A Goal-Oriented Web Browser

First Author Name
Affiliation
Address
e-mail address

ABSTRACT
Many users are familiar with the interesting but limited functionality of Data Detector interfaces like Microsoft's Smart Tags and Google's AutoLink. In this paper we significantly expand the breadth and functionality of this type of user interface through the use of large-scale knowledge bases of semantic information. The result is a Web browser that is able to generate personalized semantic hypertext, providing a goal-oriented browsing experience.

We present (1) Creo, a Programming by Example system for the Web that allows users to create a general-purpose procedure with a single example, and (2) Miro, a Data Detector that matches the content of a page to high-level user goals.

An evaluation with 34 subjects found that they were more effective using our system, and that the subjects would use features like these if they were integrated into their Web browser.

Author Keywords
Goal-oriented design, Programming by Example, Data Detectors, context ware computing, software agents, Open Mind, ConceptNet, TAP, commonsense reasoning

ACM Classification Keywords
H.5.2 User Interfaces: User-Centered Design
H.5.4 Hypertext/Hypermedia: User Issues
I.2.7 Natural Language Processing: Text Analysis
I.2.6 Learning: Concept Learning

INTRODUCTION
In this paper we describe a Programming by Example system for the Web named Creo, and a Data Detector named Miro. Working together, Creo and Miro provide the user with a goal-oriented Web browsing experience. We describe an evaluation of our software's effectiveness based on data from 34 users, and evaluations of our software's user interface during an iterative design process.

Finally, we conclude with a discussion of how large-scale knowledge bases of semantic information can be leveraged to improve Human Computer Interaction.

GOAL-ORIENTED DESIGN
A popular mantra in user interface design is to design for the user's goals, and not the user's tasks. This is a very important guideline to follow when designing a user interface. For instance, digging through options dialog boxes to configure the Bluetooth settings on a Smartphone and on a PC to set up a wireless data transfer is not a goal, it is a task. The user's goal is to share a picture they took the night before with their friends.

Goal-based design leads to highly usable interfaces. Unfortunately, this design technique does not solve all of the challenges facing Human Computer Interaction. This is because there are many situations where there is no team of interface designers in charge of shaping and molding the user experience. For instance, consider the Web. The Web contains billions of interfaces, some of which are usable. However, the user experience of interacting with the Web, as a whole, was not designed. The Web is not goal-based.

While Web browsers sit between the user and the Web, the very thin amount of interface they do provide (*Back*, *Next*, *Stop*, *Refresh*, *Home*) has little to do with the user's higher level goals.

Traditionally, goal-based design has only applied to the creation of software applications with static interfaces. Any application that attempted to provide the user with a goal-based experience on top of the broad and changing collections of interfaces like the Web would have to dynamically change the interface on its own. And the application would have to do this without the help of a designer's ethnographic aptitude. Creating a usable interface has traditionally been the designer's challenge, and not the application's challenge. Generating goal-based interfaces at runtime requires software that knows a lot more about the user than the current generation of applications.

Teaching Computers the Stuff We All Know

Computers lack common sense. Current software applications know literally nothing about human existence. Because of this, the extent to which an application understands its user is restricted to simplistic preferences and settings that must be directly manipulated. Once software applications are given access to Commonsense Knowledge, hundreds of thousands of facts about the world we live in, they can begin to employ this knowledge to understand their users' intentions and goals.

Open Mind

Since the fall of 2000, the MIT Media Lab has been collecting commonsense facts from the general public through a Web site called Open Mind [1-3]. Currently, the Open Mind Common Sense Project has collected over 772,000 facts from over 16,000 participants. These facts are submitted by users as natural language statements of the form "*tennis is a sport*" and "*playing tennis requires a tennis racket.*" While Open Mind does not contain a complete set of all the common sense knowledge found in the world, its knowledge base is sufficiently large enough to be useful in real world applications.

ConceptNet

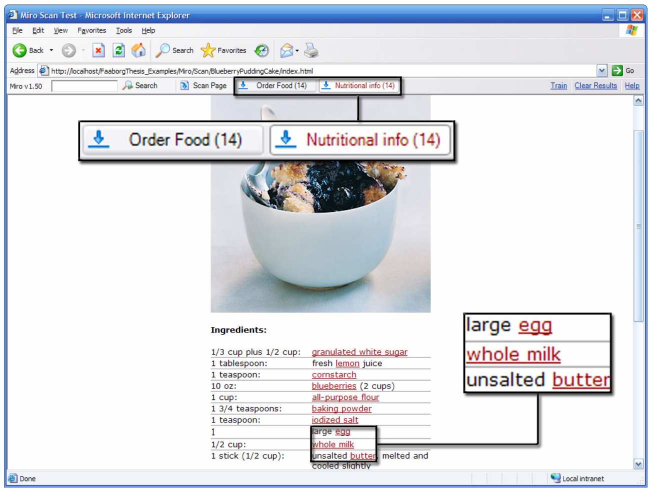
Using natural language processing, the Open Mind knowledge base was mined to create ConceptNet [4], a large-scale semantic network currently containing over 250,000 commonsense facts. ConceptNet consists of machine-readable logical predicates of the form: `(IsA "tennis" "sport")` and `(EventForGoalEvent "play tennis" "have racket")`. ConceptNet is similar to WordNet [5] in that it is a large semantic network of concepts, however ConceptNet contains everyday knowledge about the world, while WordNet follows a more formal and taxonomic structure. For instance, WordNet would identify a "*dog*" as a type of "*canine,*" which is a type of "*carnivore,*" which is a kind of "*placental mammal.*" ConceptNet identifies a "*dog*" as a type of "*pet*" [4].

Stanford TAP

The Stanford TAP (The Alpiri Project) knowledge base was created to help bootstrap the Semantic Web [6-10]. Unlike the Open Mind knowledge base, which was generated through the contributions of knowledge from volunteers on the Web, TAP was generated by creating 207 HTML scrapers for 38 Web sites rich with instance data. TAP has extracted knowledge from over 150,000 Web pages, discovering over 1.6 million entities and asserting over 6 million triples about these entities [10]. This knowledge covers a wide variety of topics, including: music, movies, actors, television shows, authors, classic books, athletes, sports, sports teams, auto models, companies, home appliances, toys, baby products, countries, states, cities, tourist attractions, consumer electronics, video games, diseases, and common drugs. The instance data found in TAP is a good complement to commonsense knowledge bases like ConceptNet or CYC [11]. For instance, "CYC knows a lot about what it means to be a musician. If it is told that Yo-Yo Ma is a cellist, it can infer that he probably owns one or more cellos, plays the cello often, etc. But it might not know that there is a famous cellist called Yo-Yo Ma" [8]. For this project, the TAP knowledge base has been modified to match the formatting of ConceptNet.

