^n \phi_i^t(c) \sum_{j=1}^{m_i} \psi_{ij}^t(c)}$$

$$\beta_{gv}^{t+1} = \frac{\sum_{i=1}^n \sum_{k=1}^{|S|} \phi_i^t(k) \sum_{j=1}^{m_i} \delta(x_{ij} = v)(1 - \psi_{ij}^t(k))}{\sum_{i=1}^n \sum_{k=1}^{|S|} \phi_i^t(k) \sum_{j=1}^{m_i} (1 - \psi_{ij}^t(k))}$$

where the following quantities are computed in the E step:

$$\begin{aligned} \phi_i^t(c) &\equiv P(s_i = c|x_i; \theta^t) \\ &= \frac{\pi_c^t \prod_{j=1}^{m_i} [\xi_c^t \beta_{cx_{ij}}^t + (1 - \xi_c^t)\beta_{gx_{ij}}^t]}{\sum_{k=1}^{|S|} \pi_k^t \prod_{j=1}^{m_i} [\xi_k^t \beta_{kx_{ij}}^t + (1 - \xi_k^t)\beta_{gx_{ij}}^t]} \end{aligned} \tag{3.1}$$

$$\begin{aligned} \psi_{ij}^t(c) &\equiv P(d_{ij} = 1|s_i = c, x_{ij}; \theta^t) \\ &= \frac{\xi_c^t \beta_{cx_{ij}}^t}{\xi_c^t \beta_{cx_{ij}}^t + (1 - \xi_c^t)\beta_{gx_{ij}}^t} \end{aligned} \tag{3.2}$$

By iterating through the E step and M step, the corpus likelihood will converge to a (local) maximum and values of parameters will be stabilized.

Extension to multiple types of features

In many cases, instances consist of multiple types of features. For example, when clustering emails we may describe each email by the set of words in its text content, plus the set of email addresses the email is sent to. If there are multiple types of features in an in-

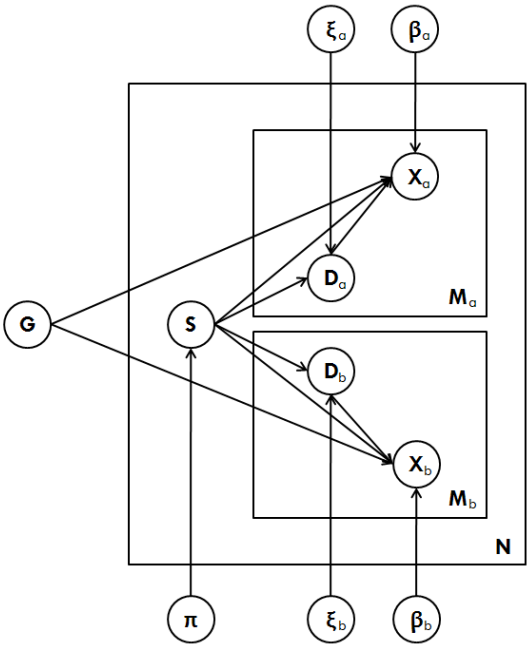


Figure 3.5: Graphical representation of the SpeClustering model with two feature types, where X_a and X_b are observations with two different feature types and D_a and D_b are boolean variables deciding whether their respective observations are generated from the specific topic S or the general topic G

stance, we can extend the SpeClustering model. Figure 3.5 shows the extended model with two feature types, a and b. Assume the original block present type-a features and decisive variables, the model adds one new block of $\{X_b, D_b\}$ for type-b features and decisive variables. $\{X_b, D_b\}$ is identical and parallel to $\{X_a, D_a\}$. In the activity-discovery-via-emails task, we can apply this model to represent an activity in terms of both its key words and the primary participants of the activity.

Parameter estimation in the extended SpeClustering model is nearly identical to the parameter estimation of single type features. The only exception is a change to the posterior probability estimate in Eq 3.1. The new posterior probability estimate in the extended model combines generative probabilities from multiple feature types. We use m_{ai} and m_{bi} to indicate the numbers of features of two feature types in the i^{th} email, and the i^{th} email is represented as $x_i = \{x_{aij}\ \forall j \in \{1 \dots m_{ai}\};\ x_{bih}\ \forall h \in \{1 \dots m_{bi}\}\}$. Eq 3.3 shows the estimate from two different feature types.

$$\phi_i^t(c) \equiv P(s_i = c|x_i;\theta^t)$$
$$= \frac{\pi_c^t \prod_{j=1}^{m_{ai}} [\xi_{ac}^t \beta_{acx_{aij}}^t + (1-\xi_{ac}^t)\beta_{agx_{aij}}^t] \prod_{h=1}^{m_{bi}} [\xi_{bc}^t \beta_{bcx_{bih}}^t + (1-\xi_{bc}^t)\beta_{bgx_{bih}}^t]}{\sum_{k=1}^{|S|} \pi_k^t \prod_{j=1}^{m_{ai}} [\xi_{ak}^t \beta_{akx_{aij}}^t + (1-\xi_{ak}^t)\beta_{agx_{aij}}^t] \prod_{h=1}^{m_{bi}} [\xi_{bk}^t \beta_{bkx_{bih}}^t + (1-\xi_{bk}^t)\beta_{bgx_{bih}}^t]} \quad (3.3)$$

Incorporating User Feedback into the SpeClustering Model

As discussed earlier, we are particularly interested in coordinating a machine's clustering model to learn from the enriched user feedback. Let's list again feedback types allowed in the first mixed-initiative clustering system and discuss how several types of user feedback are incorporated to retrain the SpeClustering model. The allowed user feedback in this system are:

1. Remove an activity cluster.
2. Confirm that an email belongs to its assigned cluster or remove an email from its assigned cluster.
3. Confirm a keyword belongs to its assigned cluster or remove a keyword from the cluster.
4. Confirm a key-person to its assigned cluster or remove a key-person from the cluster.
5. Give a short text description for an activity cluster. This feedback is considered as a combination of confirming the described activity cluster and confirming that each word in the description is a keyword to the cluster.

From the model adaptation point of view, the posterior probabilities in the SpeClustering model turn out to be highly related to the above types of feedback. To be more specific, feedback 1 and 2 are related to Eq. 3.3 and feedback 3, 4, and 5 are related to Eq. 3.2.

In this particular system, we assume the number of clusters is known and fixed. [1] When the user feedback includes removing a cluster $S = c$, we need to replace the removed cluster with a new cluster in order to keep the cluster number fixed. We develop two initialization methods to adapt the SpeClustering model to accept the cluster removal feedback. The simple initialization method inherits the previous clustering but resets the

[1] It is not realistic to assume the number of clusters is known and fixed. In the next section, the hierarchical mixed-initiative clustering system introduces various feedback types to modify cluster-to-cluster properties like adding a new cluster and merging two clusters into one, so the fixed cluster number assumption can be avoided.

initial probability value of $P(s_i|x_i;\theta^t)$ for each x_i with $s_i = c$ by distributing the probability mass uniformly among all clusters but halving the probability to cluster c. The re-distribution of the probabilistic mass is targeted to create a cluster that replaces the removed cluster, and to reduce the possibilities of documents in the removed cluster being assigned to the replacing cluster. Alternatively, the joint initialization method uses multiple feedback types to initialize the model. We first select several documents that have the highest cosine similarity with confirmed documents and keywords (where we treat keywords as a mini-document) and associate them with current clusters. We then search for a small set of similar documents that maximize inter-cluster distances and replace any cluster that is removed in the feedback.

During each EM iteration while training the SpeClustering model, we perform feedback 2 to 4 adjustments. For feedback 2, we adjust the value of $P(s_i = c|x_i;\theta^t)$ to be one if the hypothesized instance(x_i)-to-cluster(c) property is confirmed by the user or set it to zero if the bound is disapproved by the user. Proper adjustment to normalize posterior probabilities of $\{P(s_i = c'|x_i;\theta^t)\ \forall c' \neq c\}$ is also required in this case. For feedback 3 and 4, we adjust the value of $P(d_{ij} = 1|s_i = c, x_{ij} = v;\theta^t)$ to be one if the hypothesized feature(v)-to-cluster(c) property is confirmed by the user or set it to zero if the property is disapproved by the user. For feedback 5, we tokenize the description T and make each token of T a confirmed keyword as in type 3 feedback.

Figure 3.6 summarizes this mixed-initiative clustering process in which the SpeClustering model updates its parameter estimation according to the enriched user feedback.

Since each property type in the computer-to-user language, $L_{c\rightarrow u}$, corresponds to some probability expressions in the SpeClustering model, adapting the model is equivalent to adjusting some probability values according to user feedback. This system exemplifies the coordination of mixed-initiative clustering with thoroughly defined computer-to-user language, user-to-computer language and machine's clustering model.

Connection to Supervised Classification

We have described details of the SpeClustering model. However, the model is not restricted to clustering; it can also be applied to supervised classification tasks. The difference in classification is that the topic variable S is no longer a hidden variable – instead S is the classification label. We can treat the classification tasks as knowing all the type 2 user feedback and replace the estimate of posterior probabilities $P(s_i = c|x_i;\theta^t)$ with the

Algorithm: Mixed-Initiative SpeClustering
Input: Unlabeled corpus $\mathbb{C}$
Output: A SpeClustering clustering model θ, and a set of properties agreed by the machine and the user.
Languages: The computer-to-user language $L_{c\to u}$ consists of the single cluster label property type, instance-to-cluster property type, and feature-to-cluster property type. The user-to-computer language $L_{u\to c}$ consists of confirming and removing feedback types for each of the three property types defined in the computer-to-user language $L_{c\to u}$.

Method:

1. Initialize the user-to-computer communication as an empty set of modified properties, $Comm_{u\to c} = \{\}$.
2. The machine builds an initial clustering model θ_{ini}. $\theta = \theta_{ini}$.
3. The machine applies the clustering model θ to obtain the clustering result of the corpus $\mathbb{C}$. Based on the result, the machine extracts a set of properties as its communication to the user, $Comm_{c\to u} = \{property\}$. Each property in $Comm_{c\to u}$ belongs to a property type defined in the computer-to-user language $L_{c\to u}$.
4. The user gives feedback which is equivalent to modifying a subset of properties in $Comm_{c\to u}$. The subset of modified properties is annotated as $\{property^*\}$. Accumulate the user's communication to the machine as $Comm_{u\to c} \leftarrow Comm_{u\to c} \cup \{property^*\}$.
5. The computer performs model retraining, θ^{new} =SpeClustering-with-Feedback$(\mathbb{C}, \theta, Comm_{u\to c})$.
6. Update the machine's clustering model to the newly retrained model, $\theta = \theta^{new}$. Repeat step 3 to 6 until the user's satisfaction.

Algorithm: SpeClustering-with-Feedback
Input: Unlabeled corpus $\mathbb{C}$. θ as the current model. $Comm_{u\to c}$ as the collection of user's feedback.
Output: θ^{new} as the model after adaption according to user's feedback.

Method:

1. $\theta^t = \theta$.
2. Estimate posterior probabilities $\mathcal{P}^t$ of Eq 3.3 and Eq 3.2 given $\mathbb{C}$ and θ^t.
3. Adjust $\mathcal{P}^t$ according to $Comm_{u\to c}$ to obtain $\mathcal{P}^t_{adj}$.
4. Re-estimate model parameters using $\mathcal{P}^t_{adj}$ to obtain θ^{t+1}.
5. $\theta^t = \theta^{t+1}$; repeat step 2 to 5 until the model converges.
6. $\theta^{new} = \theta^t$.

Figure 3.6: The mixed-initiative SpeClustering algorithm.

true value specified by the instance label.

3.1.3 Experimental Results

Datasets

To test our non-hierarchial mixed-initiative clustering system, we used two data sets. The first is an email dataset (*EmailYH*) which contains 623 emails. It had previously been sorted into 11 folders according to the user's activities. It contains 6684 unique words and 135 individual people after pre-processing[2]. The second data set is the publicly available 20-Newsgroups collection. This data set contains text messages from 20 different Usenet newsgroups, with 1000 messages harvested from each newsgroup. We derived three datasets according to [3]. The first, *News-Similar-3*, consists of messages from 3 similar newsgroups (comp.graphics, comp.os.ms-windows.misc, comp.windows.x) where cross-posting occurs often between these three newsgroups. *News-Related-3* consists of messages from 3 related newsgroups (talk.politics.misc, talk.politics.guns and talk.politics.mideast). *News-Different-3* contains 3 newsgroups of quite different topics (alt.atheism, rec.sport.baseball, and sci.space).

We only use the text part of messages in the three newsgroup datasets because a reviewer won't have the knowledge needed to decide which author is the key-person with regard to which newsgroup. For the text part, we applied the same pre-processing we used in (*EmailYH*). There are 3000 messages in these datasets. *News-Different-3* contains 8465 unique words, *News-Related-3* contains 9998 unique words and *News-Similar-3* has 10037 unique words.

Measurement for Cluster Evaluation

We use two measurements to estimate cluster quality: folder-reconstruction accuracy, and normalized mutual information (NMI) [13].

In order to calculate the folder-reconstruction accuracy, we search through all possible alignments of cluster indices $\mathcal{I}_c$, to folder indices $\mathcal{I}_f$ in order to find the alignment resulting

[2]The pre-processing for words includes stemming (Porter stemmer), stop word removal and removal of words that appear only once in the dataset. The pre-processing for people contains reference-reconciliation over email senders and recipients, and removal of people that are involved in only one email.

in optimal accuracy, then report the accuracy under this optimal alignment:

$$Acc = max_A \frac{\sum_{i=1}^{D} \delta(A(s_i) = f_i)}{D} \quad A \in \{Map(\mathcal{I}_c) \overset{1-to-1}{\longrightarrow} \mathcal{I}_f\} \tag{3.4}$$

The normalized mutual information measurement is defined as Eq. 3.5, where $I(S;F)$ is the mutual information between cluster assignment S and folder labels F, $H(S)$ is the entropy of S and $H(F)$ is the entropy of F. It measures the shared information between S and F.

$$NMI = \frac{I(S;F)}{(H(S) + H(F))/2} \tag{3.5}$$

These two measurements are correlated but show different aspects of clustering performance. Accuracy calculates the ratio between major chunks of clusters to its reference. NMI measures the similarity between cluster partitions and reference partitions.

Autonomous Clustering

This experiment checks if the SpeClustering model can produce reasonable autonomous clustering results so we can apply the model to mixed-initiative clustering. We compared two versions of the SpeClustering algorithm with the standard multinomial naive Bayes model[39] as the baseline approach. We modified the baseline approach by allowing it to search for a good cluster initialization and to avoid situations in which one cluster gets eliminated during the EM iterations[24]. The first version is the original SpeClustering algorithm as described in Section 3.1.2. The second version, *SpeClustering-bound*, adds range constraints on parameter values ξ (ξ_c=P(D=1|S=c)) : for word features, the range is $[0.1, 0.4]$ and for person features, the range is $[0.6, 0.9]$. The reason for introducing range constraints is to avoid situations where some values of parameter ξ_c converge to 1 or 0. This is undesirable because the value of ξ_c reflects the percentage of specific features $(D = 1)$ occurring over all observations for cluster c. Both SpeClustering algorithms were initialized using the output from the baseline naive Bayes clustering.

We made 50 individual runs on *EmailYH* dataset and 20 runs each on *News-Similar-3*, *News-Related-3*, and *News-Different-3*. Table 3.1 shows the average accuracy and NMI

results of different datasets and the three clustering algorithms. Notice in all datasets, the SpeClustering algorithm performs better than the naive Bayes algorithm, and the SpeClustering-bound model performs better than SpeClustering. The naive Bayes clustering results are used to initialize its associated SpeClustering and SpeClustering-bound runs, so the performance gain are directly due to the difference between the SpeClustering probabilistic model and naive Bayes model. When examining the details of individual runs, we find that SpeClustering-bound outperforms naive Bayes in every run using the NMI measure, and in the vast majority of these runs it also outperforms Naive Bayes in terms of the accuracy measure. We can conclude that modeling a document as a mixture of a general topic and a specific topic improves autonomous clustering performance.

dateset	method	Accuracy (%)	NMI (%)
EmailYH	naive Bayes clustering	48.44 ± 7.01	48.02 ± 3.93
	SpeClustering	52.28 ± 8.61	53.25 ± 5.65
	SpeClustering-bound	$\mathbf{53.98 \pm 8.04}$	$\mathbf{56.25 \pm 4.90}$
News-Sim-3	naive Bayes clustering	46.31 ± 7.21	9.86 ± 7.34
	SpeClustering	51.38 ± 6.33	15.80 ± 6.82
	SpeClustering-bound	$\mathbf{51.98 \pm 5.91}$	$\mathbf{16.46 \pm 6.27}$
News-Rel-3	naive Bayes clustering	60.18 ± 10.64	34.36 ± 10.58
	SpeClustering	60.61 ± 11.08	36.06 ± 10.71
	SpeClustering-bound	$\mathbf{61.14 \pm 11.41}$	$\mathbf{36.92 \pm 11.04}$
News-Diff-3	naive Bayes clustering	91.24 ± 13.45	79.76 ± 14.56
	SpeClustering	93.80 ± 11.49	83.57 ± 14.27
	SpeClustering-bound	$\mathbf{96.52 \pm 6.47}$	$\mathbf{87.79 \pm 11.56}$

Table 3.1: Clustering results of different datasets and different autonomous clustering algorithms. SpeClustering and SpeClustering-bound are the SpeClustering model with unbounded and bounded parameter values. Both versions of the SpeClustering model out-perform the multinomial naive Bayes clustering model and the bounded SpeClustering model achieves the best performance.

