B978-0-12-381541-5.00004-3, 00004

CHAPTER

c0020 A goal-oriented Web browser

4

Alexander Faaborg, Henry Lieberman
MIT Media Laboratory

ABSTRACT

p0220 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 chapter 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.

p0225 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.

p0230 An evaluation with 34 subjects found that they were more efficient using our system, and that the subjects would use features like these if they were integrated into their Web browser.

s0010 INTRODUCTION

p0235 In this chapter 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 based on data from 34 users, and evaluations of our software's user interface during an iterative design process.

p0240 Finally, we conclude with a discussion of how large-scale knowledge bases of semantic information can be leveraged to improve human–computer interaction.

s0015 CONTRIBUTIONS

p0245 This chapter presents five contributions. First, the chapter demonstrates how a Programming by Example system can be used to automate repetitive tasks on the Internet, saving users time.

p0250 The central problem of programming-by-example systems is generalization. The second contribution is to show how two large knowledge bases of semantic information, MIT's ConceptNet and Stanford's TAP (The Alpiri Project), can be leveraged to improve generalization.

p0255 The chapter's third contribution is to show how a Programming by Example system can work together with a data detector, solving both recording and invocation in an integrated way.

Cypher, 978-0-12-381541-5

p0260 Commercially available data detectors like Microsoft Smart Tags and Google AutoLink limit users in both the types of data that can be detected and the services that can be performed on those types of data. This chapter's fourth contribution is to show how combining a programming-by-example system with a data detector enables users to control the services associated with their data.

p0265 Finally, this chapter demonstrates how a Web browser can proactively detect a user's potential goals while they browse the Web. Although current 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. The overall contribution of this chapter is to demonstrate how the integration of large knowledge bases of semantic information, a programming-by-example system, and a data detector can result in a goal-oriented Web browser.

s0020 TEACHING COMPUTERS THE STUFF WE ALL KNOW

p0270 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.

s0025 Open mind

p0275 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 (Singh, 2002; Singh et al., 2002; Singh, Barry, & Liu, 2004). Currently, the Open Mind Common Sense Project has collected over 806,000 facts from over 19,000 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." Although Open Mind does not contain a complete set of all the commonsense knowledge found in the world, its knowledge base is sufficient to be useful in real world applications.

s0030 ConceptNet

p0280 Using natural language processing, the Open Mind knowledge base was mined to create ConceptNet (Liu & Singh, 2004), a large-scale semantic network currently containing over 250,000 commonsense facts. ConceptNet consists of machine-readable logical predicates of the form:

u0210
```
(IsA "tennis" "sport")
```
Au2

u0215
```
(EventForGoalEvent "play tennis" "have racket")
```

p0295 ConceptNet is similar to WordNet (Fellbaum, 1998) in that it is a large semantic network of concepts, however ConceptNet contains everyday knowledge about the world, whereas 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" (Fellbaum, 1998).

s0035

Stanford TAP

p0300 The Stanford TAP knowledge base was created to help bootstrap the Semantic Web (Guha & McCool, 2002; Guha & McCool, 2003a, 2003b; Guha, McCool, & Miller, 2003; McCool, Guha, & Fikes, 2003). 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 (McCool et al., 2003). 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 (Lenat, 1995). 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" (Guha & McCool, 2002). For this project, the TAP knowledge base has been modified to match the formatting of ConceptNet.

