A COMPARISON OF CLUSTERING METHODS FOR DEVELOPING MODELS OF USER INTEREST

A Thesis

by

PRASANTH GANTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Computer Science

A COMPARISON OF CLUSTERING METHODS FOR DEVELOPING MODELS OF USER INTEREST

A Thesis

by

PRASANTH GANTA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Frank M. Shipman
Committee Members,	Richard Furuta
	Takashi Yamauchi
Head of Department,	Valerie Taylor

May 2011

Major Subject: Computer Science

ABSTRACT

A Comparison of Clustering Methods for Developing Models of User Interest. (May 2011)

Prasanth Ganta, B.Tech., Indian Institute of Technology, Guwahati

Chair of Advisory Committee: Prof. Frank Shipman

For open-ended information tasks, users must sift through many potentially relevant documents assessing and prioritizing them based on relevance to current information need, a practice we refer to as document triage. Users often perform triage through their interaction with multiple applications, and to efficiently support them in this process an extensible multi-application architecture Interest Profile Manager(IPM) was developed in the prior research at Texas A&M University. IPM infers user interests from their interactions with documents, especially the interests expressed by the user through an interpretive action like assigning a visual characteristic color, coupled with the document's content characteristics. IPM equates each specific color and application as an interest class and the main challenge for the user is to consistently maintain interest class-color scheme across applications forever which is not practical.

This thesis presents a system that can help reduce potential problems caused by these inconsistencies, by indicating when such inconsistencies have occurred in the past or are happening in the user's current triage activity. It includes (1)a clustering algorithm to group similar triage interest instances by choosing the factors that could define the similarity of interest instances, and (2)an approach to identify sequences of user actions that provide strong evidence of user's intent which can be used as constraints during clustering. Constrained and unconstrained versions of three Agglomerative Hierarchical Clustering algorithms: (1)Single-Link, (2)Complete-Link,

(3) UPGMA(Unweighted Pair Group Method with Arithmetic Mean) have been studied. The contribution of each of the three factors: (1)Content Similarity, (2)Temporal Similarity, and (3)Visual Similarity to the overall similarity between interest instances has also been examined. Our results indicate that the Single-Link algorithm performs better than the other two clustering algorithms while the combination of all three similarity factors defines the similarity between two instances better than considering any single factor. The use of constraints as strong evidence about user's intent improved the clustering efficiency of algorithms.

To my parents.

ACKNOWLEDGMENTS

I would like to acknowledge the support of my adviser, Prof. Frank Shipman in guiding and providing the necessary direction throughout this work.

I would also like to extend my thanks to committee members, fellow lab members, friends and all those people who directly and indirectly helped me in pursuing this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The continued growth in easily accessible information exposes users to an unmanageable and unknowable amount of information. Hence, users often resort to browsing sites with which they have prior experience or using a generic search engine to locate information across different domains in varied formats. Studies on Information Retrieval (IR) techniques are ongoing to bring order to this information chaos. Whether it is in the presentation of search results or the organization of content into a domain/concept model, many approaches to aiding users locate information treat each individual the same. There is an ever increasing demand to make information access more effective, by building systems that not only mimic human behavior, but also understand the expectations, goals, needs, and desires of a user in terms of specific information environment. These systems that incarnate users in terms of information access are called *user models* or *interest models* and the process of constructing, maintaining and utilizing user models is *User Modeling*.

Some user activities have specific information needs that can be satisfied by locating a single piece of information, such as to find the temperature of a place, value of a stock or the president of a country. Our current research focus is on open-ended tasks where there is generally no conclusive document providing an accurate answer. These open ended tasks are more challenging as the user is not aware initially of what to search for and where to search from. As the users skim through preliminary documents to assess their relevance to their information needs, they learn more about their activity. As a result, they identify new information needs, revise existing

The journal model is *IEEE Transactions on Automatic Control.*

information needs, find relevant alternative terminology, and unexpectedly encounter partial answers to other questions. Such learning results in new search queries and more documents to be scanned. The users' judgments in selecting documents (or data segments) to view, skim or read from those provided by search engines or other information sources determines how quickly and efficiently information needs can be satisfied. This rapid assessment of documents (or data segments) and prioritizing them based on their relevance to the current information need is *document triage* [2] or *information triage* [29]. Document triage involves how users sift through the many potentially relevant documents by prioritizing which documents to examine in more detail; identifying the most useful parts of documents; and keeping track of their progress through the search results.

A system can actively support document triage by developing models of the user's interest, determining each document's relevance to these models and recommending the documents that best match a user's interests. If such a process is successful, the user's time will be spent more efficiently by focusing on the most relevant documents. A variety of sources of information may be used and a variety of techniques may be followed when building these interest models and generating the recommendations. They can be based on the interests shown or activities done by the user, outcomes of similar information tasks performed by other users and they can also be based on similarity or relationships among the documents.

Users often interact with multiple applications while working on an open ended task: they may use a Web browser to perform searches, view the results or read the content; they may use specific reading tools like Adobe Acrobat or Microsoft Word to examine the contents of individual documents; they may use tools like Microsoft Word, PowerPoint or extensions to Web browsers to capture valuable information and take notes; and they may use organizing tools like spatial hypertext (e.g. VKB3)

[3], [40] to organize their thoughts. Though the interest models can be developed separately for each individual application with relative ease, a system supporting an efficient Document Triage process needs to compute an aggregated interest model accumulated from partial models across multiple triage-related applications. Prior research on Document Triage at Texas A&M University has developed an extensible multi-application architecture, the Interest Profile Manager (IPM) [3] that initially supported an information workspace and a document reader.

The Interest Profile Manager acts as an independent server inferring, sharing and storing user interest profiles while the triage-related applications connect to the server via a linkable software library. It supports the document triage process through four steps: (1) Recognizing the interests demonstrated by users through implicit and explicit indicators while sifting through documents, (2) Representing the recognized User Interests in a generic format and inferring the aggregated interest model to be shared across multiple applications, (3) Recognizing documents of potential interest by using information retrieval techniques to assess the relevance of each document with the interest model and, (4) Indicating the documents of potential interest by visually distinguishing them. IPM supports VKB3, a spatial hypertext workspace application and Mozilla Firefox which is a browsing and reading application.

IPM infers interest models by taking user's interactions with documents [3], especially interests expressed by the user in their interpretation of the document combined with document's content characteristics. The primary form of interpretation being considered is when a user assigns a visual characteristic like a color to a document or a document segment. In the remainder of this thesis, each such interpretive action is referred as *Interest Instance.* The IPM equates each specific color and application as an interest class. For the color coding of interests to be consistent, this interest class-color scheme has to be consistently maintained by the user during all the interest instances across multiple applications and across time. This is a root of the problem being addressed in this thesis, namely that the initial user interest models generated by the IPM are based on the colors applied to documents and document segments in each application and these colors are used to indicate when new documents match an inferred user interest.

It is not realistic for the user to remember the interest class-color scheme forever across applications. Nor is it realistic to assume the colors will remain the same across applications as different applications are likely to provide different colors for interpreting and annotating documents. Also, given the limited number of colors available in many applications that are acceptable to users, colors can be reused for multiple categories of interest.

The work discussed in this thesis investigates this issue and presents a new approach to automatically group the similar interest instances from past assignments based on an understanding of the user's intent. These can be reviewed by the user to correct his previous wrong assignments and to better the consistency in his future assignments. It makes use of a modified IR clustering technique [23] customized to suit the needs of document triage and considers three main factors while calculating the similarity between the instances: (1) Content Similarity, which is based on the similarity of the textual content of interest instances; (2) Temporal Similarity, which is based on time when the interest instances are last modified by the user and; (3) Visual Similarity, which is based on how similar the visual characteristics are for the interest instances. The system implicitly derives constraints on interest instances from the user behavior during the triage activity and supplies them as an input to the clustering algorithm.

Users often interact with multiple applications while working on an open-task and it is essential that a system implementing document triage supports multiple triage-related applications. Though IPM already supports an organizing application and a reading application, it doesn't support any note taking or authoring tools. Such tools are important in the context of document triage, as users often capture important points from the documents they encounter as notes for later use. Thus, we believe information from user authoring tools will be a valuable contributor in inferring the user's interest during triage process. Part of the work presented here is the inclusion of user activity data from two important authoring tools, Microsoft Word and PowerPoint.

The remainder of the thesis is organized as follows. The problem statement for the thesis is more formally presented in the next section. Related Work on document triage applications and clustering techniques is briefly discussed in section 3. Section 4 presents the prior work on the Interest Profile Manager. Section 5 and section 6 describe the Approach and System Design respectively. Section 7 reports the evaluation process and analysis of the results.

CHAPTER II

PROBLEM STATEMENT

Turning records of user activity in applications into a user interest model is difficult. Each application provides a unique way of interacting with information and, thus, users of an application indicate interest through a variety of application-specific interaction behaviors. The Interest Profile Manager [3] plays a key role during the document triage process by collecting and aggregating the partial interests of the user provided from multiple applications. Once the user interest is aggregated, the relative user interest for all target documents with respect to each of the interest classes is calculated by the IPM and the results are broadcast to all the registered applications. Currently, IPM takes user-assigned colors as the prime indicator of interest shown by the user to associate a document with an interest class and other document attributes are used to characterize the document.

As the user assigns more colors while expressing their interest, the knowledge base of the inferred user interests grows, hopefully resulting in the system being able to come up with better recommendations. This is true as long as the user is accurate and consistent in his interest class-color scheme. The user needs to maintain the same coloring scheme for interest classes across different applications and time. Inconsistencies in assigning these colors result in interest classes not strongly representing any single interest of the user thereby, decreasing IPM's efficiency in the prediction of similar documents. On the other hand, it is not practical for a user to remember the coloring schemes of his interest classes forever. In such systems, there is a need for a mechanism that identifies situations where the same color has been used for more than one interest or that different colors have been used for the same interest. Such identification can be used in multiple ways. It can be used to recommend changes to make the visual mapping more consistent. Alternatively, it can be used to provide legends that indicate the mappings between interests and all the associated colors. Two sub problems are identified to build such capabilities: identifying the bounds in terms of time and/or space of a mapping between an interest and a color, and grouping the resulting interests into meaningful higher-level user interests.

A. Identifying Sequences of Interest Instances that Suggest User's Real Intent

Users perform many different actions while interacting with the applications during the triage process but often only few of these actions are currently used to infer his/her interests. By taking into account time and other information concerning sequences of these few interactions, the system can determine how good the evidence is for a user's interests. For example, when a user assigns the same color to two documents (d1, d2) at almost the same time (say within less than a minute), it is strong evidence that they belong in the same interest class (c1). In another scenario, a document d3 is initially assigned a different color (interest class c2) and after a long gap (say 1 month) document d4 is also assigned the color used for c2. It is more probable that the intent in the first scenario is to define a classification including d1 and d2 than it is the intent to define a classification including d3 and d4. Identifying sequences of actions that provide the strong evidence exemplified in the first situation will help the system infer user models that match user intent as it can provide constraints for merging or not merging interest instances into interest classes.

B. An Approach to Group Similar Interest Instances Based on User's Intent

As discussed earlier, expression of interests by a user may display many inconsistencies such as two interest instances might be assigned the same color even though they are considered to be classified into distinct interest classes by the user. Similarly, two interest instances may be assigned different colors even though they belong to a single interest class from a user's perspective.

A system can help reduce potential problems caused by these inconsistencies by indicating when such inconsistencies have occurred in the past or are happening in the user's current activity. Such identification requires a mechanism to group similar interest instances by choosing the factors that could define the similarity of interest instances. This thesis presents an approach to clustering user interest instances and an approach to identify strong evidence that can be used as constraints for clustering (or not clustering) instances.

CHAPTER III

RELATED WORK

Related work falls into three main categories: (1) research into methods for identifying user interests and building interest models, (2) investigations in the field of document triage technology and practice and, (3) information retrieval clustering techniques to group similar documents.

A. Methods for Identifying User Interests and Building Interest Models

Recommendation and adaptive filtering systems are being applied for a wide range of information sources and are often successful in supporting a cooperative process for information location. Examples include Netflix recommending movies to users based on their prior movie ratings or Amazon recommending new products based on past user interaction. These systems need to understand the information need and preferences of the users with whom they are interacting. This knowledge can be acquired through variety of sources and interpreted in many ways. This user specific knowledge is usually referred as interest model or user model. User interest modeling enables a variety of services like helping and advising the user [35], [20], [41], tutoring systems [46] and error correction tools [30].

Research into methods for gathering and recognizing user interests includes analyzing explicit expression of user interests (e.g. ratings), implicit expression of user interest (e.g. scrolling time, click-through records) or a combination of both.

1. Explicit Indicators

The most obvious source for systems gaining an understanding of a user's interest is to ask them. Many recommendation systems use this approach, asking users to explicitly express their interest in an entity, for example, the rating for a movie, book or a cellphone. Several digital library systems also use this approach [34], [36].

As explicit indicators are direct information from the user, they are generally of high information value with respect to user interests, easy to understand and require no further interpretation. However, it requires extra effort from the users as they have to spend their own time to tell the system what they think about a piece of information, which may result in altering their normal reading and browsing patterns [10]. Users may not rate unless they find an incentive for their efforts [21] and can even lose interest in reading if prompted repeatedly. Moreover, users rate far fewer documents than they read [39] leaving many documents not associated to any explicit indicators. Thus the benefits of having high information value from the explicit indicators may be offset by their drawbacks.

2. Implicit Indicators

Implicit interest indicators are less intrusive but rely on methods of inferring user interests based on user behavior rather than directly obtaining it from the user. During the triage activity, a user's interest in a piece of information is also indicated by their interaction with the information: the time spent on a document or document segment while reading or editing; how much of the document they examine (e.g. how far into a document they scroll); the scrolling speed; how they categorize the document (e.g., stacking it with other interesting documents); and through other behaviors that in part rely on the tools they are using. All these activities may be recorded in the background while the user interacts with the system.

The influence of each of these implicit indicators is not yet thoroughly understood and research into their use is on-going. Morita and Shinoda [31] studied the relation of the amount of time spent reading Usenet News articles with users' interest in a controlled experimental environment and carefully chosen news domain. It is observed that the time users spend on reading Usenet news articles was the primary indication of their interest. Konstan et al [26] described how the GroupLens system for filtering Usenet news can be used to study the correlation between time spent reading an article and explicit indicators. They observed that predictions based on reading time are nearly as accurate as predictions based on explicit indicators. Claypool et al [10] extended these studies into alternative domains in a less controlled environment and by greatly expanding the number of implicit indicators examined. They found that the time spent on a page, the amount of scrolling on a page, and the combination of time and scrolling had a strong correlation with explicit interest, while individual scrolling methods and mouse-clicks alone were ineffective in predicting the explicit interests. Mac Aoidh et al [28] investigated the effects of implicit indicators in the context of geographic information systems (GIS). They examined the mouse movements and map browsing behavior of the user and found that the interests can be inferred reasonably effectively for spatial information using mouse movement data, but may not be sufficiently accurate as a stand-alone interest indicator. Other studies considered alternative user activities, like using the overlap between bookmark files to determine similarity among individuals [38], and the saving of references to an item as a strong indicator of interest [27].

Using implicit indicators for user modeling provides the system with many advantages. These include removing the extra user effort required to examine and rate items and turning every user interaction into potential indication of user interest and an opportunity for feedback. Though implicit indicators are less likely to be as accurate as explicit indicator, combining them with other implicit or explicit indicators may result in a more accurate and complete representation of user interest.

B. Document Triage

Document triage is the critical point in the information seeking process when the user first decides the relevance of a document to their information need. Different aspects of the document triage activity have been studied: Cool et al [12] investigated what document characteristics, like titles, length, embedded images, affect the user's judgments on relevance of a document to a particular information need; Bae et al [2], Marshall and Shipman [29] studied on how users interpret, structure and categorize the documents in a task context; Buchanan and Loizides [9] investigated how triage activity differs on paper from triage activity with electronic media tools.

A close look at the characteristics of documents and of the triage activity shows some limitations in current systems. First, documents are generally treated as one atomic unit but many useful documents might be long and may be dealing with multiple subtopics even though the user is interested in only a few segments. Second, the systems monitor the user activity only within a single application even though real triage activity involves user activity in multiple applications (e.g. a reader application, note taking application, organizing workspace). Prior research at Texas A&M University showed that models combining interest information from multiple applications are more effective than those that rely on information from a single application [1]. Based on the investigation of these two issues, the Interest Profile Manager [3] has been developed.