Clustering with Enriched User Feedback

We next studied the impact of enriched user feedback on the bounded SpeClustering model. In particular, we chose 5 clustering results using the multinomial naive Bayes model with the best log-likelihood among 50 runs on *EmailYH* and presented each of these to the user. We also chose one best run from 20 runs on *News-Different-3*, *News-Related-3*, and *News-Similar-3*.

The user gave feedback using the interface shown in Fig 3.1. The top left panel shows a list of documents that are clustered into the selected cluster label, the top right panel shows 5 key-persons of the cluster and the bottom right panel shows 20 keywords of the cluster. The keywords and key-persons of the cluster shown in the interface are selected using a Chi-squared measurement [57]. When a user clicks on a document in the document list, the content of the document shows in the bottom left panel. The user can give various types of feedback and the interface displays feedback the user has entered so far. The user can also go back and forth between clusters to correct conflicting assumptions she has made to achieve consistent cluster interpretations.

Like experiences from the activity extractor, displaying keywords and key-persons tremendously helps users make judgements about clusterings. In fact, to decide the meaning of a large cluster based only on examining the documents is extremely difficult. A user would tend to decide based on the first several documents she goes through even when the cluster contains more than hundreds of documents, and the biased decision often causes conflicts with later clusters. The user usually chooses to remove a cluster, if the keywords and key-persons don't show any consistency and are not meaningful to the user, or if documents in the cluster are a hodgepodge from several categories. If the keywords or key-persons of a cluster make sense to the user, the user can give various types of feedback according to the meaning of the cluster. We don't put constraints on how the user does the feedback, so the user can make decisions freely based on how she perceives the clustering results, and gives feedback using her own interpretation of the results.

We use the following notation to indicate feedback on various property types:

- CR: removing single cluster label properties
- PP: confirming or removing feedback on document-to-cluster properties
- WX: confirming or removing feedback on keyword-to-cluster properties
- HX: confirming or removing feedback on keyperson-to-cluster properties

Table 3.2 shows how many entries of feedback on different property types the user enters for each selected run. The user spends about 15 mins to finish one run from the *EmailYH* dataset and 5-10 mins to finish one run from newsgroup datasets.

We ran the SpeClustering-bound algorithm with user feedback and compared the results to the naive Bayes baseline and the SpeClustering-bound algorithm without feedback. Without adapting user feedback, the autonomous naive Bayes clustering and the SpeClustering results can be considered as outputs after one machine learning phase. The cluster-

run	doc #	CR	PP	WX	HX
Email1	623	3	99	37	30
Email2	623	3	73	35	31
Email3	623	4	92	48	26
Email4	623	7	32	28	15
Email5	623	4	91	43	28
Sim1	3000	2	39	9	-
Rel1	3000	1	29	20	-
Diff1	3000	0	16	39	-

Table 3.2: Entry numbers of different feedback types for 5 selected naive Bayes runs.

ing result produced by the SpeClustering-bound algorithm with user feedback adaptation is the mixed-initiative clustering result after one complete iteration of the interactive loop, consisting of a first machine learning phase, a machine teaching phase, a user learning phase, a user teaching phase, and a second machine learning phase.

We used the simple initialization method on *EmailYH* dataset in order to break down feedback to single types. Figure 3.7 shows the results using just one type of feedback on 5 selected runs from *EmailYH* dataset. The CR feedback is independent from other types of feedback and all other types involve feedback only from clusters that are not removed. All 5 runs with CR or PP feedback, 4 runs with WX feedback and 3 runs with HX feedback outperform both naive Bayes baseline and SpeClustering-bound without feedback.

Figure 3.8 shows the results using combination of feedback types. User's feedback gives huge improvements in all runs (19.55% average accuracy improvements from naive Bayes results to SpeClustering-bound with full feedback). SpeClustering-bound with full feedback performs best in 4 out of 5 runs. In the remaining one run, CR+PP feedback (cluster removal plus feedback on document-to-cluster properties) performs best. The quantity of PP feedback is about 1/7th to 1/9th to the whole dataset and even higher if we exclude documents in removed clusters. The combined entry numbers of WX+HX feedback (feedback on feature-to-cluster properties) are fewer than the numbers of PP feedback (feedback on document-to-cluster properties) in these runs. However, CR+WX+HX performs better than CR+PP in 2 runs, which shows that feedback on feature-to-cluster properties gives comparable information like feedback on document-to-cluster properties. More compellingly, it is also much easier to get CR+WX+HX feedback than CR+PP in terms of time efficiency. In a study [45] that measures users' time spent on labeling a

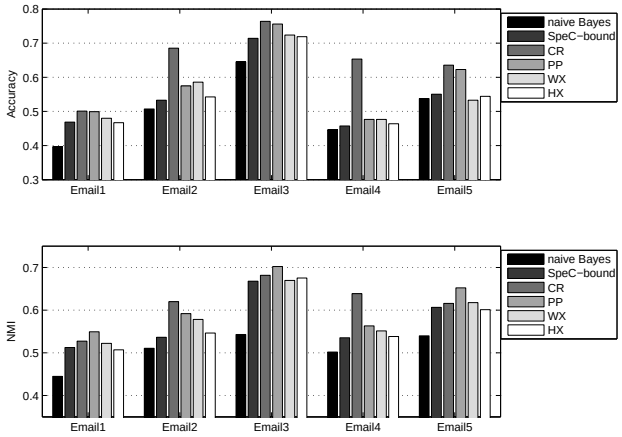


Figure 3.7: Performance of using single feedback types (CR, PP, WX and HX) on the *EmailYH* dataset. SpeC-bound is the SpeClustering-bound model without feedback. The SpeClustering-bound model with one type of feedback out-performs naive Bayes and SpeClustering-bound without feedback in 17 out of 20 runs.

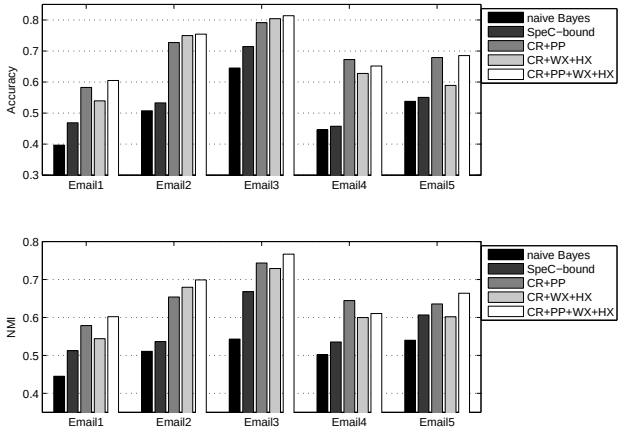


Figure 3.8: Performance of using combination of feedback types on the *EmailYH* dataset. SpeC-bound is the SpeClustering-bound model without feedback. User feedback gives huge improvements in all runs.

document or a feature, it finds a user only needs 1/5th of time to label a feature compared to the time to label a document.

For the 3 newsgroup datasets, the ratio of the amount of feedback to the corpus size is very small. In this case, the inheritance of old results, which is noisy, in the simple initialization overwhelms the training process. To remedy the problem, we used the joint

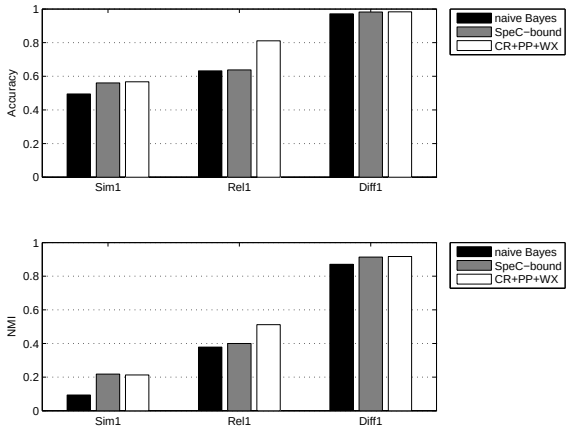


Figure 3.9: Experimental results of SpeClustering with user feedback on the newsgroup datasets. SpeCluster-bound is the model without feedback and CR+PP+WX is the SpeClustering-bound model with full user feedback. "Sim1", "Rel1", and "Diff1" refer to the selected runs (with the best autonomous naive Bayes modeling likelihood) of *News-Similar-3*, *News-Related-3*, and *News-Different-3*. Incorporating feedback gives significant improvement on the selected *News-Related-3* run, whose feedback is harvested from noisy but still meaningful clustering results.

initialization method that compares documents with no user feedback with each cluster's confirmed documents and keywords and initializes these documents to clusters with the most similar confirmed documents and keywords.

The user feedback is quite different across these three runs. For the selected run of *News-Similar-3*, the naive Bayes clustering results are extremely noisy and the cluster summarization is hardly recognizable by the user. It turns out the feedback contains the removal of two out of three clusters and the reason that one is kept is because some keywords weakly indicate the meaning of one newsgroup, but the documents in the remaining cluster contain huge chunks from each newsgroup. For the selected run of *News-Related-3*, talk.politics.guns and talk.politics.mideast are referred to two remaining clusters while talk.politics.misc has no reference due to the removal of the last cluster, which the user cannot figure out its meaning. The cluster summarization is noisy but comprehensible, so the user can make positive and negative feedback easily. For *News-Different-3*, the baseline accuracy is very high so most feedback is positive about the automatically generated summarization.

Figure 3.9 shows experimental results from user feedback on one selected run from each newsgroup dataset. It is difficult to improve on the already accurate *News-Different-*

3 run. Incorporating feedback gives no significant improvement on the selected *News-Similar-3* runs whose feedback is based on extremely noisy clusters and a user is barely able to associate meaningful criterion to any cluster. However, one sees huge improvement from using feedback on the noisy but still meaningful cluster results. The accuracy of the selected *News-Related-3* run jumps from 63.23% to 81.07%.

3.1.4 System Summary

In summary, the non-hierarchical mixed-initiative clustering system examines the idea of enriching communication languages with feature-to-cluster properties in a long-user-feedback-session setting where the interactive loop between a user and machine iterates once. This system presents an interface that enables a user to browse clustering results with extracted key-feature properties, and provide various types of feedback to guide machine retraining. With extracted key features, a user can understand the clustering results more easily. With various types of user feedback available, a user can teach a machine more intuitively. In order to achieve the coordination for mixed-initiative clustering, the SpeClustering model is used in the second machine learning phase because the model provides a natural way to adjust its parameters according to a variety of types of user feedback. The experimental results show that a mixed-initiative clustering system with enriched communication languages gains significant improvement in a personal email dataset. Given different levels of clustering difficulties, the mixed-initiative clustering system helps a machine learn a better clustering from user feedback on noisy but still meaningful autonomous clustering results.

3.2 Hierarchical Mixed-Initiative Clustering System

It is natural for users to organize personal data using hierarchies, especially in the electronic world. An obvious example is that file systems in most workstations consist of a hierarchy of user-created directories to store documents and other files. In fact, designs of hierarchical organization prevail in computer applications such as email clients, bookmark organizers, and contact management tools. However, due to the clustering mismatch problem, many applications of this category rely on users' spontaneous hierarchical organization instead of autonomous machine organization. We believe mixed-initiative human-machine approaches to hierarchical clustering hold great potential for such applications.

We extend the non-hierarchical mixed-initiative clustering system described in Section 3.1 to hierarchical clustering, and explore new types of user feedback that are natural for hierarchical cluster-to-cluster properties. Based on the experiences of developing an activity extractor described in Section 1.1, we consider a hierarchical email clustering approach composed of the following steps: (1) generating initial hierarchical clusters of depth two by using a generative clustering model for content analysis in the first level and social network analysis in the second level [3], (2) presenting the hierarchical clustering results in a user interface and recording users' modifications of the hierarchical clustering with time stamps, and (3) re-training the hierarchical clustering model according to hierarchical user feedback.

Figure 3.10 shows our user interface designed for presenting hierarchical clustering results in additional to extracted non-hierarchical properties to a user, and allowing various types of hierarchical and non-hierarchical user feedback. The left panel shows the resulting hierarchy. When a user selects a cluster in the hierarchy, the middle top panel shows a list of emails in this cluster, and the middle bottom panel would show content of an email chosen by the user (blank here for privacy reasons). The right panels show key-persons

[3] In email clustering, different approaches are studied to combine content analysis and social network analysis such as using a voting scheme [15] or analyzing social connections in the first level and content of messages in the second level [30]. However, for the general purpose of text clustering, we don't want our mixed-initiative clustering system to heavily rely on the social network perspective because this information is often unavailable or incomprehensible in other text clustering tasks. Another disadvantage of social network analysis is its lack of learning capability to user feedback, In the latter part of this chapter, we propose a weighting method to simulate the social network analysis and remedy the learning problem. Given the above reasons, we choose to analyze text content in the first level and social network in the second level.

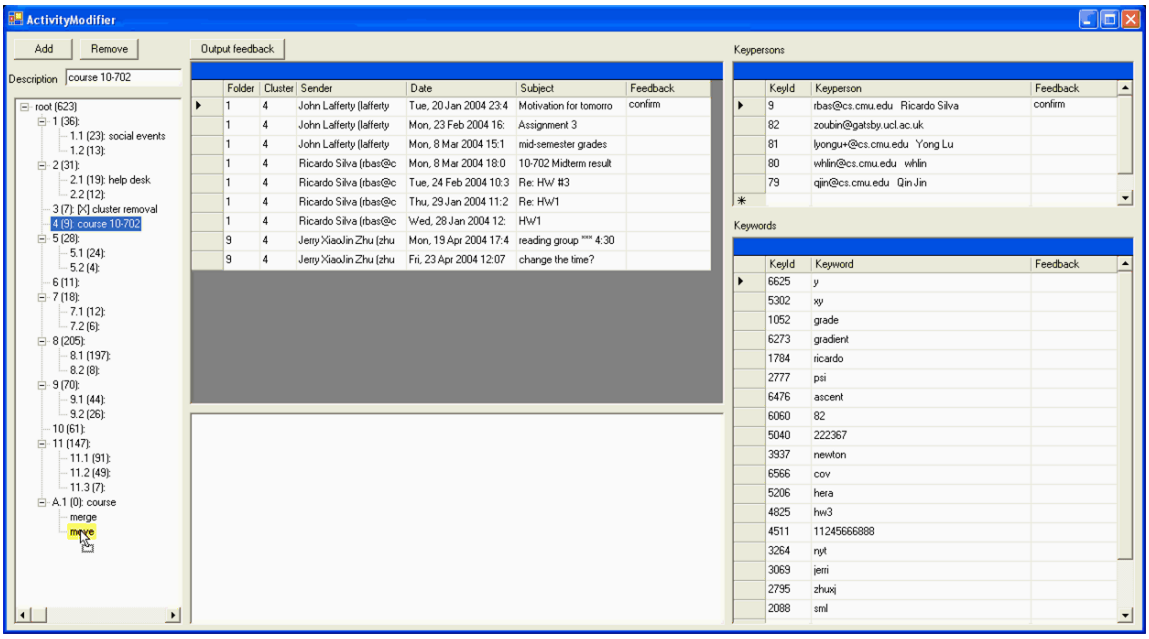


Figure 3.10: An interface that presents hierarchical clustering results to a user and allows various types of hierarchical and non-hierarchical user feedback. The left panel shows the resulting hierarchy.

and keywords associated with this cluster. In this snapshot, the user thinks cluster 4 is related to a specific course (the confirmed key-person is the TA of the course,) which should be under a general course cluster. The user has therefore added a ”course” cluster, A.1, and is moving cluster 4 underneath cluster A.1.

3.2.1 Enriching Communication Languages for Hierarchical Clustering

The resulting hierarchy shown in the left panel of the interface (Figure 3.10) is a set of cluster-to-cluster properties that describe parent-child and sibling relationships between two clusters. With this addition, the computer-to-user communication language, $L_{c \to u}$, in the hierarchical mixed-initiative clustering system consists of the following property types:

1. single cluster label properties
2. cluster-to-cluster properties

3. feature-to-cluster properties
4. instance-to-cluster properties
5. single instance content properties

In terms of enriching the user-to-computer communication language, several new types of user feedback can be added to modify the newly introduced cluster-to-cluster properties. For example, if a user finds a cluster is meaningful but is misplaced under an inconsistent parent cluster, the user can move this cluster to a more appropriate place in the hierarchy. We list possible feedback types for hierarchical cluster-to-cluster properties.

- Remove-Cluster(s): when a cluster is too noisy to be understood or a user doesn't think the idea conveyed by the cluster is significant, the user may remove this cluster and its descendants.
- Add-A-Cluster: when there is no cluster that represents a certain idea a user has in mind, the user can create a new empty cluster and place it under a reasonable parent cluster. Then a user can optionally populate this cluster by moving relevant documents into it.
- Move-A-Cluster: a user can drag and drop this cluster under a more reasonable parent cluster.
- Merge-Clusters: when a user thinks a cluster contains a repetitive idea that has been represented in another cluster, the user can merge the two clusters.
- Split-A-Cluster: when a cluster is noisy and a user thinks that it mixes up different ideas but still wants to keep the cluster, a user may request that the computer splits this cluster into smaller clusters.
- Unify-Clusters: when a user finds clusters in a branch over-specify a topic, a user can request the computer unifies all sub-clusters to be one cluster. This feedback type is especially useful when a user requests a cluster splitting to explore the possible ontology space but finds the splitting not necessary.

Another enrichment of the user-to-computer language is to enable drag-and-drop actions on non-hierarchical property types such as:

- Move-A-Document: when a user thinks a document doesn't belong to its currently assigned cluster but it should belong to another cluster, a user can drag and drop the document from its current cluster to the more appropriate cluster.

