CRISP - An Interruption Management Algorithm based on Collaborative Filtering[*]

Tammar Shrot
Department of Software Engineering
Shamoon College of Engineering, Beer-Sheva, Israel 84100
tammash@sce.ac.il

Avi Rosenfeld
Department of Industrial Engineering
Jerusalem College of Technology, Jerusalem, Israel 9116001
rosenfa@jct.ac.il

Jennifer Golbeck
Institute for Advanced Computer Studies
University of Maryland, College Park, USA 20742
golbeck@cs.umd.edu

Sarit Kraus
Department of Computer Science
Bar-Ilan University, Ramat-Gan, Israel 5290002
sarit@cs.biu.ac.il

ABSTRACT

Interruptions can have a significant impact on users working to complete a task. When people are collaborating, either with other users or with systems, coordinating interruptions is an important factor in maintaining efficiency and preventing information overload. Computer systems can observe user behavior, model it, and use this to optimize the interruptions to minimize disruption. However, current techniques often require long training periods that make them unsuitable for online collaborative environments where new users frequently participate.

In this paper, we present a novel synthesis between Collaborative Filtering methods and machine learning classification algorithms to create a fast learning algorithm, CRISP. CRISP exploits the similarities between users in order to apply data from known users to new users, therefore requiring less information on each person. Results from user studies indicate the algorithm significantly improves users' performances in completing the task and their perception of how long it took to complete each task.

Author Keywords

Interruption Management (Cost Estimation), Collaborative Filtering, Classification Algorithm

ACM Classification Keywords

H.1.2 User/Machine Systems: Software psychology

INTRODUCTION

Real-time collaboration as part of the work process has become increasingly common, through using chat tools while editing, with collaborative-editing environments like Google Docs, and with other platforms. However, users working in a distributed way may lack the ability to perceive the other's work process; thus, social cues about when to interrupt are lost. In domains like collaborative document editing where users are communicating while editing, interruptions for communication about the task are common [19]. For example, two people may be working on a document together and sending messages as they write. In person, one author would not interrupt the other if she could see he was in the middle of focused writing or editing. Online, though, she can't see this, so may send a message that pops up and disrupts his workflow.

Similarly, system dialogues can be important for users to complete tasks. However, a poorly timed message (for example, one that pops up asking a user if they want to perform a system update) can affect the user's focus level which may lead to averse effects on task performance and users' frustration levels [1]. A system that can hold interruptions until less disruptive points in a user's workflow may improve the speed and efficiency of the work.

Prior work has shown that interruptions can be extremely disruptive to users maintaining concentration on their tasks, and indeed that completely eliminating interruptions may be the best option [23]. User control over when interruptions happen and from what sources they are allowed is a trend that can be seen in smartphones, for example, where users can permit or deny apps the right to send notifications, often with

[*]This work was supported in part by ERC grant #267523.

CHI'14, April 26–May 1, 2014, Toronto, Canada.

http://dx.doi.org/10.1145/2556288.2557109

fine-grained control. However, while eliminating interruptions may be ideal for allowing users to stay on task, when working in an environment where interruptions are allowed or are a natural part of the process (such as document coediting), an algorithmic approach to minimize disruption can be helpful. Any solution requires estimating the cost of an interruption for a user. Existing methods generally require hours of observations to build a model for each person [1, 15]. For systems like Google Docs, where new users can easily be invited into the system, such a long training period can be impractical.

Our contribution is a collaborative filtering-inspired interruption management algorithm, designed to minimize the negative effects of interruptions. By leveraging data from other similar system users, a novel approach, the algorithm is able to immediately begin effectively controlling interruptions for a new user, offering a significant advantage over existing techniques that require long training phases. We demonstrate the positive effect this algorithm has on a user's speed for a collaborative editing task compared with interruptions that occur randomly or that are controlled by a rule-based algorithm.

Our algorithm models people's preferences for interruption timing and uses those insights to control interruptions. Our new approach, called CRISP (Collaborative Filtering and Rule-based Interruption management via Similarity Profile), leverages the underlying ideas of Collaborative Filtering algorithms, an environment dependent rule-based algorithm, and basic classification algorithms. The advantage of CRISP over traditional learning methods is its significant reduction in the learning time needed to model a given user. This allows us to quickly decide about the efficiency of an interruption with only limited data and can avoid pitfalls such as protracted learning periods and elicitation of private user data. It makes the algorithm ideal for systems that want this control up and running for a user very quickly after they join.

This paper presents the details of our approach for modeling user preferences. We validate this with two user studies while comparing it to random interruptions and a rule based algorithm. We found that our algorithm allows users to work faster and improves their perception of speed as well.

RELATED WORK AND BACKGROUND

Interruptions and User Performance

The importance of user attention and the impact of interruptions has been studied from a variety of perspectives in the HCI community. Adamczyk & Bailey [1] studied the impact of interruptions on users' emotional and cognitive states when they were interrupted at different times in the course of their task. They found that timing did have an impact and suggested that an "attention manager" could be an effective tool for minimizing the negative impact that interruptions have on users.

Analyzing users themselves provides insight into when interruptions are best handled. Salvucci & Bogunovich [22] found that when subjects were interrupted during high cognitive workload tasks, they deferred dealing with notifications 94% of the time. However, during low workload periods, they deferred only 6% of the time. This illustrates that users do have periods in which interruptions are easier to deal with. Other work by Iqbal & Bailey [15] modeled users working on a task and identified the "best" and "worst" times to interrupt a user. Their results showed the interruptions at the "best" time significantly improved the user experience.