As shown in Fig. 1, Interest Profile Manager acts as the central server coordinating with all triage-related applications. It accumulates implicit and explicit indicators from each application representing user's partial interests, analyzes them and infers combined user interests and finally provides information so applications can generate appropriate visualizations. There is also support for the user to show interest on

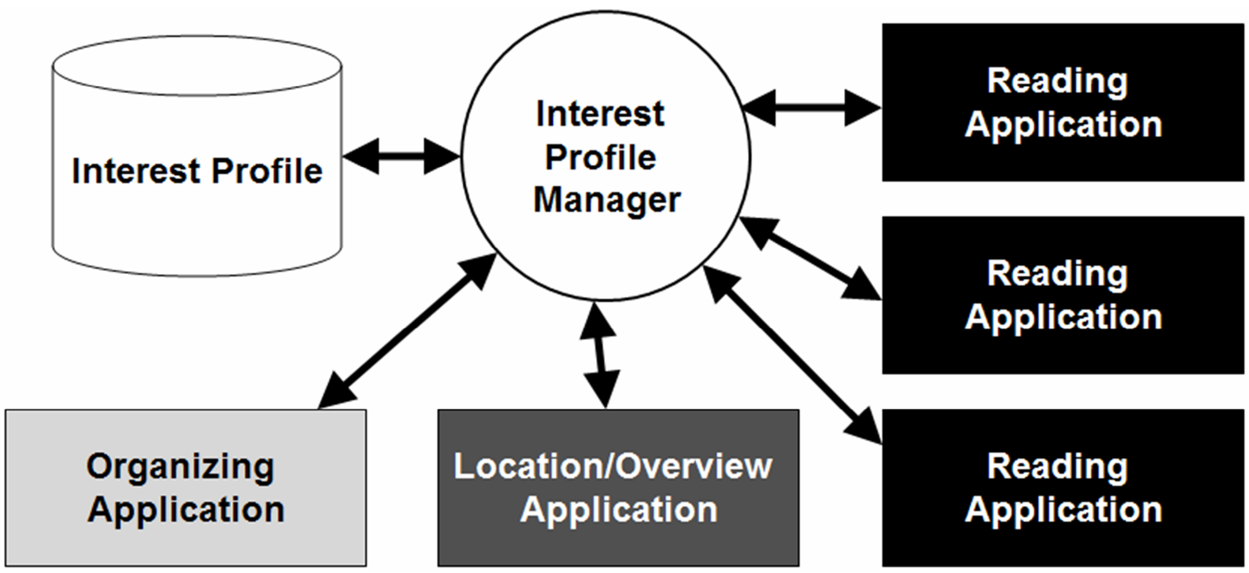


Fig. 1. IPM's Interaction with Triage-Related Applications

selected segments instead of the whole document.

C. Clustering

Clustering is the process of grouping elements together based on some measure of similarity or overall desirability. More formally, given a set of N patterns (i.e. data points), the task of clustering is to come up with a labeling scheme so that each pattern p_i is assigned a label $L(p_i) := L_i \epsilon \{1, ..., K\}$. The patterns with the same label L_j form the cluster C_j. The basic and standard approaches to clustering data can be described with the help of the hierarchy shown in Fig. 2. This taxonomy is based on a survey of clustering approaches [23], although other taxonomical representations of approaches to clustering are possible.

Partitional Clustering algorithms obtain a single partition of the data (i.e. generates one level of groupings) and many applications frequently use these computa-

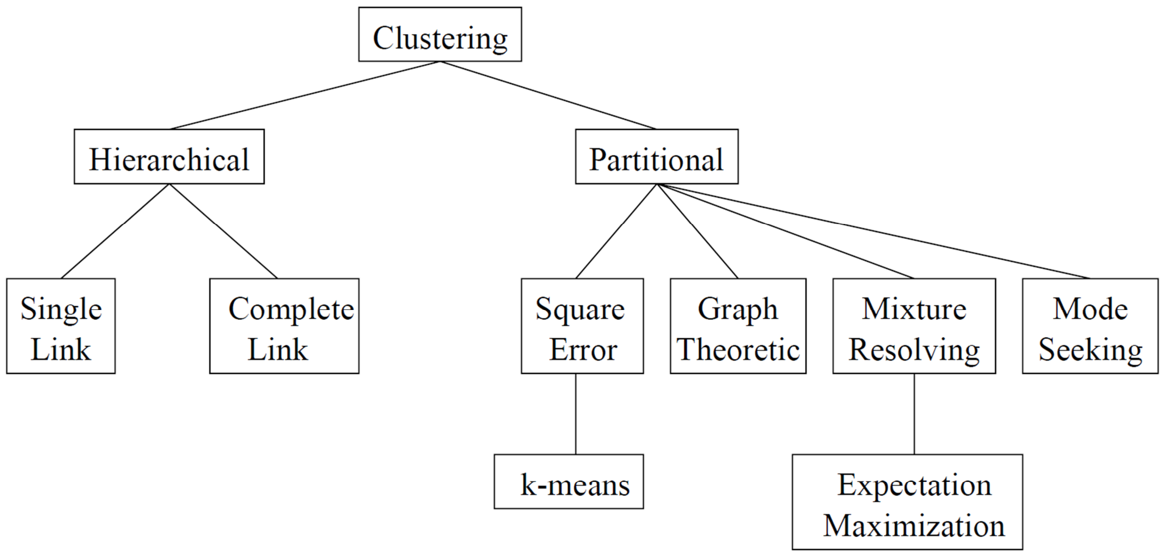


Fig. 2. A Taxonomy of Clustering Approaches

tionally efficient clustering algorithms, such as K-means. The K-means problem for a given set of n patterns, $S = (p_1, ..., p_n)$ is to form a k-block set partition of S so as to minimize the vector quantization error. The K-means algorithm finds a local minima and has linear complexity of *O(kmni)*, where *'k'* is the number of instances, *'m'* is the number of attributes, *'n'* is number of clusters and *'i'* is the number of iterations of the algorithm. However, the algorithm is sensitive to the choice of the initial starting conditions [17], [8] and hence in practice needs to be restarted many times, randomly or otherwise choosing the starting conditions for each application of the algorithm.

Hierarchical clustering algorithms are run once and create a tree dendrogram which is a tree structure containing a k-block set partition for each value of k between 1 and n. Most hierarchical clustering algorithms are variants of the (1) single-link [42]; (2) complete-link [24]; and minimum-variance [45], [32] algorithms. Though, these algorithms are useful in the domains where clusters naturally form hierarchy, they come with some additional complexity [33], [19] in time and space, since the implementation requires $O(mn^2)$ computations and $O(n^2)$ space.

Clustering with constraints [4], [6], [11], [14], [18] is another active area of research in Data Mining which allows the incorporation of background domain expertise into the algorithms. In the last few years, there has been extensive work on incorporating instance-level constraints into clustering methods [44], [25], [47]. These constraints help in creating clusters with desirable properties.

There can be different types of constraints at each level of clustering. Two types of instance level constraints were introduced by Wagstaff [43]: (1) *must-link* denoted by $c_=(x, y)$, which means two instances must be in the same cluster; and (2) *cannot-link* denoted by $c_{\neq}(x, y)$, which means two instances cannot be in the same cluster. There are also cluster level constraints like δ-constraint which requires the distances between any pair of points in two different clusters to be at least δ; the ϵ-constraint, which for any cluster C_i with two or more points, requires that for each point $x \epsilon C_i$, there must be another point $y \epsilon C_i$ such that the distance between x and y is at most ϵ [13]. In other words, δ is a minimum distance for points in different clusters while ϵ is a maximum distance for points in the same cluster.

There are two ways in which the clustering algorithms try to implement these constraints. The first one is to use them to modify the cluster assignment stage of the clustering algorithm so as to enforce their satisfaction of the constraints as much as possible. The papers [44], [15] discuss about techniques with strict enforcement while [5], [16] discuss about techniques with partial enforcement. Second, the distance function of the clustering algorithm can also be trained from the constraints either before or during the actual clustering. Many clustering algorithms using the trained distance measures [22] have been employed for constrained clustering, including single-link [7], complete link agglomerative clustering [25] and K-Means [47].

In the present work, two instance level constraints, *must-linked* and *cannot-linked*, are considered. The constrained clustering approach of training a distance function used to perform agglomerative clustering was chosen for this work.

CHAPTER IV

PRIOR WORK

Prior work at Texas A&M University on the Interest Profile Manager provides the context for the work in this proposal. Fig. 1 shows the high level block diagram of Interest Profile Manager's interactions with multiple applications during document triage process. It collects information about interest-related activity from different triage-related client applications, aggregates them, saves as an interest profile for each user and uses it to infer new documents of interest for the user. The inferred results are broadcasted across all participating client applications.

IPM is designed to act as the central server guiding the whole process while extensions are written for the triage-related applications to act as clients. It has three main modules: (1) Request Handler, (2) User Profile Handler and, (3) Inference Manager. Request Handler receives the requests from different client applications, analyzes them, calls User Profile Handler for the user profile to be updated and if required generates appropriate requests to be forwarded to Inference Manager. Inference Manager receives requests from Request Handler, interacts with User Profile Handler to get the complete user profile and infers various user interests depending on the kind of request. User Profile Handler interacts with Request Handler to collect the partial interests of the user as they appear, aggregates them to form a complete user profile, interacts with Inference Manager to provide the aggregated user profile and also has the provision for serializing interest profiles so they persist across sessions. IPM is implemented to be easily extensible so that additional applications can be added as appropriate, for example a new viewing/reading application to enable the user work with new content types.

Any application that is extended to implement interest profile client software interface can communicate with IPM server by exchanging interest profiles in a generic predefined format and can act as a client in the IPM architecture. These client applications can be implemented to support two-way communication or, one way communication in which they could merely provide information to the IPM or only receive information from the IPM. In the prior implementation, client extensions were written for two applications both supporting two-way communication: an extension for VKB3 [3], which is a spatial hypertext workspace for collecting, analyzing and organizing documents; and WebAnnotate extension for Mozilla Firefox, a reading application for web pages.

Fig. 3 shows the block diagram of different modules interacting in prior IPM implementation.

A. IPM Client for VKB3

Fig. 4 illustrates a triage scenario in the VKB3 application. In this scenario, the user is investigating Apple Corporation and is starting with 10 Google search results. The user has colored blue one document which he perceives as valuable on Apple gadgets and colored red a document on general information about Apple. It can be observed in Fig. 4 that the system has provided suggestions to the user by coloring the shadow of documents which possibly correspond to each of the interest classes expressed by the initial coloring of the first two documents. This is possible through the interaction of VKB3 with IPM. It is likely the workspace will grow to contain more collections as the user works with the documents and discovers the need for further searches; he may also create new collections to categorize documents and manage the space.

For the triage investigation, VKB 3 was extended to communicate with the Interest Profile Manager (IPM). Communication with the IPM is two-way: VKB 3 sends

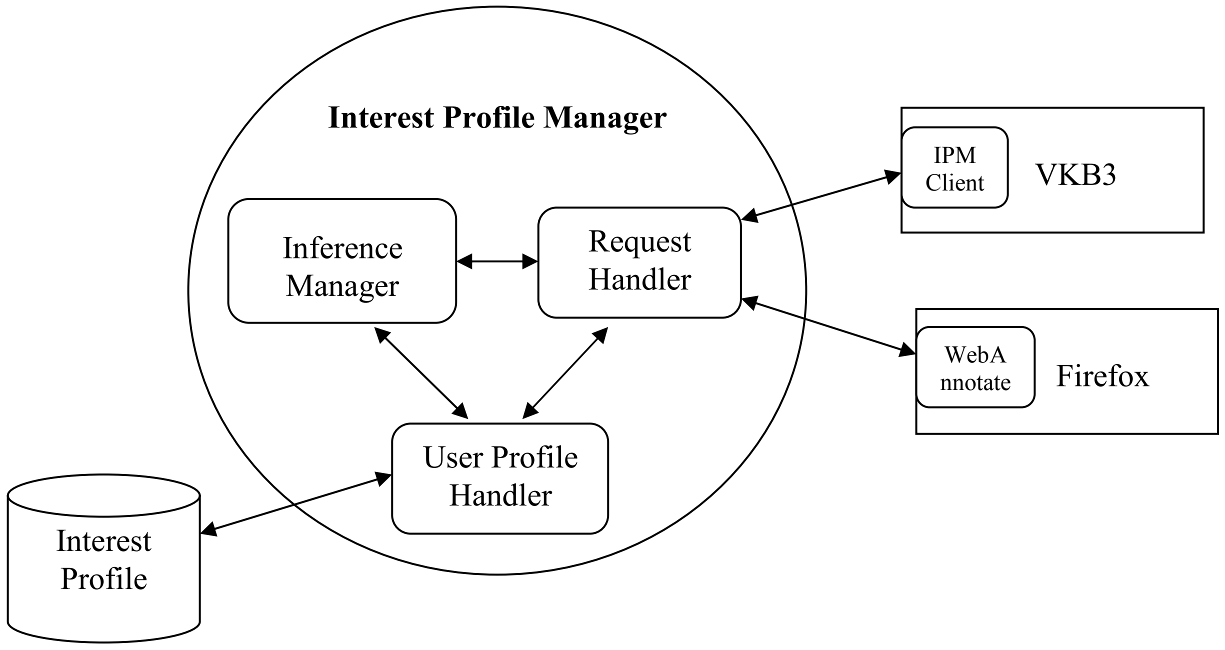


Fig. 3. IPM Module Interaction

two kinds of information to the IPM as input to the algorithm that computes user interests: (1) Records of user actions and the attribute/value pairs that characterize Web document objects in the workspace. In other words, as users open, move, color, delete, or otherwise modify document objects, records containing these actions are sent to the IPM; likewise, as document objects are added to the workspace or the attributes' values are edited, this information is also sent to the IPM. (2) VKB also receives information about user interests from the IPM, which it uses to modify the system layer of Web document objects in the workspace. The IPM infers user interest based on information collected across all of the triage applications and sends the inferred interest back to VKB (as well as to the other client applications). Each item in the IPM results includes three components: information ID (so VKB can determine which object the interest applies to), interest classification (to specify topic), and interest level (to specify intensity).

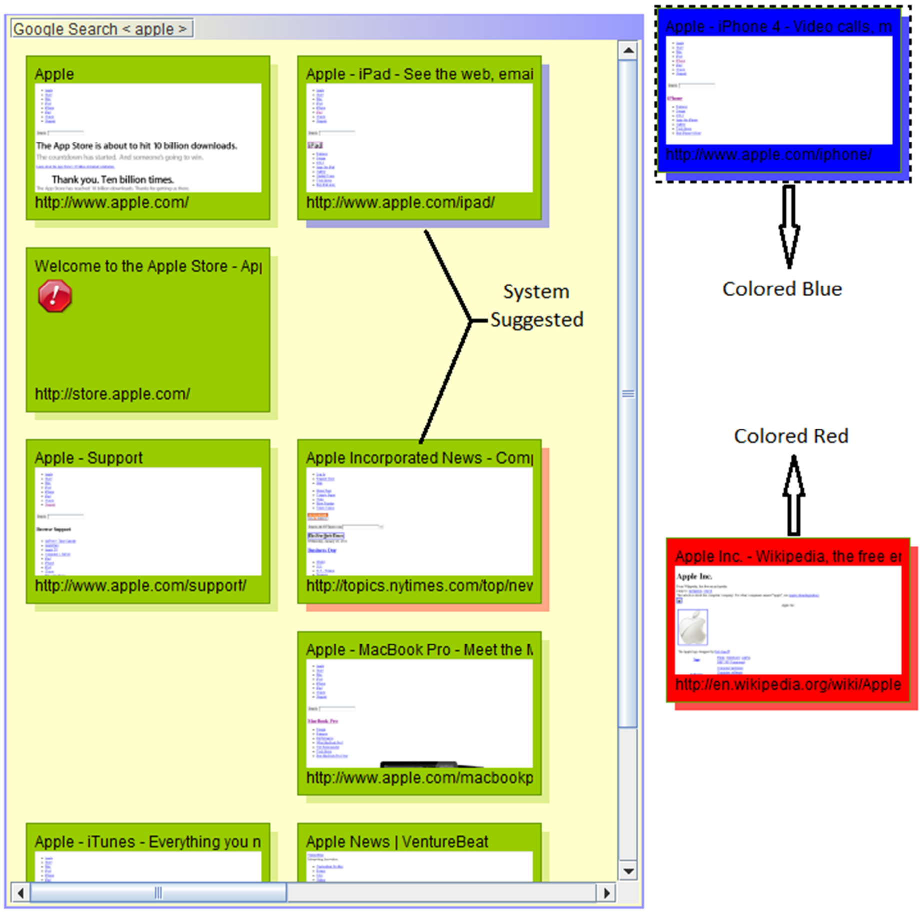

Google Search < apple >
Apple
http://www.apple.com/
Apple - iPad - See the web, emai
http://www.apple.com/ipad/
Welcome to the Apple Store - Ap
http://store.apple.com/
System Suggested
Apple - Support
http://www.apple.com/support/
Apple Incorporated News - Com
http://topics.nytimes.com/top/nev
Apple - MacBook Pro - Meet the N
http://www.apple.com/macbookp
Apple - iTunes - Everything you n
Apple News | VentureBeat
Apple - iPhone 4 - Video calls, m
http://www.apple.com/iphone/
Colored Blue
Colored Red
Apple Inc. - Wikipedia, the free e
http://en.wikipedia.org/wiki/Apple

Fig. 4. VKB3 During Document Triage

B. WebAnnotate Client for Mozilla Firefox

To further facilitate triage, an add-on was developed for Mozilla Firefox called WebAnnotate that provides basic annotation capabilities, collects data on users' interactions with documents, and uses interest data returned from the IPM to create visualizations designed to focus readers' attention on the portions of documents rele-