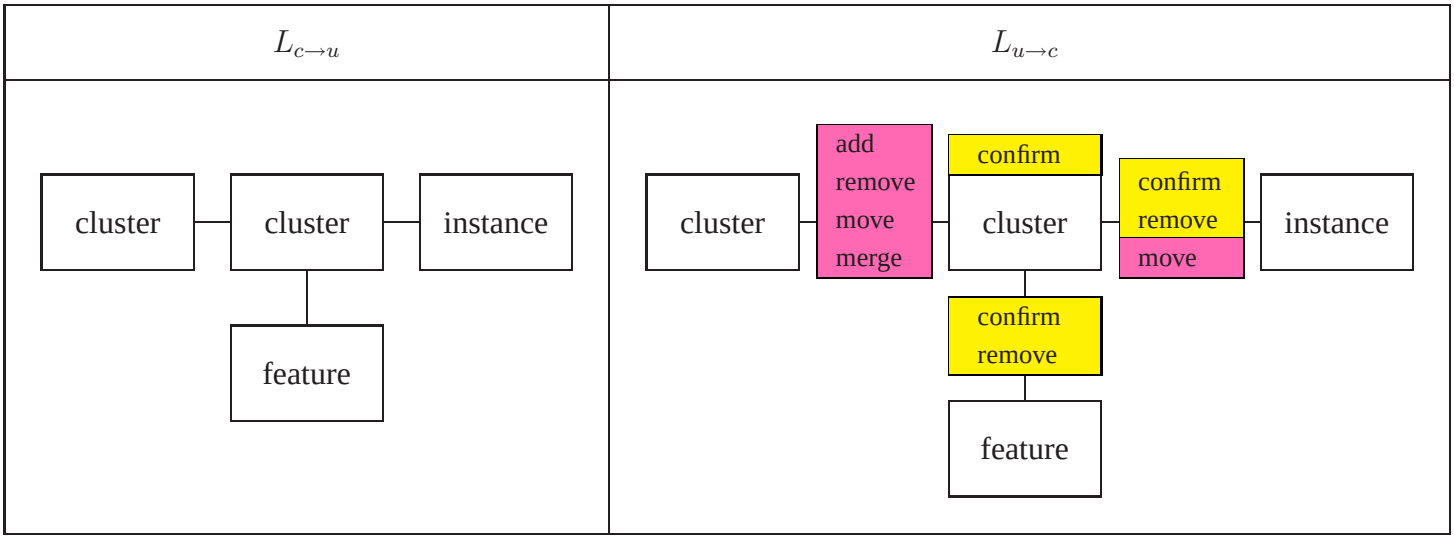


Figure 3.11: The communication languages of our first version of the hierarchical mixed-initiative clustering system. Feedback types with the pink background are new introduced feedback types to handle hierarchical clustering in this system. Feedback types with the light yellow background are inherited from the previous non-hierarchical system.

- Move-A-Feature: when a user thinks a key feature doesn't represent its associated cluster but is a good key feature for another cluster, a user can drag and drop the feature from its current cluster to the more appropriate cluster. This situation happens often when a cluster mixes documents of more than one meaningful topics.

We first integrate five of above mentioned feedback types into our hierarchical mixed-initiative clustering system while still keeping all user feedback types we have studied in the previous system. The previous feedback types consist of confirming feedback on single cluster label properties, confirming and removing feedback on instance-to-cluster properties, and confirming and removing feedback on feature-to-cluster properties. Although feedback on document-to-cluster properties is common in semi-supervised clustering, feedback on feature-to-cluster and cluster-to-cluster properties is less common. Figure 3.11 shows the communication languages used in this hierarchical mixed-initiative clustering system.

3.2.2 Coordination for Hierarchical Clustering

As mentioned above, we generate initial hierarchical clusters of depth two by using a generative clustering model in the first level and applying social network analysis for the second level to produce purer sub-clusters. The results from this approach often contain many errors in parent-child relationships and many incorrect sibling relationships, compared to the user's desires. The benefit of using social network analysis to refine first-level clusters into sub-clusters is to create separate social cliques (purer sub-clusters) so a user can understand and manipulate them more easily. After a user feedback session on this initial hierarchical result, we retrain the hierarchical model based on the user feedback.

The re-training algorithm re-uses the user-modified hierarchy but adjusts the document-to-cluster properties. It adopts the "Pachinko-machine" concept described in [29]. and trains a set of SpeClustering models [23] as classifiers for the root node and intermediate nodes in a hierarchical clustering. Each document is distributed to one sub-cluster for further training. The distribution is based on the document's posterior probabilities given the model of the parent node. Details of the SpeClustering model can be found in Section 3.1.2. Since we train separate SpeClustering classifiers for root and intermediate clusters, this is similar to performing different soft feature selections at each cluster. The SpeClustering algorithm can accommodate all non-hierarchical user feedback types shown in Figure 3.11. We will refer to this hierarchical model as the "Cascading SpeClustering model" in the rest of the thesis. Figure 3.12 illustrate the basic idea of the Cascading SpeClustering model – three SpeClustering models are trained in this example, one for the root node 1 and two for the intermediate nodes 2 and 5.

As in the non-hierarchical clustering system, we extract both word features and person features from the email corpus. The SpeClustering algorithm has an extension to assign different weightings, e.g., $\{\xi_a, \beta_a\}$ and $\{\xi_b, \beta_b\}$, to different feature sets, so it can handle word and person features jointly. In order to simulate the social network analysis that is used to obtain the initial hierarchical clustering result, we add a "PersonWeight" parameter for the second and deeper levels in the hierarchy. The value of PersonWeight multiplies counts in the person corpus, to emphasize these counts relative to word counts. Note our algorithm for retraining the hierarchical clustering in the phase of user feedback does not use social network analysis, because it is not obvious how to perform social analysis sub-clustering while respecting the constraints imposed by user feedback. This weighting of person features provides a mechanism to amplify the importance of grouping together

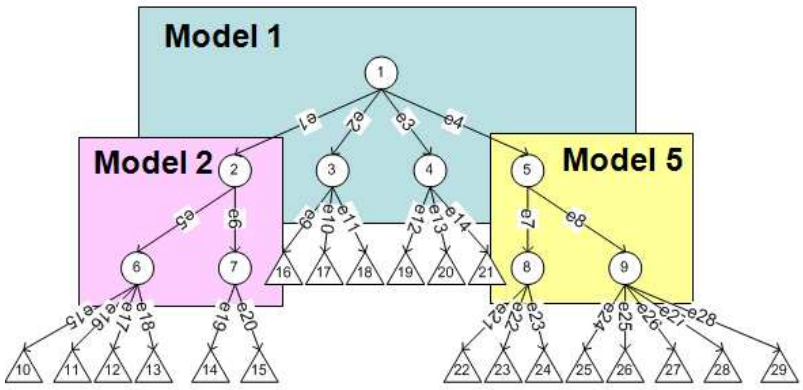


Figure 3.12: An illustration of the Cascading SpeClustering model.

emails involving the same people, and is thus analogous to social network analysis, but fits into our SpecClustering algorithm which can accept constraints imposed by enriched non-hierarchical user feedback.

In addition, we define "complete hierarchical user feedback" as a modification from an imperfect hierarchy, which contains undesirable parent-child and sibling relationships (from the user's perspective), to a reference (target) hierarchy. Since it is not practical to expect complete hierarchical feedback from a user, a user can quit whenever they want, and leave the machine to retrain the hierarchical model starting from the user-modified hierarchy and subject to their non-hierarchical feedback.

When we apply a user's hierarchical feedback for model retraining, the user modification is most likely not complete. Therefore, we add a "StructPrior" parameter that indicates the machine's belief that the user left documents at the correct locations in the hierarchy. The value of the "StructPrior" parameter is used to initialize the posterior probability distributions among sibling clusters. To be more specific, if a document x_i is assigned to a leaf cluster c in the initial hierarchical clustering and c has three sibling clusters, the re-training process initializes the posterior probability $P(c|x_i) = StructPrior$, and the posterior probabilities of each of the three sibling clusters given x_i as $(1 - StructPrior)/3$. When the value of the StructPrior parameter is 1, the algorithm preserves these document-to-cluster assignments in model initialization. When the parameter value is lower, the re-training algorithm is more likely to re-assign documents to other clusters within the hierarchy.

For each intermediate node, the SpeClustering classifier is trained using the relevant feedback entries. The algorithm extracts feedback entries relevant to this node and all its descendant clusters so descendants' feedback entries can propagate to their parent and ancestor nodes. We need to convert hierarchical feedback entries to positive/negative feedback because the SpeClustering model accepts only non-hierarchical positive/negative feedback. For example, a document-move feedback entry can be converted to a negative feedback entry for the original cluster and a positive feedback entry for the target cluster.

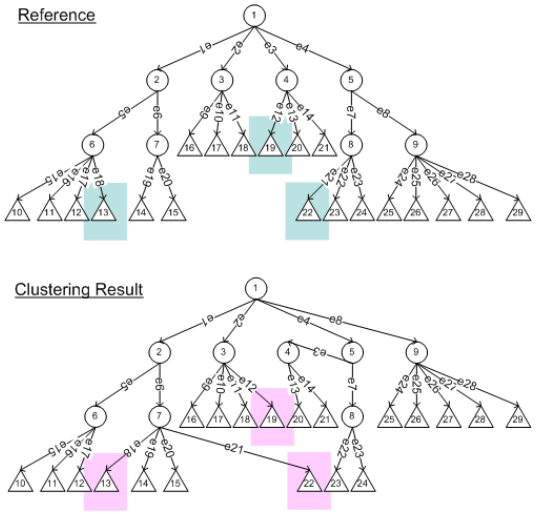
Alternative hierarchical models like the shrinkage model [36] or Hierarchical Latent Dirichlet Allocation [6] are both possible. For the shrinkage model, we would need to design model adaptation heuristics for feature feedback types. Hierarchical Latent Dirichlet Allocation was our first choice but the Markov chain processes' slow convergence is incompatible with our goal of efficient interaction between machine and user.

3.2.3 Distance Measurement between Hierarchies

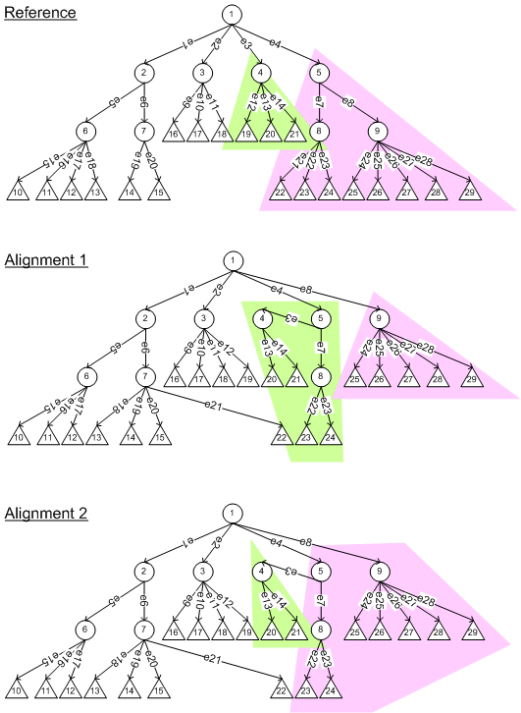
One important consideration in hierarchical mixed-initiative clustering is evaluating the clustering results (including initial hierarchical clustering results, hierarchies modified after user feedback, and re-trained hierarchical clustering results), especially when two clustering results have different hierarchical structures. Our approach defines an edit distance measure between two hierarchies, and then evaluates any proposed clustering hierarchy by its edit distance to the correct (reference) hierarchy. The reference hierarchy used in the following experiments was constructed by the user prior to the beginning of this thesis study.

To see the difficulty in comparing two hierarchies, consider the two hierarchies, Reference and Clustering Result, in Figure 3.13 and the question of how to align these two hierarchies. It is not difficult to align the left-hand side subtrees. Figure 3.13(a) shows cluster (circle node) 2, 3, 6, and 7 can be aligned and it results in clustering errors of document (triangle node) 13, 19, and 22. However, the alignment of right-hand side subtrees is not so trivial. Figure 3.13(b) shows two possible mappings from Clustering Result to Reference. Alignment 1 sticks to the hierarchical constraints imposed by Clustering Result, while Alignment 2 violates the constraints but has higher precision and recall at the document level.

Luckily, we figure that the distance between two hierarchies can be measured by the



(a) The left-hand side sub-tree (node 2, 3, 6, and 7) can be aligned between two hierarchies and results in 3 document clustering errors.



(b) Two different alignments from the right-hand side sub-tree of a clustering result to a reference.

Figure 3.13: Hierarchical structure comparison

number of editing operations needed to change one hierarchy into the other, especially when the set of hierarchical feedback types can be transformed naturally into a set of editing operations. For example, "Move-A-Cluster" feedback is equivalent to modifying the parent orientation of a parent-cluster-to-child-cluster property in which the moved cluster is the child cluster. Similarly, "Move-A-Document" feedback is changing the cluster end of the moved document's document-to-cluster property.

We define "edge modification ratio" as the minimum number of feedback steps required for complete hierarchical user feedback, divided by the total number of edges in the reference hierarchy. The Clustering Result hierarchy in Figure 3.13 needs five feedback steps that modify edge e3, e8, e12, e18, and e21 accordingly in order to match the Reference hierarchy. There are 28 edges in the Reference hierarchy, so the edge modification ratio is 0.18 (5/28) for this clustering result.

Ideally, evaluation for hierarchical mixed-initiative clustering should request user studies to examine real users' comprehension of a hierarchical mixed-initiative clustering, but user studies are very expensive. We leave the user study evaluation (see details in Chapter 5) for our full-fledged mixed-initiative clustering system that integrates both communication enrichment and mixed-initiative interaction capability. Instead, the measurement of edge modification ratio simulates a user's feedback efforts in correcting a hierarchical clustering. In other words, edge modification ratio serves as an automatic measurement for a hierarchical mixed-initiative clustering system that focuses on the communication enrichment component while skipping the interaction component, and examines whether a hierarchical clustering model and its model retraining algorithm are feasible to learn from enriched hierarchical feedback.

The exact implementation of calculating the edge modification ratio, e.g., simulated feedback efforts, consists of four steps. The first step is to find the best many-to-one mapping from learned clusters to reference clusters. The second step chooses the optimal one-to-one mapping from learned clusters to reference clusters. The third step adjusts the cluster-to-cluster property of each cluster if it meets one of the following criteria: (1) if it is a sub-optimally mapped cluster, merge it to the optimally mapped cluster, (2) if it is a optimally mapped cluster but its parent-child property is inappropriate, move it to the right position in the hierarchy, and (3) if there is no mapped clusters to a reference cluster, add a new cluster at the right hierarchical position. This step also counts how many times cluster-to-cluster adjustment is made. The fourth step counts the number of

remaining inappropriate document-to-cluster properties. The combined number of the third step and fourth step is the number of edge modification that we use to simulate a "complete hierarchical user feedback" session.

The concept of "edge modification ratio" is very similar to "tree edit distance" [4] where different feedback types are mapped into different operations and the cost function is uniform.

3.2.4 Experimental Results

We use the same email dataset (EmailYH) introduced in Section 3.1.3. There are 623 emails in this dataset that had been manually organized as a hierarchy prior to the start of the mixed-initiative clustering study. We use this hierarchy as the reference hierarchy. It consists of 15 clusters including a root, 11 leaf clusters, and 3 intermediate clusters.

In these experiments, a mixed-initiative process consists of the generation of an initial hierarchical clustering with depth two, a long user feedback session, and model retraining. In the feedback session, the initial clustering is presented to the email owner in the user interface we introduced in Figure 3.10 to browse and give feedback. The algorithm for producing initial clusters is non-deterministic. We picked the same five initial first-level clustering results used in the non-hierarchical mixed-initiative clustering experiments. These initial results have the highest likelihood values among fifty autonomous clustering results. Each of these five single-level clustering results were then extended to two-level hierarchical clusterings by applying social network analysis.

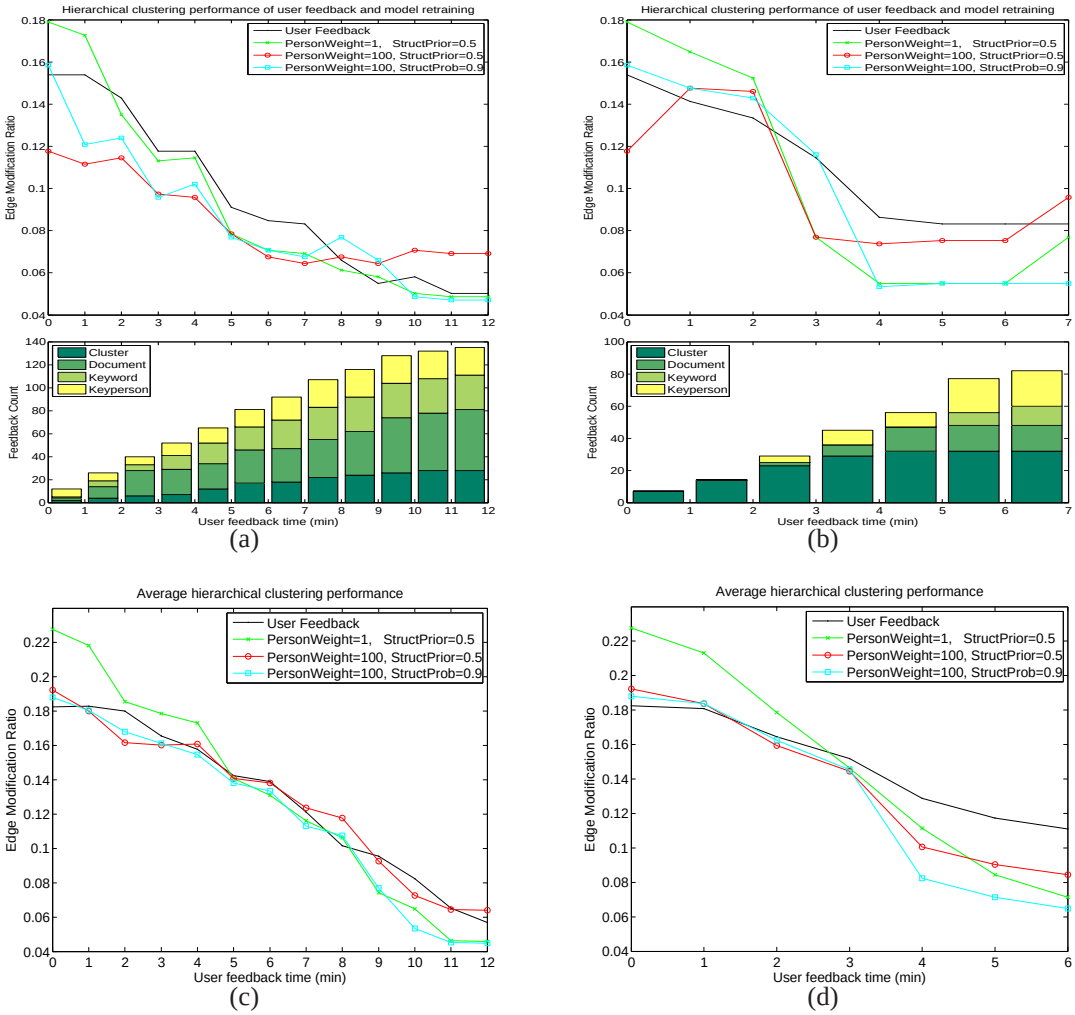
For each of these five initial hierarchical clusterings, the email owner performed what we call a "diligent" feedback session and a "lazy" feedback session on different days a couple weeks apart. In the "diligent" session, the user examines keywords and key-persons in detail, and often checks document assignments to clusters. In the "lazy" session, the user may select a few keywords and key-persons, and skims through or skips documents.