A GOAL-ORIENTED WEB BROWSER

Using the knowledge in ConceptNet and TAP, we have created a toolbar for Microsoft Internet Explorer that matches the semantic context of a Web page to potential user goals. For instance, imagine a user is viewing a Web page that contains a recipe for Blueberry Pudding Cake. The user's browser will notice a pattern of foods on the page, and present the user with two suggestions: order the foods, or view their nutritional information. When the user selects one of these buttons, all of the foods on the page turn into hyperlinks for the selected action. For instance, by pressing the "Order Food" button, each food in the recipe will be converted into a hyperlink for that food at the user's favorite online grocery store. Alternatively, the user can view the nutritional information for each of the foods at their favorite Web site for nutritional information:



Ingredients:

1/3 cup plus 1/2 cup:	granulated white sugar
1 tablespoon:	fresh lemon juice
1 teaspoon:	cornstarch
10 oz:	blueberries (2 cups)
1 cup:	all-purpose flour
1 3/4 teaspoons:	baking powder
1 teaspoon:	iodized salt
1	large egg
1/2 cup:	whole milk
1 stick (1/2 cup):	unsalted butter, melted and cooled slightly

Figure 1: Automatically associating a user's high-level goals with the content of a Web page

After being presented with this example, a critical reader likely has two significant questions: (1) *How does the browser know how to interact with the user's favorite grocery store?* And (2) *How does the browser know which of the terms in the recipe are foods?* The answer to the first question is by enabling users to train a Web browser to interact with their favorite sites using a Programming by Example system named Creo (Latin, "to create, make"). The answer to the second question is by leveraging the knowledge bases of ConceptNet and TAP to create a next generation Data Detector named Miro (Latin, "to wonder"). The following two sections discuss both of these topics in detail.

It is important to note that while this "recipe to grocery store" example is used in the next two sections for the purposes of clarity, Creo can automate interactions with any kind of site on the Web (not just grocery stores), and Miro can detect any type of data described in ConceptNet and TAP (not just foods).

PROGRAMMING BY EXAMPLE

Traditional interfaces leave the user with the cognitive burden of having to figure out what sequence of actions available to them will accomplish their goals. Even when they succeed in doing this for one example, the next time the same or a similar goal arises, they are obliged to manually repeat the sequence of interface operations. Since over time, goals tend to re-occur, the user is faced with having to tediously repeat procedures over and over. A solution to this dilemma is Programming by Example [12]. A learning system records a sequence of operations in the user interface, which can be associated with a user's high-level goal. It can then be replayed in a new situation when the goal arises again. However, no two situations are exactly alike. Unlike simple macro recordings, Programming by Example systems generalize the procedure. They replace constants in the recording by variables, that usually accept a particular kind of data.

Previous Research

The TrIAs (Trainable Information Assistants) by Mathias Bauer [12, 13] is a Programming by Example system that automates information gathering tasks on the Web. For instance, TrIAs can aggregate information from airline, hotel, weather, and map sites to help a user with the task of scheduling a trip.

Turquoise, by Rob Miller and Brad Myers [14], is a Programming by Example system that allows non-technical users to create dynamic Web pages by demonstration. For instance, users can use Turquoise to create a custom newspaper by copying and pasting information, or automate the process of aggregating multiple lunch orders into the same order.

Similar to Turquoise, the Internet Scrapbook, by Atsushi Sugiura and Yoshiyuki Koseki [12, 15], is a Programming by Example system that allows users with little programming skills to automate their daily browsing tasks. With the Internet Scrapbook, users can copy information from multiple pages onto a single personal page. Once this page is created, the system will automatically update it as the source pages change.

Web Macros, created by Alex Safonov, Joseph Konstan and John Carlis [16], allows users to interactively record and play scripts that produce pages that cannot be directly bookmarked.

A New Approach to Generalization

Knowing how to correctly generalize is crucial to the success of Programming by Example. Past systems have either depended on the user to correctly supply the generalization; or they have attempted to guess the proper generalization using a handcrafted ontology, representing knowledge of a particular, usually narrow, domain. Our contribution is to solve both the problems of generalizing procedures and proactively seeking invocation opportunities by using large knowledge bases of semantic information.

Creo

Creo allows users to train their Web browser to interact with a page by demonstrating how to complete the task. If a user decides that they are spending too much time copying and pasting the ingredients of recipes, they can easily train Creo to automate this action. To do so, the user hits the *Start Recording* button.

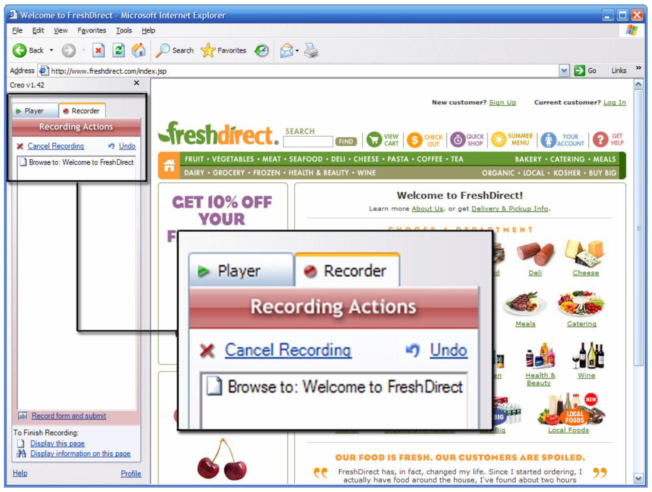

Welcome to FreshDirect - Microsoft Internet Explorer
http://www.freshdirect.com/index.jsp
Creo v1.42
Player
Recorder
Recording Actions
Cancel Recording
Undo
Browse to: Welcome to FreshDirect
Record form and submit
To Finish Recording:
Display this page
Display information on this page
Help
Profile
freshdirect
SEARCH
FIND
GET 10% OFF YOUR
Welcome to FreshDirect!
OUR FOOD IS FRESH. OUR CUSTOMERS ARE SPOILED.