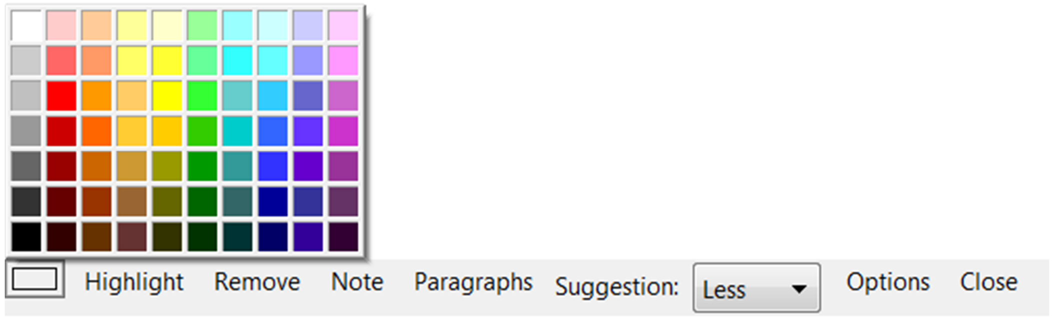
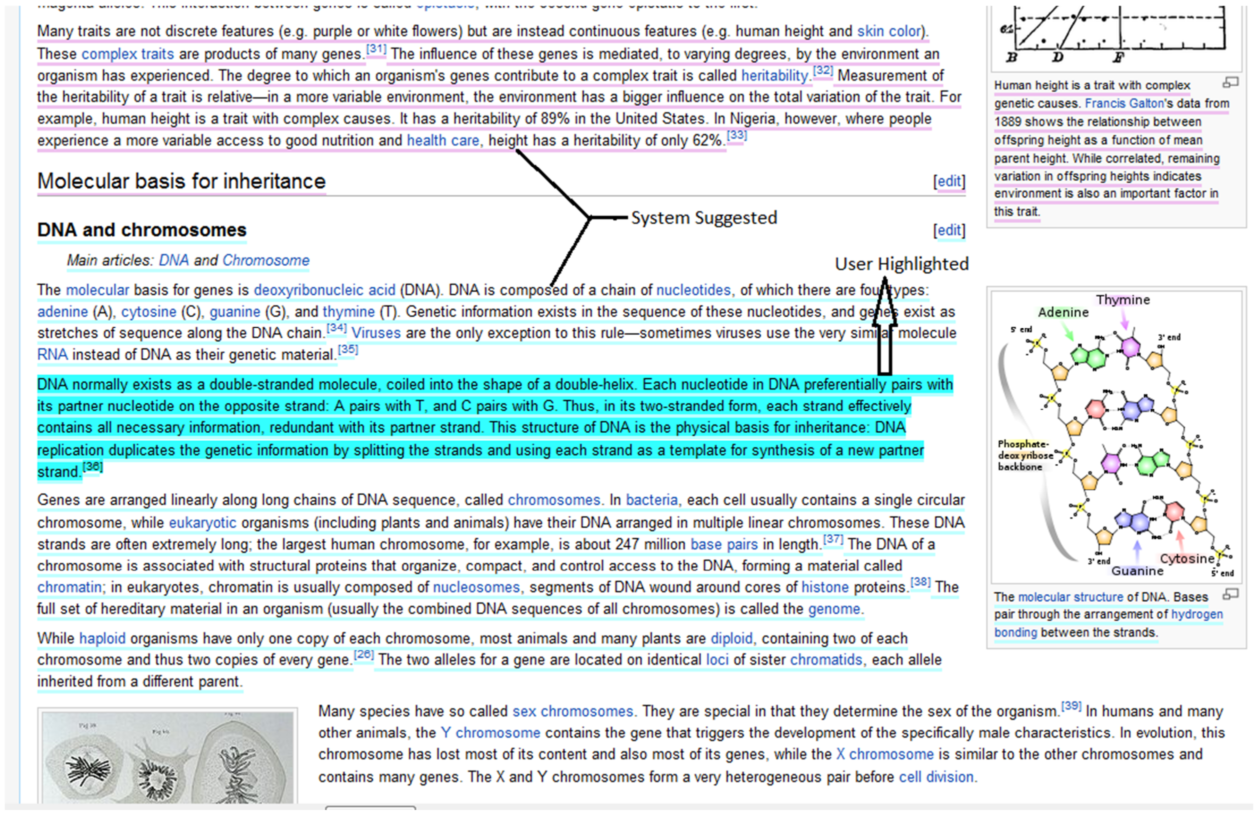


Fig. 5. Mozilla Annotation Toolbar

vant to their interests. These visualizations enable users to quickly locate what they want to read without taking the selected material out of context.

WebAnnotate supports several representative forms of annotation on HTML documents; once users activate the annotation toolbar (Fig. 5), they can highlight text in different colors and can create colored sticky notes (editable translucent text boxes that can be moved anywhere on the HTML document). It stores the reader's annotations separately from the HTML document, in the IPM, where they are used as input to the interest estimation algorithm. Whenever a user opens a HTML document, WebAnnotate checks the IPM for annotations to that document. If any are found, WebAnnotate regenerates them.

The communication between WebAnnotate and the IPM is two-way, similar to the communication between VKB and the IPM. When a user opens a Web page, WebAnnotate extracts document attributes and sends them to the IPM. WebAnnotate parses the text content into paragraphs and assigns paragraph IDs to them. This information is used by the IPM to infer and communicate potential interest on specific paragraphs to WebAnnotate. Annotation information that is sent to the IPM includes the color and type of the annotation (whether it is a highlight or sticky note)



Many traits are not discrete features (e.g. purple or white flowers) but are instead continuous features (e.g. human height and skin color). These complex traits are products of many genes.[31] The influence of these genes is mediated, to varying degrees, by the environment an organism has experienced. The degree to which an organism's genes contribute to a complex trait is called heritability.[32] Measurement of the heritability of a trait is relative—in a more variable environment, the environment has a bigger influence on the total variation of the trait. For example, human height is a trait with complex causes. It has a heritability of 89% in the United States. In Nigeria, however, where people experience a more variable access to good nutrition and health care, height has a heritability of only 62%.[33]

Human height is a trait with complex genetic causes. Francis Galton's data from 1889 shows the relationship between offspring height as a function of mean parent height. While correlated, remaining variation in offspring heights indicates environment is also an important factor in this trait.

Molecular basis for inheritance [edit]

DNA and chromosomes [edit]

Main articles: DNA and Chromosome

The molecular basis for genes is deoxyribonucleic acid (DNA). DNA is composed of a chain of nucleotides, of which there are four types: adenine (A), cytosine (C), guanine (G), and thymine (T). Genetic information exists in the sequence of these nucleotides, and genes exist as stretches of sequence along the DNA chain.[34] Viruses are the only exception to this rule—sometimes viruses use the very similar molecule RNA instead of DNA as their genetic material.[35]

DNA normally exists as a double-stranded molecule, coiled into the shape of a double-helix. Each nucleotide in DNA preferentially pairs with its partner nucleotide on the opposite strand: A pairs with T, and C pairs with G. Thus, in its two-stranded form, each strand effectively contains all necessary information, redundant with its partner strand. This structure of DNA is the physical basis for inheritance: DNA replication duplicates the genetic information by splitting the strands and using each strand as a template for synthesis of a new partner strand.[36]

Genes are arranged linearly along long chains of DNA sequence, called chromosomes. In bacteria, each cell usually contains a single circular chromosome, while eukaryotic organisms (including plants and animals) have their DNA arranged in multiple linear chromosomes. These DNA strands are often extremely long; the largest human chromosome, for example, is about 247 million base pairs in length.[37] The DNA of a chromosome is associated with structural proteins that organize, compact, and control access to the DNA, forming a material called chromatin; in eukaryotes, chromatin is usually composed of nucleosomes, segments of DNA wound around cores of histone proteins.[38] The full set of hereditary material in an organism (usually the combined DNA sequences of all chromosomes) is called the genome.

The molecular structure of DNA. Bases pair through the arrangement of hydrogen bonding between the strands.

While haploid organisms have only one copy of each chromosome, most animals and many plants are diploid, containing two of each chromosome and thus two copies of every gene.[26] The two alleles for a gene are located on identical loci of sister chromatids, each allele inherited from a different parent.

Many species have so called sex chromosomes. They are special in that they determine the sex of the organism.[39] In humans and many other animals, the Y chromosome contains the gene that triggers the development of the specifically male characteristics. In evolution, this chromosome has lost most of its content and also most of its genes, while the X chromosome is similar to the other chromosomes and contains many genes. The X and Y chromosomes form a very heterogeneous pair before cell division.

Fig. 6. WebAnnotate Interacting with IPM

as well as other terms necessary to reconstruct and describe the annotation (e.g. the anchor text or text of the note and the annotation's location). The annotation representation assumes that documents are static, an assumption that reflects the nature of the triage task. The documents' DOM structure is used to specify the highlight's anchor location; likewise, sticky notes are reattached to Web pages according to their absolute (x, y) positions. Unlike VKB, which sends events as they occur, WebAnnotate aggregates events until the user's attention turns elsewhere and the browser window loses focus.

The IPM communicates inferred user interests to WebAnnotate in a form similar to those sent to VKB; they are represented by information ID, interest classification, and interest level. Unlike VKB, the information ID consists of a document URL and a set of paragraph ids, because user interests are calculated at the paragraph level rather than the whole document level. WebAnnotate brings paragraphs to a user's attention by underlining them (i.e., users highlight; the system underlines) as shown in Fig. 6.

CHAPTER V

APPROACH

The discussion so far explained prior work on Interest Profile Manager and identified issues limiting its effectiveness in inferring the interests of the users accurately. This section discusses the approaches adopted to tackle the issues identified in the earlier state of IPM.

A. Identifying Sequences of Interest Instances that Suggest User's Real Intent

Applications provide unique ways for the user to interact with information and each user interacts with information through an application differently. Though users perform many actions during these interactions, only some of these actions may be used to derive his/her interests while other actions are not useful in the deduction of interests. The actions which contribute in deriving users interests may be explicit actions by the user like highlighting or implicit actions like scrolling speed (e.g. slow scrolling through a document against scrolling through a document of similar length quickly.) During the triage process, such actions are correlated with the user's real intent with respect to information.

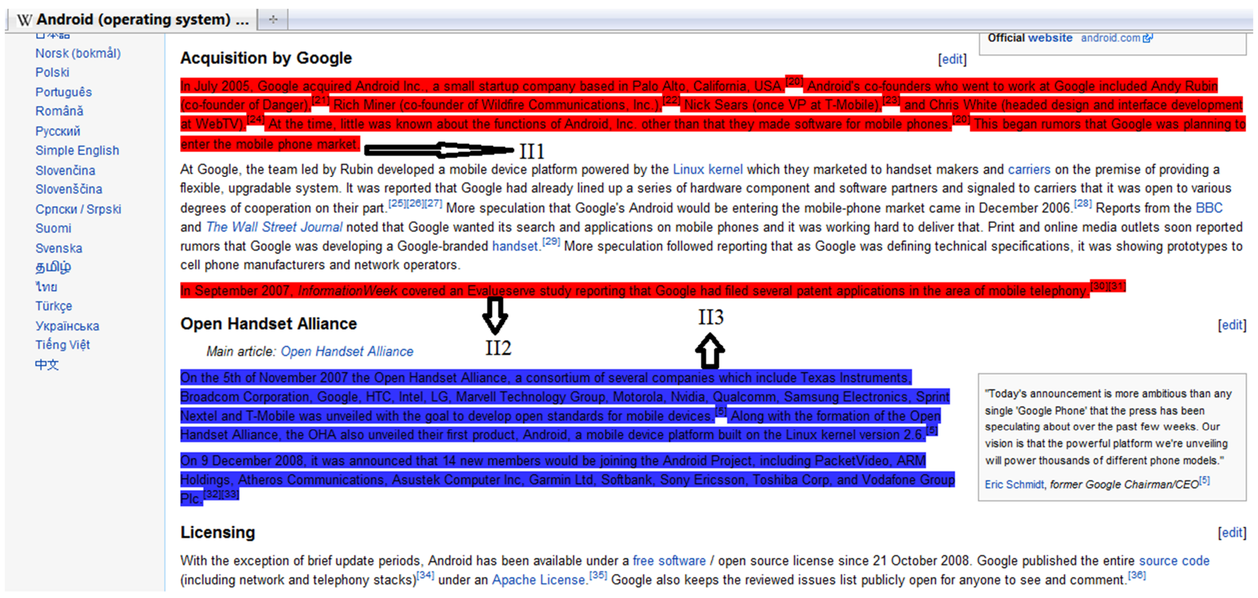
While working on a triage activity, the user has intentions for each action he/she performs. For example, the action of assigning different colors to pieces of information is indicating his/her classification of the information. A successful interest modeling mechanism needs to properly estimate the intent of a user's actions and include that estimation in the algorithm that generates recommendations. Such recommendations could be personalized for each user, given an approach to learn mappings between actions and intents, so that the suggestions are more closely aligned with his/her interest.

The user's actions during information triage may be intentional or mistake when measured against their actual intent. For example, when assigning colors to interest instances, the user might deliberately apply different colors to two interest instances or might mistakenly assign the wrong color to one of the instances classifying it to be in a different interest class though his/her intention was to put it in the same interest class. As such, the same actions may be interpreted differently in different circumstances.

We have developed a 'Constraint Builder' module which is a preprocessing step before the actual clustering algorithm (described in the next section) that analyzes all the user actions and identifies the sequences of actions that will be used in interpreting the user's intent. In particular it uses heuristics to identify sequences for which the system will assume the user's actions will be interpreted as consistent. Two types of constraints, *must-linked* [43] and *cannot-linked* [43], are developed from these identified sequences of actions and passed to the clustering algorithm as conditions to be satisfied.

The heuristics were developed based on two observations of user behavior. First, some users remember the color-to-interest class mapping in IPM for long periods of time and use them consistently while others forget (or ignore) the mapping after a while, resulting in inconsistent coloring patterns. This effect may vary based on the user's ability to remember. But, if considered in extremely short intervals of time, most users maintain consistency in their assignments. Second, the user is more likely to be consistent in assigning colors when using a single application than when using multiple applications. These two observations are used by the *Constraint Builder* module to identify constraints.

The entire timeline of the triage activity on IPM by a user is divided into tiny time intervals and his/her actions are monitored separately in each of these intervals.



Acquisition by Google [edit]

In July 2005, Google acquired Android Inc., a small startup company based in Palo Alto, California, USA.[20] Android's co-founders who went to work at Google included Andy Rubin (co-founder of Danger),[21] Rich Miner (co-founder of Wildfire Communications, Inc.),[22] Nick Sears (once VP at T-Mobile),[23] and Chris White (headed design and interface development at WebTV).[24] At the time, little was known about the functions of Android, Inc. other than that they made software for mobile phones.[20] This began rumors that Google was planning to enter the mobile phone market.

At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel which they marketed to handset makers and carriers on the premise of providing a flexible, upgradable system. It was reported that Google had already lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.[25][26][27] More speculation that Google's Android would be entering the mobile-phone market came in December 2006.[28] Reports from the BBC and *The Wall Street Journal* noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded handset.[29] More speculation followed reporting that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

In September 2007, *InformationWeek* covered an Evalueserve study reporting that Google had filed several patent applications in the area of mobile telephony.[30][31]

Open Handset Alliance [edit]

Main article: Open Handset Alliance

On the 5th of November 2007 the Open Handset Alliance, a consortium of several companies which include Texas Instruments, Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel and T-Mobile was unveiled with the goal to develop open standards for mobile devices.[5] Along with the formation of the Open Handset Alliance, the OHA also unveiled their first product, Android, a mobile device platform built on the Linux kernel version 2.6.[5]

On 9 December 2008, it was announced that 14 new members would be joining the Android Project, including PacketVideo, ARM Holdings, Atheros Communications, Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, and Vodafone Group Plc.[32][33]

"Today's announcement is more ambitious than any single 'Google Phone' that the press has been speculating about over the past few weeks. Our vision is that the powerful platform we're unveiling will power thousands of different phone models."

Eric Schmidt, *former Google Chairman/CEO*[5]

Licensing [edit]

With the exception of brief update periods, Android has been available under a free software / open source license since 21 October 2008. Google published the entire source code (including network and telephony stacks)[34] under an Apache License.[35] Google also keeps the reviewed issues list publicly open for anyone to see and comment.[36]

Fig. 7. Interest Instances Example in Mozilla

If the user assigns same color to two different instances in the same time interval and in the same application, it is highly probable that user wants to keep these instances in the same interest class. These actions by the user imply that they have to be in same interest class though the system is not sure of what else may be in the same interest class. So, a *must-linked* constraint between these two interest instances is created. Likewise, if the user assigns different colors to two instances in the same time interval and in the same application, it is highly probable that user wants these two instances to be in different interest classes. Thus, in such cases the Constraint Builder assumes that the user intends the instances to be in different interest classes though it is not sure which of the interest classes. So, a *cannot-linked* constraint between these two interest instances is created. All the identified *must-linked* and *cannot-linked* constraints are supplied as input to the clustering algorithm (described in the next section) to use while grouping similar interest instances.