The first row in Figure 3.14 shows an example of hierarchical mixed-initiative clustering, derived from a single initial clustering result plus its "diligent" or "lazy" feedback sessions. The horizontal axis in each plot corresponds to minutes of user feedback, whereas the vertical axis gives the quality of the hierarchical clustering at that point in time, measured by its edge modification ratio relative to the reference (ideal) clustering. The dot-marked (black) lines show how manual user feedback modifies the quality (the

Diligent Session

Lazy Session

Init1



Avg(Init1-4)

Figure 3.14: (a)(b): An example pair of re-trained results derived from a specific initial result and feedback counts of two feedback sessions on this initial result. (c)(d): Averages of 4 re-trained results in different feedback sessions.

edge modification ratio) of the initial hierarchical clustering over time. Lower edge modification means a user can achieve the reference hierarchy using fewer remaining feedback steps. The other lines show edge modification ratios for different clustering results obtained after model re-training with different parameter settings. Each marked point represents a retrained model learnt from the user feedback on the initial hierarchy up to that time. The cross-marked (green) lines show the edge modification ratio of results with no special weighting on the person corpus, whereas the circle-marked (red) lines show results that give the person corpus a high weight. The circle-marked (red) line and the square-marked (cyan) line in a same plot show how the level of trust of the user modified hierarchy impacts the retrained hierarchical clustering results. The closer the StructProb parameter sets to one, the more the machine assumes the user modified hierarchy is nearer to complete hierarchical user feedback. The circle-marked lines show the re-trained results with a low trust value, and square-marked lines show results with a high trust value.

The histogram below each plot shows the cumulative count of user feedback, with colors indicating the different types of feedback. The diligent feedback sessions are longer and involve larger total counts of feedback from the user. As designed, the composition of feedback is different between the diligent session and the lazy session. A user gives much less feedback on document-to-cluster properties in the lazy session than in the diligent session. The user also focuses on giving cluster-related feedback in the first 2 to 3 minutes of the lazy feedback session.

For this specific initial clustering result, the diligent user benefited from the machine's retraining when she provided less than 7 minutes' feedback. After the 8th minute, the re-trained result only maintains performance similar to the user's manual efforts. This is because if the user has meticulously corrected all document-to-cluster errors occurred in the initial hierarchy, the re-trained model can predict at best what a user has manually done. On the other hand, the re-trained results from a lazy user's feedback gets better after the user completes the cluster-to-cluster property modifications, e.g., there is no cluster-level feedback after the 4th minute. In terms of comparing the performances between different user behaviors, in this specific case, a diligent user needs to spend 11 minutes correcting the cluster hierarchy if they work alone, whereas a lazy user with the machine's assistance achieved an equivalent performance in four minutes. This result proves the advantages of adding the cluster-to-cluster property type and its corresponding feedback types to the communication languages for hierarchical mixed-initiative clustering.

Figure 3.14(c) and 3.14(d) show the aggregated re-trained results from four out of five initial hierarchies with respect to two different feedback sessions. The general trends hold that a diligent user gains marginally and a lazy user gains more from model re-training.

Notice the edge modification ratio of the black line (user feedback) at the zeroth minute in Figure 3.14 gives the performance of the two-step approach (flat clustering plus social network analysis) for generating initial hierarchical results. We have learnt from the activity extractor experiences that this approach is good at integrating different aspects of email content to produce user comprehensible sub-clusters. The reason we switch to the Cascading SpeClustering model for hierarchical model retraining is because the two-step approach cannot learn from various types of user feedback including the hierarchical feedback. In other words, the two-step approach cannot achieve the coordination with the enriched hierarchical communication languages.

However, given no user feedback and without weighting the person corpus heavily, e.g., PersonWeight equals one, the edge modification ratio is much worse than the two-step approach. This confirms our previous study that social network analysis helps generate more user-comprehensible clusters (activities) and also shows the Cascading SpeClustering model cannot beat the good heuristics without user feedback. With PersonWeight set to 100, the Cascading SpeClustering model is more capable of achieving results similar to social network analysis. Given the same StructPrior value, weighting the person corpus heavily results in a lower edge modification ratio in the early stage and using no special person weighting is better in the later stage. It shows that when the user's feedback on a hierarchy is partial, the background knowledge of social cliques is informative and when the user's feedback has fixed the hierarchical structure, the textual content is more helpful.

The StructProb parameter can be interpreted as the level of trust placed upon the user modified hierarchy where a higher value means more trust. The aggregated results in Figure 3.14(c) and 3.14(d) show that it is better to assume the document-to-cluster assignments in the user modified hierarchy are correct and initiate the re-trained model accordingly.

However, the performances of re-trained results vary greatly depending on the quality of initial autonomous clustering results and the personal style of giving feedback. The re-trained results using the 5th initial result are shown in Figure 3.15, which are a lot worse and spikier than the other four pairs of re-trained results. The reason is that the 5th initial result has two big clusters and each of these two clusters has documents belonged to

Diligent Session Lazy Session

Init5

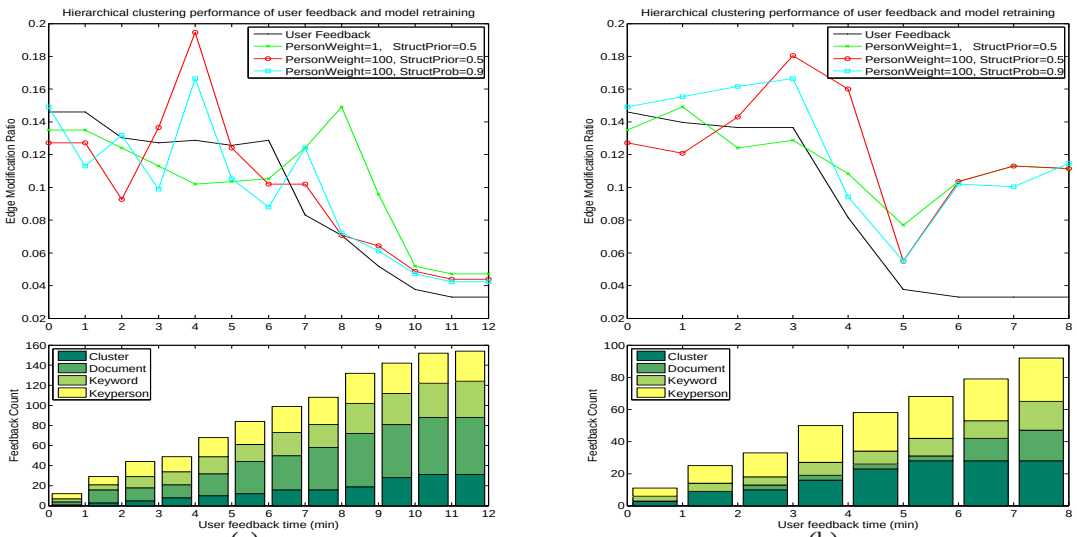


Figure 3.15: This pair of re-trained results shows that our current model re-training method doesn't always help the user.

two or three reference clusters. With the feedback types allowed by this system's user-to-computer language, the user must resort to a sub-optimal feedback strategy like confirming it as one reference folder and moving the other half of documents to a newly created cluster. When a lazy user doesn't provide enough user feedback to teach a machine, the re-trained model may converge incorrectly as in Figure 3.15(b). A better solution is to include the Split-A-Cluster feedback type in the user-to-computer language. In addition, re-training a branch of clusters in the hierarchy instead of the entire hierarchy of clusters also helps.

We re-implemented the first version of the hierarchical mixed-initiative clustering system to a fully interactive system between a user and machine. In this second revamped system, we enhance the communication languages further more by adding "Split-A-Cluster" and "Unify-Clusters" user feedback types for cluster-to-cluster properties and "Move-A-Feature" feedback type for feature-to-cluster properties. Also, a user can request the system to retrain its machine's clustering model from the root cluster, where the whole cascading SpeClustering model is updated, or from a selected intermediate cluster, where only the part of the model corresponds to the selected branch of clusters is updated. Figure 3.16

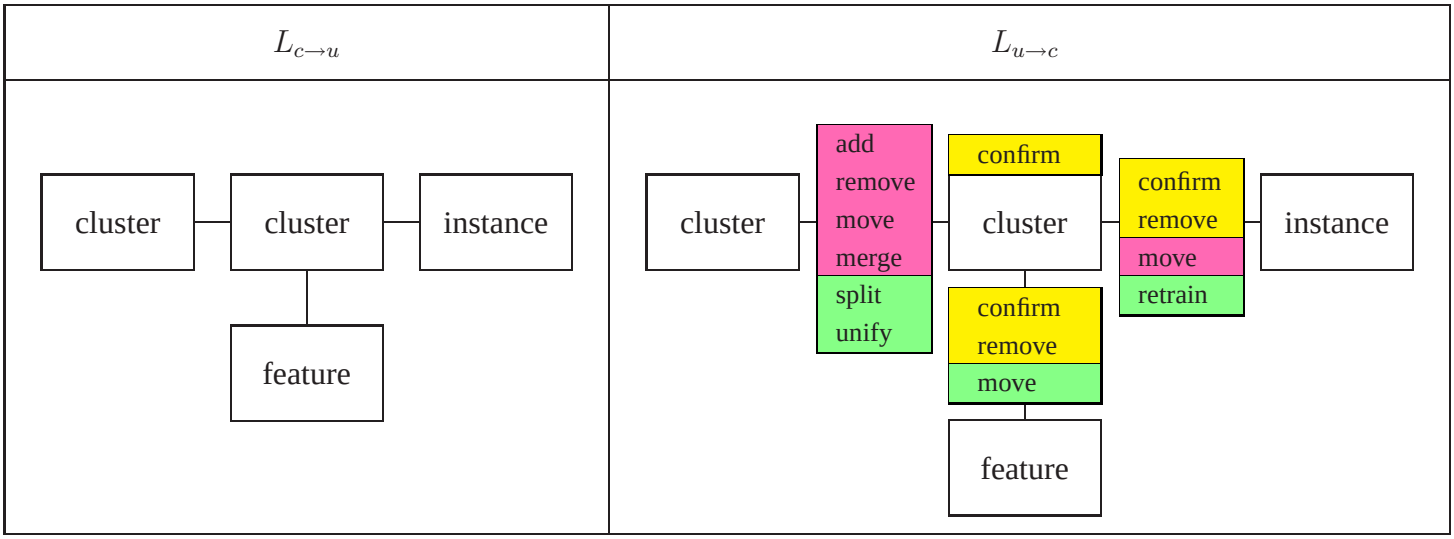


Figure 3.16: The communication languages of the second version of the hierarchical mixed-initiative clustering system. With the green background are feedback types newly introduced in this version of the hierarchical system. Feedback types with the pink background are inherited from the first hierarchical system. Feedback types with the yellow background are inherited from the non-hierarchical system.

illustrates the communication languages used in this system. With the green background are feedback types newly introduced in this version of the hierarchical system. Feedback types with the pink background are inherited from the first hierarchical system. Feedback types with the light yellow background are inherited from the non-hierarchical system. Users of this system have all basic measures for hierarchical and non-hierarchical modification. The major upgrade of this second hierarchical clustering system is the fully interactive capability, which will be discussed in detail in the next chapter.

3.2.5 System Summary

In this section, we discuss how to enrich communication languages for hierarchical mixed-initiative clustering and how to achieve coordinated hierarchical model retraining.

We build a first hierarchical mixed-initiative clustering system, and apply it to hierarchical clustering of email. A simple hierarchical clustering model, Cascading SpeClustering, is used to handle various types of user feedback including feedback on conceptual feature-to-cluster properties and cluster-to-cluster properties. The experimental results

show that the joint efforts of a machine and a user can usually achieve better edge modification ratio, or equivalently, save a user's time compared to manually editing the initial clustering. We also learn that a user's feedback style matters. A lazy user can gain more benefits from machine's retraining than a diligent user. From the worst case of re-trained results, it seems safer to re-train a specified subset of a hierarchy instead of the whole hierarchy.

3.3 Summary

In this chapter, we demonstrate how to enrich communication languages in mixed-initiative clustering by adding conceptual property types such as key features of clusters and hierarchical cluster-to-cluster properties to $L_{c \to u}$, and adding various feedback types on conceptual properties to $L_{u \to c}$.

Two example mixed-initiative clustering systems, each corresponds to one incremental step of communication enrichment, are discussed in detail. The first system enables user feedback on features for non-hierarchical clustering. In order to achieve the coordination in this mixed-initiative clustering system, we propose a new clustering model, the SpeClustering model, which provides a natural way to adjust its parameters according to user feedback on key features, and a mixed-initiative SpeClustering algorithm to adapt multiple types of user feedback altogether. On top of the first system, the second mixed-initiative clustering system accepts various types of user feedback on hierarchical clustering results. A cascading version of the mixed-initiative SpeClustering algorithm is used for coordinating hierarchical mixed-initiative clustering.

Under a long-user-feedback-session setting, we examine performances of machine's retrained clustering models using different combinations of user feedback types (with or without conceptual properties), and impacts of difficulty levels of clustering tasks and user feedback styles. The experimental results show that using communication languages with conceptual properties gains more improvement than using communication languages without conceptual properties. Given different levels of clustering difficulties, the mixed-initiative clustering system helps a machine learn a better clustering from user feedback on noisy but still meaningful autonomous clustering results. Given different user feedback styles, a lazy user can gain more benefits from machine retraining than a diligent user.

Chapter 4

Practical Issues in Mixed-Initiative Interaction with a User

Interaction is another essential element for mixed-initiative clustering. The interactive loop iterates through the machine learning, machine teaching, user learning and user teaching phases. In practice, the combined time of the machine learning phase and machine teaching phase should be as short as possible so a user doesn't have to wait too long. Also, when a non-oracle user interacts with a mixed-initiative clustering system, she may behave and expect differently from an oracle user who has no need to learn and knows how to teach, which is commonly assumed in many other machine learning tasks. This chapter discusses practical issues we encounter while building a mixed-initiative clustering system that can interact with a non-oracle user in real time.

4.1 High-Latency vs. Low-Latency Interaction

Experiments in the previous chapter are set up as one iteration of the interactive loop of mixed-initiative clustering, where a user gives feedback on machine-extracted properties in a long single user feedback session. This experimental setting is targeted to collecting sufficient user feedback for the retraining of the machine's clustering model, M_c. This setting is not designed to accept a user's retraining request at any time nor to quickly retrain the machine's model (the machine learning phase) and update its computer-to-user communication (the machine teaching phase.) Due to the expected long user feedback

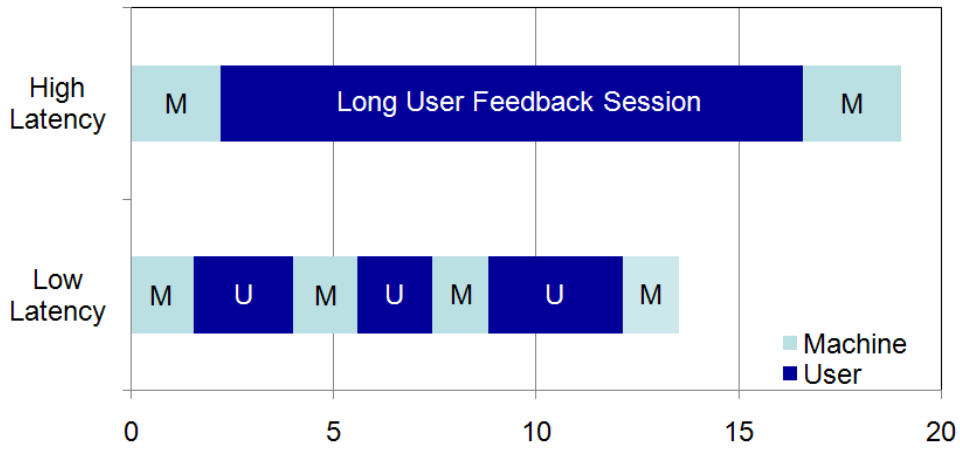


Figure 4.1: The difference between high-latency interaction and low latency interaction. The X axis is a time scale. The light blue blocks correspond to the timing where a machine takes initiative during the interactive process, and the dark blue blocks correspond to the timing of a user's initiative.

sections and sup-optimal speed of interaction, we call this interaction type **high-latency interaction**.