Several previous works have student different aspects of the interruption management problem. Arroyo & Selker [2] developed the Disruption Management Framework, a system that takes into consideration the users' motivation in how they handle interruptions. The framework was developed to support interruption mediating in multitasking environments and is constructed of 3 layers. The layers are built to take into consideration tasks' related information, users' related information and information related to the users' current behavioral pattern. Interruption was also addressed in McCrickard et al. [18], which identified relevant metrics and evaluation strategies for peripheral displays. In particular, they described a classification model based on 3 elements of awareness: interruption, reaction, and comprehension. Fogarty et al. [9] developed a sensor-based statistical model of a person‘s interruptability level. This model worked well, but required hours of video recording which limited the scope of the study. In our work we focused on developing an automated method that needs less time to collect its data.

Applications of this idea have occurred in a number of domains. Bowers et al. [4] found that in tasks where context related to the interruption, such as adding annotations to videos, interrupting a video at the time its context related to the system's question was preferred. Fischer et al. [7] studied interruption timing in mobile applications and found that when interruptions occurred at natural breakpoints, they were less disruptive than a random baseline. This emphasizes the importance of detecting users' patterns of interaction and timing such that a system can automatically identify these opportune times when they occur.

Our work builds on this research by creating a new system that automatically models users' preferences for interruption timing and uses that to control messages communicated through the system. In addition, we used the breakpoints identified by those researchers as baseline in order to validate the contribution of our system.

Cost Estimation of Interruptions

One of the most important issues concerning the initiation of interruptions is the ability to accurately estimate the cost arising from the interruption. Accurate estimation will enable interruption only when it will have a positive impact on the group's performance [8]. Interruptions have two ways to negatively affect users: a long term effect and a short term effect. Both the long and short term effects must be taken into account when calculating the cost of the interruptions.

Previous research has investigated how to estimate the cost of interruptions [13, 25]. Fleming & Cohen [8] were the first to build a user-specific model which generally takes the user's specific factors into account. They used cost estimation to create a decision making mechanism in order to decide when to initiate communication. However, they assume that they have statistical data about the users' knowledge and utility values. Our algorithm quickly builds a model with no individual background information. Tan & Richardson [26] studied the cost of interruptions to facilitate delaying when an interruption will occur. Our algorithm has similar features for deciding when to interrupt. Bailey & Iqbal [3] studied interruption managing in text editing environment after long observation they developed identified breakpoints and subtasks during workload, which considered good timing to interrupt users.

The key difference between our research and previous works is that we study cost-estimations that can work in dynamic domains in which the environment's conditions rapidly change, actions occur quickly, and users' abilities change over time. In contrast, previous work required long and expensive training sessions to gather the necessary information for the algorithm to function.

Collaborative Filtering

One of the ways our algorithm can quickly create a model for timing interruptions is by relying on data from other users as initial background. This approach borrows insights from Collaborative Filtering. Collaborative Filtering (CF) is a method of making automatic predictions (filtering) about the preferences of a user by collecting data on the preferences of many users (collaborating). There are many examples of recommendation systems via Collaborative Filtering [12, 29].

Collaborate Filtering models can be built based on users or items. User Based collaborative filtering systems find other users that have displayed similar tastes to the active user and recommend the items similar users have preferred [20]. That is, if user u_1 and user u_2 shown similar taste then items that u_1 likes will be offered to u_2 and vise verses. Users' similarity is calculated by comparing users' history and identifying similar ranks to the same items. Item-based models recommend items that are most similar to the set of items the active user has rated [17]. That is, if user u liked item i_1, then items found to be similar to i_1 will be offered to u. The assumption within CF models is that similar users will always make similar decisions, thus de-emphasizing the role of individual preferences.

Hybrid approaches are also common. Karypis [16] was the first to recommend an approach that combines the best of the Item-based and the User-based (classic Collaborate Filtering) algorithms, by first identifying a reasonably large neighborhood of similar users and then using this subset to derive the Item Based recommendation model. Vozalis et al. [27] have developed a hybrid method consisting of a number of steps.

Classification

We use two machine learning classification algorithms, the *k-nearest neighbor ($k-NN$)* algorithm [6] and a *rule-based* algorithm [5], to find people who behave most like the user, and to identify situations that are reflections of what the user might encounter. Rules are used to identify the natural breakpoints (as described in [3] model). The *k-nearest neighbors ($k-NN$)* algorithm [6] is a method for classifying objects based on closest training examples in the given labeled database. *k-nearest neighbors* is a type of instance–based learning, where the function is only approximated locally and the computation is done only at classification time. This creates a dynamic, multi-model ad-hock clustering of the subjects.

Many previous works combine several machine learning algorithms in order to achieve better results than each algorithm archives individually. Huang et al.'s [14] approach was to combine algorithms that discover global structure in the data with algorithms that discover local structures in the data. We use a similar approach. However, while most hybrid approaches use different algorithms in each phase, they use the same data in both phases. Our approach differs as we collect different types of information - the user's task information and general domain information - and use different information in separate phases. Rzeszotarski & Kittur [21] had a similar idea of using information gathered from past AMT worker in order to generate a labeled corpus that will assist in minimizing the time it takes the system to make a decision about a new user. They used a corpus of past users' behaviors in order to quickly model and evaluate task fingerprinting. They used past information gathered from AMT workers in order to capture crowdsourced behavior and make inferences about their task performance.

Our work builds on our past work by Shrot et al. [24]. However, while our work was checked online with real people against other algorithms in real word conditions, Shrot et al. only considered interruptions within a simplistic video game. Furthermore, they only checked data offline and thus could not decide if interruptions should be posed during task execution as this work does. In addition, Shrot et al. exclusively considered CF-based algorithm and did not consider incorporating previous interruption management research such as the models this work considered ([3]).

CRISP – A SYNTHESIS BETWEEN CF AND CLASSIFICATION

Our proposed algorithm, CRISP, (Figure 1) is motivated by the Collaborative Filtering hybrid approach. CRISP has three phases: a "rule based" algorithm phase (Line 2), a "user" phase (Lines 3-4 in the algorithm), and an "item" phase (Lines 5-7 in the algorithm). The first phase uses rules to identify natural breakpoints that are relatively unintrusive interruption points. The second phase uses user-specific data to identify similar users in the database and to construct an environment of similar users. The third phase of the algorithm decides if it is a good or bad time to interrupt. This leverages data from "similar" users that were discovered in the second phase.