Figure 2: Creo learns how to interact with a Web site by watching the user's demonstration

Creo turns red to indicate that it is in recording mode, and it captures the user's action of navigating to FreshDirect.com.

Next, the user searches FreshDirect.com for an example food, "diet coke." Creo detects that this was an example, and automatically generalizes the concept to "food brand":

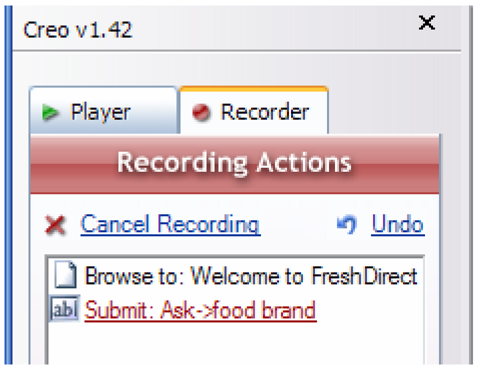


Figure 3: Creo automatically generalizes the user's input

Since these are the only two steps needed for locating a particular food at the grocery store, the user can now finish the recording, and give it a name: "Order Food." By providing a single example, "diet coke," the user has created a general-purpose recording.

In the opening example, terms like "egg," "whole milk" and "blueberries" were being linked to the grocery store, even though these are not "food brands." The reason for this is that Creo actually associates a range of generalizations with the user's input, but only displays the most general generalization for clarity. In this particular case, "food" was the second most general generalization of "diet coke," as shown in the following figure.

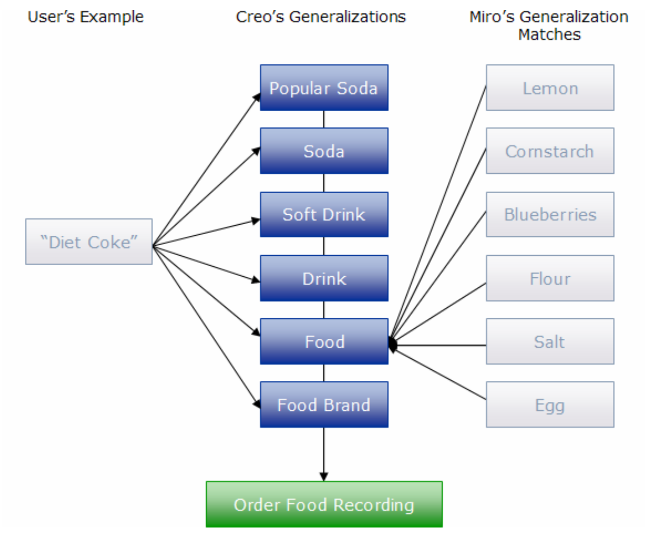
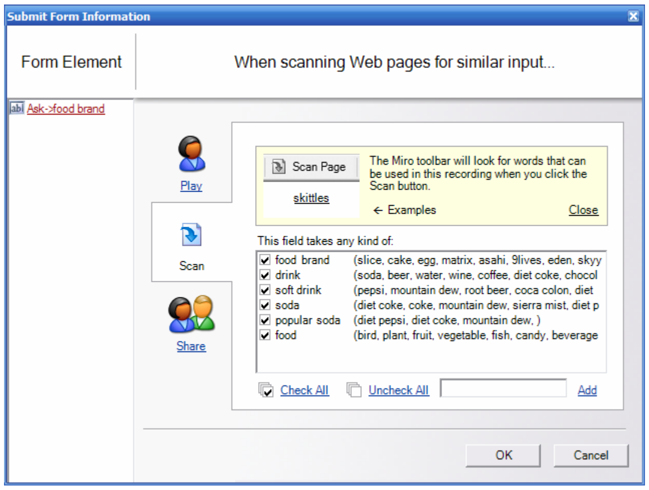


Figure 4: Foods in the recipe are matched to the user's recording

While this step is not required to create functional recordings, users can directly control the selected generalizations for a piece of input by clicking on the *Ask->Food brand* link in Figure 3 and clicking on the *Scan* tab:



The Miro toolbar will look for words that can be used in this recording when you click the Scan button.

This field takes any kind of:

✓ food brand	(slice, cake, egg, matrix, asahi, 9lives, eden, skyy
✓ drink	(soda, beer, water, wine, coffee, diet coke, chocol
✓ soft drink	(pepsi, mountain dew, root beer, coca colon, diet
✓ soda	(diet coke, coke, mountain dew, sierra mist, diet p
✓ popular soda	(diet pepsi, diet coke, mountain dew,)
✓ food	(bird, plant, fruit, vegetable, fish, candy, beverage

Figure 5: The user can control which generalizations are active with check boxes

The contextual help for this tab reads "The Miro Toolbar will look for words that can be used in this recording when you click the Scan button." By checking and un-checking items, users can directly control Creo's generalizations. For the user's example of "diet coke," Creo automatically selected the generalizations of "food brand, food, drink, soft drink, soda," and "popular soda."

Dealing with Messy Knowledge

The knowledge in ConceptNet and TAP (to a much lesser extent) is imprecise, however organized knowledge is not required for successfully matching the semantic context of a term to the generalizations of a recording's input. This is because a recording's list of generalizations attempts to be as long as possible, without being inaccurate.