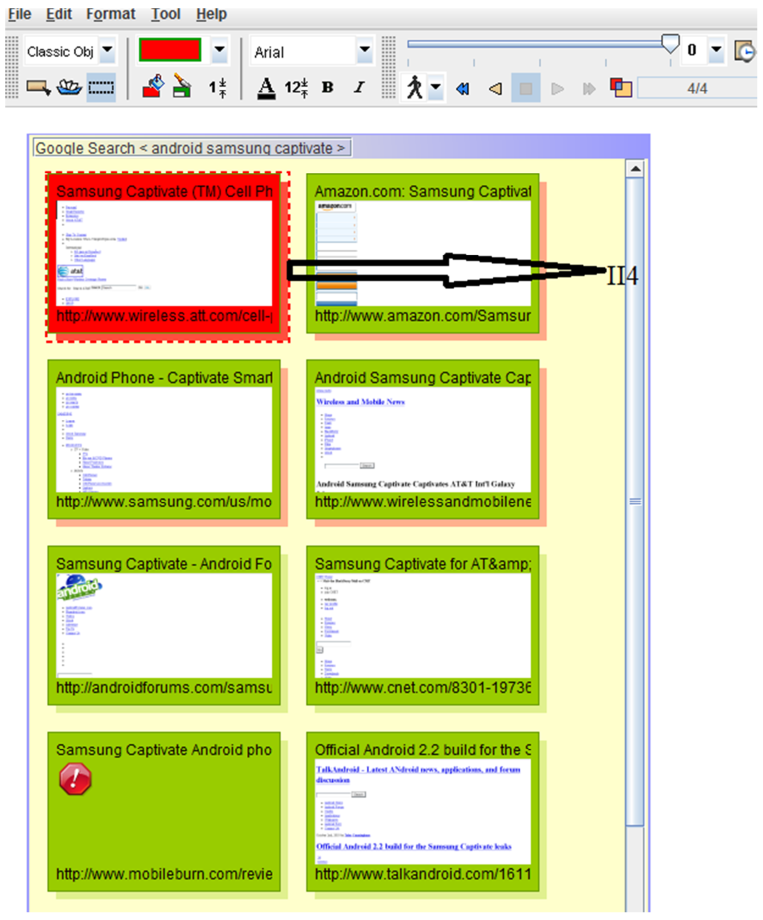

File Edit Format Tool Help
Classic Obj
Arial
0
4/4
Google Search < android samsung captivate >
Samsung Captivate (TM) Cell Ph
http://www.wireless.att.com/cell-
Amazon.com: Samsung Captival
http://www.amazon.com/Samsur
II4
Android Phone - Captivate Smart
http://www.samsung.com/us/mo
Android Samsung Captivate Cap
http://www.wirelessandmobilene
Samsung Captivate - Android Fo
http://androidforums.com/samsu
Samsung Captivate for AT&amp;
http://www.cnet.com/8301-19736
Samsung Captivate Android pho
http://www.mobileburn.com/revie
Official Android 2.2 build for the S
http://www.talkandroid.com/1611

Fig. 8. Interest Instances Example in VKB

Fig. 7 and Fig. 8 illustrate a triage scenario in which the user is investigating the Android mobile operating system. The user has initially opened Wikipedia about Android OS in the Mozilla application and created three interest instances within a duration of 1 minute as shown in Fig. 7: II1 is on the Google acquisition of Android and is colored red; II2 is also related to news about a Google acquisition in the mobile market and is colored red; and II3 is related to the creation of Open Handset Alliance and is colored blue. After 24 hours the user opens VKB, searches for 'Android Samsung Captivate' and creates an interest instance II4 (Fig. 8) by coloring one of the results red. This web document is from the AT&T Wireless web site and describes the Samsung captivate phone. Though II1, II2 and II4 are assigned the same color, II4 was created a long duration after the creation of II1 and II2, so there is a good possibility that user may have forgotten the color-interest class mapping. But II1, II2 and II3 are created within a short duration of one minute, so it is highly probable that the user has been consistent while creating these instances. As discussed earlier, the *Constraint Builder* module analyzes the actions in this scenario and creates *must-link* constraints between II1-II2, and creates *cannot-link* constraints between II1-II3, II2-II3 while nothing can be clearly inferred from the II4 instance. These constraints are forwarded to the clustering algorithm.

B. An Approach to Group Similar Interest Instances Based on User's Intent

As discussed earlier, the prior instantiation of the IPM required that the user maintains consistency in color-interest class assignment of all interest instances across multiple applications and across time. Inconsistent assignment of color to an interest class could result in drastically reducing the IPM's inferring efficiency as the representation of the interest classes would become noisy as multiple interests were merged or a single interest was repeated. In practice, there are many reasons for inconsistent color assignment. Users may misremember the color-interest class mapping; a user may forget the existence of an interest class and start similar interest class with a new color; two assigned colors may be visually very close and the user may be confused while choosing among them.

A system can help reduce these inconsistencies by making recommendations when the user is initially assigning colors to information items and by locating color-interest class mappings that are computationally ambiguous. The first type of recommendation was already provided by the visualization mechanisms based on the results of the IPM analysis although the recommendations could be improved by improving the recognition of the bounds of user interests. The second type of recommendation mechanism needs methods for grouping similar interest instances based on features other than purely color assignment.

In the current work, we are using variations of Agglomerative Hierarchical Clustering (AHC) to group similar interest instances into interest classes. Agglomerative Hierarchical clustering algorithms are a type of clustering algorithms in which successive clusters are found using previously established clusters. They create, from the bottom up, a dendrogram which is a tree structure containing a k-block set partition for each value of k between 1 and n. For example, given six patterns A, B, C, D, E, F represented as points in a two-dimensional space (shown in Fig. 9), it can be observed how an AHC clustering algorithm constructs dendrogram from these patterns, shown in Fig. 10. At each level in the dendrogram the two patterns which are most similar (closest in the two-dimensional space) are merged to form a single cluster. The Agglomerative Hierarchical Clustering approach best suited the requirements of our present problem and it was chosen over Partitional clustering approaches like K-Means, as the number of clusters 'k' (e.g. in our problem, the number of discrete user interest classes) is not available as input.

All the variations of HAC developed take as input, the interest instances which can be represented in n-dimensional feature space and, *must-linked* and *cannot-linked* constraints built from the *Constraint Builder* module (discussed in the previous section). Six versions of AHC algorithms are developed: (1) Unconstrained Complete-Link Agglomerative Hierarchical Clustering Algorithm, (2) Constrained Complete-Link Agglomerative Hierarchical Clustering Algorithm, (3) Unconstrained Single-Link Agglomerative Hierarchical Clustering Algorithm, (4) Constrained Single-

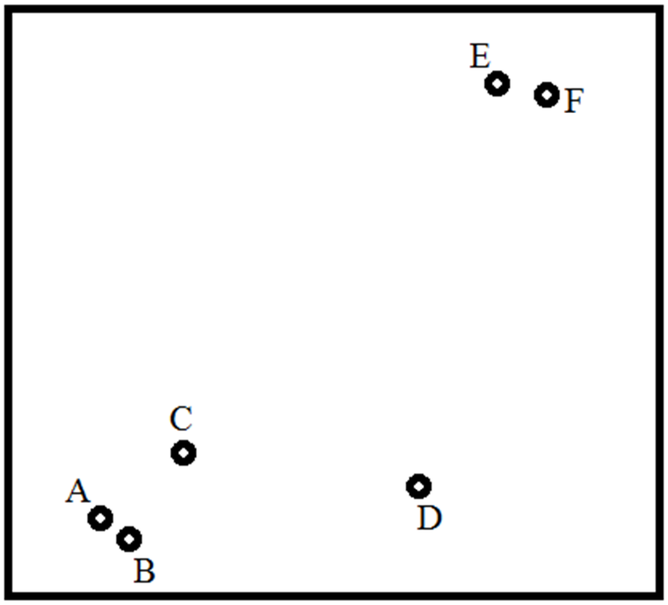


Fig. 9. Data Points in Two-Dimensional Space

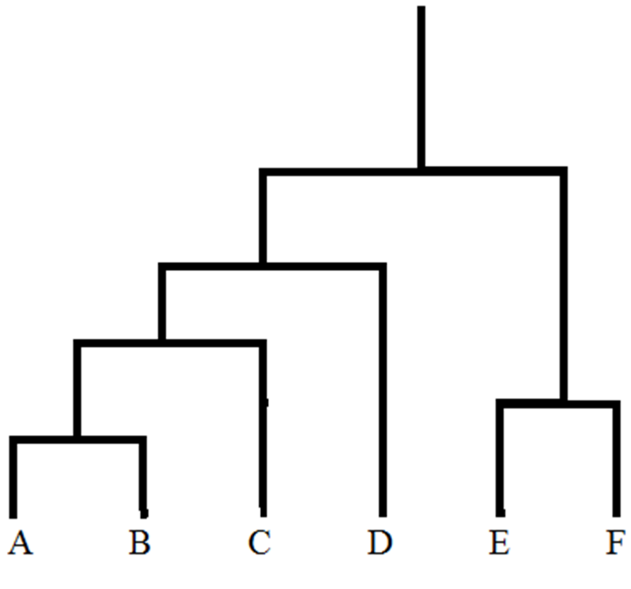


Fig. 10. Dendrogram Constructed using AHC Algorithm from Data Points in Figure 9

Link Agglomerative Hierarchical Clustering Algorithm, (5) Unconstrained UPGMA (Unweighted Pair Group Method with Arithmetic Mean) Agglomerative Hierarchical Clustering Algorithm, and (6) Constrained UPGMA Agglomerative Hierarchical Clustering Algorithm. All these algorithms have a preprocessing step that creates a proximity matrix which has the similarity score between any given two data points (indicating interest instances). Each of these algorithms is discussed below:

1. Unconstrained Complete-Link Agglomerative Hierarchical Clustering Algorithm

In the complete-link (CL) AHC algorithm, the similarity between two clusters is the similarity of their most dissimilar members, so at each step the two clusters whose merger has the smallest diameter are merged. No constraints are considered in this algorithm and, starting with the proximity matrix from the preprocessing step,the dendrogram is constructed using the CL approach at each step.

2. Constrained Complete-Link Agglomerative Hierarchical Clustering Algorithm

This approach uses the must-Linked and cannot-Linked constraints in deciding which clusters to merge during the construction of the dendrogram. In this constrained version of the above discussed algorithm, the proximity matrix from the preprocessing step is modified so that the specific items in the must-linked constraints are adjusted in feature space to be very close in distance and, the items in cannot-linked constraints are adjusted to be far apart in distance. These modifications to the proximity matrix cause the must-Linked pairs to be merged first and the cannot-Linked pairs to be merged in the dendrogram after all other mergings have occurred.

3. Unconstrained Single-Link Agglomerative Hierarchical Clustering Algorithm

In Single-link (SL) AHC algorithm, the similarity between two clusters is the similarity of their most similar members, so in each step the two clusters whose two closest members have the smallest distance are merged. No constraints are considered in this algorithm and, starting with the proximity matrix from the preprocessing step, the dendrogram is constructed using the SL approach at each step.

4. Constrained Single-Link Agglomerative Hierarchical Clustering Algorithm

This constrained version of the above discussed Single-Link AHC algorithm uses the must-Linked and cannot-Linked constraints as an input and modifies the proximity matrix from the preprocessing step so that the specific items in the must-linked constraints are adjusted in feature space to be very close in distance and, the items in cannot-linked constraints are adjusted to be far apart in distance.

5. Unconstrained UPGMA Agglomerative Hierarchical Clustering Algorithm

In the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) AHC algorithm [37], the similarity between two clusters A and B is the average of all distances between pairs of objects points 'x' in A and 'y' in B. No constraints are considered in this algorithm and starting with the proximity matrix from the preprocessing step the dendrogram is constructed using the UPGMA approach at each step.

6. Constrained UPGMA Agglomerative Hierarchical Clustering Algorithm

Similar to the earlier discussed constrained versions, this version of the above discussed UPGMA AHC algorithm uses the must-Linked and cannot-Linked constraints as an input and modifies the proximity matrix from the preprocessing step so that the specific items in the must-linked constraints are adjusted in feature space to be very close in distance and the items in cannot-linked constraints are adjusted to be far apart in distance.

In addition to exploring alternative clustering algorithms, the current work explores different approaches to generating the proximity matrix. The proximity matrix D in the preprocessing step is populated with the similarity scores between different data points (indicating interest instances). Three different factors that influence the similarity between two interest instances have been studied: (1) Content Similarity, (2) Temporal Similarity, and (3) Visual Similarity.

Content Similarity Content similarity is a measure of how much the textual content of the two interest instances overlap. There are many methods for measuring overlap varying from overlap in sentences, phrases, words, and concepts. In the current work, the content similarity is calculated by applying cosine similarity to the term vectors corresponding to two nodes.

Temporal Similarity Temporal similarity is a measure of how close different actions by the user are in time. As already discussed, if two actions are separated by very small time period it is more probable that the user is consistent with his/her actions during this period.

Visual Similarity Visual similarity is a measure of the visual distance between user actions.

The following experiments explored each of the six different clustering algorithms with different permutations of the three similarity components for computing the proximity matrix.

CHAPTER VI

SYSTEM DESCRIPTION

Fig. 11 shows the high level block diagram of the modified IPM architecture presented in the current work. Two new modules, Constraint Builder and Constrained Clustering, are developed to support the clustering of similar interest instances based on their visual, temporal and content characteristics. IPM clients for MS Word and PowerPoint have been developed in addition to the existing IPM clients for VKB3 and Firefox.

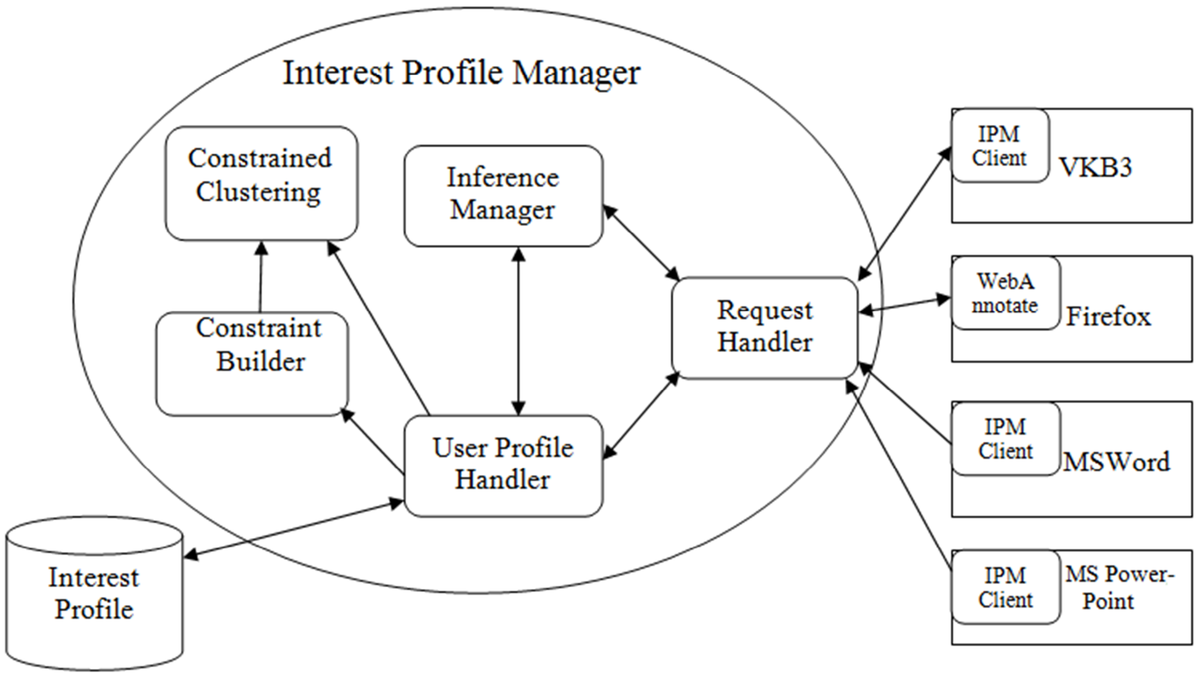


Fig. 11. Modified IPM Architecture for Clustering

A. Constraint Builder

The Constraint Builder module is a preprocessing step that analyzes user actions from the profile data supplied by User Profile Handler module and identifies the *must-linked* and *cannot-linked* constraints between the interest instances. These are used as input for the Clustering algorithm to satisfy. As discussed in the Approach section, the constraints are derived from the sequences of actions for which the user's actions can be expected to be consistent. Heuristics are used to identify these sequences of actions.

```
identifyConstraints (InterestInstances II, TimeInterval $\Delta T$)
    Constraints $C_{must}$ starts empty
    Constraints $C_{cannot}$ starts empty
    $TI(II_i) \leftarrow$ TimeInterval of $\Delta T$ to which interest instance $II_i$ belongs
    for $i \leftarrow 1$ to size(II)-1
        for $j \leftarrow (i+1)$ to size(II)
            if $TI(II_i) = TI(II_j)$ AND $AppName(II_i) = AppName(II_j)$
                if $Color(II_i) = Color(II_j)$)
                    add $(II_i, II_i)$ to $C_{must}$
                if $Color(II_i) != Color(II_j)$)
                    add $(II_i, II_i)$ to $C_{cannot}$
```

Fig. 12. Pseudo Code for Constraint Builder

Fig. 12 shows the pseudo code of how Must-Link and Cannot-Link constraints are identified. The timeline of user's triage activity is divided into fixed time intervals of length ΔT. ΔT is chosen small enough so that the users are more likely to be consistent in their interpretive actions during this interval. Must-Link constraints are identified by finding the instances from same application with same color and in the same time interval. Cannot-Link constraints are identified by finding the instances which are from same application in the same time interval but with different color.

B. Constrained Clustering

The Constrained Clustering module takes interest instances and the constraints generated by Constraint-Builder module as input and generate clusters by grouping the similar interest instances based on the content, temporal and visual characteristics of the user's activity. Six versions of the Agglomerative Hierarchical Clustering (AHC) are used for clustering the interest instances. Each of these algorithms have a common preprocessing step that computes the proximity matrix containing the similarity scores between each pair of data points (representing interest instances) considered for clustering.

Fig. 13 shows the pseudo code for three functions that are preprocessing steps before clustering. *buildProximityMatrix* functionality is used for building the proximity matrix S given the interest instances. The similarity scores reflect the distances between data points in the virtual n-Dimensional feature space. Different features considered and approaches followed for calculating the similarity score between interest instances is discussed in detail in the latter sections.