The algorithm's structure is inspired by elements of Collaborative Filtering. Similar to Collaborative Filtering methods, classification tools are used to identify the user's neighborhood (similar users) for making recommendations about

```
1. Accept a new non labeled situation $s = (ip, h)$.
2. If $ip$ is following the rule based algorithm $RULES$ Label
   the situation based on $RULES$, else:
3. Use $h$ to create a user profile $p$ for $s$.
4. Use the profile $p$ and the database $db$ to build a neighbor-
   hood $NGB$ of $l$ situations that were found to be similar
   (through user's similarity) to $s$.
5. $IP = \emptyset$
6. $\forall s' = (ip', h')$ s.t. $h' \in NGB$ $IP = IP \bigcup \{ip'\}$
7. Build a classification model $CM$ between $ip$ and $IP$ using
   a classification algorithm.
8. Label the new situation $s$ according to the classification
   decision of $CM$.
```

Figure 1. CRISP. An algorithm for deciding whether it is a good or bad timing to interrupt the user right now. s = state; ip = interruption profile; h = user's history

future interruption times. However, Collaborative Filtering methods cannot feasibly be implemented in interruption management domains because the interruption managing system in this mixed agent–user environment does not have access to users' characteristics data nor does it have access to a specific user's votes on his past interruption. Consequently, different methods must be found to model new users' and items' similarities. CRISP (Figure 1) contains information from old and different situations that were already examined. In these situations the outcome of the interruption is already known. Therefore, it is possible to label these interruptions as either good or bad interruption timing. Since our data can be labeled, our solution is to use traditional machine learning classification algorithms. The classification algorithms are used to quickly compare the users (in the second phase) and items (in the third phase) without resorting to a shared database of all users' characteristics or voting history.

The basic element of information used is the *user's state* (u_s). A *user's state* is a vector that contains numerical or other discrete values for different attributes about the user's work progress and user's state. These attributes include information such as percentage of the task accomplished and time left to complete the task. All values in the vectors' attributes are normalized to the same scale and all vectors in a given domain will have the same attributes. A *time* element is an element $t \in \{0...T_{max}\}$ that represents the time that has passed since the beginning of the task (t was in milliseconds in our experiments). A *timed state* (ts) is a pair (t, u_s) that represents the user's state at time t.

As mentioned earlier, we use two types of data – user-specific data and state-specific data. The user-specific data is the *user's latest history* data (h). User history h is a collection of k *timed states* gathered within a short period of time defined as T_{samp}. The length of these states was between 3 and 6 seconds gathered in 30 to 60 second intervals in our experiments. These intervals were chosen based on previous work ([3]) as observed subtask lengths.

The second data type - the state specific data - is the *interruption profile* (ip). ip is a *timed state* that represents the user's state immediately prior to the time of the interruption. It is also important to note that h and ip usually refer to different attributes in the vectors. I.e. h will mostly be calculated using attributes that represent the user's behavioral pattern, while ip will mostly be calculated based on local and temporal attributes that represent the current situation.

A *situation* (s) is a pair of user's latest history h and the interruption profile ip that immediately follows it. That is, $s = (h, ip)$ s.t. $ip = (t^*, u^*_s)$ and $\forall (t', u'_s) \in h$ $(t^* - T_{samp}) \leq t' < t^*$. Therefore, a situation is constructed from two distinct data types, both user specific and state specific data, giving us a wider point of view about the interruption's influence. A *labeled situation* is a pair $(s, label)$ that matches a situation with the label of whether it is a "good" situation for interruption or not, namely, whether the group's gain from this interruption is higher than the cost of the interruption.

CRISP's (Figure 1) input is a small database of labeled situations (db) and a non-labeled situation s, for which we wish to discover whether it is a "good timing" or not. In the first phase the algorithm checks the given situation based on its initial *rule based* algorithm. If the situation is matching one of the given rules it will label it according to the appropriate rule in the algorithm. In the Experimental Setup section we detail exactly what rules were implemented. If none of the rules apply, it will continue to the next phase of the algorithm.

In the second phase ("user" phase) the algorithm builds a user's similarity model between the new given situation (s) and the given labeled situations in the database (db) (Lines 3–4 in the algorithm). This phase uses only the historical data (h). The user similarity model is built according to the similarity between the "user's latest history" data in the situations.

Specifically, h is a set of k vectors that represents the user's behavior in the short period (T_{samp}) sampled before the interruption (30 to 60 seconds in our experiments). For each situation, the algorithm calculates how each attribute's value changed (on average) between the sampling in h (Line 3 in the algorithm). The user's profile is the vector of averages changes. Then, similarity between two user situations is measured as the distance between the two calculated profiles (Line 4 in the algorithm).

The assumption behind this approach is that users with similar profiles (same average change in values) undergo the same process and therefore most probably act in similar ways. This model represents the user's similarity level between the new situation (s) and the given labeled situations. Once the user's similarity model is completed, the algorithm uses the *k-nearest neighbor* ($k - NN$) algorithm [6] to choose the l most similar situations as the new situation neighborhood. This neighborhood is used in the algorithm's next stage. For the experiments l was selected as a percentage of the entire database length (10%) and in our experiments this was 60 of the 600 users in the database.

The next stage of the algorithm (Lines 5 to 8 in the algorithm) uses only interruption profile (ip). Once the user's profile and neighborhood are constructed, the algorithm builds an interruption profiles similarity model between the situations that belong in the neighborhood. A machine learning classification algorithm runs over the neighborhood's situations and returns the calculated classification.