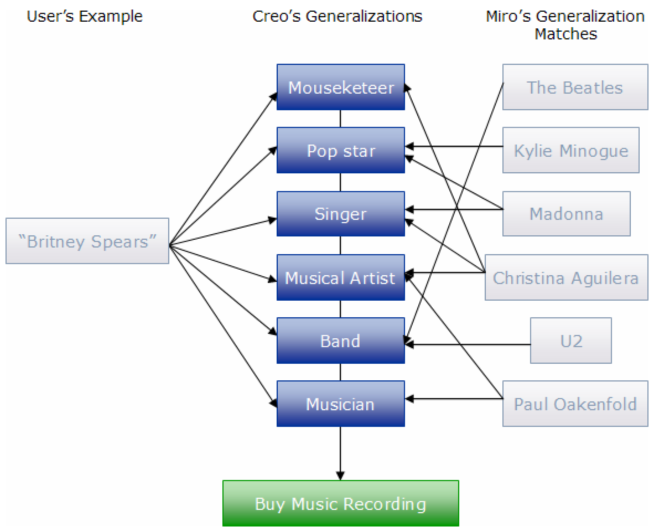


Figure 6: An example of matching messy knowledge

Because Creo has access to ConceptNet and TAP, users can create general-purpose recordings with a single example, allowing their Web browser to automate interactions with their favorite sites.

The topic of generalization also comes into play in invoking recordings: if the user creates a recording that works on certain kinds of data, seeing that data in a new situation presents an opportunity for the Web browser to invoke the recording.

DATA DETECTORS

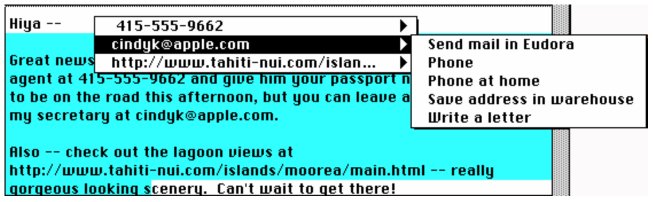
The purpose of Data Detectors is to recognize meaningful words and phrases in text, and to enable useful operations on them [17]. Data Detectors effectively turn plain text into a form of hypertext.

Previous Research

The majority of Data Detector research occurred in the late 1990s.

In 1997, Milind Pandit and Sameer Kalbag released the Intel Selection Recognition Agent [17]. The Intel Selection Recognition Agent was able to detect six types of data: geographic names, dates, email addresses, phone numbers, Usenet news groups, and URLs. These pieces of data were then linked to actions created by a programmer, like opening a Web browser to the URL, or sending an email message to an email address.

In 1998, Bonnie Nardi, James Miller and David Wright released Apple Data Detectors [18], which increased the types of data detected from six to thirteen. Apple Data Detectors were able to recognize phone numbers, fax numbers, street addresses, email addresses, email signatures, abstracts, tables of contents, lists of references, tables, figures, captions, meeting announcements, and URLs. Additionally, users could supply their own lists of terms they wanted Apple Data Detectors to recognize. Similar to the Intel Selection Recognition Agent, creating an action associated with data required programming.


```
Hiya --
Great news
agent at 415-555-9662 and give him your passport n
to be on the road this afternoon, but you can leave a
my secretary at cindyk@apple.com.

Also -- check out the lagoon views at
http://www.tahiti-nui.com/islands/moorea/main.html -- really
gorgeous looking scenery.  Can't wait to get there!
```

Figure 7: Apple Data Detectors (1998)

Also in 1998, Anind Dey, Gregory Abowd and Andrew Wood released CyberDesk [19]. CyberDesk detected eight kinds of data: dates, phone numbers, addresses, names, email addresses, GPS positions, and times. While this was less than the types supported by Apple Data Detectors, CyberDesk provided a more advanced framework for actions, including the ability to chain actions together, and to combine different pieces of data into the same action. CyberDesk also allowed for data detection on mobile devices. For instance, CyberDesk provided the ability to associate a GPS position with the action of loading a URL. Like the Intel Selection Recognition Agent and Apple Data Detectors, the only way to create new actions with CyberDesk was to program them.

The functionality of these Data Detectors has been integrated into several mainstream consumer products. Microsoft Office XP (released in 2001), provided data detection with a feature called Smart Tags, and the Google Toolbar 3.0 (released in 2005), added data detection to Web browsing, with a feature called AutoLink. Microsoft's Smart Tags currently recognizes eight types of data, although a developer can program additional data types and actions. Google's AutoLink currently recognizes three types of data: addresses, ISBNs and Vehicle Identification Numbers. The actions associated with these types of data are controlled by Google.

Back to the Future

One similarity of all of the research on Data Detectors in the late 1990s is each paper's future work section.

Programming by Example and End-User Programming

First, all of the research mentioned the importance of Programming by Example and end-user programming. The creators of the Intel Selection Recognition Agent wrote "We would like to enhance the Selection Recognition Agent along the lines of Eager [a Programming by Example system], allowing it to detect the repetition of action sequences in any application and automate these sequences" [17]. The creators of Apple Data Detectors wrote that a "goal is to complete a prototype of an end-user programming facility to enable end users to program detectors and actions, opening up the full Apple Data Detectors capability to all users" [18]. Finally, the creators of CyberDesk wrote that they were "investigating learning-by-example techniques to allow the CyberDesk system to dynamically create chained suggestions based on a user's repeated actions" [19].

Grammex (Grammars by Example) [20], released in 1999 and created by Henry Lieberman, Bonnie Nardi and David Wright, allowed users to create Data Detectors through Programming by Example. Like Creo, Grammex allowed users to define the actions to associate with data by providing demonstrations. However, Grammex was limited to the few Macintosh applications that were "recordable" (sending user action events to the agent) [20]. Similar to the Data Detectors preceding it, Grammex based its data detection on patterns of information. For instance, Grammex could learn how to detect email addresses if the user showed it several examples with the format *person@host*. Unfortunately, very few types of data outside of URLs, email addresses and phone numbers actually have a detectable structure, limiting the usefulness of such a system. This leads to the second "future work" topic mentioned by Data Detector researchers of the late 1990s: semantics.