Fig. 13 also has the pseudo code for *imposeMustLinkConstraints* and *imposeCannotLinkConstraints* functionalities which are used only by the constrained versions of AHC algorithms. Must-Link constraints are imposed differently from Cannot-Link constraints. In the *imposeMustLinkConstraints* all the pairs of interest instances involved in a Must-Link constraint are merged prior to calling the clustering algorithm while the *imposeCannotLinkConstraints* imposes the *Cannot-Link* constraints on the proximity matrix by increasing the distance (i.e, by decreasing the similarity score) between two cannot-link points. The $VisCannotLinkFactor_{i,j}$ is less than 1 and is computed based on how visually close the instances i and j are. While the two instances must involve distinct visual characteristics to be identified as Cannotlink constraints, in practice the two visual assignments might be similar. For very distinct visual characteristics of instances i and j $VisCannotLinkFactor_{i,j}$ is zero making the similarity score zero and for very visually similar characteristics it is closer to 1 decreasing the similarity score only by a small. The $VisCannotLinkFactor$ is introduced to adjust the similarity scores in the feature space depending on the likelihood of *Cannot-Link* constraint being due to user's real intent (visually very different) or it being due to user's color assignment signifying similarity (visually close).

```
buildProximityMatrix (InterestInstances II)
    for i ← 1 to |II|-1
        for j ← (i+1) to |II|
            $S_{i,j}$ ← getSimilarityScore ($II_i$, $II_j$)

imposeMustLinkConstraints (InterestInstances II, Constraints C)
    for (i, j) ε $C_{must}$
        merge instances $II_i$ and $II_j$

imposeCannotLinkConstraints (Matrix S, Constraints C)
    for (i, j) ε $C_{cannot}$
        $S_{i,j}$ ← $VisCannotLinkFactor_{i,j} * S_{i,j}$
```

Fig. 13. Preprocessing Steps Before Clustering

Six versions of the AHC algorithm are developed in the current work: (1) Unconstrained Complete-Link Agglomerative Hierarchical Clustering Algorithm, (2) Constrained Complete-Link Agglomerative Hierarchical Clustering Algorithm, (3) Unconstrained Single-Link Agglomerative Hierarchical Clustering Algorithm, (4) Constrained Single-Link Agglomerative Hierarchical Clustering Algorithm, (5) Unconstrained UPGMA Agglomerative Hierarchical Clustering Algorithm, and (6) Constrained UPGMA Agglomerative Hierarchical Clustering Algorithm.

1. Unconstrained Complete-Link Agglomerative Hierarchical Clustering Algorithm

Fig. 14 shows the pseudo code for the Unconstrained Complete-Link Agglomerative Hierarchical Clustering. It takes Interest Instances II from User Profile Handler as the only input and does not run the Constraint Builder module. It executes *build-ProximityMatrix* functionality as a preprocessing step to build the proximity matrix with similarity scores for all pairs of data points and then implements the standard Complete-Link (CL) AHC algorithm. CL algorithm merges clusters in order of proximity; the closest clusters will be merged first and the farthest clusters will be merged last. At each merge, CL creates a reduced proximity matrix, with one less row and column, and updates the similarity of other clusters with the newly merged cluster by their similarity with the most dissimilar member of the merged cluster. No constraints are considered in this algorithm.

2. Constrained Complete-Link Agglomerative Hierarchical Clustering Algorithm

Fig. 15 shows the pseudo code for the Constrained Compete-Link Agglomerative Hierarchical Clustering. The underlying CL algorithm is similar to the unconstrained version except that there are preprocessing steps involving constraints in this version. This algorithm uses an additional input Constraints C (containing Must-Link and Cannot-Link constraints) supplied by the Constraint Builder module. Preprocessing steps: imposing Must-Link constraints, building proximity matrix with similarity scores between each pair of (modified) interest instances, and imposing Cannot-Link constraints on the proximity matrix are executed in sequence. The iterative part of the CL algorithm is the same as in the unconstrained version.

```
UnconstrainedCompleteLinkAHC (InterestInstances II)
        Proximity Matrix S ← buildProximityMatrix (II)
        Clusters = {$c_i$ for each point i}
        Linkage starts empty
        Similarity Scores $\delta(c_i, c_j) = S_{ij}$
        While |Clusters| > 1
                choose closest $(c_1, c_2) = \arg\max_{c1,c2\ \varepsilon\ Clusters} \delta(c_1,c_2)$
                add $(c_1, c_2)$ to Linkage
                merge $c_1$ and $c_2$ into $c_{new}$ in Clusters
               for $c_i$ ε Clusters
                        $\delta(c_i, c_{new}) = \min\{\delta(c_i, c_1), \delta(c_i, c_2)\}$
```

Fig. 14. Pseudo Code for Unconstrained Complete-Link Agglomerative Hierarchical Clustering

```
C ← identifyConstraints (II, ΔT)

ConstrainedCompleteLinkAHC (InterestInstances II, Constraints C)
        imposeMustLinkConstraints (II, C)
        Proximity Matrix S ← buildProximityMatrix (II)
        imposeCannotLinkConstraints (S, C)
        Clusters = {$c_i$ for each point i}
        Linkage starts empty
        Similarity Scores $\delta(c_i, c_j) = S_{ij}$
        While |Clusters| > 1
                choose closest $(c_1, c_2) = \arg\max_{c1,c2\ \varepsilon\ Clusters} \delta(c_1,c_2)$
                add $(c_1, c_2)$ to Linkage
                merge $c_1$ and $c_2$ into $c_{new}$ in Clusters
               for $c_i$ ε Clusters
                        $\delta(c_i, c_{new}) = \min\{\delta(c_i, c_1), \delta(c_i, c_2)\}$
```

Fig. 15. Pseudo Code for Constrained Complete-Link Agglomerative Hierarchical Clustering

3. Unconstrained Single-Link Agglomerative Hierarchical Clustering Algorithm

Fig. 16 shows the pseudo code for the Unconstrained Single-Link Agglomerative Hierarchical Clustering. Similar to the unconstrained version of Complete-Link AHC it takes Interest Instances II from User Profile Handler as the only input and does not run Constraint Builder module. It executes *buildProximityMatrix* functionality as a preprocessing step to build the proximity matrix with similarity scores for all pairs of data points and then implements the standard Single-Link (SL) AHC algorithm. The SL algorithm merges clusters in order of proximity; the closest clusters will be merged first and the farthest clusters will be merged last. At each merge, SL creates a reduced proximity matrix, with one less row and column. It updates the similarity of other clusters with the newly merged cluster by their similarity with the most similar member of the merged cluster. In other words, in each step the two clusters whose two closest members have the smallest distance are merged. No constraints are considered in this algorithm.

4. Constrained Single-Link Agglomerative Hierarchical Clustering Algorithm

Fig. 17 shows the pseudo code for the Constrained Single-Link Agglomerative Hierarchical Clustering. The underlying SL algorithm is similar to the unconstrained version above except that it executes preprocessing steps involving constraints. This algorithm uses an additional input, Constraints C (containing Must-Link and Cannot-Link constraints) supplied by the Constraint Builder module. Preprocessing steps: imposing Must-Link constraints, building proximity matrix with similarity scores between each pair of (modified) interest instances, and imposing Cannot-Link constraints on the proximity matrix are executed in sequence. The iterative part of the SL algorithm is the same as in unconstrained version.

```
**UnconstrainedSingleLinkAHC (InterestInstances II)**
    Proximity Matrix S ← buildProximityMatrix (II)
    Clusters = {$c_i$ for each point i}
    Linkage starts empty
    Similarity Scores $\delta(c_i, c_j) = S_{ij}$
    While |Clusters| > 1
        choose closest $(c_1, c_2) = \arg\max_{c1,c2 \, \varepsilon \, Clusters} \delta(c_1,c_2)$
        add $(c_1, c_2)$ to Linkage
        merge $c_1$ and $c_2$ into $c_{new}$ in Clusters
        for $c_i$ ε Clusters
            $\delta(c_i, c_{new}) = \max\{\delta(c_i, c_1), \delta(c_i, c_2)\}$
```

Fig. 16. Pseudo Code for Unconstrained Single-Link Agglomerative Hierarchical Clustering

```
C ← identifyConstraints (II, ΔT)

**ConstrainedSingleLinkAHC (InterestInstances II, Constraints C)**
    imposeMustLinkConstraints (II, C)
    Proximity Matrix S ← buildProximityMatrix (II)
    imposeCannotLinkConstraints (S, C)
    Clusters = {$c_i$ for each point i}
    Linkage starts empty
    Similarity Scores $\delta(c_i, c_j) = S_{ij}$
    While |Clusters| > 1
        choose closest $(c_1, c_2) = \arg\max_{c1,c2 \, \varepsilon \, Clusters} \delta(c_1,c_2)$
        add $(c_1, c_2)$ to Linkage
        merge $c_1$ and $c_2$ into $c_{new}$ in Clusters
        for $c_i$ ε Clusters
            $\delta(c_i, c_{new}) = \max\{\delta(c_i, c_1), \delta(c_i, c_2)\}$
```

Fig. 17. Pseudo Code for Constrained Single-Link Agglomerative Hierarchical Clustering

5. Unconstrained UPGMA Agglomerative Hierarchical Clustering Algorithm

Fig. 18 shows the pseudo code for the Unconstrained UPGMA Agglomerative Hierarchical Clustering. It takes Interest Instances II from User Profile Handler as the only input and does not run the Constraint Builder module. It executes *buildProximityMatrix* functionality as a preprocessing step to build the proximity matrix with similarity scores for all pairs of data points and then implements the standard UPGMA AHC algorithm [37]. UPGMA algorithm merges clusters in order of proximity; the closest clusters will be merged first and the farthest clusters will be merged last. At each merge, UPGMA creates a reduced proximity matrix, with one less row and column. It updates the similarity of other clusters with the newly merged cluster to be the mean distance between elements of each cluster. No constraints are considered in this algorithm.

6. Constrained UPGMA Agglomerative Hierarchical Clustering Algorithm

Fig. 19 shows the pseudo code for the Constrained UPGMA Agglomerative Hierarchical Clustering. The underlying UPGMA algorithm is similar to the unconstrained version except that there are preprocessing steps involving constraints in this version. This algorithm uses an additional input, Constraints C (containing Must-Link and Cannot-Link constraints) supplied by the Constraint Builder module. Preprocessing steps: imposing Must-Link constraints, building proximity matrix with similarity scores between each pair of (modified) interest instances, and imposing Cannot-Link constraints on the proximity matrix are executed in sequence. The iterative part of the UPGMA algorithm is the same as in the unconstrained version.

The proximity matrix S is populated with the similarity scores between each pair of data points in the preprocessing step *buildProximityMatrix* (in Fig. 13). *getSim-*

```
UnconstrainedUPGMA-AHC (InterestInstances II)
    Proximity Matrix S ← buildProximityMatrix (II)
    Clusters = {$c_i$ for each point i}
    Linkage starts empty
    Similarity Scores $\delta(c_i, c_j) = S_{ij}$
    While |Clusters| > 1
        choose closest $(c_1, c_2) = \arg\max_{c1,c2\ \varepsilon\ Clusters} \delta(c_1,c_2)$
        add $(c_1, c_2)$ to Linkage
        merge $c_1$ and $c_2$ into $c_{new}$ in Clusters
        for $c_i\ \varepsilon$ Clusters
            $\delta(c_i, c_{new}) = \frac{|c1|}{|c1|+|c2|} * \delta(c_i, c_1) + \frac{|c2|}{|c1|+|c2|} * \delta(c_i, c_2)$
```

Fig. 18. Pseudo Code for Unconstrained UPGMA Agglomerative Hierarchical Clustering

```
C ← identifyConstraints (II, $\Delta T$)

ConstrainedUPGMA-AHC (InterestInstances II, Constraints C)
    imposeMustLinkConstraints (II, C)
    Proximity Matrix S ← buildProximityMatrix (II)
    imposeCannotLinkConstraints (S, C)
    Clusters = {$c_i$ for each point i}
    Linkage starts empty
    Similarity Scores $\delta(c_i, c_j) = S_{ij}$
    While |Clusters| > 1
        choose closest $(c_1, c_2) = \arg\max_{c1,c2\ \varepsilon\ Clusters} \delta(c_1,c_2)$
        add $(c_1, c_2)$ to Linkage
        merge $c_1$ and $c_2$ into $c_{new}$ in Clusters
        for $c_i\ \varepsilon$ Clusters
            $\delta(c_i, c_{new}) = \frac{|c1|}{|c1|+|c2|} * \delta(c_i, c_1) + \frac{|c2|}{|c1|+|c2|} * \delta(c_i, c_2)$
```

Fig. 19. Pseudo code for Constrained UPGMA Agglomerative Hierarchical Clustering

ilarityScore functionality is used to get the similarity score between any two given instances. The similarity scores between interest instances dictate the distribution of data points representing these instances in the feature space. The higher the similarity score the more closely the instances are located in the feature space and the lower the similarity score indicates that the instances are separated by a larger distance. The methods used to calculate this similarity score play vital role in deciding the final distribution of instances in clusters and thereby the clustering effectiveness. The following discussion talks in detail about the factors considered and approaches investigated in calculating the similarity score between the interest instances.

There are a variety of factors that could influence the similarity between two interest instances but in the current work studied the contributions of three specific factors collected during the user's triage activity: (1) Content Similarity, (2) Temporal Similarity, and (3) Visual Similarity.

Content Similarity Content Similarity is a measure of how much the textual content of the two interest instances overlap. In the current work, content similarity is computed by applying cosine similarity to the two term vectors corresponding to two instances.

Each document d is represented in the vector-space model, in which d is considered to be a vector in the term-space. The magnitude of the vector in each dimension is the term-frequency (TF) in the document multiplied by the term's inverse document frequency (IDF) in the document collection. The motivation behind using IDF is that terms appearing frequently in many documents have limited discrimination power, and for this reason they need to be de-emphasized. This is done by multiplying the frequency of each term i by $log(N/df_i)$, where N is the total number of documents in the collection, and

df_i is the document frequency. So a document is represented in its tf-idf vector form as $d_{tf-idf} = (tf_1 * log(N/df_1), tf_2 * log(N/df_2), ..., tf_m * log(N/df_m))$. The content similarity between two documents d_i and d_j is given by the cosine product on the two tf-idf vectors for the documents.

$$\cos(d_i, d_j) = \frac{d_i.d_j}{|d_i||d_j|}$$

Temporal Similarity Temporal Similarity is a measure of how close different actions by the user are in time. If two actions are separated by a very small time interval it is more probable that the user is consistent with his/her actions during this period. The likelihood of user's consistency decreases drastically over time. So, we chose a time decaying function as the temporal similarity between two interest instances.

$$TemporalSimilarity(\overrightarrow{v1}, \overrightarrow{v2}) = \frac{1}{e^{\lambda \Delta t}}$$

$$where, \Delta t = |TimeStamp(\overrightarrow{v1}) - TimeStamp(\overrightarrow{v2})|$$

λ is appropriately chosen for the decaying function to have the right gradient.

Visual Similarity Visual similarity is a measure of the visual distance between user actions. In the current work, color is the only visual characteristic that is modified by the user in an interest instance. Color can be represented in the three-dimensional visual space with Red, Green and Blue being the three dimensions. The visual similarity between two interest instances is calculated based on how the color vectors corresponding to these instances are distributed in the visual space. In our current work, visual similarity is non-zero only if color vectors are very closely placed in the visual space otherwise it is zero.

We investigated four versions of similarity score calculation with each of the six AHC algorithms to study the contributions of each of these similarity factors while clustering user interest instances. The first three versions include using only Content Similarity, only Temporal Similarity and only Visual Similarity as the final similarity score between interest instances. The fourth version takes all the three factors into consideration and uses the below formula as the final similarity score between two instance vectors:

$$\begin{split}SimilarityScore(\overrightarrow{v1},\overrightarrow{v2}) = & \alpha * ContentSimilarity(\overrightarrow{v1},\overrightarrow{v2}) + \\ & \beta * TemporalSimilarity(\overrightarrow{v1},\overrightarrow{v2}) * VisualSimilarity(\overrightarrow{v1},\overrightarrow{v2})\end{split} \tag{6.1}$$

Visual and Temporal similarity factors are interrelated and contribute together rather than independently. Table I discusses the relationship between these two factors and how they together influence the system's ability to infer the user's intent.

The consistency of visual characteristics assigned by the user decreases with the increasing time difference and to reflect this, the Visual Similarity component is multiplied with the Temporal Similarity which is a decaying function.

Table I. Relationship Between Visual and Temporal Factors in Inferring User's Intent

		Time	
		High Difference	Low Difference
Visual Characteristics	High Difference	Due to the high time difference, it cannot be inferred that the instances belong to different interest class.	It is most likely that the user's intent is to assign interest instances to different interest classes.
	Low Difference	Due to the high Time difference, it cannot be inferred if the instances belong to the same interest class.	It is most likely that the user's intent is to assign the interest instances to same interest class.

CHAPTER VII

EVALUATION

A study was performed to evaluate the effectiveness of the clustering techniques presented in the prior chapter and the contribution of the different factors in computing the similarity for grouping the triage interest instances of a user. The evaluation focuses on the clustering algorithms' performance in predicting groups of similar interest instances, and the contributions of content similarity, temporal similarity and visual similarity on the overall similarity of interest instances.