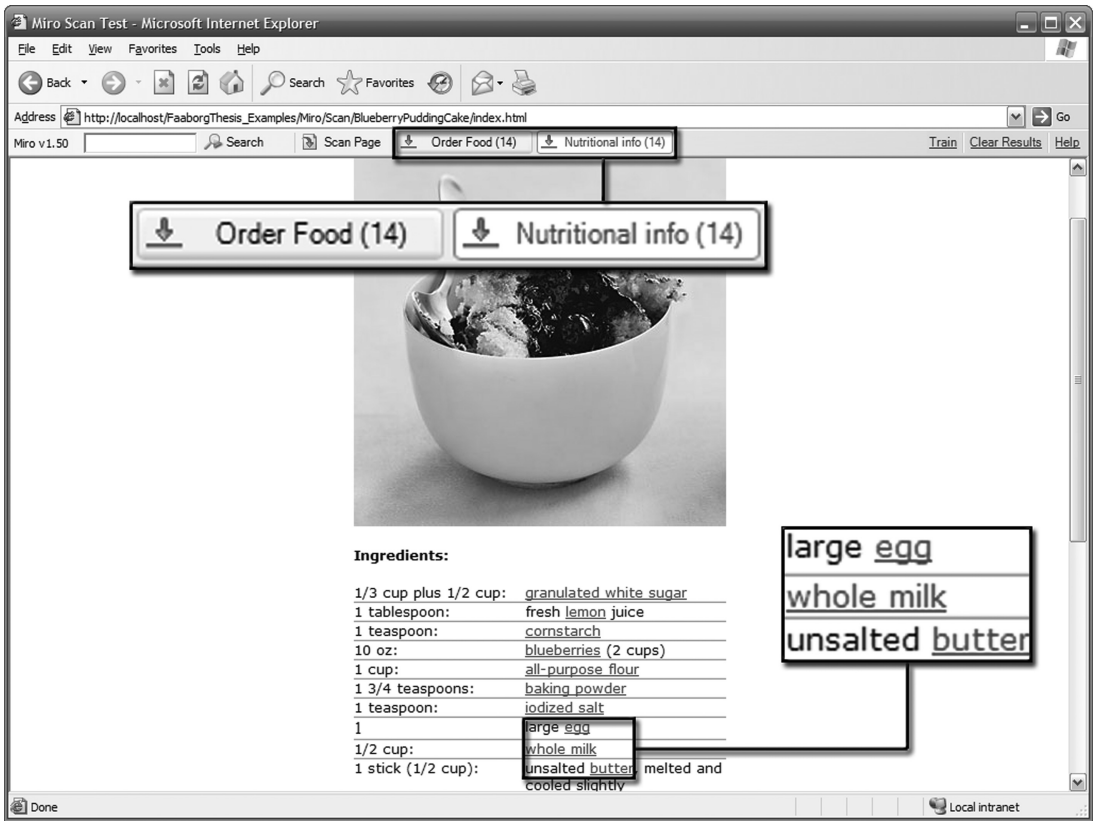
s0040

A GOAL-ORIENTED WEB BROWSER

p0305 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:

p0310 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.

p0315 It is important to note that while this "recipe to grocery store" example is used throughout the chapter for the purposes of clarity, Creo can automate interactions with other kinds of sites on the Web (not just grocery stores), and Miro can detect any type of data described in ConceptNet and TAP (not just foods).



f0010 **FIGURE 4.1**

Au24 Automatically associating a user's high-level goals with the content of a Web page.

PROGRAMMING BY EXAMPLE

s0045

p0320 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. Because goals tend to re-occur over time, the user is faced with having to tediously repeat procedures. A potential solution to this dilemma is programming by example (Lieberman, 2001). 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 with variables that usually accept a particular kind of data.

s0050

Previous research

p0325 The TrIAs (Trainable Information Assistants) by Mathias Bauer (Bauer, Dengler, & Paul, 2000; Lieberman, 2001) 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.

p0330 Turquoise, by Rob Miller and Brad Myers (Miller & Myers, 1997), is a programming-by-example system that allows nontechnical users to create dynamic Web pages by demonstration. For instance, Turquoise can be used to create a custom newspaper by copying and pasting information, or to automate the process of aggregating multiple lunch orders into the same order.

p0335 Similar to Turquoise, the Internet Scrapbook, by Atsushi Sugiura and Yoshiyuki Koseki (Sugiura & Koseki, 1998; Lieberman, 2001) is a programming-by-example system that allows users with few 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.

p0340 Web Macros, created by Alex Safonov, Joseph Konstan, and John Carlis (Safonov, 1999), allows users to interactively record and play scripts that produce pages that cannot be directly bookmarked.

s0055

A new approach to generalization

p0345 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 problems of generalizing procedures and proactively seeking invocation opportunities by using large knowledge bases of semantic information.

s0060

Creo

p0350 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.

p0355 Creo turns red to indicate that it is in recording mode, and it captures the user's action of navigating to FreshDirect (http://www.freshdirect.com).

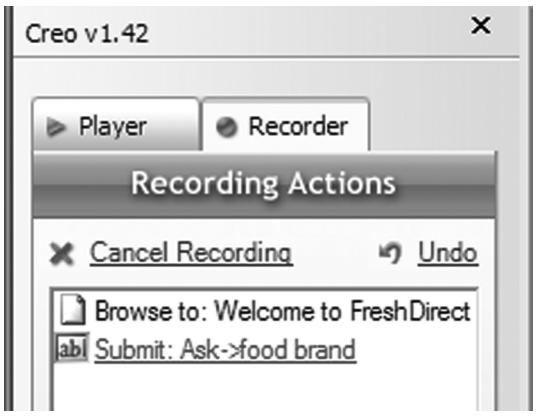
p0360 Next, the user searches FreshDirect for an example food, "Diet Coke". Creo detects that this was an example, and automatically generalizes the concept to "food brand."

p0365 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.