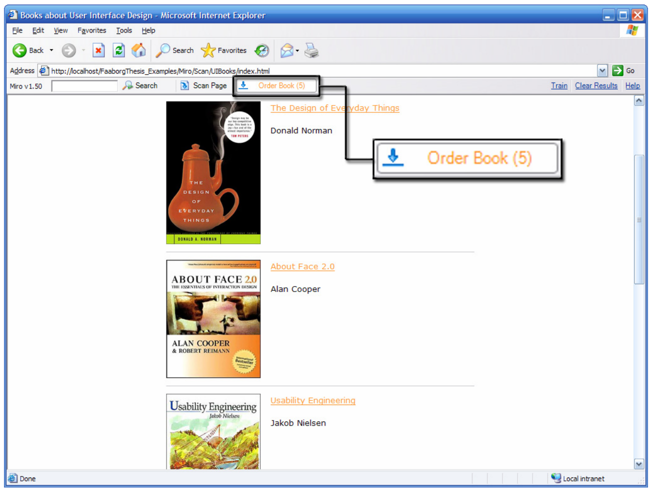
Semantics

The creators of Apple Data Detectors noted that relying on pattern detection has many limitations: "It is easy to imagine a company might choose a syntax for its product order numbers—a three digit department code followed by a dash followed by a four-digit product code—that would overlap with U.S. telephone number syntax, thus leading Apple Data Detectors to offer both telephone number and part-ordering actions…We can do little about these overlapping syntaxes without performing a much deeper, semantic interpretation of the text in which the pattern appears" [18]. The creators of CyberDesk also discussed the topic of semantic interpretation, writing that they were interested in "incorporating rich forms of context into CyberDesk, other than time, position, and meta-types" [19].

Miro

Miro expands the types of data that can be detected from the previous range of three types (Google's AutoLink) and thirteen types (Apple Data Detectors), to the full breadth of knowledge found in ConceptNet and TAP.

Earlier in the paper, Miro was shown linking foods in a recipe to the user's favorite grocery store. Below, Miro is turning plain text book titles into hyperlinks to the book at the user's favorite bookstore. Pressing the "Order Book" button in the Miro toolbar causes the personalized semantic hyperlinks to appear.



Books about User Interface Design - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/FaaborgThesis_Examples/Miro/Scan/UIBooks/index.html
Go
Miro v1.50 Search Scan Page Order Book (5)
Train Clear Results Help
The Design of Everyday Things
Donald Norman
Order Book (5)
About Face 2.0
Alan Cooper
Usability Engineering
Jakob Nielsen
Done
Local intranet

Figure 8: Miro recognizes the titles of books

It is important to note that the pages Miro reads are just normal pages on the Web. The pages do not contain any form of semantic markup. All of the semantic information is coming from the ConceptNet and TAP knowledge bases.

Leverage Commonsense Knowledge to Understand the Context of Text

Miro builds on three years of research on applying large scale knowledge bases to understanding the context of text, and using this commonsense knowledge to improve the usability of interactive applications [21].

Related Work

ARIA (Annotation and Retrieval Integration Agent) is a software agent that leverages ConceptNet to suggest relevant photos based on the semantic context of an email message [22].

ConceptNet has also been shown to be useful for determining the affective quality of text, allowing users to navigate a document based on its emotional content [23]. Also in the domain of text, by using ConceptNet to understand the semantic context of a message the user is typing, predictive text entry can be improved on mobile devices [24].

In the domain of speech recognition, this same approach can also be used to streamline the error correction user interfaces of speech recognition systems [25]. Additionally, ConceptNet can be used to detect the gist of conversations, even when spontaneous speech recognition rates fall below 35% [26].

In the domain of advising, ConceptNet has been shown to be useful for matching a novice's knowledge to an expert system [27].

Both ConceptNet and TAP have also been found to be incredibly useful in the domain of search, demonstrated by the prototypes GOOSE (Goal-Oriented Search Engine) [28] and ABS (Activity Based Search) [9], respectively.

Dealing with the Ambiguity of Natural Language

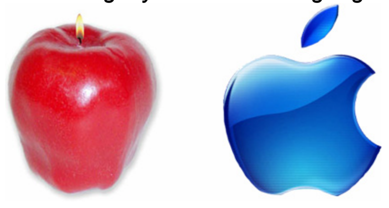


Figure 9: I want to buy an Apple

The most significant challenge that Miro faces in its task of data detection is dealing with the ambiguity of natural language. For instance, because of the way Open Mind was created, the following two statements are in ConceptNet:

```
(IsA "apple" "computer")
(IsA "apple" "fruit")
```

It is important to deal with ambiguity well, because incorrectly matching a user's goals leads to a very poor user experience:

Mr. Thurrott typed the word "nice." Up popped a Smart Tag offering to book a flight to Nice, France using Microsoft's Expedia website. When he typed the word "long," up popped a smart tag from ESPN offering more information on Oakland Athletics centerfielder Terrence Long. As Thurrott put it, "Folks, this is lame" [29].

Google's AutoLink team avoided this problem entirely by opting to only detect three kinds of data that are already designed to be unique (addresses, ISBNs and VINs).

Miro addresses this problem through a method of patterns and thresholds that can be modified by the user. For instance, the term "apple" by itself is ambiguous, but if it is surrounded by terms like Dell and Toshiba, the meaning becomes clearer. This is, of course, far from a perfect solution. For instance, if someone wrote a blog about how they spilled apple juice all over their brand new apple G5, Miro will have difficulty understanding the apples. While Miro does occasionally make mistakes, we believe the benefit it provides users is valuable nonetheless.

PUTTING END-USERS IN CONTROL OF THEIR DATA AND SERVICES

Both Microsoft and Google have received a strong outcry of criticism for their Data Detectors, Smart Tags and AutoLink [29, 30]. The equality of the criticism is surprising given the considerable difference between Microsoft and Google's current public image. Microsoft actually pulled Smart Tags as being a feature of Internet Explorer 6 shortly before the release of Windows XP due to public outcry. In an article in the Wall Street Journal, columnist Walter Mossberg wrote, "Using the browser to plant unwanted and unplanned content on these pages--especially links to Microsoft's own sites--is the equivalent of a printing company adding its own editorial and advertising messages to the margins of a book it has been hired to print. It is like a television-set maker adding its own images and ads to any show the set is receiving" [30].

Together, Miro and Creo solve this problem, by enabling end users to be in control of their data and services.

EVALUATION

In this section we describe two sets of evaluations: (1) a series of evaluations done during the iterative design process of Creo conducted with a total of 10 subjects, and (2) a final evaluation conducted with 34 subjects to assess the overall effectiveness of our system.