In contrast, **low-latency interaction** refers to mixed-initiative clustering systems that allow their users to retrain a machine's clustering model at any time and expedite the process when a machine takes initiative, especially the machine learning phase. Figure 4.1 illustrates the difference between high-latency and low latency interaction. The X axis is a time scale. The light blue blocks correspond to the periods of time when a machine takes initiative (machine learning and teaching) during the interactive process, and the dark blue blocks correspond to the periods of a user's initiative (user learning and teaching.)

4.2 Exchange of Initiatives

For a low-latency mixed-initiative clustering system, exchange of initiatives is either triggered by some specific types of user feedback or by an automatic mechanism that decides when is appropriate for a machine to interrupt a user and take its initiative [21]. We develop our second hierarchical mixed-initiative clustering system using the first approach.

As described in the final part of Chapter 3, the second version of the hierarchical

mixed-initiative clustering system is a low-latency interactive system. Figure 3.16 illustrates communication languages of this second hierarchical mixed-initiative clustering system. In the user-to-machine communication language of the system, some user feedback types demand immediate machine retraining. These specific feedback types include:

- a user requests retraining of a subtree of the cluster hierarchy,
- a user requests splitting a leaf cluster, and
- a user removes a cluster from the hierarchy.

The machine responds to the user retraining feedback by retraining its hierarchical clustering model, the cascading SpeClustering model, subject to all past feedback related to the selected cluster and its descendant clusters. When the user requests splitting a leaf cluster, the machine learns an unsupervised clustering model using documents in the leaf cluster, assigns these documents into sub-clusters, and adds this newly learned unsupervised model to the cascading SpeClustering model. When a user removes a cluster from the hierarchy, documents in the deleted cluster need to be re-distributed to remaining clusters. The machine semi-supervisedly retrains its cascading SpeClustering model for the remaining clusters, and uses the updated model for the document redistribution. There are two alternative definitions of remaining clusters: (1) the entire cluster hierarchy except the deleted cluster, or (2) all sibling clusters of the deleted cluster. We tested both definitions but couldn't find observable evidence in clustering results to favor one definition over another. The second definition is used in the final system because retraining a subtree of a hierarchical clustering model is faster than retraining the whole tree.

In addition, we implemented but did not employ a retraining-by-example feedback type that allows a user to teach a machine a few instances as labeled examples for a user-specified number of sub-clusters. After the user completes the teaching, the machine retrains its clustering model according to the labeled examples. However, at least one example for each sub-cluster is necessary, which requires additional book-keeping efforts from a user. Due to this reason, the retraining-by-example feedback type is disabled in the second version of the hierarchical mixed-initiative clustering system.

4.3 Model Retraining in the Wild

4.3.1 Issues

When we pilot tested our low-latency mixed-initiative clustering system, we found three issues with regard to retraining the machine's clustering model, M_c, that led to improvements in the final system.

The first issue is about the speed of model retraining. In low-latency mixed-initiative clustering, model retraining should take only a few seconds instead of several minutes because a user has to wait for the machine during this period of time. A rule of thumb we applied during the development of our mixed-initiative clustering system is 10 seconds threshold – whenever model retraining took more than 10 seconds to complete, we investigated methods to expedite the retraining .

> *Guideline 1:* Model retraining should be as fast as possible. Our rule of thumb is to control the time spent on each model retraining under 10 seconds.

The second issue comes from users. Users tend to be lazy and since the low-latency interaction allows a user's model retraining request at any time, it is possible that a user requests machine retraining right after giving very limited feedback or even no feedback at all. A mixed-initiative clustering system shouldn't expect its user to give sufficient user feedback before she hits the retraining button. Also, a user without machine learning background doesn't know the importance of balancing numbers of labeled examples in each cluster (these numbers are used to obtain the prior probabilities in model retraining.) In low-latency mixed-initiative clustering, a user may give many labeled examples to one cluster because she knows the cluster's topic well, and give limited or no labeled examples to another cluster because she cannot recognize the topic of this cluster. As a result, the numbers of labeled examples doesn't necessarily reflect the appropriate prior probabilities, and in many cases, unbalanced labeled examples are given by users unintentionally. With unbalanced labeled examples, a probabilistic clustering model in its likelihood optimization process may expand clusters with more labeled examples at a cost of shrinking other clusters with fewer or no labeled examples, which is undesirable. A machine in low-latency mixed-initiative clustering should prepare to learn from insufficient and unbalanced user feedback.

Guideline 2: Because a user tends to be lazy and may not have enough background knowledge of how machine learning algorithms work, it is an important capability for a machine to learn from insufficient and unbalanced user feedback.

The third issue is what is the appropriate amount of change between the previous machine-proposed clustering and the new clustering revision obtained from system retraining. Too much change between two consecutive clustering results may confuse a user instead of assisting her. This is because a user has established a (partial) topic ontology from reading the previous clustering result. In the most common case, a user submits a retraining request after giving the machine some feedback regarding its proposed topic ontology. The user wants the system to learn a model that better reflects her current ontology instead of a model that optimizes its objective function but produces an unfamiliar clustering result.

Nevertheless, a user also wants noticeable change in the new clustering result according to her feedback. For example, when a user confirms a keyword to a cluster and then requests the mixed-initiative system to retrain its clustering model, she may expect the system to learn some other keywords that are highly related to the confirmed keyword. When a user observes that enhancement in the new clustering result is specifically based on her feedback, she is more likely to trust the mixed-initiative clustering system.

The emphasis on noticeable change seemingly contradicts the emphasis on similarity between clustering revisions. However, they share the same principle that a mixed-initiative clustering system needs to help a user develop her still evolving user ontology. The similarity between clustering revisions preserves a user's existing ontology so a user can improve her ontology step by step. The noticeable change, on the other hand, focuses on how a mixed-initiative clustering system should assist a user by extrapolating her feedback because the feedback indicates the direction of a specific step to improve her user ontology. Combining these two concerns, we come up with the following guideline:

Guideline 3: The goal of a mixed-initiative clustering system is to help a user develop her still evolving user ontology. Under this goal, consecutive machine clustering revisions shouldn't make dramatic change in order to preserve a user's existing clustering ontology, but should make specific change by extrapolating user feedback.

4.3.2 Heuristic Methods

Based on these guidelines, we apply several heuristic methods to extend the low-latency interactive version of the hierarchical mixed-initiative clustering system. These heuristic methods are tied to the clustering algorithm used in our system, which may or may not be applicable to other mixed-initiative clustering systems. We don't investigate theoretical solutions for model retraining under the low-latency interactive environment in this thesis, but it is a good topic for future study.

The first heuristic utilizes the fact that computing a multinomial Naive Bayes model, which uses a single-loop EM algorithm for parameter estimation, is faster than computing a SpeClustering model, which uses a double-loop EM algorithm. The main purpose of computing a SpeClustering model instead of a multinomial Naive Bayes model is because it can learn from user feedback on feature-to-cluster properties. In low-latency interaction, a user often gives limited and unbalanced feedback, which might not include feedback on key-features. When there is no feedback on key-features, we compute the faster multinomial Naive Bayes model instead of the slower SpeClustering model. For example, when a user wants to split a cluster into sub-clusters, these yet-to-be-created sub-clusters have no feedback at all and thus no feedback on key-features. This heuristic solution implies that the machine's clustering model is replaceable because the agreement between a user and a machine in mixed-initiative clustering is established upon the shared properties, not model assumptions. As long as a model is capable of extracting the shared properties, even it can not learn from some types of properties after user feedback modification, it can be used before any un-learnable user feedback is given. The general principle is *if a user only uses a subset of the feedback language, a machine can use a cheaper machine learning algorithm.*

The second heuristic method is quite simple. We limit the number of iterations to be 10 during the EM process of computing a SpeClustering model. This heuristic not only accelerates the machine learning phase but also prevents the newly learned model from producing a clustering result that deviates too much from the previous clustering result. The value of 10 is set heuristically by observation of clustering results and also because the SpeClustering model usually converges in 20 to 30 iterations.

The third heuristic for model retraining is aimed to boost tiny clusters. Before introducing the heuristic itself, let's discuss the implication of a tiny cluster first. For mixed-

initiative clustering, each cluster should represent a salient topic, i.e; it is well supported by the data, in user ontology. A cluster with many fewer documents than its sibling clusters either doesn't represent a salient topic or is a cluster newly added by a user. For the former case, we rely on a user's cluster removal feedback to get rid of these clusters. The later case is the target case of our third heuristic – automatically boosting sizes of tiny clusters, especially newly-added ones. A newly-added cluster is typically tiny before retraining because once a user gives a mixed-initiative clustering system a few examples, she expects the system to find more similar documents for her. To be more specific, we define a cluster as a tiny cluster when it contains less than one third of the documents of its sibling clusters. The one third threshold is picked by observing which threshold value results in clusterings that the system developer liked the most.

The third heuristic boosts tiny clusters by applying (pseudo) relevance feedback techniques [2]. In information retrieval, relevance feedback reformulates a user's initial query to an expanded query by weighting words in the initial query and words in relevant documents. Pseudo relevance feedback further assumes that the top N retrieved documents are relevant even without explicit user relevance judgement. The expanded query is expected to be closer to relevant documents and further from the non-relevant ones than the initial query. The Rocchio algorithm [47] is commonly used for the query expansion purpose. The algorithm calculates the expanded query vector, $\vec{q_m}$, using the following formula:

$$\vec{q_m} = \alpha\vec{q_0} + \beta\frac{1}{|D_r|}\sum_{\vec{d_j}\in D_r}\vec{d_j} + \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d_j}\in D_{nr}}\vec{d_j}$$

where $\vec{q_0}$ is the original query vector, $\vec{d_j}$ is the tf-idf vector for each document, D_r and D_{nr} are the set of known relevant and non-relevant documents respectively, and α, β, and γ are weights that adjust the relative impact of the original query, relevant and non-relevant documents.

Similarly, in mixed-initiative clustering, we can expand limited or empty user feedback of a tiny cluster by assuming that machine-proposed keywords and documents of a tiny cluster are all positive examples unless a user says otherwise. Using the expanded user feedback, the system can find more robustly other relevant documents than using the original limited or empty user feedback. We take user feedback on key features as the original query vector while the positive and negative feedback on document-to-cluster properties

are considered as the relevant and non-relevant documents. Unlike the query expansion, there can be negative values in the original vector because a user can remove a key feature from a cluster, and there is a combination of explicit feedback given by a user and pseudo feedback. We modify the Rocchio formula to handle these differences:

$$\vec{q}_m = \alpha\vec{q}_0 + \alpha'\vec{q'}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j + \beta'\frac{1}{|D'_r|}\sum_{\vec{d}_j \in D'_r}\vec{d}_j + \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j$$

where $\vec{q}_0$ and $\vec{q'}_0$ refer to machine-proposed key-features of a tiny cluster with and without user feedback, and D_r, D_{nr} and D'_r refer to machine-proposed document-to-cluster properties with confirmation feedback, with removal feedback, and without user feedback. α, α', β, β', and γ are weights attached to $\vec{q}_0$, $\vec{q'}_0$, D_r, D'_r and D_{nr} correspondingly.

In practice, we set the values of α, β, and γ to be 1.0, and α' and β' to be 0.5. We don't fine tune these parameter values. They are obtained by suggestions in the literature [2] and satisfactory observations in system testing. After computing the expanded feedback, we use the cosine similarity measurement to find other similar documents and move them into the tiny cluster until the size of the tiny cluster is no longer smaller than one third of the sizes of its sibling clusters. However, explicit user feedback always has higher priority than the pseudo boosting. When a document is already confirmed to a cluster but found similar to the expanded feedback of a tiny cluster, it should remain in its confirmed cluster.

4.4 Managing User Feedback

In some situations, feedback items given by a user may contradict one another. For example, a user confirms a keyword for a cluster but later decides the keyword is more suitable for another cluster so she moves the keyword to the other cluster. In this example, the original feature-to-cluster property is confirmed first and negated later in the moving feedback. We use a simple rule of thumb to handle contradictions between feedback items in the low-latency mixed-initiative clustering system – the later feedback overwrites the previous one if there is a contradiction.

Due to the importance of communication enrichment, the enriched user-to-computer language of various feedback types also introduces potential feedback contradictions. Below is a general guideline for handling feedback contradictions.

Guideline 4: Feedback items given by a user may contradict one another especially when the user is allowed to give various types of feedback in mixed-initiative clustering. A mixed-initiative system developer should check potential feedback contradictions every time a new feedback type is added. A simple mechanism to resolve feedback contradictions is to let later feedback overwrite the previous contradicting one.

In practice, a user may not be familiar enough with the communicative languages of a mixed-initiative clustering system. She may give feedback that is not appropriate to the system or feedback the system is not designed for. We call this kind of user feedback "undesirable user feedback." For example, when a user requests "Split-A-Cluster" feedback on an intermediate cluster, it is hard to tell what a user wants to do with the existing child clusters of the intermediate cluster. A checking mechanism is necessary to handle this situation. In our low-latency hierarchical mixed-initiative clustering system, a short warning message will pop up and educate the user why this feedback should be avoided when a user requests undesirable user feedback. Other undesirable user feedback includes "Add-A-Cluster" under a leaf cluster, "Move-A-Document" to an intermediate cluster, and requesting model retraining on a leaf cluster.

Guideline 5: A checking mechanism is necessary to filter out undesirable user feedback and educate a user's feedback proficiency.

In addition, we only allow a leaf cluster being merged into another leaf cluster and a cluster being moved under another intermediate cluster. When a user drag and drops a cluster to a destination cluster, the system decides which user feedback type, "Merge-Clusters" or "Move-A-Cluster," is appropriate based on whether the destination cluster is a leaf cluster.

4.5 Interface Design

We re-designed the interface of the high-latency system to produce an interface that is appropriate for users who are not the system's developers. A snapshot of the redesigned interface of the low-latency mixed-initiative clustering system is shown in Figure 4.2. This

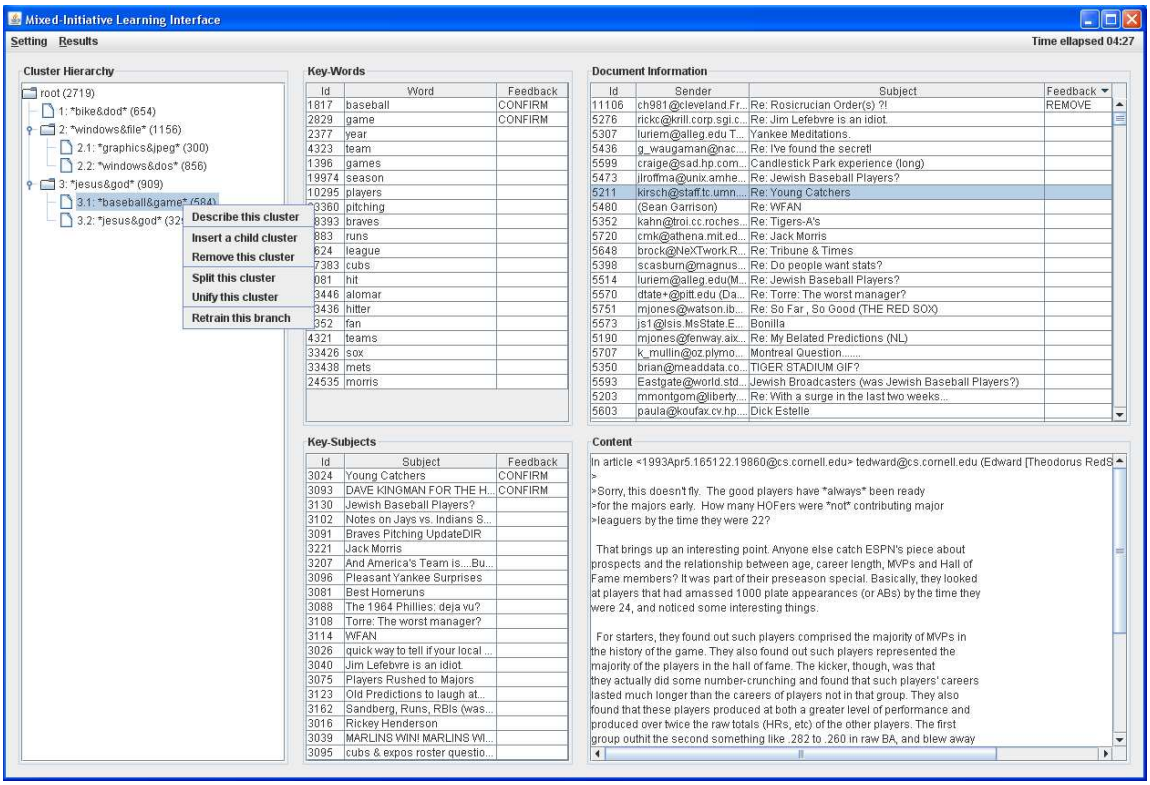


Figure 4.2: The interface for low-latency mixed-initiative clustering.

interface still contains three major areas: (1) the left panel represents the hierarchy of clusters with a menu of various feedback types that will pop up next to the selected cluster upon right clicking, (2) the middle panel represents the keywords and key-subjects of the selected cluster, and (3) the right top panel represents all documents assigned to the selected cluster and the right bottom panel shows the text content of a selected document. A user can move a selection of documents or features to a cluster or perform cluster moving and merging through drag-and-drop. The pop-up menu in the cluster panel allows additional feedback types for cluster-to-cluster properties. A combo-box selection is enabled in the feedback columns in the keyword/key-subject panels and document panel. Users can give combined feedback of various types and request a machine to retrain its clustering model of any branch of clusters at any time.