The net result is that once a new situation arrives, the algorithm needs only a very short time to gather enough data in order to decide how to treat it. This allows for a faster and more accurate classification than the base machine learning algorithms alone could provide.

EXPERIMENTAL DESIGN

Our experiments were designed to see if interruptions, as controlled by CRISP, offered users benefits over other interruption timing in terms of speed, efficiency, and user perception of task difficulty. We were inspired by Bailey & Iqbal [3] to studied this problem in the context of a text editing task. Our context was a text editing task where pop-up messages interrupted users with information they had to note down. These simulated interruptions from a collaborator might occur when working together on a document. The experiment was run on Amazon Mechanical Turk (AMT) and our AMT usage parallels previous work [21].

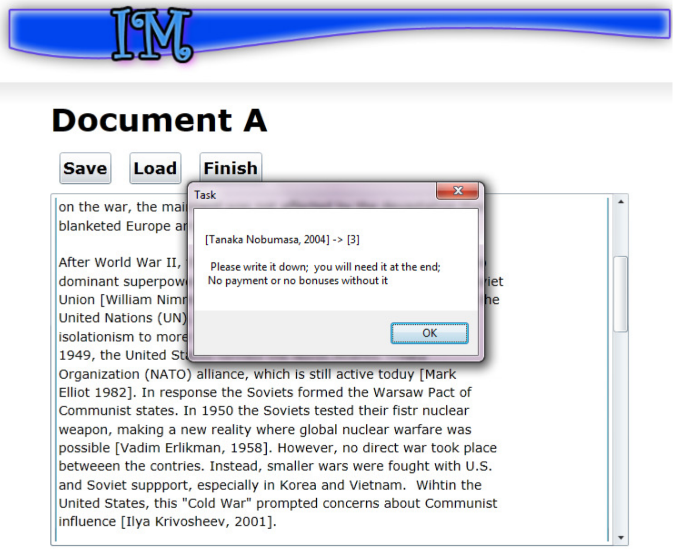
Tasks

The user's task was to locate and correct all spelling and grammatical errors within the text. For every mistake fixed, the subject earned a bonus (given in order to motivate our subjects). The assignment had a time limit (10 minutes) and the participant was instructed to fix as many mistakes possible before the end of the assignment. To simplify the logging process we disabled the subjects' ability to change the position within the text document by using the mouse. This forced the subjects to only use the keyboard, something that was easier to log. The experimental environment is shown in (Figure 2) with a message over the text being edited.

Interruptions

User interruptions have been previously categorized as being either external or internal with external interruptions being initiated by other people [10]. While this work noted that both types of interruptions occur with nearly equal frequencies, we focus on external interruptions as the system potentially has more control over these events, as opposed to internal interruptions which people initiate for themselves.

Users were interrupted as they were editing the document. There are two types of interruptions: **(1)** Unexpected interruptions and **(2)** Interruptions caused by communication (e.g. messages sent by the system or other users). Both of these interruptions can be classified as immediate interruptions as per McFarlance's taxonomy [19]. The first type of interruptions represent an unexpected and uncontrolled event that interferes with the user's regular course of action. These are interruptions that we cannot control and they have nothing to do with the user's task or the agent. They are usually rare and reflect a highly disturbing phenomenon. We simulated those interruptions by random events that erase the user's document. These occur rarely (no more than 2 events per document, usually only 1). In order to avoid disruption from these events, subjects were instructed to save frequently and load the last saved document after it was erased. Note that our algorithm does not try to control these interruptions, because they essentially represent system errors or accidental use of the system (hitting a series of keystrokes that have a bad effect). We have included them in the study since they represent events that affect user behavior. The second type of interruptions are those caused by communication. We simulated a situation where two users were collaboratively editing a document. The subject was correcting typos and the other user, whom we simulated, sent requests for the subject to update the format of references in the document.



Document A

on the war, the mai
blanketed Europe ar

After World War II,
dominant superpow
Union [William Nimr
United Nations (UN)
isolationism to more
1949, the United St
Organization (NATO) alliance, which is still active toduy [Mark Elliot 1982]. In response the Soviets formed the Warsaw Pact of Communist states. In 1950 the Soviets tested their fistr nuclear weapon, making a new reality where global nuclear warfare was possible [Vadim Erlikman, 1958]. However, no direct war took place betweeen the contries. Instead, smaller wars were fought with U.S. and Soviet suppport, especially in Korea and Vietnam. Wihtin the United States, this "Cold War" prompted concerns about Communist influence [Ilya Krivosheev, 2001].

iet
he

[Tanaka Nobumasa, 2004] -> [3]

Please write it down; you will need it at the end;
No payment or no bonuses without it

Figure 2. The experimental text-editing interface. This screen capture shows a communication interruption in the form of a pop-up window.

In the experiments, the agent that represented the other user made 10 interruptions as the subject edited the document. The interruptions appear in a popup window (Figure 2). Once the window opens, the focus moves to it and the user may not edit or move in the document. The users are instructed to make a note of the change on paper for later editing. After noting the change, the subject clicks "OK" and returns to editing.

Experimental Protocol

The experiment contains 3 documents.
Example A short, 2 paragraph (about 1000 chars) document with 3 mistakes.
Document A A long (about 4500 chars) document with 50 mistakes.
Document B A different long (about 4500 chars) document with 50 mistakes.

In order to check the efficacy of CRISP we compared the timing of its interruptions with either a random interruption algorithm (first experiment) or a rule based - breakpoints identified algorithm (second experiment). All the algorithms have to time their interruptions while following the same basic pattern:

- No communication interruption takes place in the first 30s (seconds) of the task. This time is used to gather information regarding the user behavior (h).
- The first communication interruption is scheduled somewhere between the second 30s of the task either randomly, according to breakpoint identification, or using CRISP).
- Once the first communication interruption is finished, a new 30s information gathering phase (h) begins, followed by an interruption in the following 30s interval.
- This cycle repeats until the 10 minute time limit is reached.