Evaluating the User Interface Design

While designing Creo and Miro, we realized that the critical factor to their success would not be technical limitations, since systems built on top of ConceptNet and TAP have worked fine in the past. Instead, the critical factor to their success would be usability. In particular, the user interface design of Creo was vital. Programming by Example systems often extol the virtues of enabling novice users to train their computer, but then end up being simply GUIs for Computer Scientists ([20]). We followed an iterative design process during Creo's creation, formally evaluating each iteration, before designing the next.

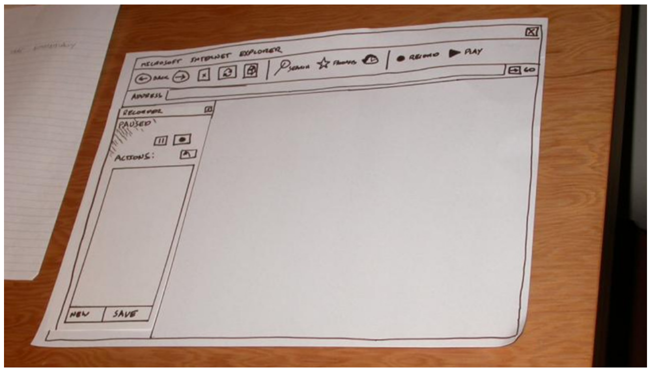


Figure 10: A paper prototype of Creo Version 1

The first version of Creo's user interface was evaluated with three users during a paper prototyping session. Users were given a briefing of the purpose of the software application, and were then asked to interact with the application, thinking out loud. This evaluation led to changes regarding how Creo provides confirmations and feedback.

The second version of Creo's user interface was evaluated with four user interface designers, using a computer prototype. Only the front end of the software was implemented, and the scenarios were entirely simulated. Each designer independently conducted a heuristic evaluation of the software, and wrote a list of usability problems they found while interacting with Creo, categorized by severity. Changes to the interface from this evaluation included reworking how Creo interacts with the file system, and making visual changes to how Creo represents its current mode.

The third version of Creo's user interface was evaluated in a usability test with three novice users, using a fully functional prototype running as part of Internet Explorer. Changes to Creo's interface from this evaluation included removing controls not relevant to the current mode to reduce visual complexity, and a re-evaluation of which controls were absolutely necessary. Additionally, Creo's generalization interface was entirely redesigned.

The fourth version of Creo's user interface was evaluated as part of a larger scale user study to determine the software's overall effectiveness.

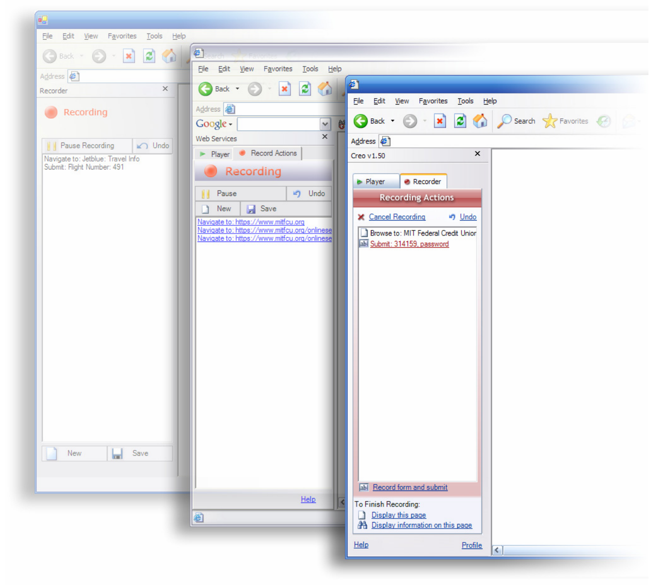


Figure 11: The evolution of Creo's design, Versions 2, 3 and 4

Determining the Software's Overall Effectiveness

The purpose of the fourth user evaluation was to (1) conclude if the overall system made users more effective at completing a task, and (2) to conclude if users understood the utility of the software, and if they would use software applications like Creo and Miro if they were included in their Web browser.

The evaluation was run with a total of 34 subjects, 17 male and 17 female. In Part 1 of the evaluation, 17 people were in the experimental group and 17 people were in the control group. The average age of the subjects was 29.3, with a range of 19 to 58. 26% of subjects had no programming experience, and all subjects were familiar with using the Web. Subjects were compensated $10.

Part 1: Evaluating the Effectiveness of Miro

In the first part of the experiment, subjects were asked to order 11 ingredients in a recipe for Blueberry Pudding Cake. The experimental group of subjects had access to the

Miro toolbar, which could recognize common foods and automatically link them to the subject's grocery store. The control group of subjects completed the same task, but used Internet Explorer with Miro turned off. Subjects in the control group were allowed to complete the task however they naturally would. All of the subjects were instructed to complete the task at a natural pace, and not to treat the experiment like a race. We hypothesized that the experimental group would be able to complete the task significantly faster than the control group.

Part 2: Evaluating the Effectiveness of Creo

In the second part of the experiment, all of the subjects were asked to create a recording with Creo that could order any type of food at a grocery store. Subjects completed this task after being shown an example of how Creo works. We showed the subjects a single demonstration of how to use Creo to look up a movie at IMDB. We chose to do this because unlike the three preceding usability studies, for this evaluation we were interested in capturing the average time it took a slightly experienced subject to create a simple recording. We hypothesized that subjects would be able to successfully complete this task in a trivial amount of time.

Quantitative Results

The experimental group completed the task in Part 1 in an average time of 68 seconds, with a standard deviation of 20 seconds. The control group completed the task in an average time of 139 seconds with a standard deviation of 58 seconds. These results are statistically significant ($p<.001$). These results are also consistent with the study conducted by the Intel Selection Recognition Agent authors, finding that interface "saved both time and effort, in some cases over 50%" [17].

The range of results from the control group in Part 1 is due to the fact that subjects were asked to complete the task however they naturally would. There was a large amount of variability in the way subjects transferred information between the recipe and the grocery store site. Some subjects relied heavily on keyboard shortcuts, using alt-tab to switch windows and tab to switch which control on the grocery store page had the focus. Some subjects double clicked to select a word, and triple clicked to select a full line. Other subjects retyped every ingredient instead of copying and pasting. Since they would often hold three to four ingredients in their own memory at a time, this usually turned out to be faster.