p0370 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 of the

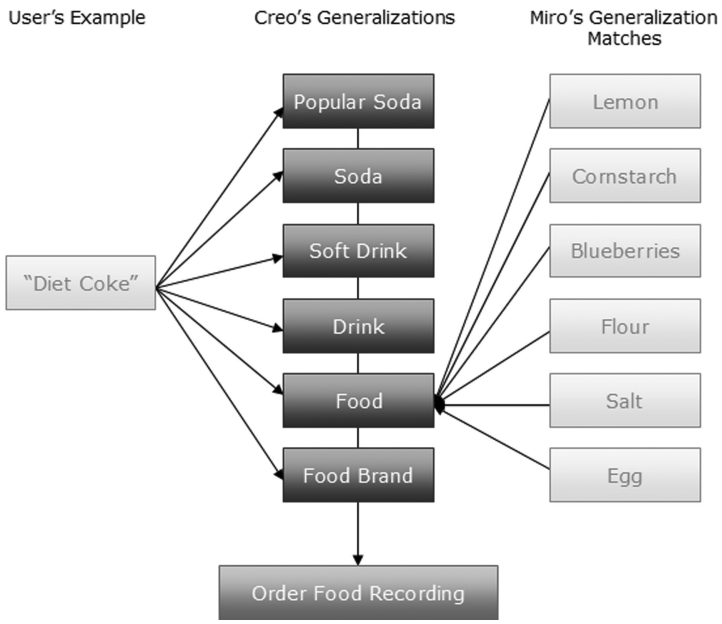
f0015 **FIGURE 4.2**

Au25 Creo learns how to interact with a Web site by watching the user's demonstration.



f0020 **FIGURE 4.3**

Au26 Creo automatically generalizes the user's input.



f0025 **FIGURE 4.4**

Foods in the recipe are matched to the user's recording.

generalizations for clarity. In this particular case, "food" was the second most general generalization of "Diet Coke," as shown in Figure 4.4.

p0375 Although 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 shown in Figure 4.5 and clicking on the *Scan* tab: Au1

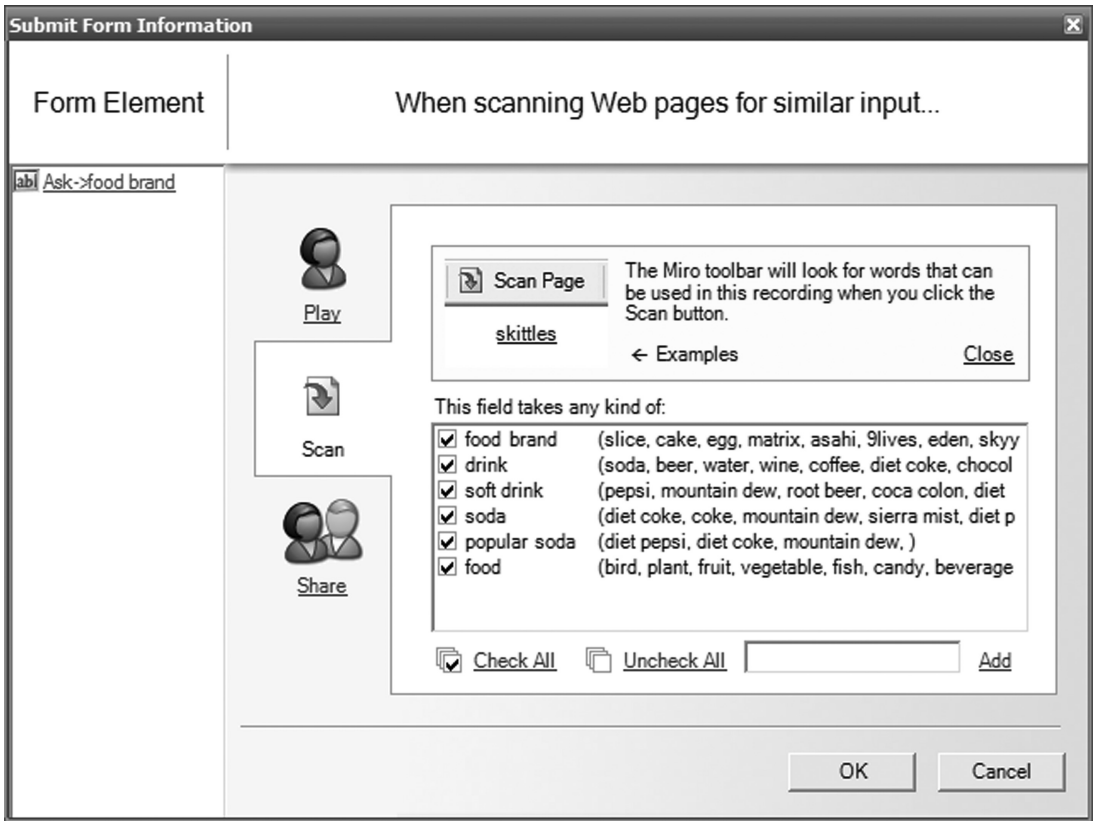
p0380 The contextual help for this tab reads, "The Miro Toolbar will look for words that can be used in this recording when you click the Scan button." By checking and unchecking items, users can directly control Creo's generalizations. For the user's example of "Diet Coke", Creo automatically selected the generalizations of food brand, food, drink, soft drink, soda, and popular soda.

p0385 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.

p0390 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.

s0065 DATA DETECTORS

p0395 The purpose of data detectors is to recognize meaningful words and phrases in text, and to enable useful operations on them (Pandit & Kalbag, 1997). Data Detectors effectively turn plain text into a form of hypertext.