The length of this pattern was chosen based on [3] observation regarding subtask length.

The protocol for each subject was as follows:

1. *Learning stage:* Running the "Example" document with random interruption.
2. *Document Editing*: Running "Document A" document with either CRISP-timed communication interruption or comparison (random or rule-based) interruptions.
3. Answering a questionnaire.
4. *Document Editing*: Running "Document B" document with either CRISP-timed communication interruption or comparison (random or rule-based) interruptions.
5. Answering a questionnaire.

All subjects completed the example document editing (step 1) first, as a training and task learning session. The order in which they completed scenarios 2 and 4 varied. 50% of the users edited Document A before editing Document B, and 50% did it the other way around. In addition, order of the interruption algorithms was randomized. This was done to negate any effects from the order of the documents or the combination of document and algorithm.

In all but the first scenario (the learning stage), once the users finished a task, they were asked to complete a questionnaire. The first question was a request to quote one of the citation changes provided by the system in one of the interrupting pop-ups. This was a filtering question meant to discover the users that did not follow the task instructions. In our experiments we found one such user that ignored the task of cooperating with the agent. This user was removed from our analysis. After answering that question, users complete the NASA-TLX (Task Load Index) assessment [11]. This survey is used to measure a subjects' perception of the mental, physical, and temporal demands of a task in addition to their perceptions of their own performance, effort, and frustration. We compared the speed with which users edited and read the document and their NASA-TLX scores among conditions to see how helpful each was.

Subjects

In the first experiment, 30 people participated. All were US citizens. Eighteen were female (60%) and 12 were male. The average age was 32.1 with a standard deviation of 9.5. The education level varied: 9 subjects' degree was a high school diploma, 17 had either a BA or BSc degree, and 4 had a post-graduate degree.

In the second experiment we had 18 people, all US citizens. Nine were female (50%), and 9 were male. The average age was 32.5 with standard deviation of 12. The education level varied here as well: 7 subjects' degree was a high school diploma, 9 had either a BA or BSc degree, 1 had a MSc degree and 1 had a post-graduate degree.

EXPERIMENTAL SETUP

This section describes how CRISP's Rule based, User and Item stages were implemented in a text editing environment. The rule component of CRISP has only one rule - if the user just re-loaded her document from a saved version, it is a good time to initiate a communication. This rule is based on the fact that loading the document moves the user to the beginning of the documents, which means she has already lost focus, and disturbing her right now will not cause a massive interruption.

The user stage of the algorithm is based on the behavioral information collected about the user. In the text editing environment the attributes that were taken for the u_s profile were as follows. All times are in milliseconds.

Last action This attribute is for the rule–based part. It saves the last meaningful action taken by the user.

Profile length The length of time this data was gathered in. The values range: 30K to 60K. Used for normalization.

Key number The number of times the user pressed any key during the gathering of information (during h).

Alpha number The number of times the user pressed any alphabetic key.

Mouse move number The number of times the user moved the mouse in this h.

Average time Mouse Move The average length of a single mouse move as record in this h.

Location distance The percentage of the document that was scanned during this h, measured by the change in document position from the start of h to its end.

Max location distance The maximum movement (in %) the user did in this document during h.

Moving indication An indicator that compares the user's past moving behavior (beginning of h) to her current moving behavior (end of h) to see if she is accelerating, slowing down or maintaining the same rhythm of work.

Fix distance Same as Location distance, but tracking the percentage of mistakes fixed.

Max fix distance Same as Max location distance, but tracking the percentage of mistakes fixed.

Fixing indication Same as Moving indication, but tracking the percentage of mistakes fixed.

Save number The number of times the user saved the document during h.

Load number The number of times the user loaded the document during h.

Interruption number The number of document erasing interruptions in h.

Average length of interruption The average time it took the user to copy the citation on page and close the pop-up window in all the communication interruption she experienced so far in this document.

Average return from interruption The average time it took the user to return and do some action in the document after she closed the pop-up window.

Average real return from interruption The time it took the user to do a meaningful action (move in the document, edit, save, etc. in contrast to just moving the mouse) after closing the pop-up window.

The item stage of the algorithm uses current state parameters in order to try and find the most similar states in the user database and learn what decision to make based on the classification model built from them. In the text editing environment the attributes that were taken for the u_s profile to be used in this stage were as follows.

Time The time that passed from the beginning of the task in milliseconds.

Location in document The current location of the user in the document (%).

Correction of document The current number of mistakes that have been corrected in the document (%).

Time from last key The time that passed from the last time the user pressed a key on the keyboard.

Time from last alpha Same as "Time from last key" only for alpha keys.

Time from last fix The time that passed from the last time the user fixed a mistake in the document.

Time from last save The time that passed from the last time the user saved the document.

Time from last load The time that passed from the last time the user loaded the document.

Time from last interruption The time that passed from the last interruption.

Total number of mouse move The total number of mouse moves the user has made so far.

Time from last mouse move The time that passed from the last time the user moved the mouse.

The agent collect all of these attributes (from both stages) of the user's state every second. This information is necessary to construct the user's state (u_s) either for user's history (h) (first stage) or interruption profile (ip) (second stage).

Algorithm Example

If Dan is editing a document and there is a message for him, the system wants to choose the optimal time to interrupt him. The algorithm first checks if Dan just completed a task, something that would signify a breakpoint [3]. If yes, it is a good time to interrupt. If not, CRISP searches for the k most similar usersin the database (in this example we will choose $k = 2$). In order to find similar users we must first calculate Dan's current behavioral profile. For example we will focus on 3 attributes that CRISP uses: Key number (normalized to 30 seconds), Fix distance, Save number. Let's assume Dan typed an average of 20.4 keystrokes per minute (yielding a value of 10.2 keystrokes normalized to a 30 second interval), fixed 4 percent of the mistakes and saved twice during his current session. Our database is constructed of 3 users: Alice, Bob and Chuck, and they have their own values for those parameters. Let's assume their values are as shown in table 1.