Comparing this interface with the previous user interface shown in Figure 3.10, we see three major changes. The first change is automatically attaching the top two keywords of each cluster to the numeric cluster label displayed in interfaces of previous systems. Al-

though machine-generated text cluster labels are noisy and the same information is duplicated in the keyword panel, text cluster labels help a user quickly judge the meaningfulness of a cluster, guess the main topic of a cluster, and maintain overall semantic consistency of clusters in a hierarchy. An asterisk symbol, "*", is added in front and at the end of the two keywords indicating that they are generated automatically by the machine. The symbol is removed if the text label is updated by a user's cluster description input.

> *Guideline 6:* Don't generate numeric indices as cluster labels. Generate text labels even the automatic text labeling algorithm is far from perfect.

The second change is the swap between the key-feature panel and the document panel, which is suggested by a third-party pilot tester of the system. The tester thinks the left to right arrangement of the cluster hierarchy panel, key-feature panel, and document panel is more cognitively consistent, from the most conceptual property type to the least conceptual property type, than the document-panel-in-the-middle arrangement.

The third change is the use of a pop-up menu in the cluster panel and combo boxes for feedback columns in the key-feature panels and document panel. The pop-up menu and combo boxes are attached right next to their target properties that provide intuitive user feedback manipulation. They also keep a flexible list of feedback types so extending or disabling feedback types can be done without modifying the interface layout.

As we learned in the previous system, moving feedback through drag-and-drop is more intuitive for users than giving positive or negative feedback. In this interface, we allow moving feedback on all cluster-to-cluster, feature-to-cluster, instance-to-cluster properties.

> *Guideline 7:* The interface design should arrange property panels to reduce a user's cognitive effort, and provide intuitive feedback interaction.

4.6 Summary

In this chapter, we distinguish low-latency interaction from high-latency interaction. Unlike the presumption of a long user feedback session in high-latency interaction, a low-latency mixed-initiative clustering system allows its user to retrain a machine's clustering model at any time and expedites the process when a machine takes initiative, especially

the machine learning phase. We examine several practical issues which occur only in the low-latency interactive environment. These issues include the timing of initiative exchange, how to retrain a machine's clustering model with insufficient and unbalanced user feedback, what is appropriate change between clustering revisions, how to resolve feedback conflict and build a user's feedback proficiency, and ideas about interface design. We summarize several guidelines that we learnt from building our low-latency mixed-initiative clustering system. We also describe three heuristic methods that are implemented for retraining a machine's clustering model in the low-latency interactive environment.

Chapter 5

User Studies on the Effectiveness of Mixed-Initiative Clustering

We demonstrate how to build a mixed-initiative clustering system with enriched communication languages in Chapter 3 and with low-latency interaction in Chapter 4. In this chapter, we evaluate the performance of the final system we built – a low-latency hierarchical mixed-initiative clustering system. Due to the user involvement in the interactive loop, evaluations of a low-latency mixed-initiative clustering system are hard to conduct offline. We propose a user study design for evaluation purposes and analyze results obtained from testers' interactions with the low-latency mixed-initiative clustering system.

5.1 Design of User Studies

The primary hypothesis of our mixed-initiative clustering study is that mixed-initiative clustering can help a user achieve better clustering results than non-mixed-initiative approaches due to the enriched communication and the interactive learning and teaching between a user and a machine. In order to examine this primary hypothesis, we present user studies involving two scenarios, a learning scenario and a teaching scenario, that emphasize different communication directions. The reasons behind the break down of two scenarios are (1) non-mixed-initiative approaches often have only unidirectional communication, and (2) an attempt to cover different purposes which a user would consider mixed-initiative clustering approaches.

In the **learning scenario** considered here, a user is not familiar with a large set of documents and wants to rapidly learn an ontology of topics discussed within this document set. The use of mixed-initiative clustering in this scenario is meant to help the user learn a good ontology through interaction with a machine, especially by presenting the user with rich clustering properties extracted by the machine. We compare our mixed-initiative learning approach with two non-mixed-initiative learning approaches, an unsupervised approach with no interaction between a user and a machine, and a manual approach that has no interaction nor the enriched machine-to-user communication.

In the **teaching scenario**, a user knows the ontology of a given document set and wants to transfer their existing knowledge of this specific ontology to a machine. Our mixed-initiative clustering system allows the user to teach a machine through modifying inappropriate properties of various types. In contrast, we consider one non-mixed-initiative approach that interacts with a machine without enriched user-to-machine communication.

A secondary problem we want to investigate is the entry point of the interactive loop in mixed-initiative clustering. This problem can be boiled down to whether a machine should propose an initial hierarchical clustering. With the machine's initial clustering, the system initializes from the machine's side. Without the machine's initial clustering, a user starts the ontology learning/teaching by splitting a root cluster with all documents into any number of sub-clusters of her choice.

There are advantages and disadvantages to beginning with a machine's initial clustering. In the user learning scenario, an initial clustering provides an initial bias and this bias can be both good and bad. It is good because the machine's proposal usually reflects statistically important properties that are sometimes not trivial for a user to observe directly from documents. It is bad because an initial proposal often limits a user's direction of exploration. In the user teaching scenario, an initial hierarchical clustering may provide a shortcut for a user if it is similar to the ontology the user has in mind. Otherwise, a user may spend more time to correct errors than to teach from scratch. It is not clear a priori which initialization method is better, so we conduct a secondary study to explore this question.

The layout of our user study design can be found in Table 5.2 along with individual learning and teaching tasks assigned to the primary study and the secondary study. Before getting into the details of individual learning and teaching tasks, let's introduce the data

set we use in this study and the idea of reference ontologies for evaluation purposes.

5.1.1 Data Subsets and Reference Ontologies

We choose to use the publicly available 20 newsgroup data set collected by Ken Lang [31] to avoid privacy issues. These documents are collected from 20 different Usenet discussion groups, each focusing on a different topic. More specifically, we used a preprocessed Matlab version prepared by Jason Rennie [46] that contains 18,774 documents. These documents are sorted into a training set and testing set according to posting dates.

These newsgroups are further split into four subsets, each with five newsgroups, such that they have an identical hierarchical graph structure and similar difficulty levels for ontology learning and teaching. The split also attempts to maintain easily separable sibling topics in the top level of the hierarchy and slightly more challenging sibling topics in the second level. Table 5.1 lists the details of four subsets, each consisting of two intermediate topics introduced for conceptual completion in addition to the five newsgroups as the leaf topics. For example, subset A introduces "computer" and "recreation" as the intermediate topics and four newsgroups, two for each, are listed as their child topics. We call these four hierarchies the *reference ontologies* of the document subsets. Reference ontologies are used to evaluate the performance of user-learnt ontologies and as the pre-determined ontologies for the ontology teaching tasks. As for the number of documents in each subset, a training subset contains about 2,800 documents and a testing subset contains about 1,850 documents.

We are fully aware that each reference ontology contains only a small set of topics. However, due to the lack of similar field work, it is still unknown whether users can work with a mixed-initiative clustering system successfully. We therefore decided to begin our studies with simple ontology learning and teaching tasks so the possibility that an ontology itself is too difficult to learn or teach is excluded, and the studies can focus on the effectiveness of different approaches.

5.1.2 User Learning Tasks

The primary study in the user learning scenario examines whether the mixed-initiative clustering approach helps a user learn a topic ontology from an unfamiliar data set better than non-mixed-initiative approaches. The comparative tasks in this study are a learning

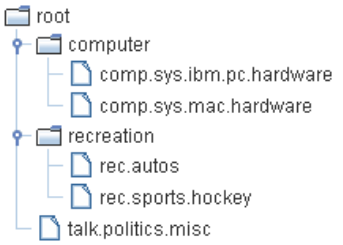
Subset	reference ontology
A	
B	
C	
D	

Table 5.1: We organize the 20 newsgroup data set into four subsets with similar hierarchical structures. The hierarchy between each five newsgroups is the reference ontology of each subset.

task using the manual approach (Task 1) and a learning task using the mixed-initiative approach (Task 2.)

Task 1 refers to the manual approach for user ontology learning. We disable the display of the clustering hierarchy and key features in the interface, e.g., the left half part in Figure 4.2. Testers need to discover the topic ontology by browsing through documents' subjects and reading the content of some documents they select. Performances of this task will give us an idea about the baseline without communication enrichment nor interaction.

In **Task 2**, testers can use the fully functional mixed-initiative clustering system to learn the topic ontology. The system first proposes a hierarchical clustering using unsupervised machine learning techniques so the tester begins their learning by understanding this initial clustering. Before they modify the initial hierarchical clustering, we ask testers to skim through the clustering and give us their judgement of meaningfulness of each cluster. If a cluster is considered meaningful, a tester is asked to give a short description. The initial judgment reflects how a user perceives an autonomous clustering generated by the unsupervised clustering approach plus enriched computer-to-user communication, e.g., keywords and key-subjects extracted for each cluster. After completing this initial judgement, testers continue to work with the mixed-initiative clustering system to refine the initial clustering into a final ontology they like. The final ontology records the user's learning result of the mixed-initiative clustering approach.

The secondary user learning study examines if the mixed-initiative clustering system should propose an initial hierarchical clustering in the learning scenario. It compares Task 2 from the primary study to a new Task 2u. In **Task 2u**, testers still have assistance from the fully functional mixed-initiative clustering system, but the mixed-initiative clustering system does not propose an initial clustering. A tester begins with a root cluster that contains all documents and has greater freedom to interact with the mixed-initiative system to learn an ontology.

In all learning tasks, testers are given one of the four document subsets listed in Table 5.1. We don't provide any prior information such as the number of topics or the structure of the hierarchy to testers.

Evaluation Measurement

In order to evaluate the performance of user learning, we use the idea of precision and recall in information retrieval to compute the similarity between a learnt ontology and its

reference ontology. Topic retrieval scores count the number of relevant topics, instead of relevant documents, that appear in both learnt and reference ontologies. The precision score calculates the ratio of topics in the learnt ontology that also appear in the reference ontology. The recall score calculates the ratio of topics in the reference ontology also recovered by the learnt ontology. Each reference ontology in our experimental setting, which is unknown to testers when they perform learning tasks, contains seven topics including two intermediate topics and five leaf topics. In contrast, the number of topics a user may come up with can vary widely. For instance, the minimum number is 3 and the maximum number is 24 in our records. When a learnt ontology contains many topics, it has a higher chance to achieve high recall but low precision. On the contrary, a learnt ontology with only a few topics may achieve high precision but low recall. The topic F-measure attempts to balance the trade-off between precision and recall.

$$\text{F-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{5.1}$$

It is possible that a tester identifies a topic that is not word for word identical to any reference topic but is semantically equivalent to a topic in the reference ontology. Due to the subjective nature of topic matching, we rely on testers' self-assessment to decide which topics in the reference ontologies were recovered in their user ontology from the learning sessions. We argue that self-assessment is probably the best way to evaluate performance of the user learning tasks because it is hard or equally subjective to rebuke a tester's claim that she or he learns a topic. Although testers may have a tendency to inflate the number of topics found, as long as this tendency is uniform among comparative tasks, the relative comparison is still meaningful. The last column in Table 5.3 shows the topic retrieval scores of some sample ontologies.

5.1.3 User Teaching Tasks

The primary user teaching study examines whether mixed-initiative clustering helps a user teach an ontology to a machine better than non-mixed-initiative approaches. The comparative tasks in this study are a task simulating the teaching style in semi-supervised clustering (Task 3) and a teaching task using the mixed-initiative clustering system (Task 4.) Testers are told which reference ontology they need to teach the system in advance for these tasks.

They are also told the time limit for each task is ten minutes but they can stop whenever they feel the teaching is sufficient.

Task 3 corresponds to the teaching style in semi-supervised clustering. The system simply presents the pre-determined reference ontology but not the document-to-cluster assignments in the beginning. Testers have to select documents and drag-and-drop them to proper topics in the reference ontology, which is exactly like giving the machine a small amount of labeled examples in semi-supervised clustering. The user-to-machine communication in this task is limited to a single iteration of feedback on document-to-cluster properties.

Task 4 is teaching with the mixed-initiative clustering system. To begin with, a reference ontology is shown to testers via a paper hint card. An initial hierarchical clustering is proposed by the machine in this task, so a tester's teaching is for the purpose of correcting errors in the machine-proposed clustering. Testers can use the fully functional mixed-initiative clustering system to adjust the initial clustering into a clustering that is as close to the ideal clustering of the reference ontology as possible.

The secondary user teaching study examines whether the mixed-initiative clustering system should propose an initial hierarchical clustering in the teaching scenario. **Task 4u** is the comparative task against Task 4 in this study. This task is identical to Task 4 except that there is no initial hierarchical clustering proposed by a machine, so a tester can teach the machine from a clean slate.

Table 5.2 summarizes the differences among these tasks and marks tasks used in the primary study and the secondary study.

Task	scenario	approach	initialization choice	primary study	secondary study
1	a user learns an ontology	manual	n/a	✓	
2		mixed-initiative	machine proposes an initial clustering	✓	✓
2u			user starts from scratch		✓
3	a user teaches a known ontology to a machine	semi-supervised	n/a	✓	
4		mixed-initiative	machine proposes an initial clustering	✓	✓
4u			user starts from scratch		✓

Table 5.2: Summary of user study tasks. The primary study examines whether mixed-initiative clustering is better than non-mixed-initiative approaches under two scenarios. The secondary study examines whether a mixed-initiative clustering system should propose an initial hierarchical clustering to its user.

Evaluation Measurement

A direct measurement for user teaching is how fast a user can complete a teaching task. If a user can spend less time to teach a machine using approach A than approach B, approach A is better than B in terms of teaching efficiency. An indirect measurement is the clustering accuracy of a machine's retrained model on held-out testing subsets. This measurement indicates how much a machine can learn from different styles of feedback a user gives through different user teaching approaches.

5.1.4 Participants

We target computer science students with machine learning background so they are familiar with the idea and application of clustering technologies. Participants are assigned to one of two groups.

Primary group is for the primary study focusing on the comparison with and without assistance of mixed-initiative clustering. Testers in this group performed Task 1 and Task 2 under the user learning scenario, and Task 3 and Task 4 under the user learning scenario.

Secondary group is for the secondary study addressing the initialization problem. Testers in this group performed user learning Task 2 and Task 2u, and user teaching Task 4 and Task 4u.

A total of 16 testers were recruited. Eight of them were native English speakers and the other eight were non-native English speakers. We tried our best to equally balance testers' language proficiency and subset-to-task assignments.

5.1.5 Experimental Procedure

A user study session consisted of a short demonstration, four tasks (two comparative tasks for each scenario), and one questionnaire. In the beginning, the principal investigator gave a short demonstration to help testers become familiar with our mixed-initiative clustering system. The demonstration included (1) how the interface represents hierarchical clustering information and (2) what types of user feedback testers can give to the system. The demonstration used a held-out email data set so testers would not gain prior knowledge of the text corpus used in the actual tasks.

After the demonstration, each tester performed two comparative learning tasks depending on which study they were assigned to. In order to control ordering effects, we counterbalanced the order of the two learning tasks. At the end of learning tasks, testers were asked to draw the user ontology and answer task-related questions in the questionnaire. Once two tasks were completed, the principal investigator would show them reference ontologies of both tasks and ask them to assess their performances in terms of topic retrieval. After completing two learning tasks, each tester performed two comparative teaching tasks, again, according to their assignment to the primary or secondary user study. The order of teaching tasks was also alternated to avoid testers performing better in the second attempt. However, a tester always finished two learning tasks first and then worked on two teaching tasks second to prevent them from acquiring any knowledge about the reference ontologies prior to the learning tasks. At the end of teaching tasks, they were asked to answer task-related questions in the questionnaire. A tester had 10 minutes each to complete the four tasks. In order to control for difficulty across tasks, we used a Latin square design in which we alternated the association between tasks and data subsets.

5.2 Results

5.2.1 User Learning Results in the Primary Group

Using the manual approach, testers wrote down topics they found during the task on a paper and summarized the learnt ontology at the end of the task. The learnt ontologies using the manual approach were coherent conceptually. However, due to the lack of means to summarize the large number of documents, testers tended to find topics based on a few documents they happened to observe. The second ontology in table 5.3 is an example of a manually learnt ontology.

Two ontologies were recorded in the mixed-initiative learning task. An initial ontology was recorded according to how the tester perceived the initial clustering generated by the unsupervised clustering approach. If some clusters proposed by the machine were not meaningful to the tester, they could cross these clusters out (shown as "[X]" in Table 5.3.) At the end of the learning task, the final ontology, learnt mixed-initiatively by the tester, was also recorded. In brief, the initial and final ontology are referred to as the "unsupervised ontology" and "mixed-initiative ontology". An unsupervised ontology and

a mixed-initiative ontology learnt by a tester are also shown in Table 5.3. In this example, the unsupervised clustering approach helped the tester pick up ontology fragments, but the machine-proposed hierarchy was noisy and inconsistent to the tester. For instance, "car" and "comp.hardware" were grouped together under the same branch (they shared the keyword "drive".) On the other hand, the mixed-initiative ontology was more conceptually coherent than the initial fragmented unsupervised ontology.