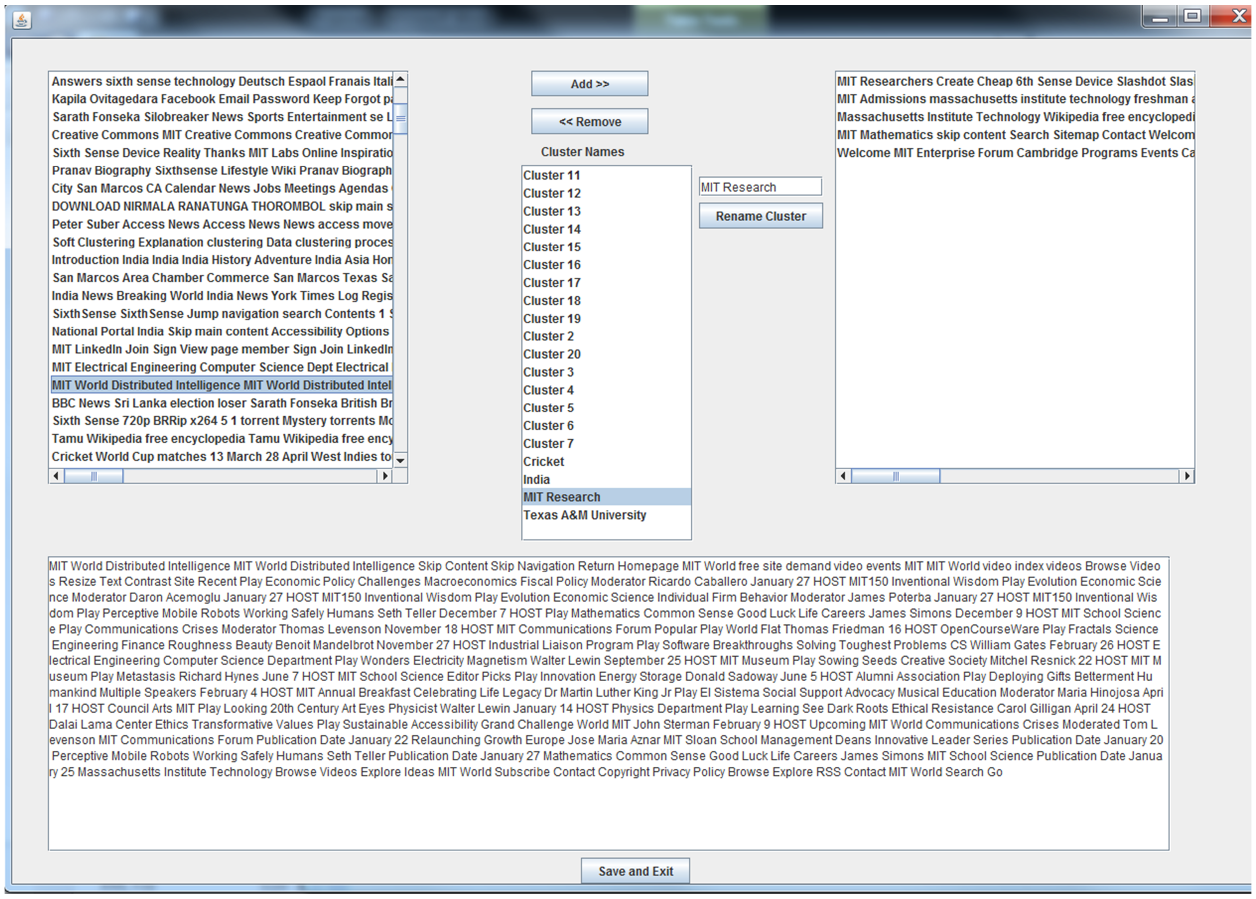
A. Experimental Design

The study was conducted on seven participants, 5 graduate students and 2 working professionals ranging in age from 23 to 31. They use computers regularly and are familiar with internet searching and browsing. The participants were asked to perform the evaluation task on their local machines for their convenience as it required them to work in multiple sessions over an extended time period. The evaluation study contained two tasks, the User Triage task and the Manual Grouping task and each user had to complete both to finish the evaluation.

In the User Triage activity task, each participant is provided with an installer to setup the IPM environment on their local machine. The environment included the central IPM system, VKB3, the WebAnnotate extension for the Firefox, and the Add-ins for Microsoft Word and PowerPoint. Each participant was asked to pick a topic of their choice and do research on using VKB3, Mozilla, Word and PowerPoint applications. To successfully complete the first task, a participant had to complete more than 250 IPM events across least 4 different sessions (on different days) and generating minimum of 10 events in each of the four IPM client applications. They were instructed to browse through at least 10 different interest classes and were given 2 weeks time to complete this task. For this evaluation, the interpretive actions by the user considered IPM events were the assigning of a color to WebDocument in VKB3, highlighting of a segment or a writing note in Mozilla Firefox, composing or editing text in Word or PowerPoint.

In the Manual Grouping task, all the interest instances created during the first task are shown together without the participant-assigned visual characteristics. The participants were then asked to manually cluster them. This cluster distribution indicates how the user would ideally group his/her interest instances and is used as the ground truth against which the results from different algorithms are compared. We have developed a simple UI tool to facilitate the user in performing this manual clustering task. The snapshot of the application is shown in Fig. 20. The left JList has all the unassigned interest instances, the middle JList has the clusters and the right JList has the assigned instances corresponding to the highlighted cluster. In the second task, the user had to assign all the instances to one of the clusters by emptying the left list. 'Add' button is used to assign (move) an instance to a cluster (right list). The text area below displays the full content of the selected instance to help the user decide in which cluster it belongs.

While the users are performing the Document Triage task, , all the actions establishing an interest instance were logged along with the characteristics of the document or segment involved, the action's temporal data and the visual characteristics assigned for that instance. In the Manual Grouping task, the final state of the manual distribution of instances to various clusters is saved. Each participant was asked to send the two XML files corresponding to each task, IPM_Profile.xml and Cluster_Profile.xml, containing their activity during the evaluation. The clustering algorithms discussed use these XML files as input and construct the interest instances.



```
Answers sixth sense technology Deutsch Espaol Franais Itali
Kapila Ovitagedara Facebook Email Password Keep Forgot pa
Sarath Fonseka Silobreaker News Sports Entertainment se L
Creative Commons MIT Creative Commons Creative Commor
Sixth Sense Device Reality Thanks MIT Labs Online Inspiratio
Pranav Biography Sixthsense Lifestyle Wiki Pranav Biograph
City San Marcos CA Calendar News Jobs Meetings Agendas
DOWNLOAD NIRMALA RANATUNGA THOROMBOL skip main s
Peter Suber Access News Access News News access move
Soft Clustering Explanation clustering Data clustering proces
Introduction India India India History Adventure India Asia Hor
San Marcos Area Chamber Commerce San Marcos Texas Sa
India News Breaking World India News York Times Log Regis
SixthSense SixthSense Jump navigation search Contents 1
National Portal India Skip main content Accessibility Options
MIT LinkedIn Join Sign View page member Sign Join LinkedIn
MIT Electrical Engineering Computer Science Dept Electrical
MIT World Distributed Intelligence MIT World Distributed Intel
BBC News Sri Lanka election loser Sarath Fonseka British Br
Sixth Sense 720p BRRip x264 5 1 torrent Mystery torrents Mo
Tamu Wikipedia free encyclopedia Tamu Wikipedia free ency
Cricket World Cup matches 13 March 28 April West Indies to
```

```
Cluster 11
Cluster 12
Cluster 13
Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 2
Cluster 20
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cricket
India
MIT Research
Texas A&M University
```

```
MIT Researchers Create Cheap 6th Sense Device Slashdot Slas
MIT Admissions massachusetts institute technology freshman a
Massachusetts Institute Technology Wikipedia free encyclopedi
MIT Mathematics skip content Search Sitemap Contact Welcom
Welcome MIT Enterprise Forum Cambridge Programs Events Ca
```

```
MIT World Distributed Intelligence MIT World Distributed Intelligence Skip Content Skip Navigation Return Homepage MIT World free site demand video events MIT MIT World video index videos Browse Video
s Resize Text Contrast Site Recent Play Economic Policy Challenges Macroeconomics Fiscal Policy Moderator Ricardo Caballero January 27 HOST MIT150 Inventional Wisdom Play Evolution Economic Scie
nce Moderator Daron Acemoglu January 27 HOST MIT150 Inventional Wisdom Play Evolution Economic Science Individual Firm Behavior Moderator James Poterba January 27 HOST MIT150 Inventional Wis
dom Play Perceptive Mobile Robots Working Safely Humans Seth Teller December 7 HOST Play Mathematics Common Sense Good Luck Life Careers James Simons December 9 HOST MIT School Scienc
e Play Communications Crises Moderator Thomas Levenson November 18 HOST MIT Communications Forum Popular Play World Flat Thomas Friedman 16 HOST OpenCourseWare Play Fractals Science
 Engineering Finance Roughness Beauty Benoit Mandelbrot November 27 HOST Industrial Liaison Program Play Software Breakthroughs Solving Toughest Problems CS William Gates February 26 HOST E
lectrical Engineering Computer Science Department Play Wonders Electricity Magnetism Walter Lewin September 25 HOST MIT Museum Play Sowing Seeds Creative Society Mitchel Resnick 22 HOST MIT M
useum Play Metastasis Richard Hynes June 7 HOST MIT School Science Editor Picks Play Innovation Energy Storage Donald Sadoway June 5 HOST Alumni Association Play Deploying Gifts Betterment Hu
mankind Multiple Speakers February 4 HOST MIT Annual Breakfast Celebrating Life Legacy Dr Martin Luther King Jr Play El Sistema Social Support Advocacy Musical Education Moderator Maria Hinojosa Apri
l 17 HOST Council Arts MIT Play Looking 20th Century Art Eyes Physicist Walter Lewin January 14 HOST Physics Department Play Learning See Dark Roots Ethical Resistance Carol Gilligan April 24 HOST
Dalai Lama Center Ethics Transformative Values Play Sustainable Accessibility Grand Challenge World MIT John Sterman February 9 HOST Upcoming MIT World Communications Crises Moderated Tom L
evenson MIT Communications Forum Publication Date January 22 Relaunching Growth Europe Jose Maria Aznar MIT Sloan School Management Deans Innovative Leader Series Publication Date January 20
 Perceptive Mobile Robots Working Safely Humans Seth Teller Publication Date January 27 Mathematics Common Sense Good Luck Life Careers James Simons MIT School Science Publication Date Janua
ry 25 Massachusetts Institute Technology Browse Videos Explore Ideas MIT World Subscribe Contact Copyright Privacy Policy Browse Explore RSS Contact MIT World Search Go
```

Fig. 20. Snapshot of Manual Cluster UI Tool

The participants in the study went about the triage task in significantly different ways. The topics they selected widely varied from movies to technology-related topics like HCI and C++ox, to Texas A&M University and even to pets. On average, it took each participant between three and half and four hours of aggregated time over 2 weeks to complete the first task and forty five minutes to one hour to complete the second task. The number and distinct type of visual characteristics (color) used, the number of documents that had to be accessed to create 250+ IPM events, and the number of IPM events generated with each application varied between participants.

B. Analysis

The data analysis focused on evaluating three different aspects of grouping interest instances: (1) whether the user's intentions during the triage activity can be derived from his/her actions to improve the overall clustering i.e., by comparing Constrained vs. Unconstrained algorithms, (2) the performance of Complete Link, Single Link, and UPGMA clustering algorithms to group instances, and (3) the contributions of content similarity, temporal similarity and visual similarity on the overall similarity of interest instances. We use the above three evaluation goals to organize our results.

We computed the standard quality measures Precision, Recall and F-measure to compare the clustering results with ground truth. These measures view clustering as a series of decisions, one for each of the N*(N-1)/2 pairs of documents in the collection. A true positive (TP) decision assigns two similar documents to the same cluster; a true negative (TN) decision assigns two dissimilar documents to different clusters. Two types of errors can happen. A False Positive (FP) decision assigns two dissimilar documents to the same cluster. A False Negative (FN) decision assigns two similar documents to different clusters. The precision P, Recall R and F-measure F_β are calculated as shown below:

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \qquad F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

We have used the F_2 measure for all the comparisons during our evaluation as it penalizes false negatives more strongly than false positives.

The next subsection compares the results of the constrained and unconstrained versions of the clustering algorithms. This is followed by a comparison of the three approaches to clustering. Both of these sections assume a similarity metric that combines content similarity, temporal similarity, and visual similarity. The last section examines how similarity metrics including only one of these components compare to the similarity metric combining all three.

1. Constrained vs Unconstrained Clustering

One aim of this research was to identify sequences of actions that provide strong evidence of user's intent during the triage process and use those sequences as a feedback in generating the user models. In particular, these sequences were used to form constraints to enhance clustering. Constrained and Unconstrained versions of three AHC algorithms are studied where the constrained version uses Must-Link and Cannot-Link constraints derived from user actions while the unconstrained version does not consider them.