Table 1. Data for example users

	Key number	Fix distance	Save number
Alice	8.5	2	1
Bob	11.4	6	2
Chuck	36.1	8	2

By comparing the distance between the vectors it is clear that Alice and Bob are much more similar to Dan than Chuck. Chuck is a much more active user. Therefore, since we want the $k = 2$ most similar users, we choose Alice and Bob as the users that constitute Dan's neighborhood.

Next, we search Alice and Bob's data for situations most similar to the one Dan is currently in. Similar situations are defined as being around the same location within the document that Dan is currently editing (Location in the Document), with a similar number of mistakes fixed (Correction in the document) and done at a similar work rate (time from last key, time from last alpha, or time from last fix). These values are combined as a weighed sum as described previously. We search for the situation within the cluster with the highest similarity to Dan's current situation. CRISP then checks if this situation was a good or bad interruption, and decides if it will interrupt Dan or not accordingly. If it was a good time, the algorithm will interrupt. If it was a bad time, it will not.

Bootstrapping the Experiment

We bootstrapped CRISP with a small initial database, that contains a set of pre-labeled situations. In order to create that initial database we ran an offline data collection phase. In this offline data collection phase we had 20 subjects. All were US Citizens. Thirteen were female (65%) and 7 were male. The average age was 33.5 with a standard deviation of 13.3. The education level varied: 12 subjects' highest degree was a high school diploma, 6 had either a BA or BSc degree, and 2 had a post-graduate degree. As previously stated, Amazon's Mechanical Turk was used to select participants. In our experiment we only allow USA citizens (that were also born in the USA) in order to disable noise due to English not being the subjects' main language. Each subject open the experiment on her machine and the size of the window was not fixed. Yet, we limit the maximal number of characters a subject can see in any given moment.

Interruptions were generated randomly each 30 $\sim$ 60 seconds, and the data was gathered. Each subject performed the entire experimental protocol as described above, but both scenario 2 and scenario 4 had random interruptions. The subject's state and status were sampled every 10 seconds and also before and after each interruption.

At the end of the research protocol each communication interruption was labeled as either "good" or "bad" according to the effects it had on the users' final outcomes at the end of the experiment (the state of the user at the end of the documents, including errors caught and how far they have made it through the document), their emotional state (measured by the NASA-TLX survey), and according to the magnitude of interruption and frustration it caused the user. Unlike Shrot et al. [24], the labeling process was not automatic, since this approach is too simplified and cannot work in a complex domain such as text editing. Instead, the communication was labeled manually according to the effect it had on the user's performance and emotional state (based on Bailey & Iqbal [3]'s model). The labels were determined based on an analysis of the differences in the information gathered before and after the interruptions (i.e. the difference in typing speed as could be observed by the "Key number" attribute mentioned in Experimental Setup section) as well as the differences between how well the subject completed the task and the score she gave that task in the NASA-TLX survey. The data resulting from this offline data collection phase was used to bootstrap the live experiments.

RESULTS

First Experiment: CRISP vs. Random

The results of our experiment showed significant improvements in actual performance and user perception of performance with the CRISP algorithm over random interruptions.

Fixing Errors

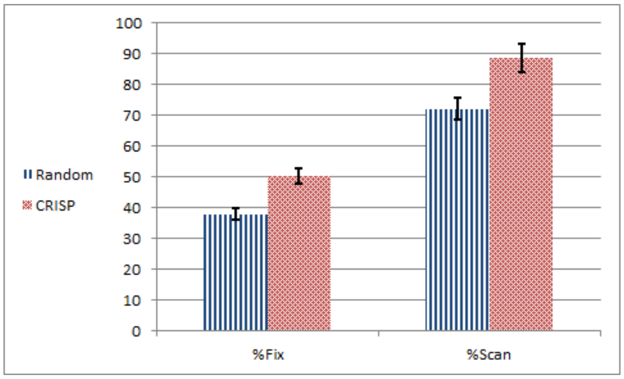


Figure 3. The percentage of the mistakes fixed and document scanned as a function of the different algorithms (CRISP vs Random).

The best and most obvious indicator of success is how much of the task was completed. We measure this by how many of document's mistakes were fixed. Users performed significantly better with CRISP (see Figure 3). They fixed an average of 50.2% of the errors when interruptions were made using CRISP compared with only 38.0% of errors with the random interruptions. These results are statistically significant (two-tailed paired-t-test, $p < 0.01$).

Scanning the Document

A second indicator of success is that with CRISP the users were able to scan significantly more of the document before the 10-minute deadline (two-tail paired-t-test, $p < 0.01$). Since users were only using the keyboard and could not scroll with the mouse, we could measure the percentage of the document scanned by tracking the location of the cursor. We measured how far into the document each user went. With CRISP, users covered 88.7% of the document, compared with only 72.2% with the randomized interruptions. Results are shown in Figure 3.

User Perception Results

An interesting results is that, unlike in the performance indicators that are significantly better using CRISP, most of the NASA-TLX indicators show no significant difference between the random timing algorithm and CRISP.

The only exception was the Temporal indicator, where the users were asked *"How hurried or rushed was the pace of the task?"*. In this indicator, as seen in Figure 4, using CRISP, users felt less rushed during the task. On a 20 point scale, they rated the time pressure with CRISP at a 14.6, but gave a significantly higher difficulty score of 16.3 to the task with random interruptions (two-tailed paired-T-test, $p < 0.01$).