On average, 13.0 topics were learnt by testers in the primary group using the manual approach, and among these learnt topics, 4.0 topics were shared with reference ontologies of 7 topics. 10.5 topics were proposed by the machine's unsupervised hierarchical clusterings, and testers further refined them into mixed-initiative ontologies with an average of 8.1 topics. Unsupervised ontologies shared 4.4 topics, and mixed-initiative ontologies shared 4.8 topics with reference ontologies. Table 5.2.1 shows the averages and standard deviations of topic retrieval scores of ontologies learnt using different approaches by the primary group testers. With assistance from our mixed-initiative clustering system, testers obtained higher topic recall, which means recognizing more topics in reference ontologies, and higher precision, which means obtaining more concise ontologies, than other approaches. [1]

One thing we want to point out is that the reference ontologies are not the only reasonable ontologies. For example, the reference ontology of subset B contains a intermediate "computer" topic and two newsgroup topics corresponding to "IBM hardware" and "Mac hardware." Three testers (one using the manual approach and two using the mixed-initiative approach) learnt sub-topics of individual hardware components such as "cpu", "RAM" and "monitor." Based on our measurements, these user-learnt topics are not counted as retrieved topics because they are not in the reference ontology. However, testers did express their satisfaction in this alternative learning result and one of these three testers even argued his learnt topics should be counted right.

When we asked testers to choose their preferred learning approach, seven out of eight testers chose the mixed-initiative approach. The remaining one gave equal preference to the manual approach and the mixed-initiative approach. None chose the unsupervised approach.

[1] In the t-test, the precision and f-measure scores of the mixed-initiative approach both reach 5% level of significance over the manual approach and the semi-supervised approach, while its recall score reaches 10% level of significance over the manual approach.

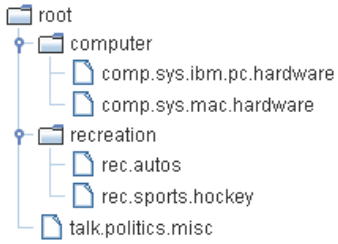
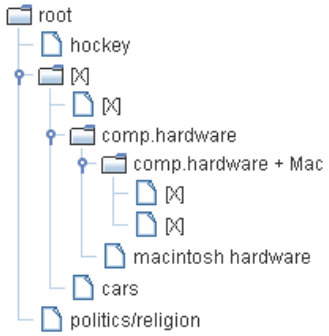
Approach	ontology	score
reference ontology		
manual		R=4/7 P=4/13 F=0.40
un-supervised		R=5/7 P=5/10 F=0.59
mixed-initiative		R=6/7 P=6/6 F=0.92

Table 5.3: This table presents a reference ontology and several sample ontologies learnt by testers using different approaches. Topic retrieval scores of precision (P), recall (R), and F-measure (F) are shown in the last column for each learnt ontology.

User learning	manual	unsupervised	mixed-initiative
precision	0.42 ± 0.27	0.43 ± 0.09	$\mathbf{0.67} \pm 0.28$
recall	0.57 ± 0.15	0.63 ± 0.11	$\mathbf{0.68} \pm 0.13$
macro-average F-measure	0.43 ± 0.13	0.50 ± 0.10	$\mathbf{0.66} \pm 0.19$
micro-average F-measure	0.48	0.51	$\mathbf{0.67}$

Table 5.4: The topic retrieval scores of different user learning approaches in the primary group. The macro-average F-measure is obtained from averaging the F-measure scores of individual user-learnt ontologies, and the micro-average F-measure is calculated using the averaged precision and recall scores. The result shows that with the assistant of our mixed-initiative clustering system, a user can obtain both higher topic recall and precision scores than non-mixed-initiative approaches.

5.2.2 User Teaching Results in the Primary Group

Figure 5.1 shows the user teaching performances of eight primary group testers in terms of user teaching efficiency and machine learning performances. The X-axis indicates the amount of time that a tester spent on teaching a machine, and the Y-axis presents the clustering accuracy of a machine's model after learning from a tester's instruction. Although testers were told to finish the task in ten minutes and the investigator reminded them to wrap up at the ninth minute, we didn't shut down the system when the time was up so some testers actually took longer than ten minutes to complete their teaching. With the assistance of the mixed-initiative system, a tester on average completed the teaching task in 7.2 minutes. Five out of eight testers finished earlier than the given ten-minute limit because they felt the machine had learned the given reference ontology. This early completion didn't happen when they were asked to teach a machine by giving labeled examples. Testers spent up to 9.9 minutes on average to complete the teaching task using the semi-supervised approach. Many testers described that they didn't feel a sense of completion so they just kept teaching until the time limit was used up, and one tester abandoned the task early on because he disliked the semi-supervised style of teaching. Two testers spontaneously told the principal investigator that having machine learning knowledge was important to the semi-supervised teaching approach, which didn't happen when testers taught the machine using the mixed-initiative approach. The model retraining performance of the mixed-initiative system is marginally better, 83.63%, than the teaching-by-example approach, 82.75%. However, the standard deviation of model performance is larger when retraining with mixed-initiative feedback (5.9% vs. 5.7%). Some early-stopped mixed-initiative teaching obtains poor retraining performance because the testers thought their

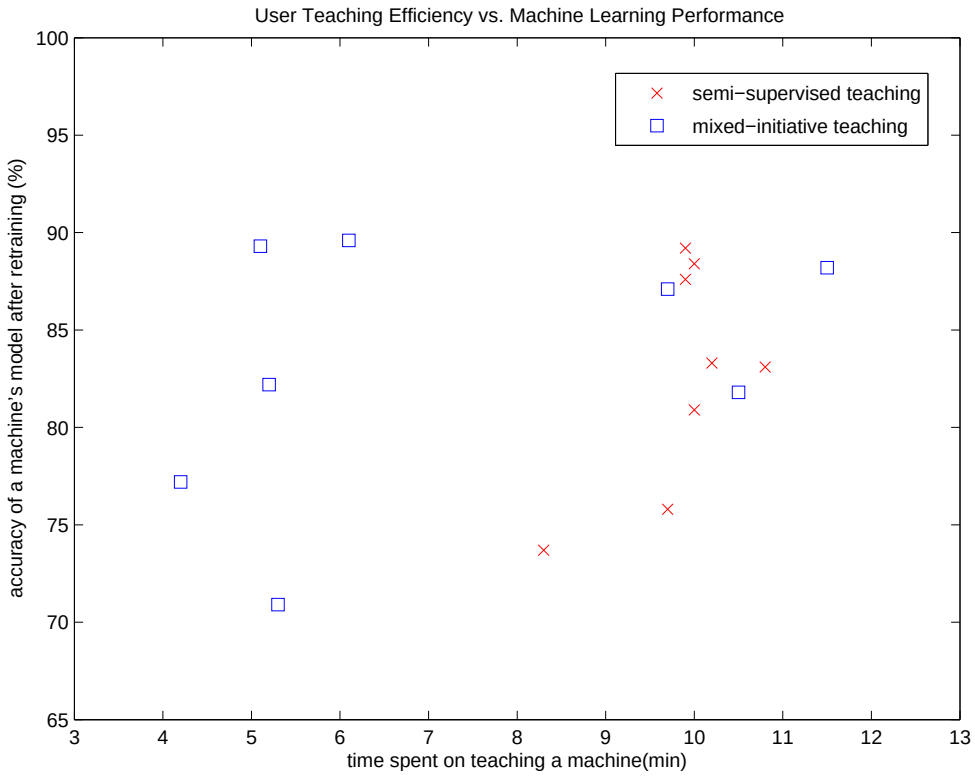


Figure 5.1: User teaching performance of the primary group.

teaching was enough but in fact it was not. If the main purpose of a future mixed-initiative clustering system is to facilitate user teaching, we suggest to include a probing mechanism, such as allowing a user to reserve some documents and ask a machine to label them after one iteration of user teaching, so a user can gauge whether she successfully transfers the knowledge. According to the result in Figure 5.1, the mixed-initiative style of teaching facilitates efficient teaching, but only marginally improves the clustering model retraining at the machine's side. [2]

When asked to choose which teaching approach they would prefer, eight testers out of eight favored our mixed-initiative clustering approach.

[2]In the t-test, the teaching efficiency of using mixed-initiative clustering reaches 5% level of significance but the machine retraining performance doesn't.

5.2.3 Feedback Composition

Analysis of feedback composition reveals what feedback types are actually used by users under different scenarios. With the assistance of the mixed-initiative clustering system, three feedback types were available to testers, and testers were free to choose whatever feedback they wanted to give at any time during the tasks. The semi-supervised style of teaching, on the other hand, limited testers to document-to-cluster feedback. We recorded a feedback log file for each task except the manual learning task. The break-down analysis for the primary group is shown in Table 5.5.

Task	cluster-to-cluster feedback	feature-to-cluster feedback	document-to-cluster feedback
2: mixed-initiative learning	19.1	5.4	1.3
3: semi-supervised teaching	n/a	n/a	51.3
4: mixed-initiative teaching	15.1	20.4	0.8

Table 5.5: Average numbers of different feedback types given by the primary group of testers. It shows testers preferred giving feedback on conceptual properties.

The comparison between the semi-supervised style of teaching (Task 3) and the mixed-initiative style of teaching (Task 4) strongly indicates that testers didn't like to give document-to-cluster feedback, e.g., teaching by example, if they could teach in a more conceptual level such like feedback on feature-to-cluster or cluster-to-cluster properties. In fact, document-to-cluster feedback was almost ignored (only 0.8 times per mixed-initiative teaching task) when there were other choices. The enrichment of communication languages in our mixed-initiative clustering provided exactly these desirable choices for users.

The comparison between Task 2 and Task 4 highlights the difference between user learning and user teaching. In the learning task, testers used more cluster-to-cluster feedback, e.g., moving clusters around or merging/splitting clusters, to explore the space of possible user ontologies. When testers intended to teach a machine an ontology, they chose to give more feature-to-cluster feedback, e.g., assigning keywords or key-subjects to clusters.

5.2.4 Secondary Group Initialization Choice

The secondary group of testers performed the comparative tasks 2, 2u, 4 and 4u because we want to answer the question "should a machine propose an initial hierarchial clustering?"

In Task 2 and 4, testers were presented with an initial hierarchical clustering proposed by the machine, which we refer to as "machine initialization," while Task 2u and 4u started from the user's side with no initial hierarchical clustering, which we refer to as "user initialization."

Table 5.6 shows the topic retrieval scores and user preference of the initialization choice in the learning scenario. With an initial hierarchical clustering proposed by a machine, testers on average found 9.0 topics, in which 5.25 topics could be found in reference ontologies of 7 topics. Without a machine's initial proposal (user initialization), testers found fewer topics, 7.5, and only 4.37 topics shared in reference ontologies. Machine initialization achieved lower precision but better recall and overall F-measure than user initialization. When asked about their preference, five testers in the secondary group favored an initial hierarchical clustering proposal while three favored no such proposal.

User learning	machine initialization	user initialization
precision	0.62 ± 0.18	$\mathbf{0.68} \pm 0.27$
recall	$\mathbf{0.75} \pm 0.13$	0.63 ± 0.15
F-measure	$\mathbf{0.67} \pm 0.14$	0.62 ± 0.15
user preference	$\mathbf{5}/8$	$3/8$

Table 5.6: The machine initialization has better overall F-measure and is favored by more testers than the user initialization.

User teaching	machine initialization	user initialization
time (min)	7.66 ± 3.77	$\mathbf{5.89} \pm 2.87$
accuracy (%)	78.07 ± 7.89	$\mathbf{83.70} \pm 5.66$
user preference	$2.5/8$	$\mathbf{5.5}/8$

Table 5.7: User initialization achieves better user teaching performance than machine initialization and is favored by testers.

Table 5.7 shows the result of the initialization choice for user teaching. When testers started teaching from scratch, the retrained model achieved 83.70% average accuracy in predicting held-out testing sets. The performance degraded to 78.07% when testers had to correct errors in the machine's initial hierarchical clusterings. The average time spent on teaching the machine when testers started without an initial clustering was also shorter, 5.89 minutes compared to 7.66 minutes with initial clusterings. Unlike the user learning study where more testers favored starting with a machine's initial hierarchical clustering,

testers favored no such initial clustering in the user teaching study. This result gives a intuitive guideline about the initialization of mixed-initiative clustering: if a user already has enough knowledge of the data, it is good to give the user more control; if a user has limited prior knowledge about the task, it is good for a machine to propose an initial clustering.

Language proficiency		native speakers	non-native spks
user learning	machine init.	2/4	3/4
	user init.	2/4	1/4
user teaching	machine init.	0/4	2.5/4
	user init.	4/4	1.5/4

Table 5.8: The impact of language proficiency on the user preference of the initialization choice. The nominator number indicates how many native or non-native speakers prefer this initialization choice, and the denominator indicates there are four native and four non-native speakers in this study.

Another interesting finding is about the impact of language proficiency. There are four native speakers and four non-native speakers in the secondary study. From Table 5.8, it seems that native speakers are more likely to prefer no initial clustering, and non-native speakers are more likely to prefer an initial clustering proposed by a machine. However, more testers are needed to statistically justify this finding.

5.2.5 Suggestions from Testers

Testers were asked to list other feedback types they would like the system to have, and existing feedback types they think the system should get rid of in the questionnaire. Before discussing individual suggestions, we want to remind readers that our testers are computer science students with machine learning background. They are more aware of related information technologies than average users are. As a consequence, some of their suggestions are on the complicated end of the spectrum.

The most popular suggestion is the integration of a "keyword search" function. Four testers suggested it based on two different purposes. Some testers wanted to use the keyword search function as a judgement tool for adding a new topic into ontologies. The idea behind is that if a topic does exist in a data set, there should be enough documents containing the most prominent keywords of the topic as the supportive evidence. Some

other testers wanted to teach a machine a topic by giving the machine example documents that contain one or more topic-related keywords. Two other suggestions also exploit the feature-to-cluster properties in different ways: (1) two testers suggested the system should allow users to directly add keywords to a cluster, and (2) one tester wanted the interface to include information of how many documents affiliated with a feature.

Enhancing transparency of a machine's clustering model is the second most popular suggestion. Three testers suggested to present a machine's confidence score for each cluster so they can prioritize their learning efforts. One tester suggested to measure and present a homogeneous score for each cluster. The homogeneous score can assist a user in judging whether a cluster is pure enough or need further splitting.

Two testers suggested automatic retraining instead of our current user-controlled initiative exchange mechanism. However, one of the two testers who made this suggestion also revealed his worries about possible constant interruptions if the system can retrain its clustering model automatically. Two testers suggested retraining-by-example feedback, which we implemented but did not employ in these user studies because the retraining-by-example feedback is hard to perform in practice.

Two testers concerned about the manual decision of the number of clusters in cluster splitting. One suggested that a machine should decide automatically. Another tester suggested to show multiple clustering results, each with a different number of clusters, and let a user pick which one she likes.

Some suggestions are about highlighting specific information. One testers wanted the system to highlight keywords in the content of documents. One wanted to see the differences between two clustering revisions highlighted. One suggested to display tag clouds for document subjects in each cluster.

With regards to necessity of existing feedback types, two testers thought feedback types of confirming and removing a document are not necessary because it is more convenient to move documents around. One testers thought that confirming or removing feature-to-cluster and document-to-cluster properties are too complicated and time consuming. One tester stated that the initial clustering proposal barred his exploration.

Other suggestions include better automatic topic naming than the top two keywords, allowing multi-cluster merging in addition to current pair-wise merging, multi-label clustering capability, and sticky keywords so they can be manipulated later. Many testers also commented our system is easy to use.

5.3 Summary

In this chapter, we first discuss how to design user studies so we can evaluate the feasibility of a mixed-initiative clustering system, and then prove the advantages of our low-latency mixed-initiative clustering system by results from the user studies.

We design the user studies to test two scenarios, a learning scenario where a user tries to learn a topic ontology from an unfamiliar data set, and a teaching scenario where a user knows the ontology in advance and wants to transfer this knowledge to a machine.

The results of the primary user study show that our mixed-initiative clustering system helps users in both scenarios. Users can learn more relevant and concise ontologies using the mixed-initiative clustering system than a manual approach or an unsupervised clustering approach. Users can also teach more efficiently when using our mixed-initiative system than when they use the teaching-by-example approach in semi-supervised clustering. However, despite this reduced demand on the user's time, the iterative mixed-initiative process converges to a machine's clustering model with similar accuracy to the teaching-by-example approach. From users' point of view, mixed-initiative learning and teaching are significantly favored over non-mixed-initiative approaches. The analysis of feedback composition clearly indicates that users prefer giving conceptual-level feedback such as adjusting the clustering hierarchy and confirming keywords of a cluster over detailed feedback such as labeling documents. Users want enriched communication in the interactive process.

As the secondary study shows, it is best to choose the initialization method according to how much prior knowledge a user knows. If a user has limited prior knowledge about the mixed-initiative clustering task, providing an initial clustering may better assist the user. Otherwise, it is better to give the user control over the initial clustering. Language proficiency may be another factor that affects the initialization choice.

Testers also gave our system many useful suggestions. The most prominent ones are the integration of a keyword search function, which is handy for users in both learning and teaching scenarios, and enhancing transparency of a machine's clustering model such as presenting a confidence score or a homogeneous score for each cluster.

The success of applying the mixed-initiative clustering approach under both the teaching and learning scenario validates our main thesis that mixed-initiative clustering can help a user achieve better clustering results than non-mixed-initiative approaches due to

the enriched communication and the interactive learning and teaching between a user and a machine. These studies also raise more rigorous research challenges. To name a few, can mixed-initiative approaches help a user in learning or teaching more difficult ontologies? Can users without machine learning background use mixed-initiative learning systems? How to theoretically consider a user's cognitive load into the design decisions of mixed-initiative learning systems? We hope the design of user studies and findings in our experimental results provide a good starting point for pursuing these challenging research directions.

Chapter 6

Conclusion

Mixed-initiative learning has become an emerging research area in machine learning communities. By our definition, a mixed-initiative learning task should have the following characteristics. First, a user and a machine collaborate in a mixed-initiative task by making interleaved contributions to the task. Second, the user and the machine can both update their model assumptions by learning from the other's contributions. We consider our study on mixed-initiative clustering as a case study of mixed-initiative leaning. In particular, we study mixed-initiative "text" clustering where a user and a machine work collaboratively to identify a topic ontology within a large set of documents.