f0030 **FIGURE 4.5**

Au27 The user can control which generalizations are active with check boxes.

s0070 ## Previous research

p0400 The majority of data detector research occurred in the late 1990s.

p0405 In 1997, Milind Pandit and Sameer Kalbag released the Intel Selection Recognition Agent (Pandit & Kalbag, 1997). 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 a URL or sending an email message to an email address.

p0410 In 1998, Bonnie Nardi, James Miller and David Wright released Apple Data Detectors (Nardi, Miller, & Wright, 1998), which increased the types of data detected from 6 to 13. 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.

f0035 **FIGURE 4.6**

Au28 Apple Data Detectors (1998).

p0415 Also in 1998, Anind Dey, Gregory Abowd and Andrew Wood released CyberDesk (Dey, Abowd, & Wood, 1998). CyberDesk detected eight kinds of data: dates, phone numbers, addresses, names, email addresses, GPS positions, and times. Although 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.

p0420 Like the Intel Selection Recognition Agent and Apple Data Detectors, the only way to create new actions with CyberDesk was to program them.

p0425 The functionality of these Data Detectors has been integrated into several consumer products. Released in 1999 by aQtive, onCue monitored information copied to the clipboard and suggested relevant Web services and desktop applications. Like earlier data detectors, onCue did not perform any level of semantic analysis, and it simply associated words with various search engines, an encyclopedia, and a thesaurus. However, onCue differed from previous data detectors in that it was also able to detect different structures of information, such as lists and tables, and then suggest relevant ways to visualize that information, including dancing histograms, pieTrees, and the charts available in Microsoft Excel. Both the service and recognizer components in the onCue framework (called Qbits) required a developer to program (Dix, Beale, & Wood, 2000). 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.

s0075 Back to the future

p0430 One similarity of all of the research on data detectors in the late 1990s is each paper's future work section.

s0080 *Programming by example and end-user programming*

p0435 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" (Pandit & Kalbag, 1997). 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" (Nardi et al., 1998). 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" (Dey et al., 1998).

p0440 Grammex (Grammars by Example) (Lieberman, Nardi, & Wright, 1998), 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) (Lieberman et al., 1998). 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.

s0085 ### *Semantics*

p0445 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" (Nardi et al., 1998). 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" (Dey et al., 1998).

s0090 ## Miro

p0450 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.

p0455 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.

s0095 # LEVERAGING COMMONSENSE KNOWLEDGE TO UNDERSTAND THE CONTEXT OF TEXT

p0460 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 (Lieberman, Liu, Singh, & Barry, 2004).

s0100 ## Related work

p0465 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 (Lieberman & Liu, 2002).

p0470 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 (Liu, Selker, & Lieberman, 2003). Also in the domain of text analysis, by using ConceptNet to understand the semantic context of a message the user is typing, predictive text entry can be improved on mobile devices (Stocky, Faaborg, & Lieberman, 2004).

p0475 In the domain of speech recognition, this same approach can also be used to streamline the error correction user interfaces of speech recognition systems (Lieberman, Faaborg, Daher, & Espinosa, 2005). Additionally, ConceptNet can be used to detect the gist of conversations, even when spontaneous speech recognition rates fall below 35% (Eagle & Singh, 2004).

p0480 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) (Liu, Lieberman, & Selker, 2002) and ABS (Activity Based Search) (Guha et al., 2003).

s0105 ## Dealing with the ambiguity of natural language

p0485 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:

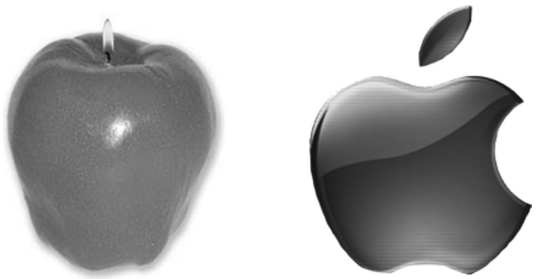
u0220
```
(IsA "apple" "computer")
```
u0225
```
(IsA "apple" "fruit")
```
Au2

p0500 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."*
>
> **(Kaminski, 2001)**

p0505 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).

p0510 Miro begins to address this problem by leveraging the semantic context of surrounding terms. For instance, the term "apple" by itself is ambiguous, but if it is surrounded by terms like "Dell" and "Toshiba," the meaning becomes clearer. However, algorithms to re-rank a term's semantic value based on the surrounding context are far from perfect. In general, our current algorithm performs much better on semistructured data (like a list of items) compared to parsing paragraphs of text. For instance, if someone wrote a blog entry about how they "spilled apple juice all



f0040 **FIGURE 4.7**

Au29 The ambiguity of "apple."

over a brand new apple MacBook Pro," Miro will have difficulty understanding the apples. Although Miro does occasionally make mistakes, we believe the benefit it provides users is valuable nonetheless. Using large knowledge bases of semantic information to determine the specific semantic value of a particular term remains an interesting challenge for future research.