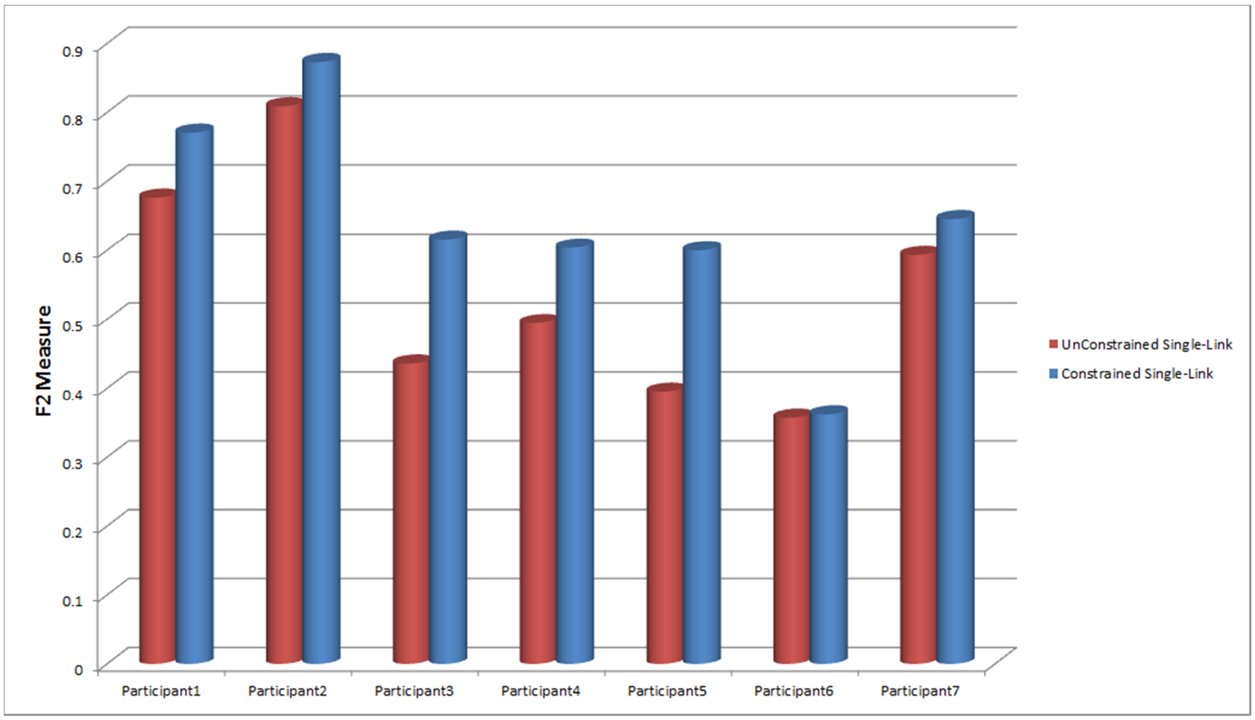


Fig. 21. Constrained vs UnConstrained Single-Link AHC

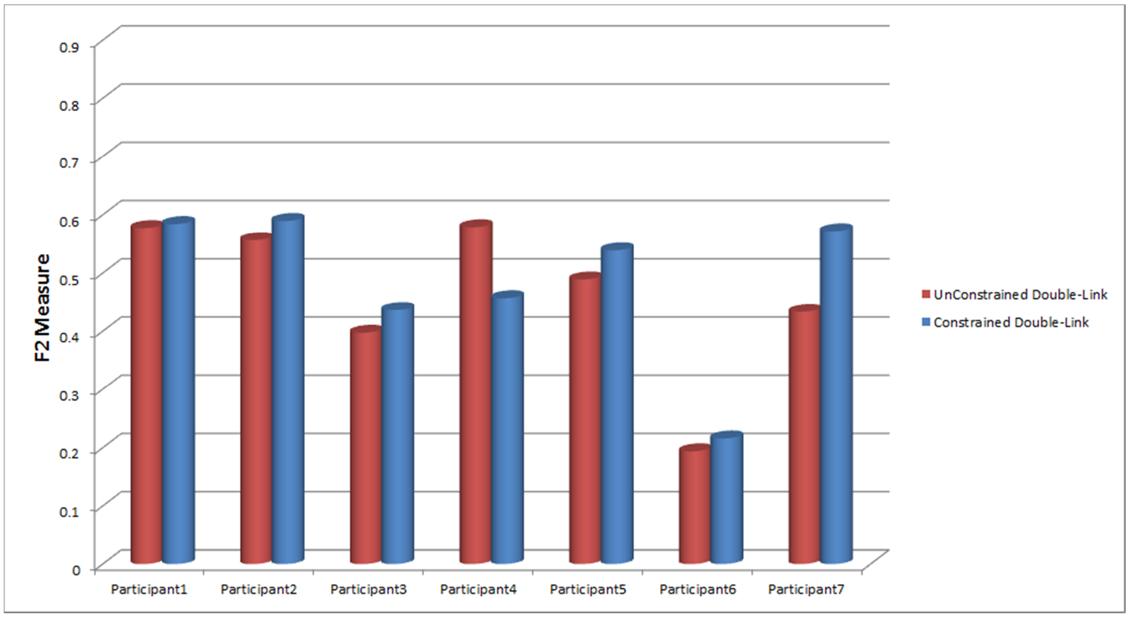


Fig. 22. Constrained vs UnConstrained Complete-Link AHC

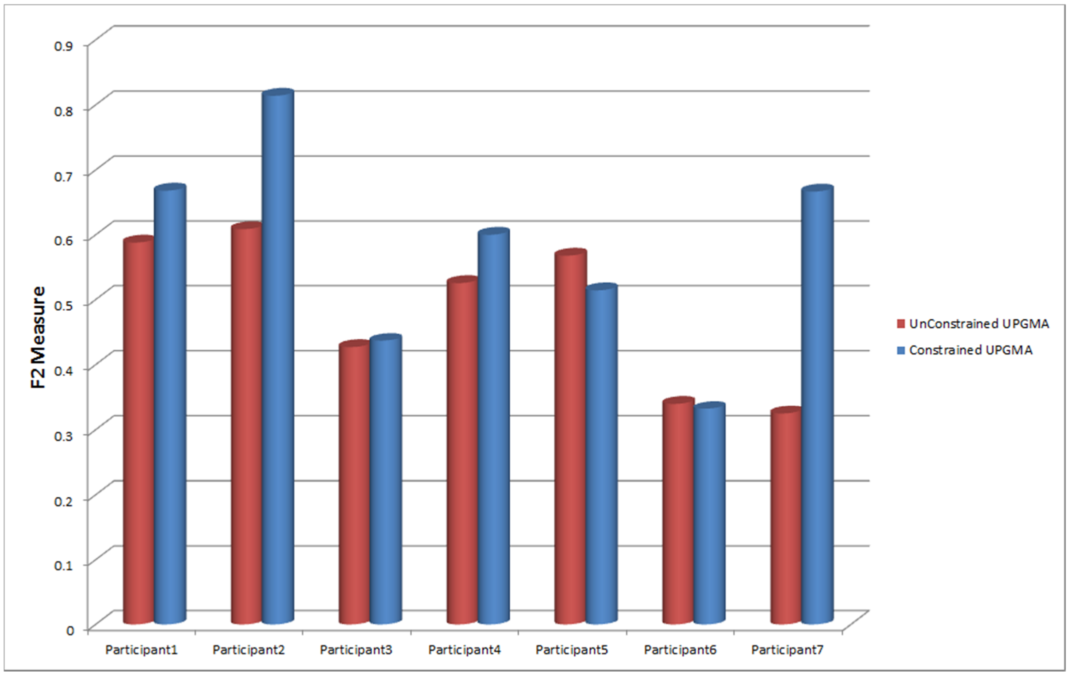


Fig. 23. Constrained vs UnConstrained UPGMA AHC

Fig. 21, 22 and 23 show the performance of Constrained vs Unconstrained algorithms for the Single-Link, Complete-Link and UPGMA AHC algorithms respectively. The X-axis shows the results from each of the seven participants and the average across the seven participants. Each individual is presented because personal work practices tend to be idiosyncratic and an average alone would hide the variance resulting from this. The F_2 measure of the results for each algorithm is plotted on the Y-axis. The constrained algorithm performs as well or better than the equivalent unconstrained algorithm in 18 of 21 cases (all except one dataset in Complete-Link and two datasets in the UPGMA case). This shows that considering constraints improves the performance in majority of the cases. Exceptions are due to the algorithm being ineffective (depending on the data) in propagating the constraints to adjust the proximities of other data points and due to the user's use of color being ambiguous even in short time intervals resulting in the creation of wrong constraints. But, these are rare instances and the results support the argument that including constraints helps improve the performance of a clustering algorithm. Fig. 24 also shows that on average the constrained version performs better than the unconstrained version in all three algorithms.

Table II. Paired T-test on Constrained vs Unconstrained Algorithms

FeaturesCompared	**P-Value**
Constrained vs Unconstrained	0.003603

These results show that some of the user's intentions during the triage activity can be effectively captured and represented as constraints by analyzing the sequences of his/her actions. These constraints prove to be a valuable asset in building the clusters of similar interest instances. Table II shows the P-Value from the paired T-tests on the results suggesting their statistical significance.

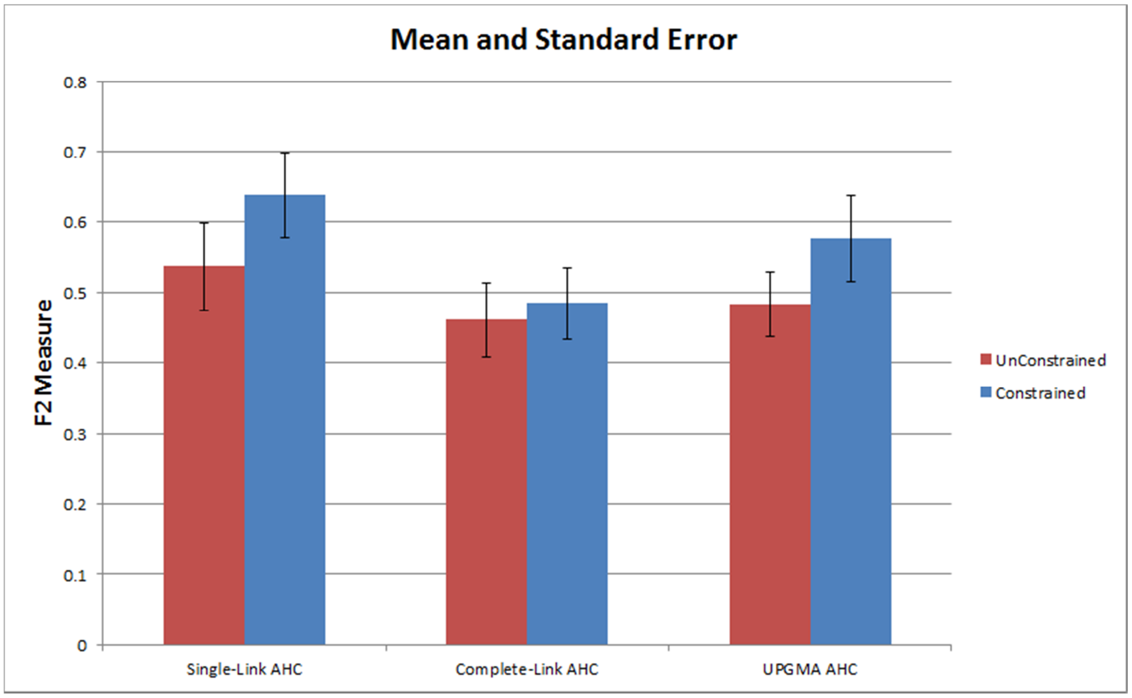


Fig. 24. Mean and Standard Error of Constrained and Unconstrained AHC Algorithms

2. Comparison of Clustering Algorithms

The prior section shows the value of including constraints when clustering interest instances. This section examines the performance of the three alternative clustering algorithms when including constraints. Fig. 25 shows the performance of the three algorithms with constraints. The blue bars show the F_2 measure from the results of Single-Link AHC algorithm on each dataset, the red bars correspond to Complete-Link AHC algorithm and the green bars correspond to UPGMA AHC algorithm. The Single-Link AHC algorithm outperforms the other two algorithms except in the case of

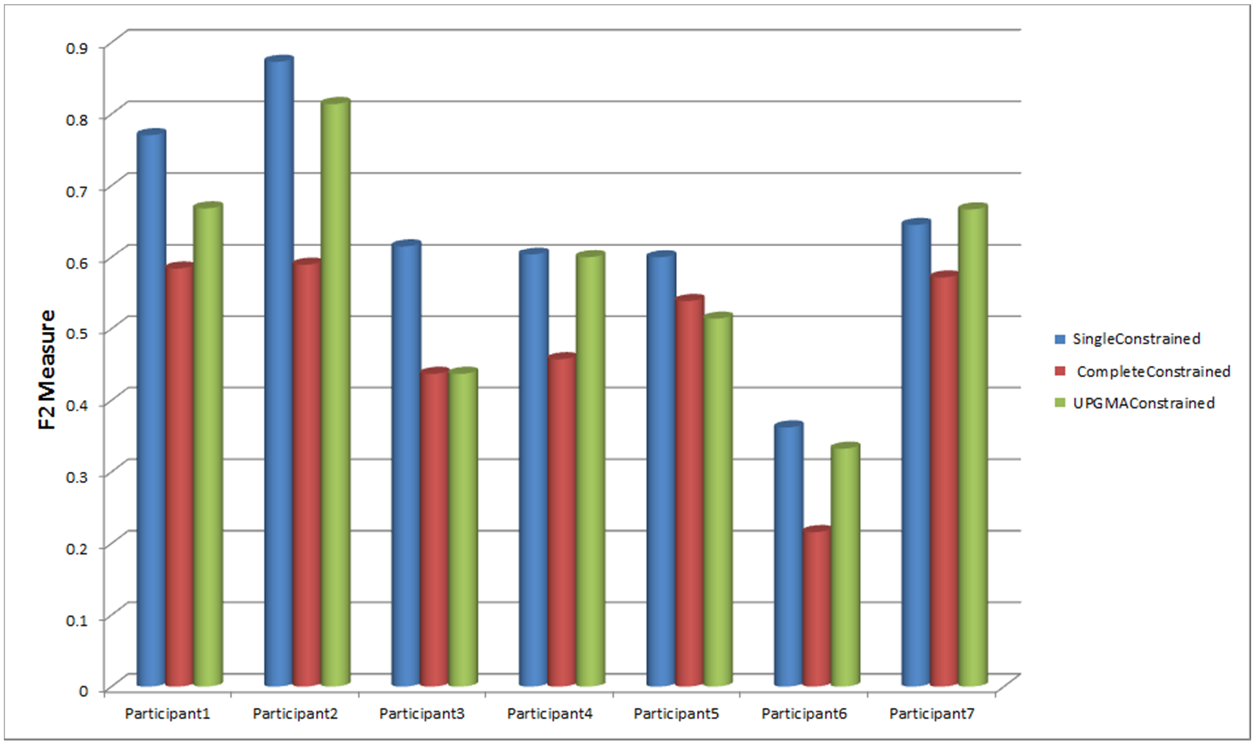


Fig. 25. Comparison of Agglomerative Hierarchical Clustering Algorithms

dataset from participant7 in which its performance is very close to the top performing UPGMA. Fig. 26 also shows that on average the Single-Link AHC performs better than the Complete-Link and UPGMA AHC algorithms. Table III shows the P-Value from the paired T-tests on the results suggesting their statistical significance.

The better performance by the Single-Link algorithm can be attributed to the characteristics of the triage instances and the fact that similarity between instances is often related to the context during which they are created by the user. We use temporal and visual characteristics to infer contextual relations. The differences created by these characteristics are valuable when comparing instances that are near to one another and to identify instances that are not near one another. Not much can be inferred about the relative contextual relations between instances when they

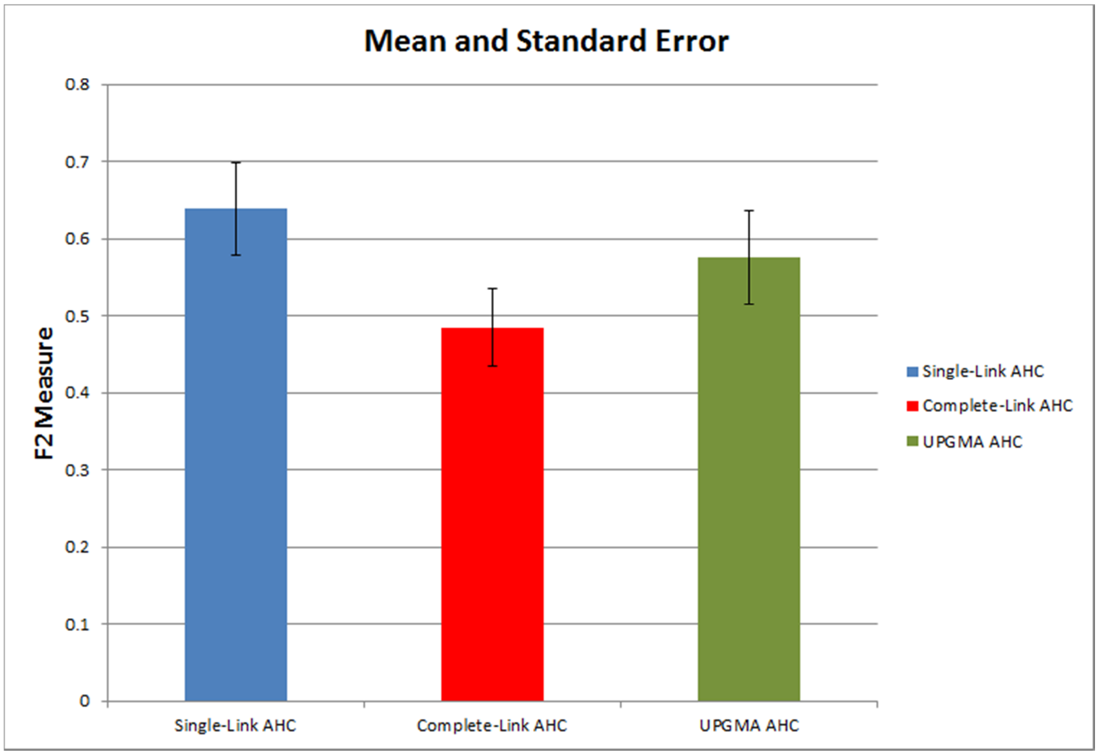


Fig. 26. Mean and Standard Error of Three AHC Algorithms

are separated by a large time interval. Each step of Single-Link AHC acts based on the distance between their two closest members, which is where the discriminatory power of the similarity metric is strongest. While the points in each cluster may be directly or indirectly related both in context and content, not much could be inferred many times about their similarity with multiple points in other clusters except by comparing their content. So, ignoring such comparisons and considering only the two closest members while deciding the similarity between clusters proves to be effective. In comparison, the Complete-Link algorithm, which bases its decision on the most distant members within the two clusters, and UPGMA, which takes the average of all pairwise distances, are less effective than Single-Link.

Table III. Paired T-test on Different AHC Algorithms

FeaturesCompared	**P-Value**
Single-Link vs Complete-Link	0.001619302
Single-Link vs UPGMA	0.049476008

3. Evaluation of Factors Contributing to the Similarity of Instances

The third goal of this research was to examine the potential factors contributing to the similarity estimation of interest instances during triage activity. In particular, we studied the contributions of three main factors: (1) Content Similarity, (2) Temporal Similarity, and (3) Visual Similarity. The performance of each was considered in isolation, by comparing only-Content, only-Visual, only-Temporal metrics with the combination of all three as similarity measures.

Fig. 27 shows the performance of four different versions of SLAHC algorithm with constraints. The only thing that varied in these four versions is the similarity measure calculation between any two given instances used during all steps of clustering. The first version, shown in red in the graph, uses only content similarity as the complete similarity measure between the interest instances. The second variation, shown in green, uses only the temporal similarity while the third variation, shown in violet, uses only the visual similarity. The fourth variation, shown in blue, uses a weighted aggregation of all these three similarities as shown below.

$$\begin{aligned} SimilarityScore(\overrightarrow{v1},\overrightarrow{v2}) =& \alpha * ContentSimilarity(\overrightarrow{v1},\overrightarrow{v2}) + \\ & \beta * TemporalSimilarity(\overrightarrow{v1},\overrightarrow{v2}) * VisualSimilarity(\overrightarrow{v1},\overrightarrow{v2}) \end{aligned} \tag{7.1}$$

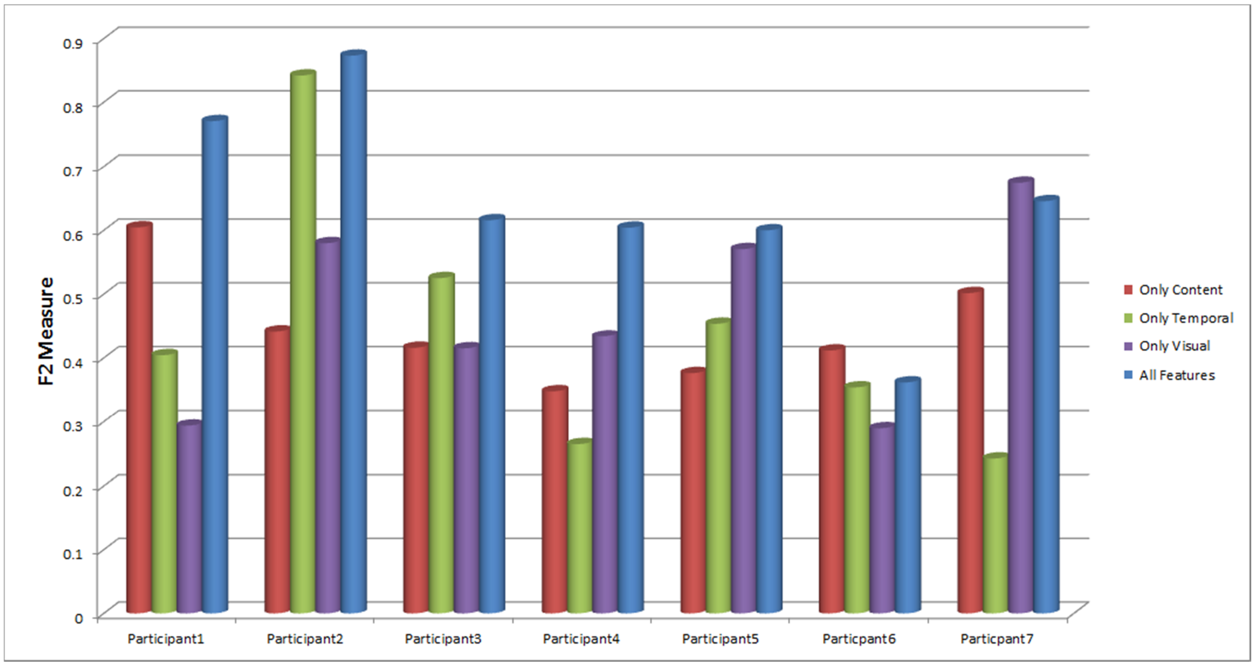


Fig. 27. Comparison of Similarity Contribution from Different Factors

Fig. 26 shows the mean and standard deviation of the four versions of these algorithms on the 7 datasets.