The motivation of this research comes from problems we identified while applying unsupervised and semi-supervised clustering techniques in a previous study. Traditionally, text clustering is studied using unsupervised techniques where a machine builds the model alone, but an unsupervised clustering result is usually different from a user's ideal clustering result. We call this difference "clustering mismatch". Semi-supervised clustering is proposed to solve the clustering mismatch problem in which an oracle user can provide a small amount of labeled examples. However, to be qualified as an oracle user, a user needs to be a domain expert, which means she doesn't need to learn, and a knowledge engineer, which means she already knows how to teach. In practice, it is not realistic to assume that a user is an oracle user. We raise the "user teaching problem" to challenge the assumption that an oracle user is a knowledge engineer. To solve the user teaching problem, one should seek intuitive user feedback types so a user can communicate her conceptual ideas to a machine rather than requiring a user to give instance-based information, which is typically requested by a semi-supervised clustering algorithm. On the other hand, the

"user learning problem" challenges the assumption that an oracle user is a domain expert. To solve the user learning problem, a machine has to provide enough information to assist a user when the user is still in the process of developing her ontology for a specific clustering task.

Our study on mixed-initiative clustering solves the user teaching problem and the user learning problem by considering a non-oracle user as a collaborator with a machine. The first essential component required for building a mixed-initiative clustering system is the capability that a machine and a user can learn from each other and teach each other interactively. Next, it is important to investigate what types of information are effective and efficient for teaching and learning in this interactive loop. Including those effective and efficient types of information for communication between a non-oracle user and a machine, which we refer to as communication enrichment, is another essential component of mixed-initiative clustering.

Given these two essential components, the main thesis of this study is *mixed-initiative clustering can help a user achieve better clustering results than non-mixed-initiative approaches due to the* ***enriched communication*** *and the* ***interactive learning and teaching*** *between a user and a machine.* In order to examine our thesis, we have studied communication enrichment and interactive learning and teaching for mixed-initiative clustering, built a system with all new knowledge we have learnt by studying the two essential components, and then completed user studies to evaluate whether non-oracle users can eventually achieve better clustering results using this system. We summarize our contributions in this study as follows.

6.1 Contributions

The first contribution of this study is providing a framework for mixed-initiative clustering. This framework consists of a machine learning phase, a machine teaching phase, a user learning phase, and a user teaching phase. These phases are connected in an interactive loop that allows bi-directional communication between a user and a machine. The bi-directional communication languages define types of information exchanged in an interface. The **computer-to-user communication language**, $L_{c \to u}$ defines types of properties a machine extracts and presents in the interface, and the **user-to-computer language**, $L_{u \to c}$, defines types of user feedback supported by the interface. Each user feedback type

corresponds to a specific way to confirm or correct machine-proposed properties. Given this framework, we can formally define communication enrichment as adding new types of properties and allowing new types of user feedback in an interface. In particular, we consider communication enrichment by introducing **conceptual properties** such as key features of clusters and hierarchical relationships between clusters in order to help a non-oracle user learn and teach. By including multiple property and feedback types in communication languages, communication robustness is built up as well. We also identify the necessity of **coordination** between the communication languages and the machine's clustering model from this framework.

The second contribution comes from successfully building several systems using our proposed framework. Two systems are built with incrementally enriched communication languages. We demonstrate in Chapter 3 how to enrich communication languages in mixed-initiative clustering by adding two conceptual property types to $L_{c \to u}$, and various feedback types on conceptual properties to $L_{u \to c}$. The first conceptual communication introduced is feedback on key features of each cluster. For example, if a machine extracts "espn" as a keyword for a "sport" cluster, a user can confirm this feature-to-cluster property. The second conceptual communication enrichment is accepting user feedback on hierarchical clustering. For example, if a machine misplaces a "baseball" cluster underneath a "finance" cluster, a user can correct this inappropriate parent-child relationship by moving the "baseball" cluster to a suitable place in the hierarchy. With conceptual properties, a user can understand the clustering results more easily. With various types of user feedback available, a user can teach a machine more intuitively. In order to achieve the coordination for mixed-initiative clustering, we propose a new clustering model, the SpeClustering model, which provides a natural way to adjust its parameters according to user feedback on key features, and a mixed-initiative SpeClustering algorithm to adapt multiple types of user feedback. For coordination in hierarchical mixed-initiative clustering, a cascading version of the mixed-initiative SpeClustering algorithm trains a set of SpeClustering models as classifiers for the root node and intermediate nodes. The successful building of mixed-initiative clustering systems validates our framework and also demonstrates the possibility to develop machine learning algorithms to work with conceptual properties.

The third contribution comes from the study of the other essential component, low-latency interaction, in mixed-initiative clustering. **Low-latency interaction** refers to a mixed-initiative learning environment in which a user can retrain a machine's model at

any time and a machine can complete its initiative of learning and teaching in a few seconds. We examine several practical issues occurred only in the low-latency interactive environment. These issues include the timing of initiative exchange, how to retrain a machine's clustering model with insufficient and unbalanced user feedback, how to resolve feedback contradictions, how to build a user's feedback proficiency, and how to design a mixed-initiative interface. Below is our summarization of guidelines for general mixed-initiative learning problems:

1. Model retraining should be as fast as possible. If a user only uses a subset of the feedback language, a machine can use a cheaper machine learning algorithm. Our rule of thumb is to control the time spent on each model retraining under 10 seconds.
2. Because a user tends to be lazy and may not have enough background knowledge of how machine learning algorithms work, it is an important capability for a machine to learn from insufficient and unbalanced user feedback.
3. The goal of a mixed-initiative clustering system is to help a user develop her still evolving user ontology. Under this goal, consecutive machine clustering revisions shouldn't make dramatic change in order to preserve a user's existing clustering ontology, but should make specific change by extrapolating user feedback.
4. Feedback items given by a user may contradict one another especially when the user is allowed to give various types of feedback in mixed-initiative clustering. A mixed-initiative system developer should check potential feedback contradictions every time a new feedback type is added. A simple mechanism to resolve feedback contradictions is to let later feedback overwrite the previous contradicting one.
5. A checking mechanism is necessary to filter out undesirable user feedback and educate a user's feedback proficiency.
6. Don't generate numeric indices as cluster labels. Generate text labels even the automatic text labeling algorithm is far from perfect.
7. The interface design should arrange property panels to reduce a user's cognitive effort, and provide intuitive feedback interaction.

Using the knowledge learnt from the second and third contributions, we build a final mixed-initiative clustering system that integrates communication enrichment and low-latency interaction capabilities. User studies are conducted to examine the effectiveness of this full-fledged system.

The fourth and last contribution comes from **the design of user studies** and user study results. Evaluating a mixed-initiative clustering system is not easy. We solve the evaluation problem by designing comparative tasks under separate user learning and user teaching scenarios. This separation covers most of use cases for mixed-initiative clustering, and allows comparison with non-mixed-initiative approaches. In the **user learning scenario**, a user develops a topic ontology from an unfamiliar data set, and in the **user teaching scenario**, a user knows the ontology and wants to transfer this knowledge to a machine. **Results of user studies confirm our main thesis** that mixed-initiative clustering can help a user achieve better clustering results than non-mixed-initiative approaches. The analysis of feedback composition clearly indicates that users prefer conceptual properties and like to give conceptual-level feedback. From users' point of view, mixed-initiative learning and teaching are significantly favored over non-mixed-initiative approaches. We also learn that a mixed-initiative clustering system should decide its initialization choice based on how much prior knowledge its user possesses.

6.2 Future Work

There are many future research opportunities in mixed-initiative clustering and the more general research area of mixed-initiative learning.

To expand the study on mixed-initiative clustering, we can apply techniques developed in this dissertation to intelligent information retrieval. Some search engines have successfully applied autonomous clustering techniques to groups similar retrieved results together into clusters [54]. Given an autonomous clustering on retrieved results, mixed-initiative clustering techniques can further assist users' following exploratory searching activities.

Another future research topic we are particularly interested in is how to develop a mixed-initiative collaborative filtering/recommendation system. We believe our framework of mixed-initiative clustering can be applied to this new mixed-initiative learning task. Let's use the popular online DVD subscription service, Netflix, as an example. If we want to build a mixed-initiative recommendation system to replace Netflix's current recommendation system, we can analyze this new mixed-initiative learning task in terms of its communication languages and collaborative filtering algorithm. As mentioned in Section 2.3, the coordinated computer-to-user language, user-to-computer language and machine's clustering model can be considered as the signature of a mixed-initiative learn-

ing system. The Netflix's recommendation pages already present its subscribers various types of information about movies it recommends, such as a short summary, cast, directors, genres, comments from other subscribers, reasons of recommendation, and prediction on user ranking, which define a well-enriched computer-to-user language, $L_{c \to u}$. The other direction of communication enrichment is slightly enriched as well. Its user-to-computer language, $L_{u \to s}$, allows subscribers to edit their genre preference (confirming and disapproving genre properties) in addition to rank individual movies (correcting the machine's prediction on ranking properties.) The trend of emphasizing genres in its recommendation explanations and allowing user edition of genre preference fits our finding that conceptual properties are good to have in any human-computer communication languages. However, this user-to-computer language still misses feedback types on many property types that Netflix web pages also provide. One way to further enrich the user-to-computer language is to assume each link a subscriber clicks as implicit confirmation feedback. For example, a subscriber follows a link of an actress indicates her interest in this actress. She may also be interested in other movies this actress played. The next challenge is how to develop a suitable collaborative filtering algorithm, M_c, to learn from newly introduced user feedback types, especially the implicit feedback types. We can imagine building this mixed-initiative recommendation system by fixing $L_{c \to u}$ and bootstrapping $L_{u \to c}$ and M_c, i.e., introducing one feedback type at a time to the user-to-computer communication language and coordinating the collaborative filtering algorithm to learn from this new feedback type.

Other research opportunities include but not limited to understanding user behaviors and expectation in a mixed-initiative learning task, design guidelines based on users' cognitive load, and systematic analysis on machine learning algorithms and their coordinated property types and feedback types.

Given our experiences in mixed-initiative clustering, we believe that approaching mixed-initiative learning as a bi-directional learning and teaching task between a user and a machine helps users more substantially than studying it as an interactive user-teaching and machine-learning task.

Bibliography

[1] Jaime Arguello and Carolyn Rose. InfoMagnets: making sense of corpus data. In *NAACL '06: Proceedings for the North American Chapter of the Association for Computational Linguistics*, pages 253–256, 2006. 2.4

[2] Ricardo Baeza-yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*, chapter 5.2.1 and 5.3.2. Addison Wesley, 1999. 4.3.2

[3] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, 2004. 2.4, 3.1.3

[4] Philip Bille. Tree edit distance, alignment distance and inclusion. Technical report, IT University, 2003. 3.2.3

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *Journal of Machine Learning Research*, 2003. 3.1.2

[6] David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems 16*, 2004. 3.2.2

[7] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, 1998. 1.2.1

[8] Adam Cheyer, Jack Park, and Rich Giuli. IRIS: Integrate. Relate. Infer. Share. In *1st Workshop on the Semantic Desktop*. International Semantic Web Conference (ISWC '05), 2005. 1.1

[9] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scat-

ter/gather: a cluster-based approach to browsing large document collections. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1992. 2.4

[10] Sanjoy Dasgupta and John Langford. Active learning tutorial. ICML '09 tutorial, `http://hunch.net/~active_learning/`, 2009. 2.1

[11] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. In *Journal of the Royal Statistical Society*, volume 39, pages 1–38, 1977. 3.1.2

[12] Marie desJardins, Eric Eaton, and Kiri L. Wagstaff. Learning user preferences for sets of objects. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*. ACM, 2006. 2.4

[13] Byron E. Dom. An information-theoretic external cluster-validity measure. Technical report, Research Report RJ 10219, IBM, 2001. 3.1.3

[14] Pinar Donmez and Jaime G. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*. ACM, 2008. 2.1

[15] Mark Dredze, Tessa Lau, and Nicholas Kushmerick. Automatically classifying emails into activities. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*. ACM, 2006. 3

[16] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR '08: Proceedings of ACM Special Interest Group on Information Retreival*. ACM, 2008. 2.4, 3.1.2

[17] Jerry Alan Fails and Dan R. Olsen, Jr. Interactive machine learning. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003. 2.4

[18] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. Cueflik: interactive concept learning in image search. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. ACM, 2008. 2.4

[19] Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. Document classification through interactive supervision of document and term labels. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice*

of Knowledge Discovery in Databases. Springer-Verlag New York, Inc., 2004. 2.4, 3.1.2

[20] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1996. 2.4

[21] Eric Horvitz. Principles of mixed-initiative user interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1999. 2.4, 4.2

[22] Yifen Huang and Tom Mitchell. Toward mixed-initiative email clustering. In *Agents that Learn from Human Teachers, AAAI 2009 Spring Symposium*, 2009. 2.4

[23] Yifen Huang and Tom M. Mitchell. Text clustering with extended user feedback. In *SIGIR '06: Proceedings of ACM Special Interest Group on Information Retreival*. ACM, 2006. 2.4, 3.1.2, 3.2.2

[24] Yifen Huang, Dinesh Govindaraju, Tom Mitchell, Vitor R. Carvalho, and William Cohen. Inferring ongoing activities of workstation users by clustering email. In *First Conference on Email and Spam*, 2004. 3.1.3

[25] SRI International. CALO: Cognitive assistant that learns and organizes. `http://caloproject.sri.com/`, 2008. 1.1

[26] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of 16th International Conference on Machine Learning*, 1999. 1.2.1

[27] Rosie Jones, Andrew Mccallum, Kamal Nigam, and Ellen Riloff. Bootstrapping for text learning tasks. In *IJCAI '99 Workshop on Text Mining: Foundations, Techniques and Applications*, 1999. 3.1.2

[28] Andruid Kerne, Eunyee Koh, Vikram Sundaram, and Michael J. Mistrot. Generative semantic clustering in spatial hypertext. In *DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering*, 2005. 2.4

[29] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, 1997. 3.2.2

[30] Nicholas Kushmerick, Tessa Lau, Mark Dredze, and Rinat Khoussainov. Activity-centric email: a machine learning approach. In *AAAI'06: proceedings of the 21st national conference on Artificial intelligence*. AAAI Press, 2006. 3

[31] Ken Lang. CMU text learning group data archives: the 20 newsgroups data set. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html. 5.1.1

[32] Bing Liu, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. Text classification by labeling words. In *AAAI'04: Proceedings of the 19th national conference on Artifical intelligence*. AAAI Press / The MIT Press, 2004. 2.4, 3.1.2

[33] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7), 1994. 2.4

[34] Dorin Marcu, Mihai Boicu, and Gheorghe Tecuci. User-agent interactions in mixed-initiative learning. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, 2001. 2.4

[35] Andrew McCallum, Gideon Mann, and Gregory Druck. Generalized expectation criteria. Technical Report Technical Report 2007-62, University of Massachusetts, Amherst, 2007. 3.1.2

[36] Andrew K. Mccallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998. 3.2.2

[37] Tom M. Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7), 1994. 2.4

[38] Tom M. Mitchell, Sophie H. Wang, Yifen Huang, and Adam Cheyer. Extracting knowledge about users' activities from raw workstation contents. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, 2006. 1.1

[39] Kamal Nigam, Andrew K. Mccallum, Sebastian Thrun, and Tom M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, 1998. 1.2.1, 3.1.2, 3.1.3

[40] B. Peintner, P. Viappiani, and N. Yorke-Smith. Preferences in interactive systems: technical challenges and case studies. *AI Magazine*, 29(4), 2008. 2.4

[41] Adam Perer and Ben Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. ACM, 2008. 2.4

[42] Adam Perer and Ben Shneiderman. Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 2008. 2.4

[43] Peter Pirolli, Patricia Schank, Marti Hearst, and Christine Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1996. 2.4

[44] Pearl Pu and Li Chen. User-involved preference elicitation for product search and recommender systems. *AI Magazine*, 29(4), 2008. 2.4

[45] Hema Raghavan, Omid Madani, and Rosie Jones. InterActive feature selection. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, 2005. 2.4, 3.1.2, 3.1.3

[46] Jason Rennie. 20 newsgroups. http://people.csail.mit.edu/jrennie/20Newsgroups/. 5.1.1

[47] J. J. Rocchio. Relevance feedback in information retrieval. *SMART Retrieval System Experimens in Automatic Document Processing*, 1971. 4.3.2

[48] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. 2.1

[49] Simone Stumpf, Erin Sullivan, Erin Fitzhenry, Ian Oberst, Weng-Keen Wong, and Margaret Burnett. Integrating rich user feedback into intelligent user interfaces. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 2008. 2.4

[50] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting meaningfully with machine learning systems: Three experiments. In *Int. J. Human-Computer*

Studies, volume 67, pages 623–669, 2009. 2.4

[51] Gheorghe Tecuci, Mihai Boicu, Cindy Ayers, and David Cammons. Personal cognitive assistants for military intelligence analysis: Mixed-initiative learning, tutoring, and problem solving. In *the First International Conference on Intelligence Analysis*, 2005. 2.4

[52] Andrea L. Thomaz and Cynthia Breazeal. Transparency and socially guided machine learning. In *the 5th International Conference on Developmental Learning*, 2006. 2.4

[53] Andrea L. Thomaz, Guy Hoffman, and Cynthia Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *the 15th IEEE International Symposium on Robot and Human Interactive Communication*, 2006. 2.4

[54] Vivisimo. http://www.vivisimo.com. 6.2

[55] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, 2001. 2.4

[56] Hui Yang and Jamie Callan. Human-guided ontology learning. In *Second Workshop on Human-Computer Interaction and Information Retrieval*, 2008. 2.4

[57] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, 1997. 3.1.3