s0110 PUTTING END USERS IN CONTROL OF THEIR DATA AND SERVICES

p0515 Both Microsoft and Google have received a strong outcry of criticism for their Data Detectors, Smart Tags, and AutoLink (Kaminski, 2001; Mossberg, 2001). The equality of the criticism is surprising given the considerable difference between Microsoft's and Google's current public images. Microsoft actually pulled the feature Smart Tags from Internet Explorer 6 shortly before the release of Windows XP because of 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" (Mossberg, 2001).

p0520 Together, Miro and Creo solve this problem by enabling end users to define the services associated with particular types of data.

s0115 IMPLEMENTATION

p0525 This section briefly covers the implementation of Creo and Miro. Further information can be found in Alexander Faaborg's masters thesis, *A Goal-Oriented User Interface for Personalized Semantic Search* (Faaborg, 2005).

AN EXTENSION FOR INTERNET EXPLORER 6?

p0530 Although the author of this chapter is currently a principal designer on Firefox at Mozilla, he notes that development work on Creo and Miro began long before the release of Firefox 1.0. While the system detailed in this chapter was built to be part of IE6, Firefox's excellent support for extension development would have made implementing Creo and Miro considerably easier today. Examples of end-user programming systems for the Web implemented as Firefox extensions (and in general implemented after Firefox's initial release in late 2004) can be found throughout this book, in Chapter 1, Chapter 2, Chapter 3, Chapter 5, Chapter 6, Chapter 9 and Chapter 15.

s0125 ## Implementation of Creo

s0130 ### *Recording actions on the web*

p0535 Unlike many of the previous Programming by Example systems for the Web, which are implemented using a proxy server, Creo is integrated directly into a Web browser. Creo's integration with Internet Explorer provides two core functions: (1) *monitoring*, the ability to directly capture the user's actions, and what the user is currently looking at; and (2) *impersonation*, the ability to recreate actions inside the Web browser and make it appear as if an actual user was completing them.

p0540 The basic set of actions that Creo must be able to monitor and impersonate consists of capturing navigation events (monitoring), navigating (impersonation), scraping a form (monitoring), filling out a form (impersonation), being instructed to scrape text from a Web site (monitoring), and scraping text from a Web site (impersonation).

p0545 From a Web site's perspective, there is no difference between the user completing actions by controlling their Web browser, and Creo completing actions by controlling the Web browser. Aside from the fact that Creo is faster (which actually caused problems with some Web sites, so it was subsequently slowed down), Creo does a perfect job of impersonating the user's actions. This, of course, does not include CAPTCHA tests or completing any other type of higher level perceptual or cognitive challenges.

s0135 ### *Generalizing information*

p0550 What differentiates programming-by-example systems like Creo from basic macro recorders is their ability to generalize information. First, Creo determines if the input should be generalized or remain static based on three heuristics: (1) if the input consists of personal information, (2) if the name of the text field matches a predetermined list of fields that should remain static, and (3) the number of generalizations of the input. If Creo determines that the input should be generalized, it looks up the relevant IsA relationships for the input in ConceptNet and TAP. For instance, the input "Diet Coke" is found in ConceptNet and TAP in statements like:

u0230
```
(IsA "diet coke" "food brand")
```
Au2

p0560 As shown in the earlier example, the full list of generalizations of Diet Coke consists of food brand, food, drink, soft drink, soda, and popular soda. These generalizations are written to the recording's XML file, and leveraged by Miro when determining if the recording should be activated based on the semantic context of a Web page.

s0140 ## Implementation of Miro

p0565 Miro determines the user's potential goals based on the Web page they are looking at by matching the generalizations of terms and phrases on the page to the set of generalizations associated with recordings created using Creo. For instance, the recipe page shown earlier activated the "Order Food" and "Nutritional Information" recordings because many of the terms on the Web page generalized to "food," and this generalization described a variable in both the "Order Food" and "Nutritional Information" recordings.

p0570 When Miro converts a plain text term or phrase into a hyperlink for a particular recording, the hyperlink does not reference a resource found on the Internet. Instead, the hyperlink references a URI that instructs the Web browser to invoke the recording, with the term or phrase used as a variable.

s0145 Limitations of Creo and Miro

p0575 Creo's current implementation results in a number of limitations to its use. First, Creo cannot record interactions with Flash, Java Applets, or other non-HTML elements of Web pages. This limitation is similar to the challenges facing the development of third-party Software Agents for client-side applications. To be able to automate a procedure, the agent must be able to capture the user's actions.

p0580 Secondly, Creo is currently not able to generalize navigation events. However, many of the Programming by Example systems for the Web discussed earlier have implemented this ability.

p0585 The third limitation of Creo is its ability to automate procedures that change based on the variables provided. Creo is able to automate multistep, multivariable procedures, like purchasing stock, ordering a pizza, or sending PayPal. However, Creo cannot currently automate procedures that change based on dependencies of the variables provided, like making the travel arrangements for a trip.