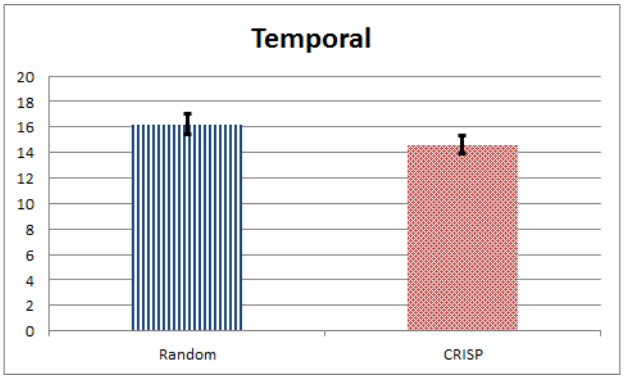


Figure 4. The Temporal value (0 to 20 scale) given in the survey as a function of the different algorithms (CRISP vs Random).

Second Experiment: CRISP vs. Rule-Based Breakpoints

In this experiment we wanted to compare our algorithm to some of the state of the art work done in this field. One common and high-performing technique is to identify natural "breakpoints" in the user's workflow. Points at which user's attention has broken from the current task are good times for interruption since there is minimal disruption that will occur. Many researchers have done work on creating rules for identifying breakpoints and they have demonstrated their effectiveness ([1, 3, 7, 15, 22]).

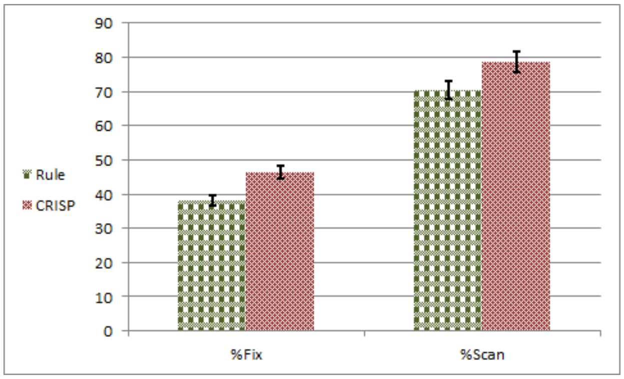


Figure 5. The percentage of mistakes fixed and document scanned as a function of the different algorithms (CRISP vs Rule–Based).

In this second experiment we ran the same protocol as before, only this time instead of comparing CRISP to randomly timed interruptions, we compared it to a rule–based algorithm that timed its interruptions to known breakpoints in the workflow. We used the breakpoints that were identified in prior work ([3]): finishing a sub-task, saving one's work, and extreme attention shifts. Within our task finishing a sub-task was fixing a textual mistake, saving one's work was identical to their work, and extreme attention shifts were represented by delete interruptions and when the user loaded the document.

As can be seen in Figures 5 and 6, the results are similar to those of the first experiment. Users performed much better (scanned more and fixed more) with CRISP, and they felt less rushed during the task (two-tailed paired-T-test, $p < 0.01$ for all results). There were no significant differences for almost all of the other NASA-TLX indicators.

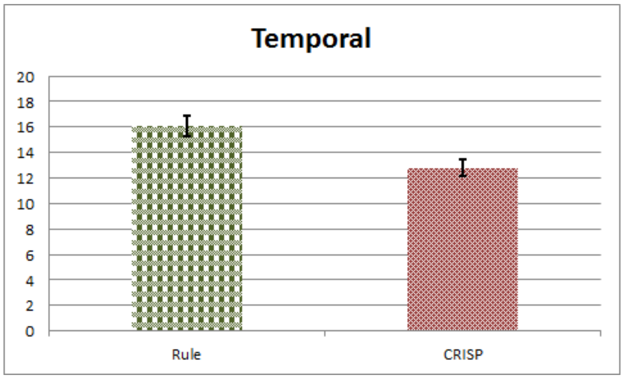


Figure 6. The Temporal value (0 to 20 scale) given in the survey as a function of the different algorithms (CRISP vs Rule–Based).

Yet, there is one interesting exception: the "Performance" indicator. In this experiment we found a significant difference between CRISP and the rule–based algorithm. Users ranked the rule–based algorithm significantly *higher* (one-tailed paired-T-test, $p < 0.05$), indicating a belief that they preformed better with the rule–based algorithm than with CRISP. In reality (as can be seen in Figure 5) the reverse is true. Despite this slight difference in users' perception, we demonstrated in this section that CRISP helped users perform significantly better in their text correction task compared to two different sets of users using the Random and a Rule-Based algorithm (Figures 3 and 5), and the CRISP users performed significantly better in the temporal category in the NASA-TLX temporal category.

There are several limitations to our study. First in order to simplify the calculation we limit the editing environment in two ways: **(1)** We disabled the ability to move in the document using the mouse. **(2)** We asked the users to write down the changes rather than applying them. As these changes are unnatural and not evident in actual text editing tasks, the results of this study may be limited. An additional limitation is that the system only supported two participants: The user and the interruption management system. In many real-word systems, such as Google Docs, the environment will often contain more participants. Nonetheless, we posit that different users can be models through CRISP. Last, the experiments were conducted using AMT, which may or may not accurately model users in all settings. We hope to further study these potential limitations in the future.

DISCUSSION AND FUTURE WORK

This paper introduces a novel approach to limit the disruptive impact that interruptions have on users when they are working in a system. We introduced an algorithm, CRISP, that models user behavior and uses this to control the timing of messages sent by the system or other users. In our experiments, we found that when compared to randomly timed interruptions and those controlled by a standard rule-based approach, CRISP-controlled interruptions allowed users to work more quickly. Users also reported feeling less time pressure with CRISP. This indicates that interruption control techniques build from user actions can improve a user's workflow when interruptions might arise.

As this paper demonstrates, CRISP excels in modeling user behavior with only limited data through identifying how similar users react to interruptions using collaborative filtering (CF) models. While CF models do generally compare similar users based on their individual differences, assuming much more information exists about a given user, the general comparisons between users made by CF models could potentially be further differentiated– something that could significantly aid in predicting how different users react to interruptions [28]. In order to address such individual differences directly and differently from a CF approach, cognitive models of users must be created through either extensively learning users in a given task, or through observing external user traits such as working memory capacity [28]. Creating hybrid approaches combining CF approaches with individual cognitive models is an interesting challenge that we leave for future work.