In Part 2, subjects completed the task in 26 seconds, with a standard deviation of 5 seconds. This means that even for interacting with a list of 11 items, it would be faster to train Creo first, and then use Miro to turn the information into hyperlinks. In the following chart, the time for Part 2 is represented as an overhead cost for the experimental group's time for Part 1.

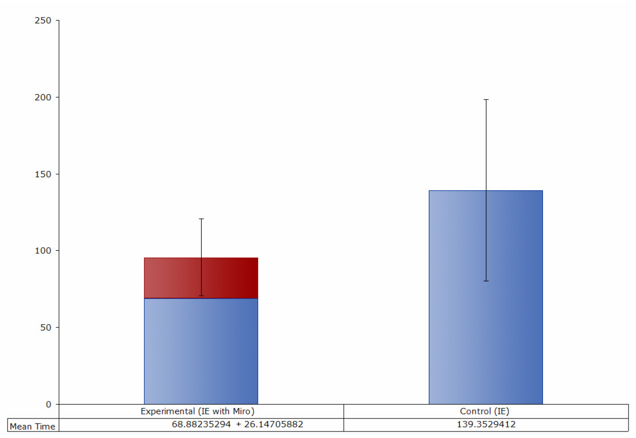


Figure 12: The time it took the control and experimental groups to complete the task

Debriefing Questionnaire

The debriefing questionnaire contained several Likert scale questions asking the subject's impressions of the software's usability (shown below), and if they would actually use the software.

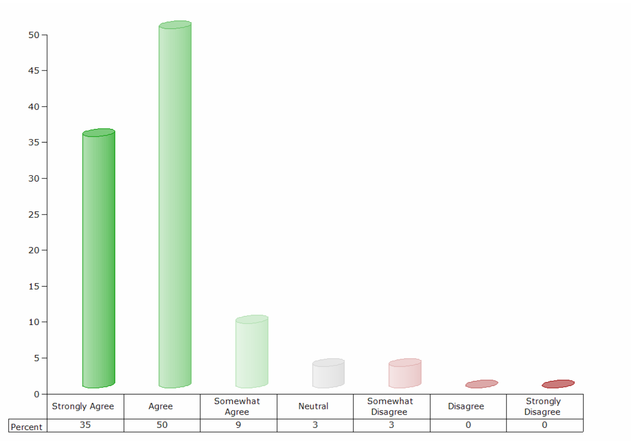


Figure 13: Did the subjects find Miro easy to use

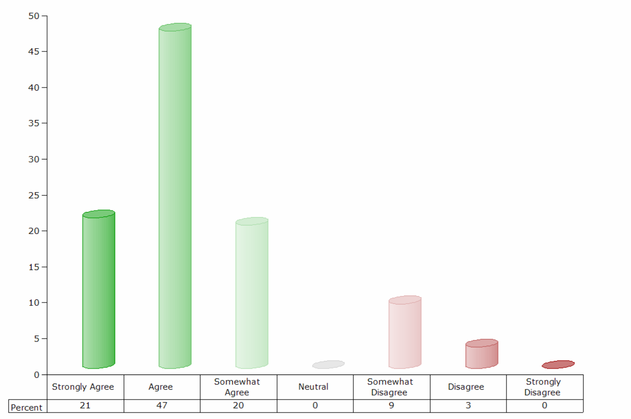


Figure 14: Did the subjects find Creo easy to use

Asked if they would use the software, 85% of subjects responded they would use Creo, and 100% of subjects responded they would use Miro. We have implemented a way for users to easily share the functionality of recordings they create with Creo without sharing any of their personal information, (which Creo automatically detects and stores separately). So it is technically possible for a subset of users to use Creo, and for everyone to use Miro.

To analyze how Creo and Miro make users more effective compared to using a conventional Web browser, this evaluation focused on a single example of using Creo and Miro. We did not study the breadth tasks that Creo and Miro can perform for two reasons, (1) the ConceptNet and TAP knowledge bases are rapidly growing, and (2) the respective teams at MIT and Stanford responsible for the creation of these knowledge bases have already performed evaluations of their breadth [1-4, 7-10].

FUTURE WORK

When Things Go Wrong

While the ConceptNet and TAP knowledge bases are very large, they are certainly not complete. To assist the user with situations where Miro fails to detect a specific piece of information, we have developed a Training Wizard. This wizard consists of a three-step process: (1) ask the user what information should have been detected, (2) ask the user what the information is (by having them fill out a sentence), and (3) ask the user which recording from Creo should have been activated. In most cases, Miro can provide intelligent defaults for at least two of these three steps, creating a collaborative learning interface between Miro and the user. In the first step, Miro performs predictive text entry on what the user types, based on the terms on the current Web page. In the second step, Miro attempts to describe the concept itself. In some cases Miro will know what the concept is, but not how it relates to the current set of recordings created by Creo. In the third step, Miro attempts to check which recordings should have been activated based on the information in the previous step. This is useful when providing new pieces of information. For instance, once the user tells Miro that "Eastern Standard Tribe" is a book, Miro knows what to do with books.

For the situations where a recording breaks due to a change with a Web site, we have a developed a debugging mode.

Learning from the Web

We are exploring using natural language processing to enable Miro to learn new pieces of information by reading Web pages. Miro takes the text of the page the user is on and (1) performs sentence boundary detection, (2) isolates the nouns and words used for pattern matching, (3) lemmatizes the text and, (4) matches the text against 11 different patterns. For instance, the sentence "The Killers is a really great band" can be easily parsed to `(IsA "the killers" "band")`. When Miro finds a new piece of information that matches the current set of recordings, it displays the activated recording (as if the knowledge came out of ConceptNet or TAP), and then watches to see if the user clicks on it. We believe an approach like this could be used to quickly grow broad knowledge bases, if a system like Miro were to be used by a large number of users.

Improving Mobile Browsing

We are also researching ways to use the recordings created with Creo to improve mobile browsing. Users don't want to *browse* on mobile devices, they want to complete actions. Adeo is a mobile application we have developed that allows users to invoke the recordings they have created with Creo. Adeo reduces complex procedures to the minimal amount of required input and output.