p0590 Because of the breadth of ConceptNet and TAP, Miro is able to avoid many terminological issues like the different spelling of words, synonyms, and in some cases, concepts described in phrases. However, the knowledge in ConceptNet and TAP is by its very nature common and generic. Subsequently, Miro is unable to detect specialized domain information, such as particular part numbers, job codes, or customer numbers, unless this information is provided in an additional knowledge base.

p0595 Whereas Creo is able to automate recordings that take multiple variables, the current implementation of Miro is not yet able to combine multiple pieces of information from a Web page into a single invocation.

s0150 EVALUATION

p0600 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 how Creo and Miro can improve a user's efficiency when completing a task.

s0155 Evaluating the user interface design

p0605 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. We followed an iterative design process during Creo's creation, formally evaluating each iteration, before designing the next. The first version of Creo's user interface was evaluated with three users during a paper prototyping

session. The second version of Creo's user interface was evaluated with four user interface designers, using a computer prototype. 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.

s0160 ## Determining the software's ability to improve the user's efficiency

p0610 The purpose of the fourth user evaluation was to (1) conclude if the overall system made users more efficient when completing a task, (2) conclude if users understood the utility of the software, and (3) determine whether they would use software applications like Creo and Miro if they were included in their Web browser.

p0615 The evaluation was run with 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 years, with a range of 19 to 58 years; 26% of subjects had no programming experience, and all subjects were familiar with using the Web. Subjects were compensated $10.

s0165 ### *Part 1: Evaluating Miro*

p0620 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.

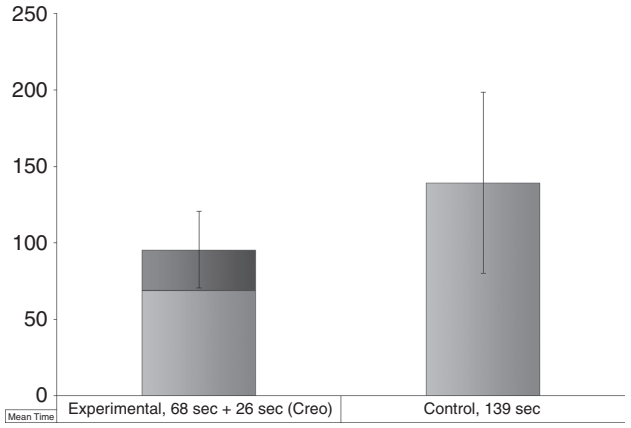
s0170 ### *Part 2: Evaluating Creo*

p0625 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 train Creo to look up a movie at The Internet Movie Database (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.

s0175 ### *Results*

p0630 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%" [Pandit 98].

p0635 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



f0045 **FIGURE 4.8**

Au30 The time it took the control and experimental groups to complete the task.

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 Figure 7, the time for Part 2 is represented as an overhead cost for the experimental group's time for Part 1.

p0645 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.

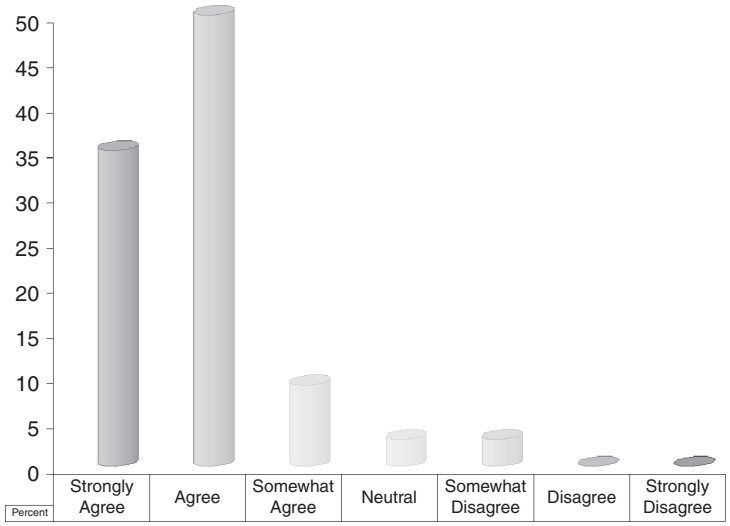
p0650 Asked if they would use the software, 85% of subjects responded that they would use Creo, and 100% of subjects responded that they would use Miro. We have implemented a way for users to easily share the functionality of recordings they create with Creo without sharing any of their personal information (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.

s0180 Limitations of the evaluation

p0655 Although subjects responded favorably to debriefing questions asking if they would use Creo and Miro if they were integrated into their Web browsers, it remains an open question if users in real-world environments would devote the necessary time to apply these tools.