Our algorithm, CRISP, has been presented and tested in the context of document editing, but the general technique could be applied in other contexts. As long as data from other users' interactions is available, our collaborative-filtering inspired approach can be effective. This makes CRISP particularly applicable in collaborative online applications. Applications that could benefit from this technique are common and growing in popularity. Collaborative editing in systems like Google Docs or real-time collaborative code editing are examples of places that users can interrupt and unintentionally disrupt one another's work flow. A system control to hold messages until a less disruptive time (mediating communication) could make these experiences smoother. Similarly, system messages can interrupt users as well. These may be notifications of updates or other system messages in online editing environments, but even the common pop-up messages requesting subscriptions or offering help on websites could be better controlled. Users may be more likely to quickly dismiss these windows if they interrupt their workflow, but they may be more likely to read them if they come at a less disruptive point in a user's interaction with a site.

Our contribution described here is this new approach to interruption management that uses a collaborative filtering-inspired algorithm which we have demonstrated is effective in exemplar environments. Our goal was not to show this is the best possible interruption management algorithm, but rather to demonstrate that it works well. The benefit of a quick startup time compared to other algorithms makes the technique an important addition to the domain of interruption management algorithms. For future work, the techniques we presented may be combined or integrated with other algorithms with a goal of achieving top performance. In addition, it will be interesting to investigate other domains and problems. For example, tasks with higher cognitive demand than the one used here may show greater impact from interruption.

REFERENCES

1. Adamczyk, P. D., and Bailey, B. P. If not now, when?: the effects of interruption at different moments within task execution. In *CHI* (2004), 271–278.
2. Arroyo, E., and Selker, T. Attention and intention goals can mediate disruption in human-computer interaction. In *Human-Computer Interaction–INTERACT*. 2011, 454–470.
3. Bailey, B. P., and Iqbal, S. T. Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *TOCHI 14*, 4 (2008), 21.
4. Bowers, C. P., Byrne, W., Cowan, B. R., Creed, C., Hendley, R. J., and Beale, R. Choosing your moment: interruptions in multimedia annotation. In *Human-Computer Interaction*. 2011, 438–453.
5. Clancey, W. J. The epistemology of a rule-based expert system - a framework for explanation. *Artificial Intelligence 20*, 3 (1983), 215–251.
6. Dasarathy, B. *Nearest neighbor (NN) norms: nn pattern classification techniques*. IEEE Computer Society Press tutorial. 1991.
7. Fischer, J. E., Greenhalgh, C., and Benford, S. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *MobileHCI* (2011), 181–190.
8. Fleming, M., and Cohen, R. A user modeling approach to determining system initiative in mixed-initiative ai systems. In *User Modeling*. 2001, 54–63.
9. Fogarty, J., Hudson, S. E., Atkeson, C. G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. C., and Yang, J. Predicting human interruptibility with sensors. *TOCHI 12*, 1 (2005), 119–146.
10. González, V. M., and Mark, G. Constant, constant, multi-tasking craziness: managing multiple working spheres. In *CHI* (2004), 113–120.
11. Hart, S. G., and Staveland, L. E. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload 1*, 3 (1988), 139–183.
12. Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. An algorithmic framework for performing collaborative filtering. In *SIGIR* (1999), 230–237.
13. Horvitz, E., and Apacible, J. Learning and reasoning about interruption. In *Proceedings of the 5th international conference on Multimodal interfaces* (2003), 20–27.
14. Huang, K., Yang, H., King, I., and Lyu, M. R. Machine learning: modeling data locally and globally. *IEEE Transactions on Neural Networks 19*, 2 (2008), 260–272.
15. Iqbal, S. T., and Bailey, B. P. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI* (2005), 1489–1492.
16. Karypis, G. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management* (2001), 247–254.
17. Linden, G., Smith, B., and York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE 7*, 1 (2003), 76–80.
18. McCrickard, D. S., Chewar, C. M., Somervell, J. P., and Ndiwalana, A. A model for notification systems evaluationassessing user goals for multitasking activity. *TOCHI 10*, 4 (2003), 312–338.
19. McFarlane, D. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human-Computer Interaction 17*, 1 (2002), 63–139.
20. Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 54–88.
21. Rzeszotarski, J. M., and Kittur, A. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (2011), 13–22.
22. Salvucci, D. D., and Bogunovich, P. Multitasking and monotasking: The effects of mental workload on deferred task interruptions. In *CHI* (2010), 85–88.
23. Shneiderman, B., and Bederson, B. B. Maintaining concentration to achieve task completion. In *Proceedings of the 2005 conference on Designing for User eXperience* (2005), 9.
24. Shrot, T., Rosenfeld, A., and Kraus, S. Leveraging users for efficient interruption management in agent-user systems. In *IAT*, vol. 2 (2009), 123–130.
25. Tambe, M. Electric elves: What went wrong and why. *AI Magazine 29*, 2 (2008), 23.
26. Tan, M. K. S., and Richardson, A. Please do not disturb: Managing interruptions and task complexity. In *PACIS* (2011).
27. Vozalis, M., and Margaritis, K. G. On the combination of collaborative and item-based filtering. In *3rd Hellenic Conference on Artificial Intelligence (SETN)* (2004).
28. Werner, N. E., Cades, D. M., Boehm-Davis, D. A., Chang, J., Khan, H., and Thi, G. What makes us resilient to interruptions? understanding the role of individual differences in resumption. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 55 (2011), 296–300.
29. Xu, J., Zhang, L.-J., Lu, H., and Li, Y. The development and prospect of personalized tv program recommendation systems. In *Proceedings Fourth International Symposium on Multimedia Software Engineering* (2002), 82–89.