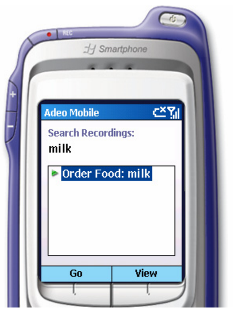

Adeo Mobile
Search Recordings:
milk
Order Food: milk

Figure 15: Adeo streamlines mobile browsing

CONCLUSION

One ethnographic observation from our user studies is notable: subjects with programming experience had more difficulty using Creo than subjects without any programming experience. While at first this seems counterintuitive, we believe it has to do with the subject's expectations. Specifically, technical subjects had more difficulty believing that Creo could generalize their single example. This is because subjects with a programming background were familiar with how computers function. They had never seen a computer behave this intelligently before.

Creo and Miro, like many other interactive applications [21-28], would not be able to generalize information and anticipate their users goals without access to the knowledge stored in ConceptNet and TAP. This paper has demonstrated the effect these knowledge bases can have on the research areas of Programming by Example and Data Detection. However, we believe many other types of interactive applications can benefit from access to this knowledge as well.

Usability is improved by making it easier for humans to understand computers. However the reverse is true as well. ConceptNet and TAP improve usability by making it easier for computers to understand humans.

ACKNOWLEDGEMENTS

The authors would like to thank many people. Unfortunately, since this is a blind review, we can't thank them yet. They were very helpful though.

REFERENCES

1. Singh, P. *The Public Acquisition of Commonsense Knowledge*. Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access. Palo Alto, California. (2002).
2. Singh, P., Barry, B., Liu, H. *Teaching Machines about Everyday Life*. BT Technology Journal. (2004).
3. Singh, P., Lin, T., Mueller, E., Lim, G., Perkins, T., Zhu, W.L. *Open Mind Common Sense: Knowledge Acquisition from the General Public*. Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems. Irvine, California. (2002).
4. Liu, H., Singh, P. *ConceptNet: a Practical Commonsense Reasoning Toolkit*. BT Technology Journal. (2004).
5. Fellbaum, C. *WordNet: An Electronic Lexical Database*. MIT Press. Cambridge, Massachusetts. (1998).
6. *TAP: Building the Semantic Web*. http://tap.stanford.edu/.
7. Guha, R., McCool, R. *TAP: a Semantic Web Platform*. Computer Networks: The International Journal of Computer and Telecommunications Networking. Volume 42, Issue 5. (2003).
8. Guha, R., McCool, R. *A System for Integrating Web Services into a Global Knowledge Base*. http://tap.stanford.edu/sw002.html.
9. Guha, R., McCool, R., Miller, E. *Semantic Search*. Proceedings of the 12th International Conference on World Wide Web. Budapest, Hungary. (2003).
10. McCool, R., Guha, R., Fikes, R. *Contexts for the Semantic Web*. http://tap.stanford.edu/contexts.pdf.
11. Lenat, D. *CYC: a Large-Scale Investment in Knowledge Infrastructure*. Communications of the ACM. Volume 38, Issue 11. (1995).
12. Lieberman, H. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann. San Francisco, California. (2001).
13. Bauer, M., Dengler, D., Paul, G. *Instructible Information Agents for Web Mining*. Proceedings of the 5th International Conference on Intelligent User Interfaces. New Orleans, Louisiana. (2000).
14. Miller, R., Myers, B. *Creating Dynamic World Wide Web Pages by Demonstration*. Technical Report CMU-CS-97-131 (and CMU-HCII-97-101), CMU School of Computer Science. (1997).
15. Sugiura, A., Koseki, Y. *Internet Scrapbook: Automating Web Browsing Tasks by Demonstration*. Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology. San Francisco, California. (1998).
16. Safonov, A. *Web Macros by Example: Users Managing the WWW of Applications*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 99). Pittsburgh, Pennsylvania. (1999).
17. Pandit, M., Kalbag, S. *The Selection Recognition Agent: Instant Access to Relevant Information and Operations*. Proceedings of the 2nd International Conference on Intelligent User Interfaces. (1997).
18. Nardi, B., Miller, J., Wright, D. *Collaborative, Programmable Intelligent Agents*. Communications of the ACM, Vol. 41, No. 3. (1998).
19. Dey, A., Abowd, G., Wood, A. *CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services*. Proceedings of the 3rd International Conference on Intelligent User Interfaces. San Francisco, California. (1998).
20. Lieberman, H., Nardi, B., Wright, D. *Grammex: Defining Grammars by Example*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 98). Los Angeles, California. (1998).
21. Lieberman, H., Liu, H., Singh, P., Barry, B. *Beating Some Common Sense into Interactive Applications*. AI Magazine, Winter 2004. (2004).
22. Lieberman, H., Liu, H. *Adaptive Linking between Text and Photos Using Common Sense Reasoning*. Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002. Malaga, Spain. (2002).
23. Liu, H., Selker, T., Lieberman, H. *Visualizing the Affective Structure of a Text Document*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 03). Ft. Lauderdale, Florida. (2003).
24. Stocky, T., Faaborg, A., Lieberman, H. *A Commonsense Approach to Predictive Text Entry*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 04). Vienna, Austria. (2004).
25. Lieberman, H., Faaborg, A., Daher, W., Espinosa, J. *How to Wreck a Nice Beach You Sing Calm Incense*. Proceedings of the International Conference on Intelligent User Interfaces (IUI 2005). San Diego, California. (2005).
26. Eagle, N., Singh, P. *Context Sensing Using Speech and Common Sense*. Proceedings of the NAACL/HLT 2004 workshop on Higher-Level Linguistic and Other Knowledge for Automatic Speech Processing. (2004).
27. Kumar, A., Sundararajan, S., Lieberman, H. *Common Sense Investing: Bridging the Gap Between Expert and Novice*. Proceedings of the Conference on Human Factors in Computing Systems (CHI 04). Vienna, Austria. (2004).
28. Liu, H., Lieberman, H., Selker, T. *GOOSE: A Goal-Oriented Search Engine With Commonsense*. Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002. Malaga, Spain. (2002).
29. Kaminski, C. *Much Ado About Smart Tags*. http://www.alistapart.com/articles/smarttags/.
30. Mossberg, W. *Microsoft Will Abandon Controversial Smart Tags*. http://ptech.wsj.com/archive/ptech-20010628.html.