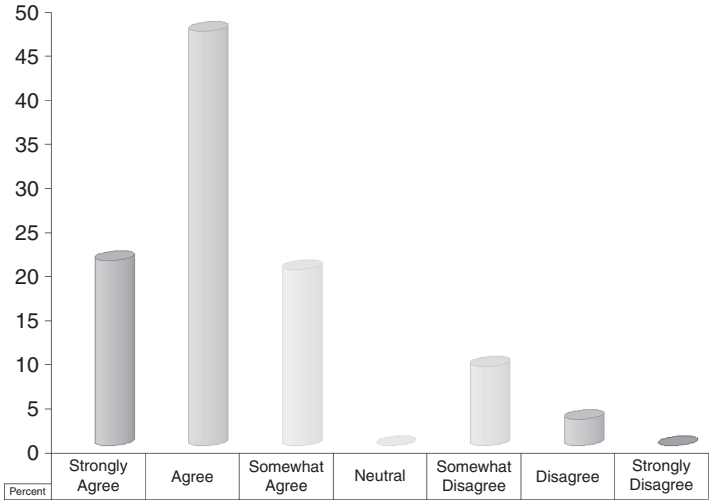
p0660 The 34 users in our study were demographically diverse in age and gender. However, the fact that 74% of the subjects reported some level of programming experience may limit this study's external validity. Unexpectedly, we found that subjects with programming experience had more difficulty using Creo than subjects without any programming experience. Although at first this seems counter-intuitive, we believe it has to do with the subject's expectations. Specifically, subjects with technical experience had more difficulty believing that Creo could generalize their single example. This is because they were familiar with how computers normally function.

p0665 To analyze how Creo and Miro make users more efficient 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 of 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



f0050 **FIGURE 4.9**

Au31 Did the subjects find Miro easy to use?



f0055 **FIGURE 4.10**

Au32 Did the subjects find Creo easy to use?

(Singh, 2002; Singh et al., 2002; Singh et al., 2004; Liu & Singh, 2004; Guha & McCool, 2002; Guha & McCool, 2003b Guha et al., 2003; [McCool 04]. We believe the task users were asked to perform, while only in a single domain, represents a common use of Creo and Miro. However, further studies should be conducted to assess the overall effectiveness of Creo and Miro in real-world situations.

s0185 # FUTURE WORK

s0190 ## When things go wrong

p0670 Although 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.

p0675 For the situations where a recording breaks because of a change with a Web site, we have a developed a debugging mode.

s0195 ## Learning from the web

p0680 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.

s0200 # SUMMARY

p0685 In their 1998 article, the creators of Apple Data Detectors described the goal-oriented nature of their system: "When users invoke it on a region of text in a document, they are saying, in effect, 'Find the important stuff in here and help me do reasonable things to it'... Direct manipulation is a wasteful,

frustrating way for users to interact with machines capable of showing more intelligence" (Nardi et al., 1998). Creo and Miro further enhance this type of goal-oriented user interface, by (1) enabling users to define their own services by example, and (2) increasing the types of data that can be detected to any information stored in the semantic knowledge bases ConceptNet and TAP.

p0690 Creo and Miro, like many other interactive applications (Lieberman & Liu, 2002; Lieberman et al., 2004; Lieberman et al., 2005; Liu et al., 2002; Liu et al., 2003; Stocky et al., 2004; Eagle & Singh, 2004) would not be able to generalize information and anticipate their users' goals without access to the knowledge stored in ConceptNet and TAP. This chapter 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.

s0205 Acknowledgments

p0695 Thanks to James Hendler, Pattie Maes, and Rob Miller for their advice. Thanks to Push Singh and Hugo Liu for ConceptNet, and Rob McCool and R.V. Guha for TAP. The authors would also like to thank Alan Dix.

References

Au14

Bauer, M., Dengler, D., & Paul, G. (2000). Instructible Information Agents for Web Mining. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI 00)*.

Dey, A., Abowd, G., & Wood, A. (1998). CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI 98)*.

Dix, A., Beale, R., & Wood, A. (2000). Architectures to make Simple Visualisations using Simple Systems. In *Proceedings of Advanced Visual Interfaces (AVI 00)*.

Eagle, N., & Singh, P. (2004). Context Sensing Using Speech and Common Sense. In *Proceedings of the NAACL/HLT 2004 Workshop on Higher-Level Linguistic and Other Knowledge for Automatic Speech Processing*.

Faaborg, A. (2005). *A Goal-Oriented User Interface for Personalized Semantic Search*. Masters Thesis. Massachusetts Institute of Technology. http://agents.media.mit.edu/projects/semanticsearch/.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Massachusetts: MIT Press.

Guha, R., & McCool, R. (2002). A System for Integrating Web Services into a Global Knowledge Base. http://tap.stanford.edu/sw002.html.

Guha, R., & McCool, R. (2003a). TAP: Building the Semantic Web. http://tap.stanford.edu/.

Guha, R., & McCool, R. (2003b). TAP: a Semantic Web Platform. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, *42*(5).

Guha, R., McCool, R., & Miller, E. (2003). Semantic Search. In *Proceedings of the 12th International Conference on World Wide Web (WWW 03)*.