As expected, Fig. 27 shows that the participants in the studies went about the triage task in significantly different ways. A closer look helps understand each user's instance creating patterns, the kind of data selected for creating an instance, and their consistency in applying visual characteristics to instances. The graph indicates that participant7 to a greater extent and participant2 and participant5 to a lesser extent were consistent in using the visual characteristics as their only-Visual version of the algorithm performed better than it did for other participants. The good performance of only-Temporal version for Participant2 and Participant3 suggests that his/their browsing patterns and instance creation patterns were temporally concentrated which means all documents and segments corresponding to same interest class were generally accessed during the same time. The good performance of only-Content versions and bad performance of only-Temporal and only-Visual versions for Participant1 and Participant6 suggests that they were not consistent in their use of visual characteristics and did not temporally segment their access of documents based on interest class. In these cases, the text selected during the creation of instances provides the strongest evidence of the interest classes that they belong to.

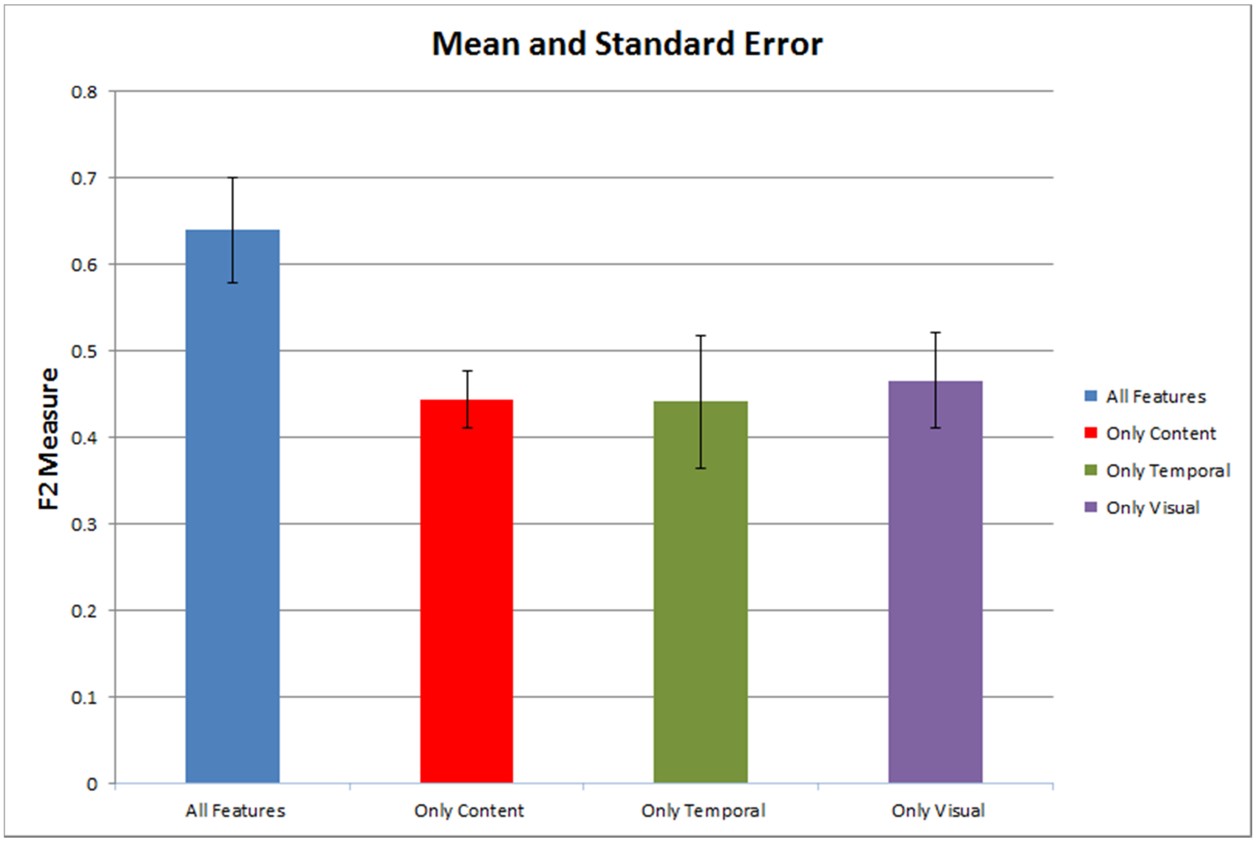


Fig. 28. Mean and Standard Error of SLAHC with Four Similarity Measures

None of the first three versions of algorithms using single factor could perform consistently better across all seven datasets. These results have asserted our claim that a single similarity measure cannot completely define the relationship between the instances. People have idiosyncratic work practices and a single measure will not work across these various practices. The results also show that the fourth metric, which is an aggregation of the three single-component similarity measures, performs consistently better in all the seven datasets. It performs best for the first five datasets and, in the cases of Participant6 and Participant7 where content and visual factors dominate, it stands second best.

Fig. 28 also shows that on average the combined-feature metric performs better than the other three algorithms. This indicates that across a population of users, this similarity measure can better define the relationship between the triage interest instances. Table IV shows the P-Value from the paired T-tests on the results suggesting their statistical significance.

Table IV. Paired T-test on Algorithms with Different Similarity Features

FeaturesCompared	**P-Value**
All Features vs Only-Content	0.011265902
All Features vs Only-Temporal	0.020553995
All Features vs Only-Visual	0.037797025

CHAPTER VIII

CONCLUSION AND FUTURE WORK

This thesis presents an approach to address issues faced by triage supporting systems trying to recognize distinct user interests based on user activity that arise due to the inconsistencies in user's interpretive actions. These inconsistencies if not taken care, affect the interest model inferring capability of the systems. A modified IR clustering technique customized to suit the needs of document triage was developed to group similar interest instances. Three factors were considered in calculating the similarity between the instances: (1) Content Similarity, (2) Temporal Similarity, and (3) Visual Similarity. We have also developed an approach that derives constraints on interest instance clustering from user behavior. The IPM infrastructure has also been extended with the development of IPM clients for MS Word and PowerPoint. IPM now can aggregate actions recorded from four client applications: VKB3, Mozilla and the newly added Word and PowerPoint.

We have evaluated the effectiveness of alternative clustering algorithms and found that the Single-Link AHC algorithm generally performs better than the Complete-Link and UPGMA algorithms in the context of clustering interest instances and the similarity measure combining all three similarity factors. Combining the three similarity factors defined well the relationship between interest instances even across the diverse work practices found among our evaluation participants. We also found that the use of constraints derived from user's intent improved the overall effectiveness of the clustering algorithms.

In the current work we used temporal, visual and content characteristics to estimate the similarity between two interest instances. This work can be extended to include more implicit indicators like the time spent on a document or segment while reading or editing; how much of the document they examine (e.g. how far into a document they scroll); the scrolling speed; how they categorize the document (e.g., stacking it with other interesting documents); and through other behaviors that in part rely on the tools they are using. Considering these indicators can give more information about the user behavior and his/her interests. Though, they are less likely to be useful on their own, combining them with the other existing factors may result in a more accurate prediction of the similarity between interest instances. These implicit indicators may also help better understand a user's intent from his/her browsing patterns which can be used to derive new constraints to satisfy by the clustering algorithm.

Implementing more clustering algorithms, both hierarchical and Partitional and analyzing their results will also help better understand the characteristics of the similarity relation between two triage interest instances in the feature space. The extension of IPM infrastructure to support more applications like Adobe Acrobat Reader, Chrome and Windows Explorer browsers as clients will also help IPM collect more user activity and build better interest models for the user.

REFERENCES

[1] R. Badi, S. Bae, J.M. Moore, K. Meintanis, A. Zacchi, H. Hsieh, F.M. Shipman, and C.C. Marshall, ”Recognizing user interest and document value from reading and organizing activities in document triage,” in. *Proc. IUI*, pp. 218-225, Jan. 2006.

[2] S. Bae, R. Badi, K. Meintanis, J.M. Moore, A. Zacchi, H. Hsieh, C.C. Marshall and F.M. Shipman, ”Effects of display configurations on document triage,” in *Proc. INTERACT*, pp. 130-143, Sep. 2005.

[3] S. Bae, D. Kim, K. Meintanis, J.M. Moore, A. Zacchi, F.M. Shipman, H. Hsieh and C.C.Marshall, ”Supporting document triage via annotation-based multi-application visualizations,” in *Proc. JCDL*, pp. 177-186, Jun. 2010.

[4] S. Basu, A. Banerjee and R.J. Mooney, ”Semi-supervised clustering by seeding,” in *Proc. ICML*, pp. 27-34, Jul. 2002.

[5] S. Basu, M. Bilenko and R.J. Mooney, ”A probabilistic framework for semi-supervised clustering,” in *Proc. ACM SIGKDD*, pp. 59-68, Aug. 2004.

[6] S. Basu, I. Davidson and K. Wagstaff, *Constrained clustering: Advances in algorithms, theory, and applications.* Chapman & Hall/CRC Press Data Mining and Knowledge Discovery Series, 2008.

[7] M. Bilenko and R.J. Mooney, ”Adaptive duplicate detection using learnable string similarity measures,” in *Proc. ACM SIGKDD*, pp. 39-48, Aug. 2003.

[8] P. Bradley, U. Fayyad and C. Reina, ”Scaling clustering algorithms to large databases,” in *4th Intl. Conf. on KDD*, pp. 9-15, Aug. 1998.

[9] G. Buchanan and F. Loizides, "Investigating document triage on paper and electronic media," in *Proc. ECDL*, pp. 416-427, Sep. 2007.

[10] M. Claypool, P. Le, M. Waseda and D. Brown, "Implicit interest indicators," in *Proc. IUI*, pp. 33-40, Jan. 2001.

[11] D. Cohn, R. Caruana and A. Mccallum, "Semi-supervised clustering with user feedback," in *Tech. Rep. TR 2003-1892*, Cornell University, Feb. 2003.

[12] C. Cool, N.J. Belkin and P.B. Kantor, "Characteristics of text affecting relevance judgments," in *Proc. National OnlineMeeting*, pp. 77-84, May 1993.

[13] I. Davidson and S. Basu, "A Survey of clustering with instance level constraints," presented as a tutorial at *IEEE ICDM*, 2005 and *ACM KDD*, 2006.

[14] I. Davidson, M. Ester and S.S. Ravi, "Efficient incremental clustering with constraints," in *Proc. 13th ACM KDD*, pp. 240-249, Aug. 2007.

[15] I. Davidson and S.S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *PKDD*, pp. 59-70, oct. 2005.

[16] I. Davidson and S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," in *Proc. SDM*, pp. 138-149, Apr. 2005.

[17] I. Davidson and S.S. Ravi, "Towards dfficient and improved hierarchical clustering with instance and cluster-level constraints," in Tech. Report, CS Department, SUNY-Albany, 2005. Available: www.cs.albany.edu/?davidson.

[18] I. Davidson and S.S. Ravi, "Identifying and generating easy sets of constraints for clustering," in *Proc. 21st AAAI*, pp. 336-341, Jul. 2006.

[19] I.S. Dhillon, J. Fan and Y. Guan, "Efficient clustering of very large document collections," in *Data Mining for Scientific and Engineering Applications*, Kluwer Academic Publishers, 2001.

[20] T.W. Finin, "Help and advice in task oriented systems," in *Proc. 7th IJCAI*, Aug. 1982.

[21] J. Grudin, "Groupware and social dynamics: eight challenges for developers," in *CACM*, vol. 37, no. 1, pp. 92-105, Jan. 1994.

[22] T. Hertz, A. Bar-Hillel and D. Weinshall, "Boosting margin based distance functions for clustering," in *Proc. 21st ICML*, pp. 50-57, Jul. 2004.

[23] A.K. Jain, M.N. Murty and P.J. Flynn, "Data clustering: A review" in *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, Sep. 1999.

[24] B. King, "Step-wise clustering procedures," in *J. Am. Stat. Assoc.*, vol. 62, No. 317, pp. 86-101, Mar. 1967.

[25] D. Klein, S.D. Kamvar and C.D. Manning, "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering," in *Proc. 19th ICML*, pp. 307-314, Jul. 2002.

[26] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon and J. Riedl, "GroupLens: Applying collaborative filtering to Usenet news," in *CACM*, vol. 40, no. 3, pp. 77-87, Mar. 1997.

[27] H. Lieberman, "Autonomous interface agents," in *Proc. ACM CHI*, pp. 67-74, Apr. 1997.

[28] E. Mac Aoidh, M. Bertolotto and D. C. Wilson, "Analysis of implicit interest indicators for spatial data," in *Proc. 15th ACM GIS*, pp. 1-4, Nov. 2007.

[29] C.C. Marshall and F. Shipman, "Effects of hypertext technology on the practice of information triage," in *Proc. HT*, pp. 124-133, April 1997.

[30] K.F. McCoy, C.A. Pennington, and L.Z. Suri, "English error correction: A syntactic user model based on principled mal-rule scoring," in *Proc. 5th UM*, pp. 59-66, Jan. 1996.

[31] M. Morita and Y. Shinoda, "Information filtering based on user behavior analysis and best match text retrieval," in *Proc. SIGIR Conf. on R&D*, pp. 272-281, Jul. 1994.

[32] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms which use cluster centers," in *Comput. J.*, vol. 26, No. 4, pp. 354-359, 1984.

[33] M. Nanni, "Speeding-up hierarchical agglomerative clustering in presence of expensive metrics," in *PAKDD*, pp. 137-138, May 2005.

[34] D. Nichols, D. Pemberton, S. Dalhoumi, O. Larouk, C. Belisle and M. DEBORA Twidale, "Developing an interface to support collaboration in a digital library," in *Proc. ECDL*, pp. 239-248, Sep. 2000.

[35] F. Qiu and J. Cho, "Automatic identification of user interest for personalized search," in *Proc. WWW*, pp. 727-736, May 2006.

[36] M. E. Renda, and U. Straccia, "A personalized collaborative digital library environment: a model and an application," in *Information Process Management*, vol. 41, no. 1, pp. 5-21, Jan. 2005.

[37] Richard C. Dubes and Anil K. Jain, *Algorithms for clustering data.* Prentice Hall, 1988.

[38] J. Rucker and M. Polanco, "Personalized navigation for the web," in *CACM*, vol. 40, no. 3, pp. 73-76, Mar. 1997.

[39] B. Sarwar, J. Konstan, A. Borchers, J. Herlocker, B. Miller and J. Reidl, "Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system," in *Proc. CSCW*, pp. 345-354, Nov. 1998.

[40] F. Shipman, H. Hsieh, R. Airhart, P. Maloor and J.M. Moore, "The visual knowledge builder: a second generation spatial hypertext," in *Proc. HT*, pp. 113-122, Aug. 2001.

[41] J. Shrager and T. Finin, "An expert system that volunteers advice," in *Proc. AAAI*, pp. 339-340, Aug. 1982.

[42] P.H.A. Sneath and R. R. Sokal, *Numerical taxonomy: The principles and practice of numerical classification.* London,UK: Freeman, 1973.

[43] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints" in *Proc. 17th ICML*, pp. 1103-1110, Jun. 2000.

[44] K. Wagstaff, C. Cardie, S. Rogers and S. Schroedl, "Constrained K-Means clustering with background knowledge," in *Proc. ICML*, pp. 577-584, Jun. 2001.

[45] J.H.JR. Ward, "Hierarchical grouping to optimize an objective function," in *J. Am. Stat.Assoc.*, vol. 58, No. 301, pp. 236-244, Mar. 1963.

[46] G. Weber and M. Specht, "User modeling and adaptive navigation support in WWW-based tutoring systems," in *Proc. 6th UM*, pp. 289-300, Jun. 1997.

[47] E.P. Xing, A.Y. Ng, M.I. Jordan and S. Russell, "Distance metric learning, with application to clustering with side-information," in *NIPS*, vol. 15, pp. 505-512, Dec. 2003.

VITA

Prasanth Ganta received his Bachelor of Technology degree in Computer Science and Engineering from Indian Institute of Technology, Guwahati in 2006. He entered the Computer Science program at Texas A&M University in August 2008 and received his Master of Science degree in May 2011. His research interests include Information Retrieval, Data Management and Intelligent User Interfaces.

Prasanth Ganta
Dept. of Computer Science and Engineering
Texas A&M University
301 Harvey R. Bright Building
College Station, TX 77843-3112
EMAIL: prasanth.ktgm@gmail.com

The typist for this thesis was the author.