Kaminski, C. Much Ado About Smart Tags. Available at http://www.alistapart.com/articles/smarttags/.

Lenat, D. (1995). CYC: a Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, *38* (11).

Lieberman, H., Nardi, B., & Wright, D. (1998). Grammex: Defining Grammars by Example. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 98)*.

Lieberman, H. (2001). *Your Wish is My Command: Programming by Example*. San Francisco, California: Morgan Kaufmann.

Lieberman, H., & Liu, H. (2002). Adaptive Linking between Text and Photos Using Common Sense Reasoning. In *Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, (AH 02)*.

Lieberman, H., Liu, H., Singh, P., & Barry, B. (2004). Beating Some Common Sense into Interactive Applications. *AI Magazine*, Winter.

Lieberman, H., Faaborg, A., Daher, W., & Espinosa, J. (2005). How to Wreck a Nice Beach You Sing Calm Incense. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI 05)*.

Liu, H., Lieberman, H., & Selker, T. (2002). GOOSE: A Goal-Oriented Search Engine With Commonsense. In *Proceedings of the Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, (AH 02)*.

Liu, H., Selker, T., & Lieberman, H. (2003). Visualizing the Affective Structure of a Text Document. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 03)*.

Liu, H., & Singh, P. (2004). ConceptNet: a Practical Commonsense Reasoning Toolkit. *BT Technology Journal*.

McCool, R., Guha, R., & Fikes, R. (2003). Contexts for the Semantic Web. http://tap.stanford.edu/contexts.pdf.

Miller, R., & Myers, B. (1997). *Creating Dynamic World Wide Web Pages by Demonstration*. Technical Report CMU-CS-97-131 (and CMU-HCII-97-101). CMU School of Computer Science.

Mossberg, W. Microsoft Will Abandon Controversial Smart Tags. http://ptech.wsj.com/archive/ptech-20010628.html.

Nardi, B., Miller, J., & Wright, D. (1998). Collaborative, Programmable Intelligent Agents. *Communications of the ACM*, *41*(3).

Pandit, M., & Kalbag, S. (1997). The Selection Recognition Agent: Instant Access to Relevant Information and Operations. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI 97)*.

Safonov, A. (1999). Web Macros by Example: Users Managing the WWW of Applications. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 99)*.

Singh, P. (2002). The Public Acquisition of Commonsense Knowledge. In *Proceedings of AAAI Spring Symposium on Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*.

Singh, P., Lin, T., Mueller, E., Lim, G., Perkins, T., & Zhu, W. L. (2002). Open Mind Common Sense: Knowledge Acquisition from the General Public. In *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*.

Singh, P., Barry, B., & Liu, H. (2004). Teaching Machines about Everyday Life. *BT Technology Journal*.

Stocky, T., Faaborg, A., & Lieberman, H. (2004). A Commonsense Approach to Predictive Text Entry. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 04)*.

Sugiura, A., & Koseki, Y. (1998). Internet Scrapbook: Automating Web Browsing Tasks by Demonstration. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology (UIST 98)*.

s0210

CREO/MIRO

Intended users:	All Users
Domain:	All Web Sites
Description:	Creo is a Programming by Example system for a Web browser that uses Commonsense knowledge to generalize interactions. Miro is a companion “data detectors” system that analyzes a Web page and automatically detects opportunities for invoking Creo procedures.
Example:	Use Creo to record a procedure to order a food item from an online grocery store. Then, if you browse a recipe, Miro can automatically order the recipe ingredient items from the store.
Automation:	Yes. Mainly used to automate repetitive activities.
Mashups:	Yes.
Scripting:	Yes.
Natural Language:	Yes. Natural language on Web pages is analyzed by tools integrated with the Commonsense inference.
Recordability:	Yes.
Inferencing:	Yes. A novel ability is the use of a Commonsense knowledge base and an instance knowledge base to generate candidate generalizations. For example “Diet Coke” can be generalized to “a cola”, “a soft drink”, “a drink”, etc.
Sharing:	Yes.
Comparison to other systems:	Comparable to CoScripter and other Programming by Example systems in the browser environment.
Platform:	Microsoft Internet Explorer. PC and compatibles.
Availability:	Available only for sponsor organizations of the MIT Media Lab and selected academic collaborators.

Author Query Form

Book: No Code Required: Giving Users Tools to Transform the Web
Chapter No: 00004

Query Refs.	Details Required	Author's response
AU1	OK as edited?	
AU2	Pls check style	
AU14	For all hardcopy refs, pls provide page range.	
AU24	This Figure is not cited in text. pls. check	
AU25	This Figure is not cited in text. pls. check	
AU26	This Figure is not cited in text. pls. check	
AU27	This Figure is not cited in text. pls. check	
AU28	This Figure is not cited in text. pls. check	
AU29	This Figure is not cited in text. pls. check	
AU30	This Figure is not cited in text. pls. check	
AU31	This Figure is not cited in text. pls. check	
AU32	This Figure is not cited in text. pls. check	